

जावा प्रोग्रामिंग

जगजीत चड्ढा



जावा प्रोग्रामिंग

जावा प्रोग्रामिंग

जगजीत चड्ढा

भाषा प्रकाशन
नई दिल्ली – 110002

© प्रकाशक

I.S.B.N. : 978-81-323-7167-0

प्रथम संस्करण : 2022

भाषा प्रकाशन

22, प्रकाशदीप बिल्डिंग, अंसारी रोड,
दरियागंज, नई दिल्ली – 110002

द्वारा वर्ल्ड टेक्नोलॉजीज नई दिल्ली के सहयोग से प्रकाशित

अनुक्रम

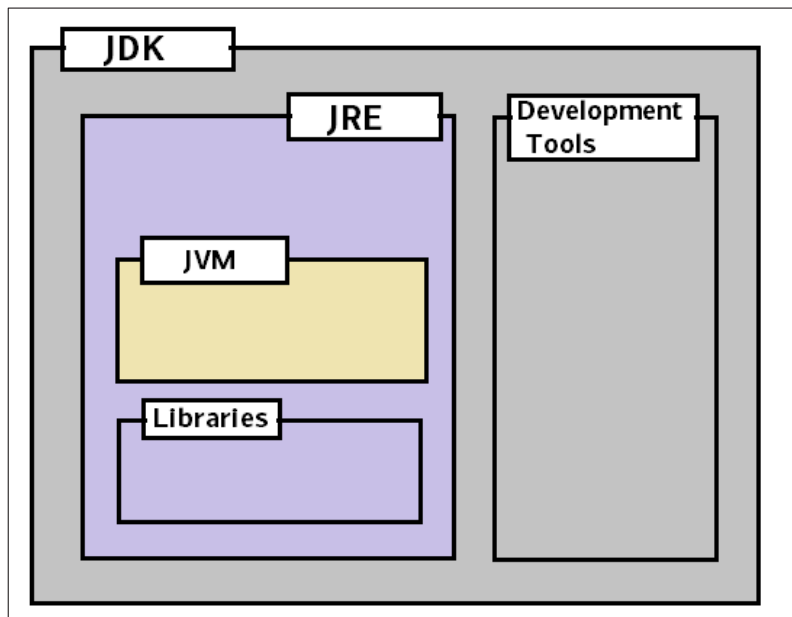
1. जावा का परिचय	1
2. जावा में कंडीशनल स्टेटमेंट, लूप्स, मेथड्स और एरेज़	40
3. ऑब्जेक्ट ओरिएंटेड जावा प्रोग्रामिंग	106
4. जावा में कंटेनर, सॉर्टिंग और थ्रेड्स	158
5. मिसलेनियस एप्लीकेशन्स	213

जावा का परिचय

जावा एक वस्तु-उन्मुख सामान्य प्रयोजन उच्च-स्तरीय प्रोग्रामिंग भाषा है, जिसका उपयोग जटिल प्रोग्राम बनाने के लिए किया जाता है। यह एक बार लिखने, कहीं भी चलाने की नीति का अनुसरण करता है, क्योंकि संकलित जावा कोड का उपयोग किसी भी प्लेटफॉर्म पर किया जा सकता है। यह एक परिचयात्मक अध्याय है जो संक्षेप में जावा के सभी महत्वपूर्ण घटकों का परिचय देगा।

जावा एक सामान्य-उद्देश्य वाली कंप्यूटर प्रोग्रामिंग भाषा है जो समवर्ती, वर्ग-आधारित, वस्तु-उन्मुख है, और विशेष रूप से यथासंभव कम कार्यान्वयन निर्भरता के लिए डिज़ाइन की गई है। इसका उद्देश्य एप्लिकेशन डेवलपर्स को "राइट वन्स, रन एनीव्हेयर" (डब्ल्यूओआरए) देना है, जिसका अर्थ है कि संकलित जावा कोड उन सभी प्लेटफॉर्मों पर चल सकता है जो पुनः संकलन की आवश्यकता के बिना जावा का समर्थन करते हैं।

उदाहरण के लिए, आप यूनिक्स पर एक जावा प्रोग्राम लिख और संकलित कर सकते हैं और इसे माइक्रोसॉफ्ट विंडोज, मैकइंटोश, या यूनिक्स मशीन पर स्रोत कोड में बिना किसी संशोधन के चला सकते हैं। डब्ल्यूओआरए एक जावा प्रोग्राम को बाइटकोड नामक एक मध्यवर्ती भाषा में संकलित करके प्राप्त किया जाता है। बाइटकोड का प्रारूप प्लेटफॉर्म-स्वतंत्र है। जावा वर्चुअल मशीन (जेवीएम) नामक एक वर्चुअल मशीन का उपयोग प्रत्येक प्लेटफॉर्म पर बाइटकोड चलाने के लिए किया जाता है।



जेडीके बनाम जेआरई बनाम जेवीएम।

गारबेज कलेक्शन (Garbage Collection)

प्रोग्रामर निर्धारित करता है कि ऑब्जेक्ट कब बनाए जाते हैं, और जावा रनटाइम मेमोरी को पुनर्प्राप्त करने के लिए जिम्मेदार होता है जब ऑब्जेक्ट अब उपयोग में नहीं होते हैं। एक बार जब किसी वस्तु का कोई संदर्भ नहीं रहता है, तो अगम्य स्मृति कचरा संग्रहकर्ता द्वारा स्वचालित रूप से मुक्त होने के योग्य हो जाती है।

मेमोरी लीक के समान कुछ तब भी हो सकता है जब किसी प्रोग्रामर के कोड में किसी ऐसी वस्तु का रेफरेंस होता है जिसकी अब आवश्यकता नहीं है, आमतौर पर जब जिन वस्तुओं की अब आवश्यकता नहीं होती है उन्हें कंटेनरों में संग्रहित किया जाता है जो अभी भी उपयोग में हैं। यदि किसी गैर-मौजूद वस्तु के लिए विधियों को कहा जाता है, तो "नलपाइंडरएक्सेप्शन" दिखा दिया जाता है।

गारबेज कलेक्शन किसी भी समय हो सकता है। आदर्श रूप से, यह तब होगा जब कोई प्रोग्राम निष्क्रिय हो। यदि एक नई वस्तु आवंटित करने के लिए ढेर पर अपर्याप्त फ्री मेमोरी है, तो यह ट्रिगर होने की गारंटी है; यह एक प्रोग्राम को क्षण भर के लिए रुकने का कारण बन सकता है। जावा में एक्सप्लिसिट मेमोरी प्रबंधन संभव नहीं है।

जेडीके, जेआरई और जेवीएम

जावा डेवलपमेंट किट

जावा डेवलपमेंट किट (जेडीके) एक सॉफ्टवेयर डेवलपमेंट एनवायरनमेंट है जिसका उपयोग जावा एप्लिकेशन और एप्लेट्स को विकसित करने के लिए किया जाता है। इसमें जावा रनटाइम एनवायरनमेंट (जेआरई), एक इंटरप्रेटर/लोडर (जावा), एक कंपाइलर (जावैक), एक आर्काइवर (जार), एक दस्तावेज जनरेटर (जावाडोक) और जावा विकास में आवश्यक अन्य टूल्स शामिल हैं।

जावा रनटाइम एनवायरनमेंट

जेआरई "जावा रनटाइम एनवायरनमेंट" के लिए प्रयोग किया जाता है और इसे "जावा आरटीई" के रूप में भी लिखा जा सकता है। जावा रनटाइम एनवायरनमेंट जावा एप्लिकेशन को निष्पादित करने के लिए न्यूनतम आवश्यकताएं प्रदान करता है; इसमें जावा वर्चुअल मशीन (जेवीएम), कोर क्लास और सहायक फाइलें शामिल हैं।

जावा वर्चुअल मशीन

यह है:

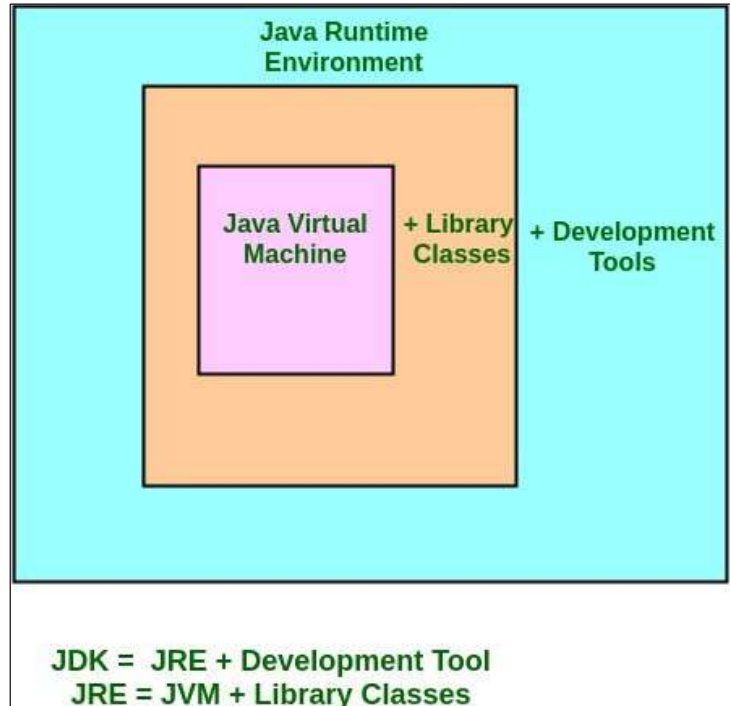
- एक विनिर्देश जहां जावा वर्चुअल मशीन का कार्य निर्दिष्ट किया गया है। लेकिन इम्प्लीमेंटेशन प्रोवाइडर एल्गोरिदम चुनने के लिए स्वतंत्र है। इसका क्रियान्वयन सन और अन्य कंपनियों द्वारा प्रदान किया गया है।
- एक इम्प्लीमेंटेशन एक कंप्यूटर प्रोग्राम है जो जेवीएम विनिर्देश की आवश्यकताओं को पूरा करता है।
- रनटाइम इंस्टेंस जब भी आप जावा क्लास को चलाने के लिए कमांड प्रॉम्प्ट पर जावा कमांड लिखते हैं, तो जेवीएम का एक इंस्टेंस बन जाता है।

जेडीके, जेआरई और जेवीएम के बीच अंतर

इन तीनों के बीच के अंतर को समझने के लिए, आइए निम्नलिखित पर विचार करें:

- जेडीके - जावा डेवलपमेंट किट (संक्षेप में जेडीके) एक किट है जो जावा प्रोग्राम को विकसित और निष्पादित (चलाने) करने के लिए एनवायरनमेंट प्रदान करती है। जेडीके एक किट (या पैकेज) है जिसमें दो चीजें शामिल हैं।

- डेवलपमेंट उपकरण (अपने जावा प्रोग्राम को विकसित करने के लिए एक वातावरण प्रदान करना)।
- जेआरई (अपने जावा प्रोग्राम को निष्पादित करने के लिए)।



जेडीके का उपयोग केवल जावा डेवलपर द्वारा किया जाता है।

- जेआरई - जावा रनटाइम एनवायरनमेंट (जेआरई कहने के लिए) एक इंस्टॉलेशन पैकेज है जो आपकी मशीन पर केवल जावा प्रोग्राम (या एप्लिकेशन) को चलाने (विकसित नहीं) के लिए पर्यावरण प्रदान करता है। जेआरई का उपयोग केवल उनके द्वारा किया जाता है जो केवल जावा प्रोग्राम यानी आपके सिस्टम के अंतिम उपयोगकर्ता चलाना चाहते हैं।
- जेवीएम - जावा वर्चुअल मशीन (जेवीएम) जेडीके और जेआरई दोनों का एक बहुत ही महत्वपूर्ण हिस्सा है क्योंकि यह दोनों में निहित या अंतर्निहित है। जेआरई या जेडीके का उपयोग करके आप जो भी जावा प्रोग्राम चलाते हैं, वह जेवीएम में चला जाता है और जेवीएम जावा प्रोग्राम लाइन को लाइन से निष्पादित करने के लिए जिम्मेदार होता है इसलिए इसे दुइंटरप्रेटर के रूप में भी जाना जाता है।

जेआरई और जेडीके कैसे काम करता है?

जेआरई में क्या शामिल है?

जेआरई में निम्नलिखित घटक होते हैं:

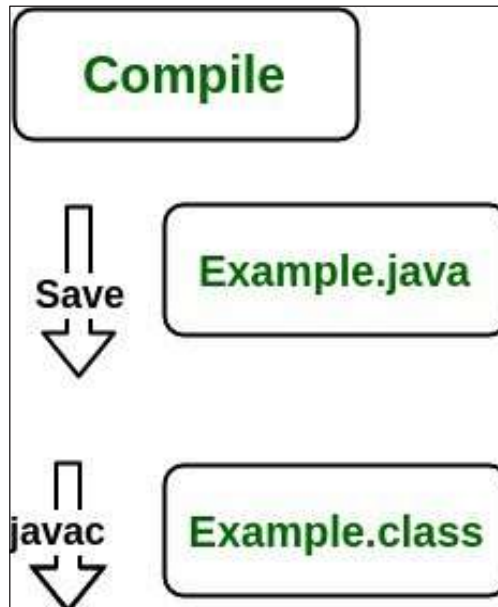
- परिनियोजन तकनीक, जिसमें परिनियोजन, जावा वेब स्टार्ट और जावा प्लग-इन शामिल हैं।
- एक्सट्रेक्ट विंडो टूलकिट (एडब्ल्यूटी), स्विंग, जावा 2डी, एक्सेसिबिलिटी, इमेज आई/ओ, प्रिंट सर्विस, साउंड, ड्रैग एंड ड्रॉप (डीएनडी) और इनपुट विधियों सहित यूजर इंटरफेस टूलकिट।

- इंटरफेस डेफिनिशन लैंग्वेज (आईडीएल), जावा डेटाबेस कनेक्टिविटी (जेडीबीसी), जावा नेमिंग एंड डायरेक्ट्री इंटरफेस (जेएनडीआई), रिमोट मेथड इनवोकेशन (आरएमआई), रिमोट मेथड इनवोकेशन ओवर इंटरनेट इंटर-ऑर्ब प्रोटोकॉल (आरएमआई-आईआईओपी) और स्क्रिप्टिंग सहित इंटीग्रेशन लाइब्रेरी।
- अंतर्राष्ट्रीय समर्थन, इनपुट/आउटपुट (आई/ओ), एक्सटेंशन मैकेनिज्म, बीन्स, जावा मैनेजमेंट एक्सटेंशन (जेएमएक्स), जावा नेटिव इंटरफेस (जेएनआई), मैथ, नेटवर्किंग, ओवरराइड मैकेनिज्म, सिक््योरिटी, सीरियलाइजेशन और एक्सएमएल के लिए जावा सहित अन्य बेस लाइब्रेरी प्रसंस्करण (एक्सएमएल जेएक्सपी)।
- लैंग और यूटिल बेस लाइब्रेरी, जिसमें लैंग और यूटिल, मैनेजमेंट, वर्जनिंग, ज़िप, इंस्ट्रूमेंट, रिफ्लेक्शन, क्लेक्शंस, कंसीडर यूटिलिटीज, जावा आर्काइव (जेएआर), लॉगिंग, प्रेफरेंस एपीआई, रेफ ऑब्जेक्ट्स और रेगुलर एक्सप्रेशंस शामिल हैं।
- जावा वर्चुअल मशीन (जेवीएम), जिसमें जावा हॉटस्पॉट क्लाइंट और सर्वर वर्चुअल मशीन शामिल हैं।

जेआरई कैसे काम करता है?

यह समझने के लिए कि जेआरई कैसे काम करता है, आइए हम उदाहरण के रूप में सहेजी गई जावा स्रोत फ़ाइल पर विचार करें। फ़ाइल को बाइट कोड के एक सेट में संकलित किया जाता है जो ".class" (डॉट क्लास) फ़ाइल में संग्रहित होता है। यहाँ यह "Example.class" होगा।

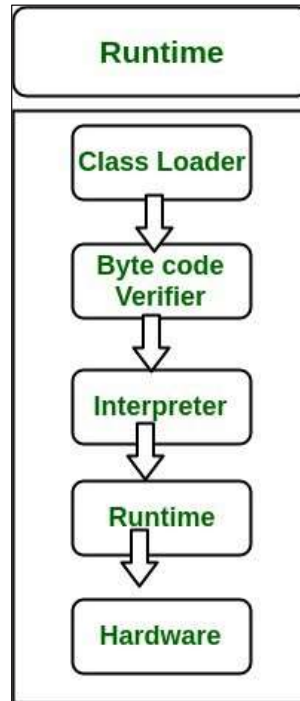
निम्नलिखित आरेख दर्शाता है कि संकलन के समय पर क्या किया जाता है।



निम्नलिखित क्रियाएं रनटाइम पर होती हैं:

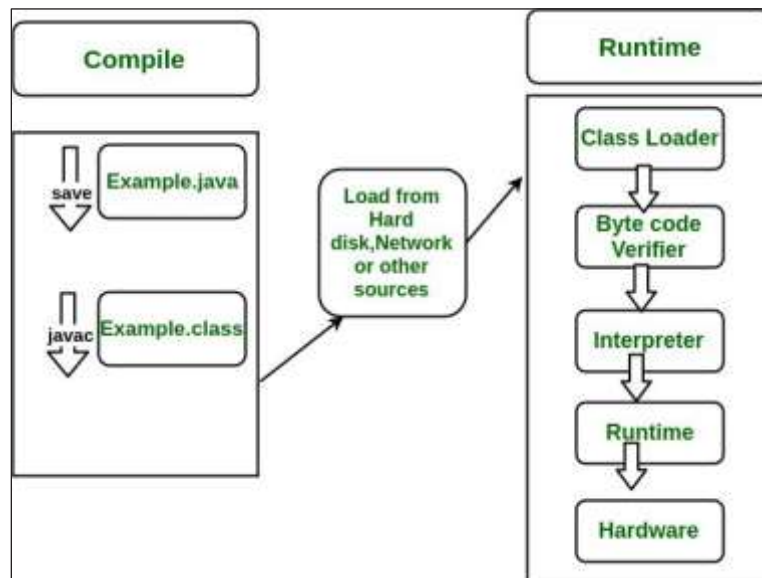
- क्लास लोडर: क्लास लोडर प्रोग्राम के निष्पादन के लिए आवश्यक सभी आवश्यक कक्षाओं को लोड करता है। यह स्थानीय फ़ाइल सिस्टम के नामस्थान को नेटवर्क के माध्यम से आयात किए गए नामस्थान से अलग करके सुरक्षा प्रदान करता है। ये फ़ाइलें हार्ड डिस्क, नेटवर्क या अन्य स्रोतों से लोड की जाती हैं।
- बाइट कोड सत्यापनकर्ता: जेवीएम बाइट कोड सत्यापनकर्ता के माध्यम से कोड डालता है जो प्रारूप की जांच करता है और एक अवैध कोड की जांच करता है। अवैध कोड, उदाहरण के लिए, वह कोड है जो वस्तुओं पर पहुंच अधिकारों का उल्लंघन करता है या पॉइंटर्स के कार्यान्वयन का उल्लंघन करता है।

बाइट कोड सत्यापनकर्ता सुनिश्चित करता है कि कोड जेवीएम विनिर्देश का पालन करता है और सिस्टम अखंडता का उल्लंघन नहीं करता है।

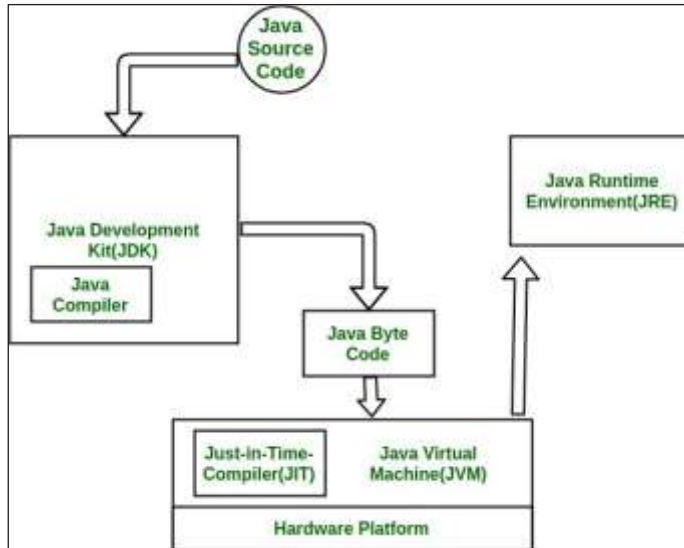


- इंटरप्रेटर: रनटाइम पर बाइट कोड इंटरप्रेटर द्वारा लोड, चेक और चलाया जाता है। इंटरप्रेटर के निम्नलिखित दो कार्य हैं:
 - बाइट कोड निष्पादित करें।
 - अंतर्निहित हार्डवेयर के लिए उपयुक्त कॉल करें।

दोनों कार्यों को इस प्रकार दिखाया जा सकता है:



जेडीके और जेआरई के बीच की इंटरैक्शन को समझने के लिए निम्नलिखित आरेख पर विचार करें।



जेवीएम जावा प्रोग्राम के रनटाइम पर जेआरई का एक उदाहरण बन जाता है। इसे व्यापक रूप से एक रनटाइम इंटरप्रेटर के रूप में जाना जाता है। जेवीएम बड़े पैमाने पर उन प्रोग्रामर्स से आंतरिक कार्यान्वयन को निकालने में मदद करता है जो जेडीके से अपने कार्यक्रमों के लिए लाइब्रेरीज़ का उपयोग करते हैं।

विंडोज़ पर जावा कैसे इनस्टॉल करें

चरण

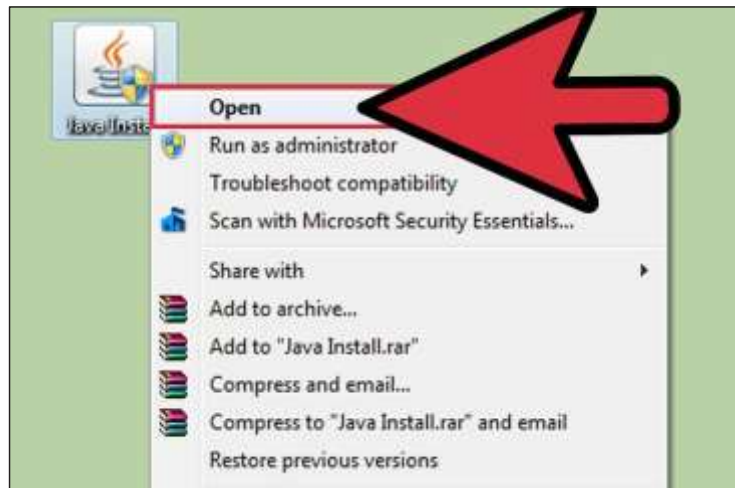


1. ब्राउज़रों के लिए जावा रनटाइम एनवायरनमेंट (जेआरई) है। डेवलपर टूल (जेडीके) इंस्टॉल करने के निर्देशों के लिए है। जावा भी जावास्क्रिप्ट से अलग है। यदि आपको जावास्क्रिप्ट सक्षम करने की आवश्यकता है।



2. जावा होम पेज पर जाएं। जावा सिस्टम फाइलों को इंस्टॉल करता है जो सभी ब्राउज़र उपयोग करते हैं, इसलिए विशिष्ट ब्राउज़रों के लिए विशेष निर्देशों का पालन करने की कोई आवश्यकता नहीं है। आप जावा होम पेज से जावा इंस्टॉलर पर जा सकते हैं।

- जावा इंस्टॉलर इंस्टॉलेशन प्रक्रिया के दौरान फाइलों को डाउनलोड करेगा। यदि आपको नेटवर्क कनेक्शन के बिना किसी डिवाइस पर जावा इंस्टॉल करने की आवश्यकता है, तो मैन्युअल डाउनलोड पृष्ठ पर उपलब्ध ऑफ़लाइन इंस्टॉलर डाउनलोड करें।
- आपकी ब्राउज़र सेटिंग्स के आधार पर, आपको जावा इंस्टॉलेशन के शुरू होने से पहले डाउनलोड को स्वीकार करने की आवश्यकता हो सकती है।
- मैक ओएस के लिए एक्स 10.6, जावा प्रीइंस्टॉलड आता है। ओएस एक्स 10.7 और इसके बाद के वर्ज़न के लिए, जावा पूर्व-इंस्टॉल नहीं है। जावा को इंस्टॉल करने के लिए आपको ओएस एक्स 10.7.3 या नए की आवश्यकता होगी। आपको 64-बिट ब्राउज़र जैसे कि सफारी या फ़ायरफ़ॉक्स (यानी क्रोम नहीं) का उपयोग करना चाहिए।
- लिनक्स के लिए, जावा को डाउनलोड करने, मैन्युअल रूप से इंस्टॉल करने और फिर काम करने के लिए सक्षम करने की आवश्यकता है। लिनक्स में जावा इंस्टॉल करने के बारे में विस्तृत निर्देश के लिए है।



3. इंस्टॉलर प्रारंभ करें। एक बार जब इंस्टॉलर डाउनलोड करना समाप्त कर लेता है, तो इसे इंस्टॉलेशन शुरू करने के लिए चलाएं। ओएस एक्स पर, इंस्टॉलेशन शुरू करने के लिए .dmg फ़ाइल पर डबल-क्लिक करें।

- इंस्टॉलेशन शुरू करने से पहले किसी भी ब्राउज़र विंडो को बंद कर दें, क्योंकि इंस्टॉलेशन पूरा होने के बाद उन्हें फिर से शुरू करना होगा।



4. इंस्टॉलेशन चरणों का पालन करें। इंस्टॉलेशन प्रोग्राम के प्रत्येक स्क्रीन को पढ़ें। जब तक आप बॉक्स को अनचेक नहीं करते हैं, जावा ब्राउज़र टूलबार जैसे अतिरिक्त सॉफ्टवेयर को इंस्टॉल करने का प्रयास करेगा। यदि आप अपने ब्राउज़र को बदलना नहीं चाहते हैं, तो प्रत्येक स्क्रीन को ध्यान से पढ़ना सुनिश्चित करें।



5. इंस्टॉलेशन का परीक्षण करें। जब आप जावा इंस्टॉलेशन करना समाप्त कर लें, तो यह सुनिश्चित करने के लिए कि सब कुछ ठीक हो गया है, इंस्टॉलेशन का परीक्षण करें। आप जावा वेबसाइट पर जावा परीक्षण एप्लेट पा सकते हैं, या "जावा परीक्षण" की खोज करके और पहले परिणाम का चयन कर सकते हैं।

- आपको प्लगइन को चलने की अनुमति देनी होगी, और लोड होने से पहले आपसे कई बार पूछा जा सकता है। ऐसा इसलिए है, क्योंकि सामान्य तौर पर, जावा एक खतरनाक उपकरण हो सकता है जो दूसरों को आपके कंप्यूटर तक पहुंच प्रदान कर सकता है यदि आप सावधान नहीं हैं। हमेशा सुनिश्चित करें कि आप उस वेबसाइट पर भरोसा करते हैं जिस पर आप जावा एप्लेट चला रहे हैं।

लिनक्स पर जावा कैसे इंस्टॉल करें

तरीका 1. नॉन-आरपीएम लिनक्स पर इंस्टॉल करना



1. लिनक्स के लिए डाउनलोड जावा पेज खोलें। आपको यहां सूचीबद्ध कई विकल्प दिखाई देंगे।



2. लिनक्स पर क्लिक करें। यह पृष्ठ के मध्य में एक लिंक है। ऐसा करने से जावा इंस्टालेशन फाइल डाउनलोड होने के लिए प्रॉम्प्ट होगी।

- यदि आप 64-बिट जावा इंस्टॉल करना चाहते हैं तो आप लिनक्स एक्स 64 वर्ज़न पर भी क्लिक कर सकते हैं।



3. फ़ाइल का नाम नोट करें। जावा का नवीनतम वर्ज़न वर्ज़न 8 है, लेकिन आपको अद्यतन वर्ज़न संख्या की भी आवश्यकता है, जो "8u" खंड के बाद फ़ाइल नाम में लिखा गया है।

- उदाहरण के लिए, आपकी फ़ाइल का नाम "jre-8u151" हो सकता है, यह दर्शाता है कि यह वर्ज़न 8 है, अद्यतन 151 है।



4. कमांड लाइन खोलें। यह चरण आपके लिनक्स के वर्ज़न के आधार पर अलग-अलग होगा, लेकिन आप आमतौर पर टर्मिनल ऐप खोलकर या स्क्रीन के ऊपर या नीचे एक बार पर क्लिक करके कमांड लाइन पा सकते हैं।

```

charlie
File Edit View Search Terminal Help
charlie@charlie:~$ cd /usr/java
charlie@charlie:~/java$ java -version

Command 'java' not found, but can be installed with:

sudo apt install default-jre
sudo apt install openjdk-11-jre-headless
sudo apt install openjdk-8-jre-headless

charlie@charlie:~/java$ sudo update-alternatives --ins
[sudo] password for charlie:
update-alternatives: using /usr/java/jdk-10.0.2/bin/java
charlie@charlie:~/java$ sudo update-alternatives --ins
update-alternatives: using /usr/java/jdk-10.0.2/bin/javac
charlie@charlie:~/java$ sudo update-alternatives --ins
1
update-alternatives: using /usr/java/jdk-10.0.2/bin/javaw

```

5. इंस्टॉलेशन निर्देशिका बदलें। कंसोल में सीडी टाइप करें, स्पेसबार को एक बार दबाएं, और फिर टाइप करें पथ में, उदा, /यूएसआर/जावा/ और एंटर दबाएं।

```
[root@ip-172-31-25-239 java]# ls
jdk-8u141-linux-x64.tar.gz
[root@ip-172-31-25-239 java]# tar -zxvf jdk-8u1
gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now
[root@ip-172-31-25-239 java]# tar -xvf jdk-8u14
gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now
[root@ip-172-31-25-239 java]# file jdk-8u141-l1
jdk-8u141-linux-x64.tar.gz: HTML document, ASCII
g lines, with CRLF line terminators
[root@ip-172-31-25-239 java]# clear
```

6. इंस्टॉलेशन कमांड दर्ज करें। tar zxvf टाइप करें, स्पेसबार को एक बार दबाएं और फिर फाइल का पूरा नाम टाइप करें। यह जावा संस्करण और जब आप इसे डाउनलोड करते हैं, के आधार पर अलग-अलग होंगे।

- आप टार में टाइप करेंगे zxvf jre-8u151-linux-i586.tar.



7. एंटर दबाएँ। ऐसा करने से आपके कंप्यूटर पर "jre1.8.0_[up .] नाम के फोल्डर में जावा इंस्टाल हो जाएगा- date]" जहां "[अपडेट]" अपडेट वर्जन नंबर है।

विधि 2. लिनक्स पर आरपीएम इंस्टॉल करना



1. लिनक्स के लिए जावा डाउनलोड पेज खोलें। आपको यहां सूचीबद्ध कई विकल्प दिखाई देंगे।



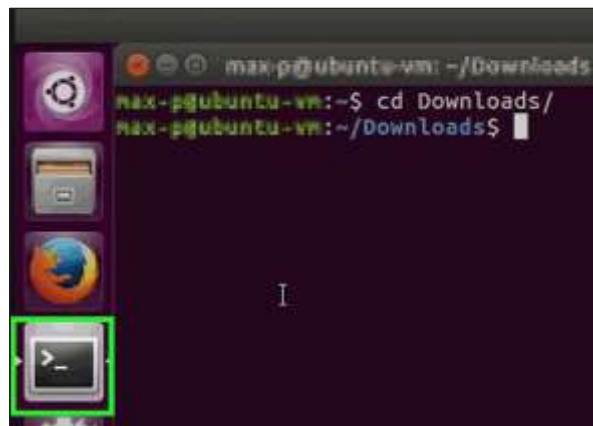
2. लिंक आरपीएम पर क्लिक करें। यह पृष्ठ के मध्य में है। ऐसा करने से जावा आरपीएम इंस्टॉलेशन फाइल को डाउनलोड करने के लिए कहेगा।

- यदि आप 64-बिट जावा इंस्टॉलेशन करना चाहते हैं तो लिंक आरपीएम एक्स 64 वर्ज़न पर भी क्लिक कर सकते हैं।



3. फ़ाइल का नाम नोट करें। जावा का नवीनतम वर्ज़न वर्ज़न 8 है, लेकिन आपको अद्यतन वर्ज़न संख्या की भी आवश्यकता है, जो "8u" खंड के बाद फ़ाइल नाम में लिखा गया है।

- उदाहरण के लिए, आपकी फ़ाइल का नाम "jre-8u151" हो सकता है, यह दर्शाता है कि यह वर्ज़न 8 है, अद्यतन 151 है।



4. कमांड लाइन खोलें। यह चरण आपके लिंक के आधार पर भिन्न होगा, लेकिन आप आमतौर पर टर्मिनल ऐप खोलकर या स्क्रीन के ऊपर या नीचे एक बार पर क्लिक करके कमांड लाइन पा सकते हैं।

```
dadarsh-fossbytes: ~  
-fossbytes:~$ sudo su  
guest-rk3inc: █
```

5. रूट कमांड दर्ज करें। `sudo su` में टाइप करें और एंटर दबाएं `↵`। यह आपके उपयोगकर्ता पासवर्ड का अनुरोध करने के लिए कमांड लाइन को संकेत देगा।

```
Waldo% sudo su -  
Password: █
```

6. अपना खाता पासवर्ड दर्ज करें। अपने अकाउंट का पासवर्ड टाइप करें और एंटर दबाएं। जब तक आपके पास अपने खाते पर रूट पहुंच है, ऐसा करने से आप जावा इंस्टॉल कर सकेंगे।

- यदि आपके पास अपने खाते पर रूट पहुंच नहीं है, तो आपको उस खाते के लिए पासवर्ड दर्ज करना होगा जिसकी रूट पहुंच है।

```

charlie
File Edit View Search Terminal Help
charlie@charlie:~$ cd /usr/java
charlie@charlie:/usr/java$ java -version

Command 'java' not found, but can be installed with:

sudo apt install default-jre
sudo apt install openjdk-11-jre-headless
sudo apt install openjdk-8-jre-headless

charlie@charlie:/usr/java$ sudo update-alternatives --ins
[sudo] password for charlie:
update-alternatives: using /usr/java/jdk-10.0.2/bin/java
charlie@charlie:/usr/java$ sudo update-alternatives --ins
update-alternatives: using /usr/java/jdk-10.0.2/bin/javac
charlie@charlie:/usr/java$ sudo update-alternatives --ins
1
update-alternatives: using /usr/java/jdk-10.0.2/bin/javaw

```

7. इंस्टॉलेशन निर्देशिका बदलें। कंसोल में सीडी टाइप करें, स्पेसबार को एक बार दबाएं, और फिर पथ में टाइप करें, उदा, /यूएसआर/जावा/ और एंटर दबाएं।



```

rpm -ivh jre-8u91-linux-x64.rpm
Preparing...
1:jre1.8.0_91
Unpacking JAR files...
  plugin.jar...
  javaws.jar...
  deploy.jar...
  rt.jar...

```

8. इंस्टॉलेशन कमांड दर्ज करें। rpm -ivh टाइप करें, स्पेसबार को एक बार दबाएं, फाइल का पूरा नाम टाइप करें और एंटर दबाएं। यह आपके कंप्यूटर पर जावा इंस्टॉल करेगा।

- फाइल का नाम इस बात पर निर्भर करेगा कि आपने फाइल कब डाउनलोड की है। अक्टूबर 2017 तक, आप आरपीएम में rpm -ivh jre-8u151-linux-i586.rpm टाइप करेंगे और एंटर दबाएं।

9. डाउनलोड को अपग्रेड करें। rpm -Uvh jre-8u73-linux-i586.rpm टाइप करें और एंटर दबाएं। यह जावा पैकेज के अपडेट की जांच करेगा और यदि संभव हो तो उन्हें लागू करेगा।

विधि 3. उबंटू (ओपन जेडीके) पर इंस्टॉल करना



1. कमांड लाइन खोलें। कीबोर्ड पर Ctrl+Alt+T दबाएं, या स्क्रीन के बाईं ओर एक सफेद ">_" वाले ब्लैक बॉक्स आइकन पर क्लिक करें।

```
sandy@sandy-Insipram-3121:~$ sudo apt-get update
[sudo] password for sandy:
E: Could not get lock /var/lib/apt/lists/lock - open (11: Resource temporarily unavailable)
E: Unable to lock directory /var/lib/apt/lists/
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is it being used by another process?
sandy@sandy-Insipram-3121:~$ sudo apt-get upgrade
[sudo] password for sandy:
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is it being used by another process?
sandy@sandy-Insipram-3121:~$
```

2. अपडेट कमांड दर्ज करें। `sudo apt-get update && sudo apt-get upgrade -y` टाइप करें और एंटर दबाएं। यह पैकेज सूची को ताज़ा करेगा और आपके लिए सभी उपलब्ध अद्यतनों को इंस्टॉल करेगा।

```
File Edit View Search Terminal Help
jaski@jaski:~/Downloads$ sudo dpkg -i google-chrome
[sudo] password for jaski:
```

3. पूछे जाने पर अपना पासवर्ड भरें। यदि आपसे आपका यूजर पासवर्ड मांगा जाता है, तो उसे टाइप करें और एंटर दबाएं।

```

$ java -version
The program 'java' can be found in the following packages:
  default-jre
  * gcj-4.8-jre-headless
  * openjdk-7-jre-headless
  * gcj-4.6-jre-headless
  * openjdk-8-jre-headless
Try: sudo apt-get install <selected package>
bar@bar1:~$ sudo apt-get update
[sudo] password for bar:

```

4. सुनिश्चित करें कि आपके पास जावा पहले से इंस्टॉल नहीं है। `java -version` टाइप करें और एंटर दबाएं। यदि आप एक पंक्ति देखते हैं जो कहती है कि "प्रोग्राम 'जावा' निम्नलिखित पैकेजों में पाया जा सकता है" दिखाई देता है, तो जावा आपके कंप्यूटर पर इंस्टॉल नहीं है।

- यदि जावा इंस्टॉल है, तो आपको एक लाइन दिखाई देगी जो इसके बजाय जावा के वर्तमान वर्ज़न की रिपोर्ट करती है।

```

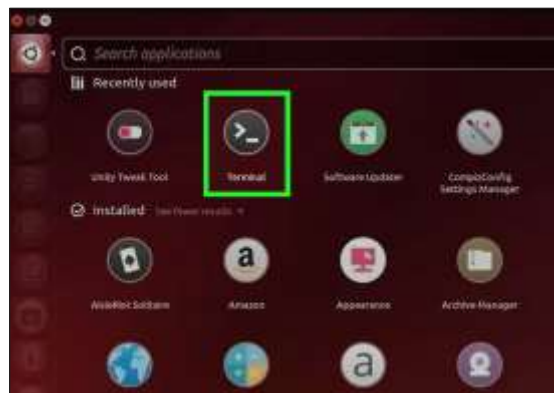
bar@bar1:~$ sudo apt-get install default-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java libatk-wrapper-java libatk-wrapper-java-jni libgl1
  openjdk-8-jre-headless
Suggested packages:
  default-java-plugin lcedtin-g-plugin fonts-ipafont-gothic
  fonts-ipafont-mincho fonts-why-microbel fonts-why-zenbel
The following NEW packages will be installed:
  ca-certificates-java default-jre default-jre-headless fonts-dejavu-extra
  java-common libatk-wrapper-java libatk-wrapper-java-jni libgl1
  openjdk-8-jre-headless
0 upgraded, 10 newly installed, 0 to remove and 11 not upgraded.
Need to get 28.4 MB of archives.
After this operation, 100 MB of additional disk space will be used.
Do you want to continue? [Y/n]

```

5. इंस्टॉलेशन कमांड टाइप करें। कमांड लाइन में `sudo apt-get install default-jre` टाइप करें, फिर एंटर दबाएं। यह आपके उबंटू कंप्यूटर पर डिफॉल्ट निर्देशिका में जावा को इंस्टॉल करेगा।

- यदि यह काम नहीं करता है, तो इसके बजाय `sudo apt-get install openjdk-8-jre` दर्ज करने का प्रयास करें।

विधि 4. पीपीए के माध्यम से उबंटू 16.04 पर इंस्टॉल करना



1. सबसे पहले, यह एक तृतीय पक्ष पैकेज है, आपके डिस्ट्रो का मेटेनर इस पैकेज का ऑडिट नहीं कर सकता, सावधानी के साथ उपयोग करें। कहा जा रहा है, पहले `Ctrl+Alt+T` दबाकर एक टर्मिनल खोलें।

```

root@kali:~# sudo apt-get update
[sudo] password for root:
E: Could not get lock /var/lib/apt/lists/lock - open (11: Resource temporarily unavailable)
E: Unable to lock directory /var/lib/apt/lists/
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
root@kali:~# sudo apt-get upgrade
[sudo] password for root:
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
root@kali:~#

```

```

root@kali:~# whoami
ryek
root@kali:~# sudo -l
[sudo] password for ryek:
root@kali:~# whoami
root

```

2. सुनिश्चित करें कि आपके पास एक अद्यतन प्रणाली है। `sudo apt-get update` && `sudo apt-get upgrade -y` टाइप करें, आपको पासवर्ड के लिए संकेत मिल सकता है, इसे टाइप करें और एंटर दबाएं, जैसे ही आप टाइप करते हैं, कोई बिंदु या तारक दिखाई नहीं देगा, यह सामान्य है।

- तकनीकी रूप से वैकल्पिक होते हुए भी, कुछ भी इंस्टॉल करने से पहले इस चरण की हमेशा अनुशंसा की जाती है, अपने सिस्टम को अपडेट रखने से कई समस्याओं को रोकने में मदद मिलेगी।

```

root@kali:~# sudo apt-add-repository ppa:webupd8team/java
[sudo] password for root:

```

3. अपने सिस्टम में पीपीए रिपॉजिटरी जोड़ें। `sudo add-apt-repository ppa:webupd8team/java`, टाइप करें, और फिर एंटर दबाएं।

```

root@kali:~# sudo apt-get update
[sudo] password for root:
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease [83.2 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease
Fetched 180 kB in 1s (131 kB/s)
Reading package lists... Done

```

4. अपनी पैकेज सूचियों को फिर से अपडेट करें। `sudo apt-get update` टाइप करें और सूचियों के ताज़ा होने की प्रतीक्षा करें।


```

sudo apt-get install oracle-java9-installer
Reading package lists... Done
Building dependency tree
Reading state information... Done
oracle-java9-installer is already the newest version (9b177-1-webupd8-8).
The following packages were automatically installed and are no longer required:
libltdl0 libltdl3 a:libseccomp libseccomp2 libnfc-common1:libnfc
linux-headers-4.4.0-71 linux-headers-4.4.0-71-generic linux-headers-4.4.0-72
linux-headers-4.4.0-72-generic linux-headers-4.4.0-75
linux-headers-4.4.0-76-generic linux-headers-4.4.0-78
linux-headers-4.4.0-78-generic linux-headers-4.4.0-81
linux-headers-4.4.0-81-generic linux-image-4.4.0-71-generic
linux-image-4.4.0-72-generic linux-image-4.4.0-75-generic
linux-image-4.4.0-76-generic linux-image-extra-4.4.0-71-generic
linux-image-extra-4.4.0-72-generic linux-image-extra-4.4.0-75-generic
linux-image-extra-4.4.0-78-generic linux-image-extra-4.4.0-79-generic
linux-image-extra-4.4.0-81-generic snap-confine ttf-wqy-microslat
Use 'sudo apt autoremove' to remove them.
The following packages will be upgraded:
  0 to be upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Usefully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n]
Setting up oracle-java9-installer (9b177-1-webupd8-8) ...
Getting settings from /var/cache/oracle_jdk9_installer/wgetrc
Downloading Oracle Java 9...
2017-07-20 22:22:38... http://download.java.net/java/jdk9/archive/127/binaries/jdk-9-linux
enabling download.java.net (download.java.net)... 209.95.138.153, 209.95.138.146
connecting to download.java.net (download.java.net):209.95.138.153:80... connected.
HTTP request sent, awaiting response... 404 Not Found
2017-07-20 22:22:40 ERROR: 404: Not Found.
Download failed

```

5. पैकेज इंस्टॉल करें। `sudo apt-get install oracle-java9-installer -y` टाइप करें।

- आपको पासवर्ड के लिए संकेत दिया जा सकता है, इसे टाइप करें और `↵` एंटर दबाएं, कोई बिंदु या तारांकन नहीं दिखाई देगा, यह सामान्य है।

```

Terminal
update-alternatives: using /usr/lib/jvm/java-9-oracle/bin/unpack200 in auto mode
usr/bin/unpack200 (unpack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-9-oracle/bin/wsgen in auto mode
bin/wsgen (wsgen) in auto mode
update-alternatives: using /usr/lib/jvm/java-9-oracle/bin/wsimport in auto mode
sr/bin/wsimport (wsimport) in auto mode
update-alternatives: using /usr/lib/jvm/java-9-oracle/bin/xjc in auto mode
n/xjc (xjc) in auto mode
update-alternatives: using /usr/lib/jvm/java-9-oracle/lib/jexec in auto mode
bin/jexec (jexec) in auto mode
update-alternatives: using /usr/lib/jvm/java-9-oracle/bin/javaws in auto mode
/usr/bin/javaws.real (javaws.real) in auto mode
Oracle JDK 9 installed

#####Important#####
To set Oracle JDK9 as default, install the "oracle-java9-set-default" package.
E.g. - sudo apt install oracle-java9-set-default.
Setting up oracle-java9-set-default (9b152+9b152arn-1-webupd8-8) ...
Setting up oracle-java9-set-default (9b152+9b152arn-1-webupd8-8) ...
$ java -version
java version "9-ea"
Java(TM) SE Runtime Environment (build 9-ea+152)
Java HotSpot(TM) 64-Bit Server VM (build 9-ea+152, mixed mode)

```

6. ओरेकल के जावा को डिफॉल्ट बनाएं। कई उबंटू डेरिवेटिव्स में, ओपनजेडीके का उपयोग करने के लिए डिफॉल्ट जावा होने के लिए सेट किया गया है, यदि आप चाहते हैं कि ओरेकल के जावा को डिफॉल्ट रूप से उपयोग किया जाए तो आपको `sudo apt install oracle-java9-set-default..` टाइप करना होगा।

मैक पर जावा कैसे इंस्टॉल करें

1. जावा वेबसाइट पर जाएं और इंस्टॉलर डाउनलोड करें।

जावा को इंस्टॉल करने के लिए, आपको सबसे पहले ओरेकल से इंस्टॉलर प्रोग्राम डाउनलोड करना होगा। "फ्री जावा डाउनलोड" बटन पर क्लिक करें।



फिर आपको अंतिम उपयोगकर्ता लाइसेंस अनुबंध को पढ़ने और उससे सहमत होने के लिए कहा जाता है।



जावा को डाउनलोड करने के लिए आप किस वेब ब्राउज़र का उपयोग कर रहे हैं और इसके कॉन्फिगरेशन के आधार पर, आपको इंस्टॉलर फ़ाइल को डाउनलोड करने या सहेजने के लिए कहा जा सकता है।

यदि आप सफारी का उपयोग कर रहे हैं, तो जावा इंस्टॉलर स्वचालित रूप से आपके डाउनलोड फ़ोल्डर में डाउनलोड हो जाएगा। आप ऊपर दाईं ओर छोटे "डाउनलोड" आइकन में इसकी प्रगति देख सकते हैं।



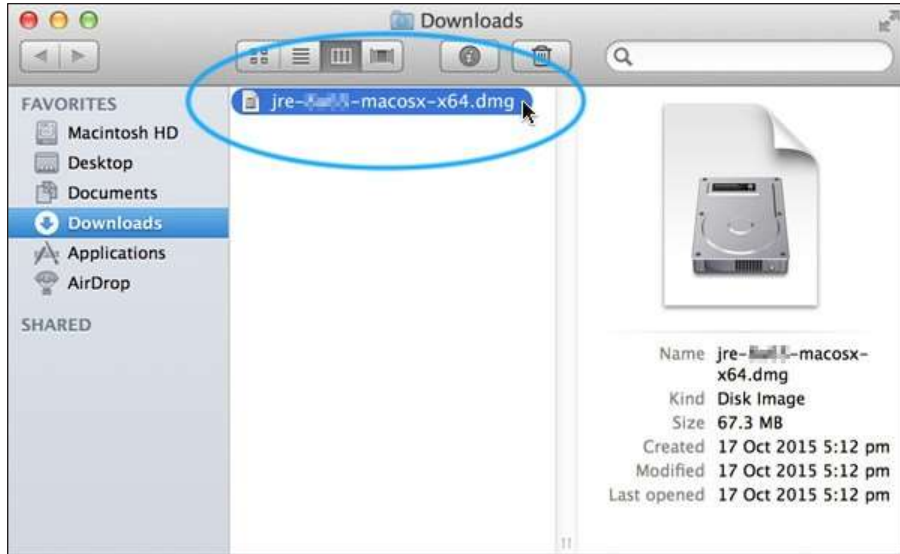
यदि आप सफारी का उपयोग कर रहे हैं, तो एक बार जब यह डाउनलोड हो जाता है, तो आप "शो इन फाइंडर" के लिए लिटिल मैग्निफाइंग ग्लास पर क्लिक कर सकते हैं, जो तब "डाउनलोड" फ़ोल्डर खोलेगा।



यदि आप किसी भिन्न ब्राउज़र का उपयोग कर रहे हैं, तो आपको फाइंडर का उपयोग करके अपने "डाउनलोड" फ़ोल्डर में जाना होगा।

2. इंस्टॉलर को अनपैक करें और रन करें।

आप जिस वेब ब्राउज़र का उपयोग कर रहे हैं और आपके ब्राउज़र के कॉन्फ़िगरेशन के आधार पर, आपका ब्राउज़र स्वचालित रूप से ".dmg" फ़ाइल खोल सकता है जिसे अभी डाउनलोड किया गया था।



यदि यह स्वचालित रूप से नहीं खुलता है, तो डाउनलोड पूरा होने के बाद, अपना डाउनलोड फ़ोल्डर खोलें और "dmg" इंस्टॉलर फ़ाइल ढूँढ़ें।

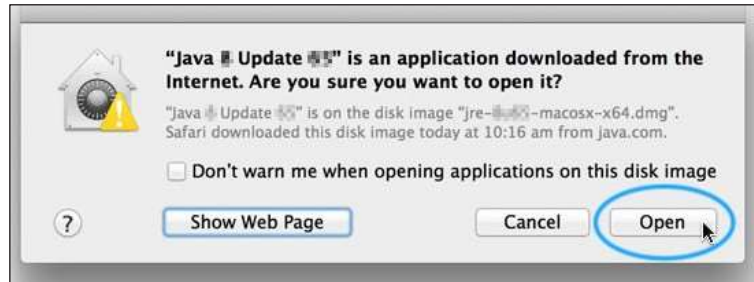
सुनिश्चित करें कि यह वास्तव में वही फ़ाइल है जिसे आपने अभी जावा वेबसाइट से डाउनलोड किया है। इंस्टॉलर कंटेनर को माउंट/खोलने के लिए इसे डबल क्लिक करें।



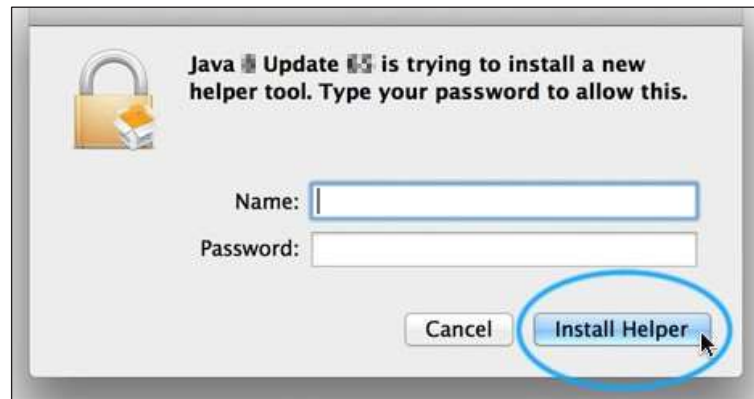
आपके द्वारा डाउनलोड की गई "dmg" कंटेनर फ़ाइल में इंस्टॉलर प्रोग्राम है। इंस्टॉलर को चलाने के लिए इसे डबल क्लिक करें।

यदि आप वास्तव में "जावा एक्स अपडेट एक्सएक्स" प्रोग्राम चलाना चाहते हैं, तो आपको संकेत दिया जाएगा, इसमें वास्तविक संस्करण संख्याएं होंगी।

यह सुनिश्चित करने के लिए संकेत को ध्यान से पढ़ें कि आप वास्तव में वह फ़ाइल चला रहे हैं जिसे आपने अभी डाउनलोड किया है। फ़ाइल का नाम और डाउनलोड करने की तारीख और समय की जाँच करें। जब आप तैयार हों, तो "ओपन" बटन पर क्लिक करें।



इंस्टॉलर प्रोग्राम को आपके कंप्यूटर पर सिस्टम फाइल डालने की आवश्यकता होगी और डिफ़ॉल्ट रूप से इसे ऐसा करने की अनुमति नहीं होगी। इंस्टॉलर प्रोग्राम को सिस्टम फाइलों को संशोधित करने और बनाने की अनुमति देने के लिए आपको अपना उपयोगकर्ता नाम और पासवर्ड दर्ज करना होगा और फिर "ओके" बटन पर क्लिक करना होगा।



3. जावा इंस्टॉल करें।



अब आप जावा इंस्टॉलर चला रहे हैं। आगे बढ़ने के लिए "अगला" बटन पर क्लिक करें।

इंस्टॉलर तब जावा के नवीनतम वर्ज़न को डाउनलोड और इंस्टॉल करेगा।



जब इंस्टॉलेशन समाप्त हो जाती है, तो आपको बताया जाता है कि "बंद करें" बटन पर क्लिक करने के बाद, आपका ब्राउज़र खुल जाएगा ताकि आप सत्यापित कर सकें कि जावा काम कर रहा है।



4. जावा को सत्यापित करें।

जावा इंस्टॉलर आपका डिफ़ॉल्ट वेब ब्राउज़र खोलेगा और आपको अपने जावा इंस्टॉलेशन को सत्यापित करने के लिए संकेत देगा।

आपको एक अलग ब्राउज़र (जैसे सफारी या फ़ायरफ़ॉक्स) का उपयोग करने की आवश्यकता है जो अभी भी जावा का समर्थन करता है।

आपको "जावा वरज़न सत्यापित करें" बटन दिखाई देगा। जावा की अपनी इंस्टॉलेशन को सत्यापित करने के लिए इसे क्लिक करें।



आमतौर पर, आपको जावा प्लगइन को रन करने की अनुमति देने के लिए कहा जाएगा। निम्नलिखित स्क्रीनशॉट सफारी से लिया गया है।



यदि आप इस एप्लिकेशन को चलाना चाहते हैं तो आपको जावा द्वारा एक बार फिर से संकेत दिया जाता है। आपके कंप्यूटर को अप्रतिबंधित पहुंच प्रदान करने के बारे में दी गई चेतावनी पर ध्यान दें। जावा एप्लिकेशन चलाते समय आपको हमेशा बहुत सावधान रहना चाहिए; सुनिश्चित करें कि आप केवल उन्हीं कंपनियों के एप्लिकेशन चलाते हैं जिन पर आप भरोसा करते हैं। विशेष रूप से, हमेशा प्रकाशक का नाम देखें।

इस उदाहरण में, हम जावा बनाने वाली कंपनी द्वारा बनाए गए जावा एप्लिकेशन के बारे में बात कर रहे हैं ("ओरेकल अमेरिका, इंक."); तो शायद इसे रन करना काफी सुरक्षित है। जब आप तैयार हों, तो चेतावनी पर "रन" बटन पर क्लिक करें।



आपके द्वारा इस जावा ऐप को चलाने की अनुमति देने के बाद, यह सत्यापित करेगा कि आपने जावा इंस्टॉल किया है और (बशर्ते कि इंस्टॉलेशन के दौरान सब कुछ काम कर गया हो) आपको इस आशय का एक संदेश दिखाएगा।



जावा के लिए डिफ़ॉल्ट कॉन्फ़िगरेशन स्वचालित रूप से अपडेट की जांच करना और उपलब्ध होने पर उन्हें इंस्टॉल करना है। यह जावा पैच और आपके कंप्यूटर को सुरक्षित रखने में मदद करता है।

जेडीके को कैसे इंस्टॉल करें

विंडोज़ पर

यह जावा डेवलपमेंट किट (जेडीके) आपको जावा प्रोग्राम को कोड करने और चलाने की अनुमति देता है। यह संभव है कि आप एक ही पीसी पर कई जेडीके वर्ज़न इंस्टॉल करें। लेकिन इसकी अनुशंसा की जाती है कि आप केवल नवीनतम वर्ज़न इंस्टॉल करें।

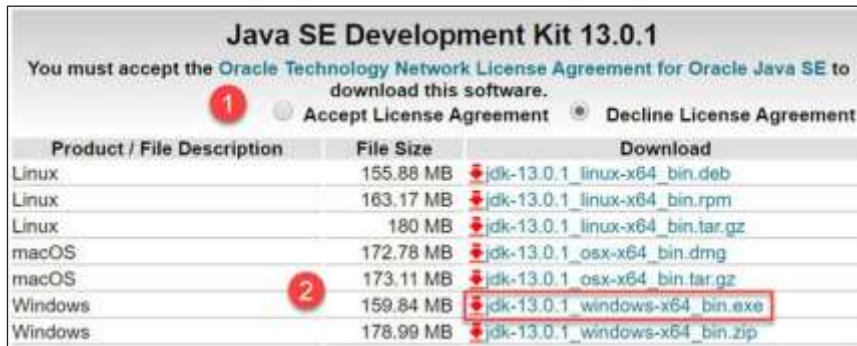
विंडोज में जावा इंस्टॉल करने के चरण निम्नलिखित हैं:

चरण 1. लिंक पर जाएं। जावा नवीनतम संस्करण के लिए डाउनलोड जेडीके पर क्लिक करें।



चरण 2. अगला,

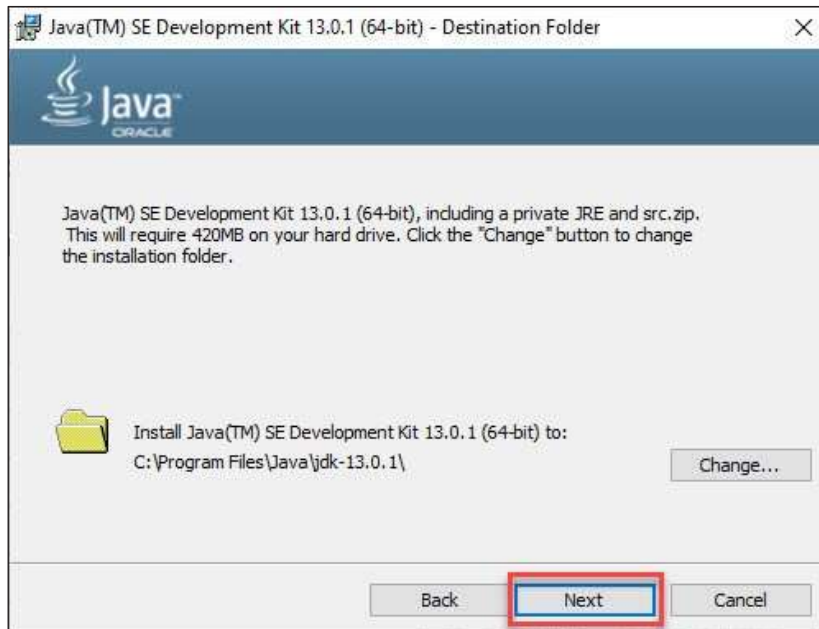
- लाइसेंस एग्रीमेंट को स्वीकार करें।
- विंडोज के लिए जावा के अपने वर्ज़न (32 या 64 बिट) के लिए नवीनतम जावा जेडीके डाउनलोड करें।



चरण 3. डाउनलोड पूरा होने के बाद, जेडीके इंस्टॉल करने के लिए exe (एक्ज़) रन करें। अगला पर क्लिक करें।



चरण 4. जावा इंस्टॉल के लिए पथ का चयन करें और अगला क्लिक करें।



चरण 5. इंस्टॉलेशन पूर्ण होने के बाद क्लोज़ पर क्लिक करें।



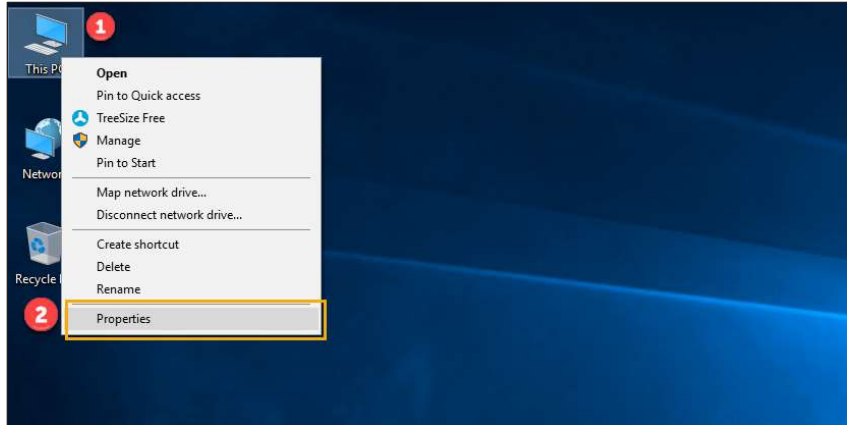
जावा में पर्यावरण चर कैसे सेट करें: पथ और क्लासपाथ

पथ वेरिएबल जावैक, जावा आदि जैसे एक्जिक्यूटिव का स्थान देता है। पथ को निर्दिष्ट किए बिना प्रोग्राम चलाना संभव है, लेकिन आपको निष्पादन योग्य का पूरा पथ देना होगा जैसे C:\Program Files\Java\jdk-13.0.1\ साधारण जावैक ए.जावा के बजाय बिन \ जावैक ए.जावा।

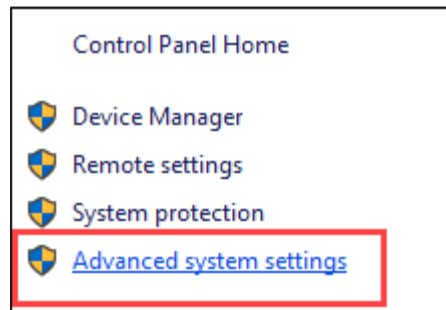
क्लासपाथ वेरिएबल लाइब्रेरी फाइलों का स्थान देता है।

आइए पथ और क्लासपाथ को सेट करने के चरणों को देखें।

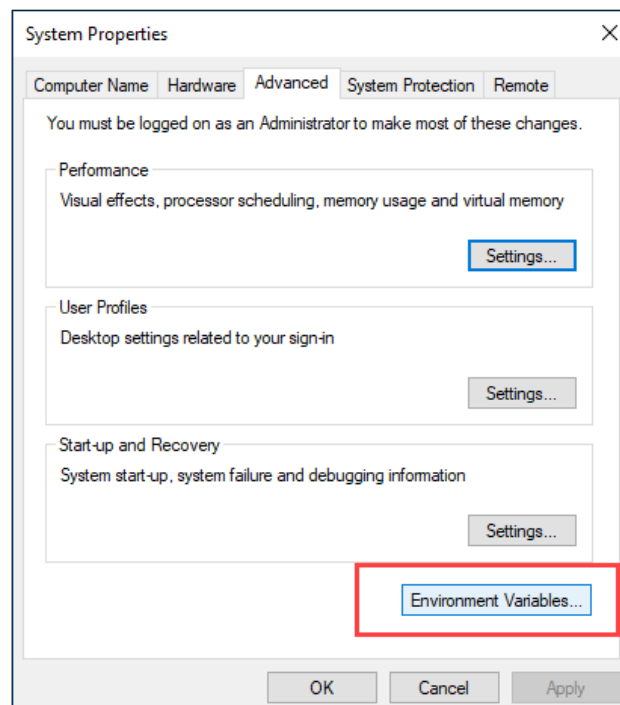
चरण 1. माई कंप्यूटर पर राइट क्लिक करें और गुणों का चयन करें।



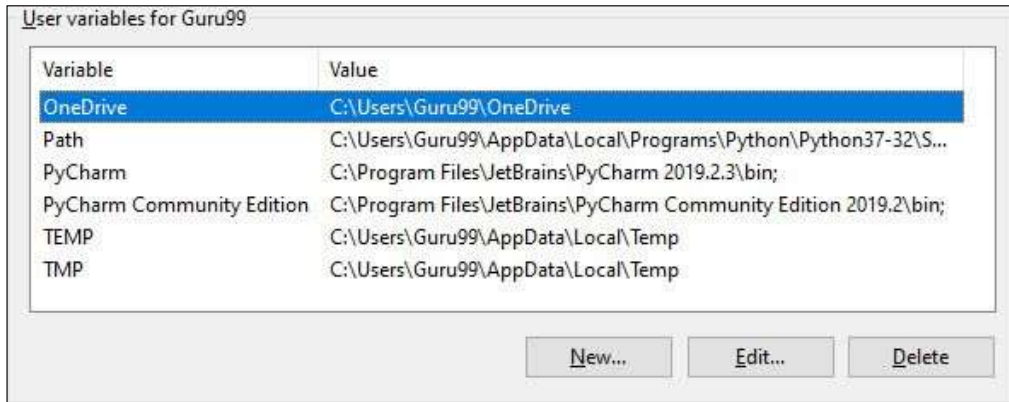
चरण 2. उन्नत सिस्टम सेटिंग्स पर क्लिक करें।



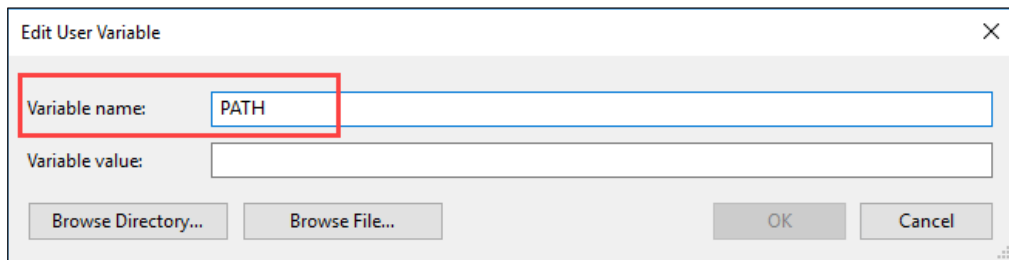
चरण 3. एनवायरनमेंट वेरिएबल पर क्लिक करें।



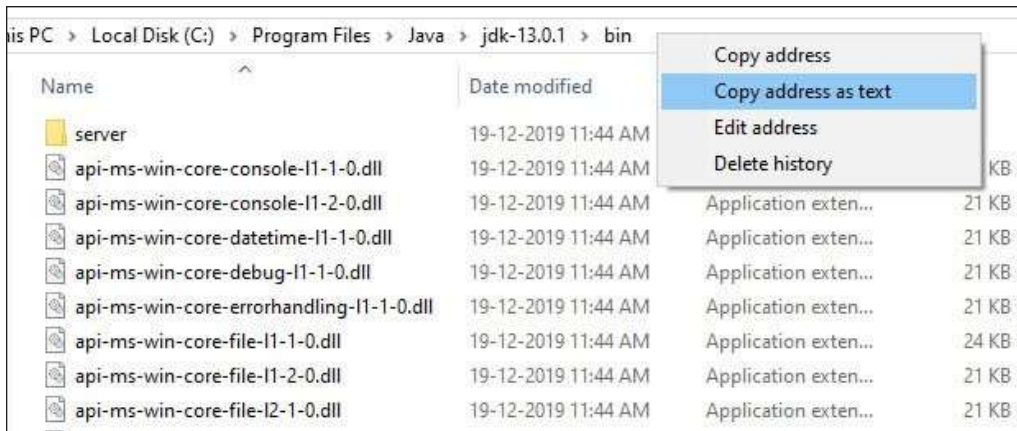
चरण 4. उपयोगकर्ता वेरिएबल के नए बटन पर क्लिक करें।



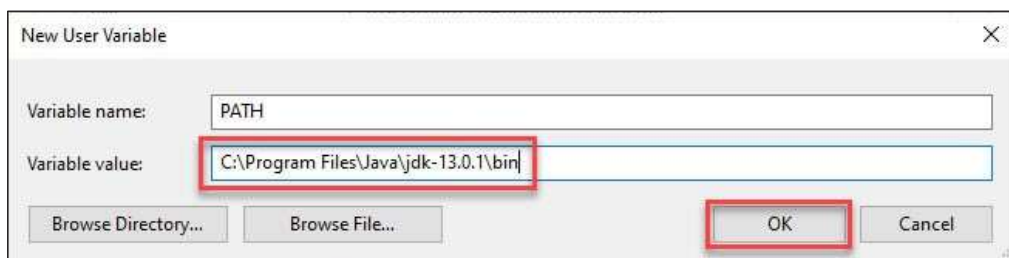
चरण 5. वेरिएबल नाम में पीएटीएच टाइप करें।



चरण 6. बिन फ़ोल्डर के पथ की प्रतिलिपि बनाएँ जो जेडीके फ़ोल्डर में इंस्टॉल है।



स्टेप 7. बिन फ़ोल्डर के पथ को वेरिएबल वैल्यू में पेस्ट करें और ओके बटन पर क्लिक करें।



यदि आपके पास पहले से ही आपके पीसी में एक पाथ वेरिएबल बनाया गया है, तो पाथ वेरिएबल को इसमें संपादित करें:

```
PATH = <JDK installation directory>\bin;%PATH%;
```

यहां, %PATH% मौजूदा पाथ वेरिएबल को हमारे नए मान में जोड़ता है।

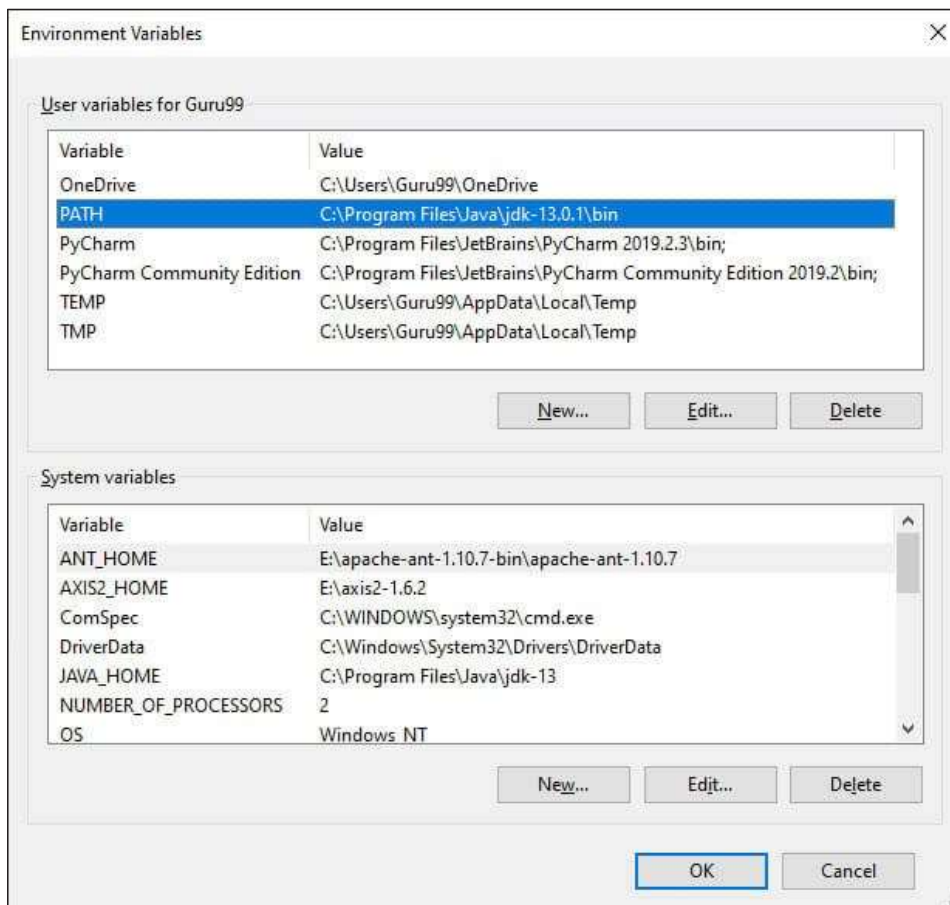
चरण 8. क्लासपाथ सेट करने के लिए आप इसी तरह की प्रक्रिया का पालन कर सकते हैं।



यदि इंस्टॉलेशन के बाद जावा इंस्टॉलेशन काम नहीं करता है, तो आप क्लासपाथ को इसमें बदलें:

```
CLASSPATH = < JDK installation directory>\lib\tools.jar;
```

चरण 9. ओके बटन पर क्लिक करें।



चरण 10. कमांड प्रॉम्प्ट पर जाएं और जावैक कमांड टाइप करें।

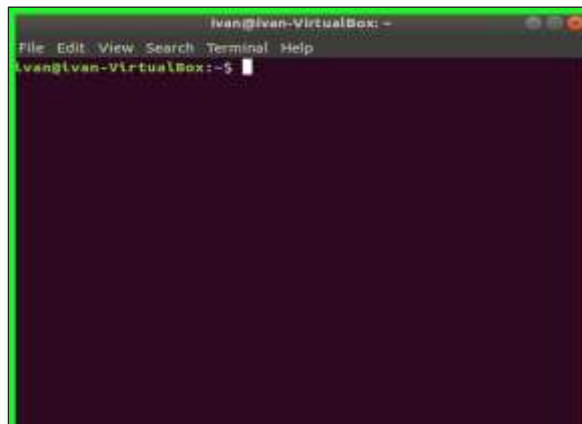
यदि आप नीचे की तरह एक स्क्रीन देखते हैं, तो जावा इंस्टॉल है।

```

C:\Users\Ivan>java
Usage: java [-options]  []
where possible options include:
  -D<propertyname>=<propertyvalue>  Read options and filenames from file
  -D<key=value>                      Options to pass to annotation processors
  --add-modules <module(s)>[:<module(s)>]*
    Root modules to resolve in addition to the initial modules, or all modules
    on the module path if <module> is All-MODULE-PATH.
  --boot-class-path <paths>, -bootclasspath <paths>
    Override location of bootstrap class files
  --class-path <paths>, -classpath <paths>, -cp <paths>
    Specify where to find user class files and annotation processors
  -d <directory>
    Specify where to place generated class files
  -deprecation
    Output source locations where deprecated APIs are used
  --enable-preview
    Enable preview language features. To be used in cooperation with either --source or --release.
  --encoding <encoding>
    Specify character encoding used by source files
  --make-libs-dir <dir>
    Override location of make-libs standard path
  --outdir <dir>
    Override location of installed extensions
  
```

लिनक्स पर

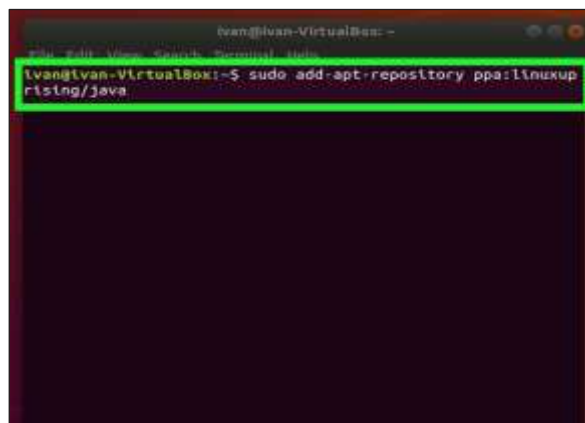
विधि 1. उबंटू या लिनक्स मिंट का उपयोग करना



```

Ivan@Ivan-VirtualBox: ~$
  
```

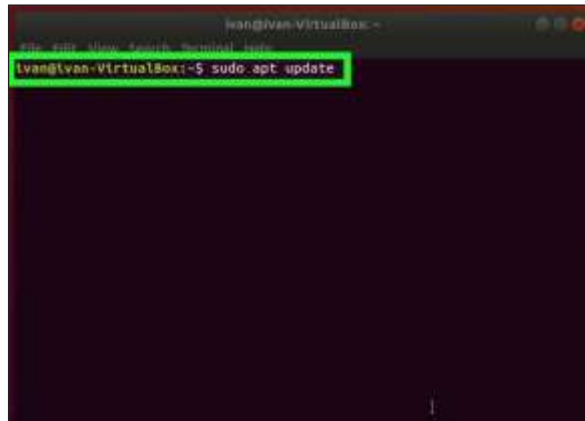
1. नई टर्मिनल विंडो खोलने के लिए Ctrl+Alt+T दबाएं। यदि आप पहले से ही कमांड प्रॉम्प्ट पर हैं, तो बस अगले चरण पर जाएं।



```

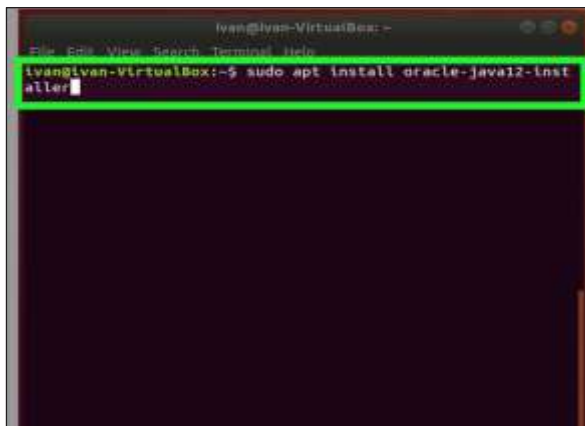
Ivan@Ivan-VirtualBox: ~$ sudo add-apt-repository ppa:linuxuprising/java
  
```

2. sudo add-apt-repository ppa:linuxuprising/java टाइप करें और एंटर दबाएं। यह लिनक्स अपराइजिंग रिपॉजिटरी जोड़ता है।



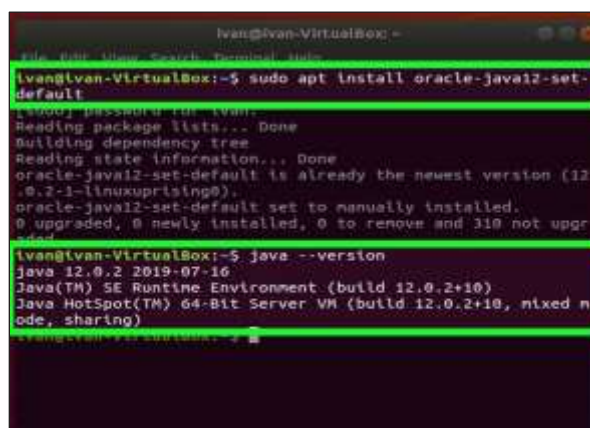
```
ivan@ivan-VirtualBox:~$ sudo apt update
```

3. `sudo apt update` टाइप करें और एंटर दबाएँ। अब जब रिपॉजिटरी तैयार हो गई है, तो आप इसका उपयोग जेडीके को इंस्टॉल करने के लिए कर सकते हैं।



```
ivan@ivan-VirtualBox:~$ sudo apt install oracle-java12-installer
```

4. `sudo apt install oracle-java12-installer` टाइप करें और एंटर दबाएं। जेडीके अब इंस्टॉल है।



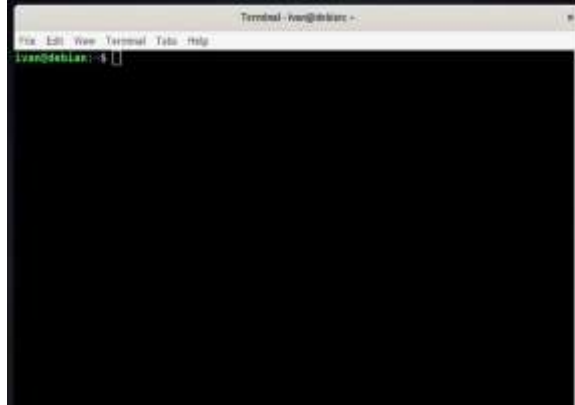
```
ivan@ivan-VirtualBox:~$ sudo apt install oracle-java12-set-default
[sudo] password for ivan:
Reading package lists... Done
Building dependency tree
Reading state information... Done
oracle-java12-set-default is already the newest version (12.0.2-1-linuxuprising0).
oracle-java12-set-default set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 110 not upgraded.

ivan@ivan-VirtualBox:~$ java --version
java 12.0.2 2019-07-16
Java(TM) SE Runtime Environment (build 12.0.2+10)
Java HotSpot(TM) 64-Bit Server VM (build 12.0.2+10, mixed mode, sharing)
```

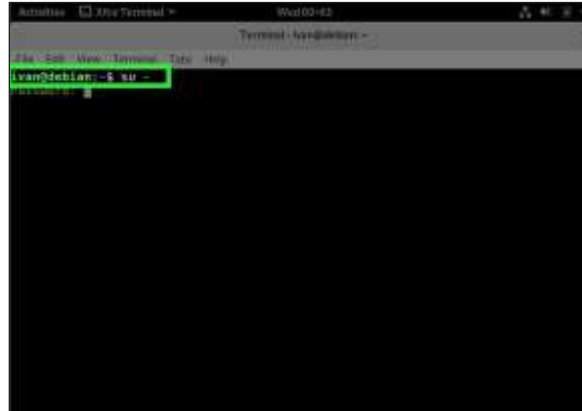
5. `sudo apt install oracle-java12-set-default` टाइप करें और एंटर दबाएं। यह जावा 12 को डिफॉल्ट जावा वरज़न के रूप में सेट करता है।

- डिफॉल्ट वरज़न को दोबारा जान्ने के लिए, कमांड प्रॉम्प्ट पर `java --version` टाइप करें और एंटर दबाएं।

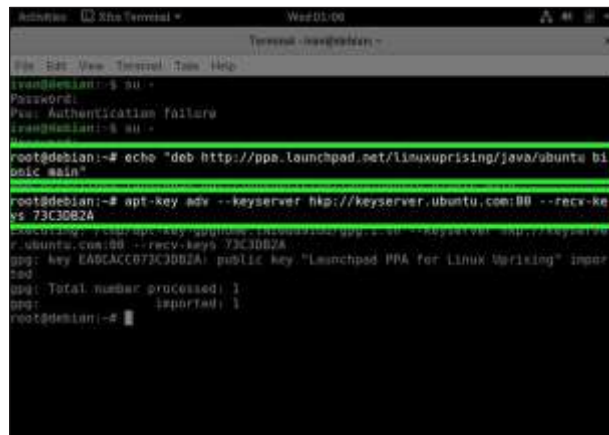
विधि 2. डेबियन उपयोग करना



1. नई टर्मिनल विंडो खोलने के लिए Ctrl+Alt+T दबाएं। यदि आप पहले से ही कमांड प्रॉम्प्ट पर हैं, तो बस अगले चरण पर जाएं।



2. su टाइप करें और ↵ एंटर दबाएं।



3. लिनक्स अपराइजिंग रिपॉजिटरी जोड़ें। ऐसे:

- echo "deb http://ppa.launchpad.net/linuxuprising/java/ubuntu bionic main" टाइप करें और ↵ एंटर दबाएं।

- apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 73C3DB2A टाइप करें और **↵** एंटर दबाएं।

```

root@debian:~# apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 73C3DB2A
Executing: /tmp/apt-key-gpghome.fH2ua91Gc/gpg.1.sh --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 73C3DB2A
gpg: key E48C4CC873C3DB2A: public key "Launchpad PPA for Linux Upriting" imported
gpg: Total number processed: 1
gpg:      imported: 1

root@debian:~# apt-get update
Ign:1 cdrom://buster_ubuntu_18.04_ubuntu_18.04.0_20190706-10.24_buster InRelease
Err:2 cdrom://buster_ubuntu_18.04_ubuntu_18.04.0_20190706-10.24_buster InRelease
   Please use apt-cdrom to make this CD-ROM recognized by APT. apt-get update cannot be used to add new CD-ROMs
Hit:3 http://security.debian.org/debian-security buster/updates InRelease
Reading package lists... Done
Err:4 The repository 'cdrom://buster_ubuntu_18.04_ubuntu_18.04.0_20190706-10.24_buster InRelease' does not have a Release file.
W: Updating from such a repository can't be done securely, and is therefore disabled by default.
W: See apt-secure(8) manpage for repository creation and user configuration details

```

4. apt-get update टाइप करें और **6ca**, एंटर दबाएं।

```

root@debian:~# apt-get install oracle-java12-installer

```

5. apt-get install oracle-java12-installer टाइप करें और एंटर दबाएं।

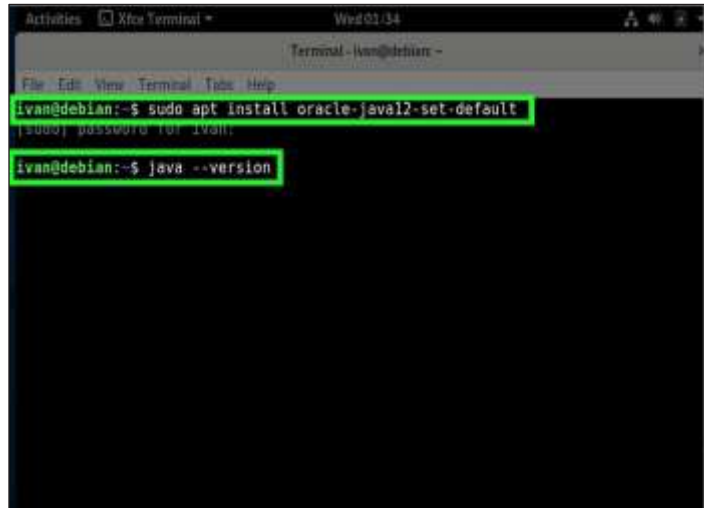
```

root@debian:~# sudo apt install oracle-java12-set-default
Reading package lists... Done
Building dependency tree
Reading state information... Done

root@debian:~# exit

```

6. exit (एग्जिट) टाइप करें और एंटर दबाएं। जेडीके अब इंस्टॉल है।



```

Terminal - Ivan@debian: ~
File Edit View Terminal Tabs Help
ivan@debian:~$ sudo apt install oracle-java12-set-default
[sudo] password for ivan:
ivan@debian:~$ java --version

```

7. `sudo apt install oracle-java12-set-default` टाइप करें और एंटर दबाएं। यह जावा 12 को डिफॉल्ट जावा वर्ज़न के रूप में सेट करता है।

- डिफॉल्ट वर्ज़न को दोबारा जांचने के लिए, `java --version` कमांड प्रॉम्प्ट पर टाइप करें और एंटर दबाएं।

मैक पर

अपने मैक पर जावा डेवलपमेंट किट (जेडीके) को इंस्टॉल करने से आप जावा अनुप्रयोगों को लिखने और संकलित करने की अनुमति देंगे। जेडीके की स्थापना बहुत सीधी है, और इसमें नेटवीन नामक एक डेवलपमेंट एनवायरनमेंट शामिल हैं। आप अपना कोड लिखने और परीक्षण के लिए इसे संकलित करने के लिए नेटवीन का उपयोग करेंगे।

भाग 1. जेडीके इंस्टॉल करना



1. जेडीके डाउनलोड पेज पर जाएं। अपना वेब ब्राउज़र खोलें और oracle.com/downloads/index.html पर जाएं।



2. जेडीके इंस्टॉलर डाउनलोड करें। एक बार जब आप डाउनलोड पृष्ठ पर हों, तो आपको इंस्टॉलर फ़ाइलों पर नेविगेट करना होगा:

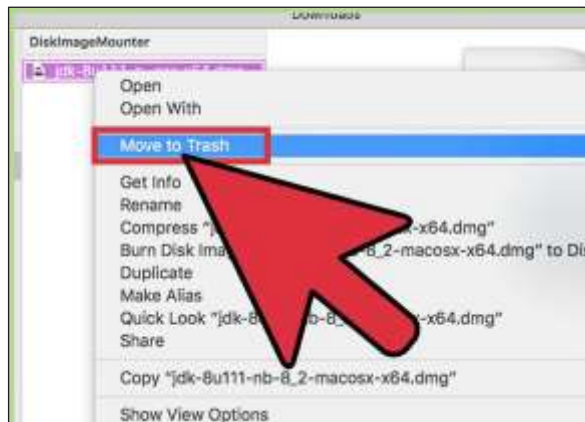
- "जावा" विकल्प पर क्लिक करें।
- "जावा एसई" पर क्लिक करें।
- नेटबीन के साथ "जेडीके 8" के बगल में "डाउनलोड" बटन पर क्लिक करें।
- "एक्सेप्ट लाइसेंस एग्रीमेंट" सलेक्ट करें और फिर शीर्ष अनुभाग में "मैक ओएस एक्स" के लिए डाउनलोड लिंक पर क्लिक करें। यह नेटबीन्स देवलपमेंट एनवायरनमेंट के साथ जावा एसडीके का नवीनतम रिलीज़ है।



3. डाउनलोड किए गए इंस्टॉलर पर डबल-क्लिक करें। इंस्टॉलर .dmg प्रारूप में है। इसे डबल-क्लिक करने से इंस्टॉलेशन इंटरफ़ेस खुल जायेगा ।



4. जेडीके को इंस्टॉल करने के लिए संकेतों का पालन करें। इंस्टॉलेशन आगे बढ़ने से पहले आपसे एडमिनिस्ट्रेटर पासवर्ड दर्ज करने के लिए कहा जाएगा ।

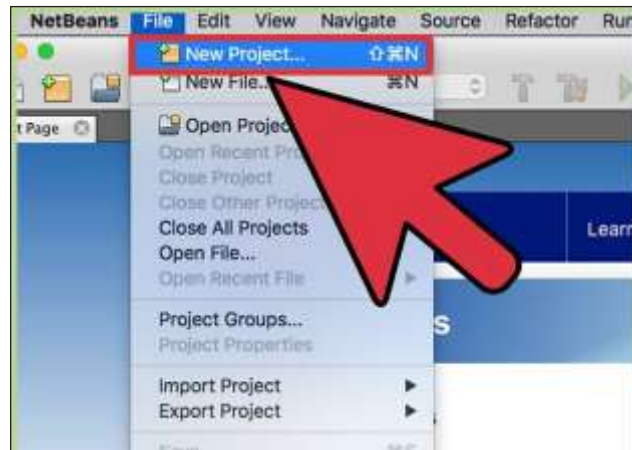


5. इंस्टॉलेशन के बाद डीएमजी फ़ाइल हटाएं (वैकल्पिक)। यह आपको डिस्क स्थान बचाने में मदद करेगा, क्योंकि जेडीके इंस्टॉल होने के बाद आपको इसकी आवश्यकता नहीं है।

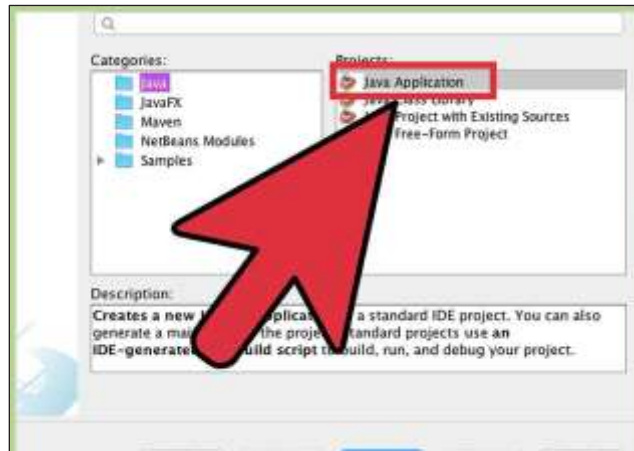
भाग 2. अपना पहला प्रोग्राम बनाना



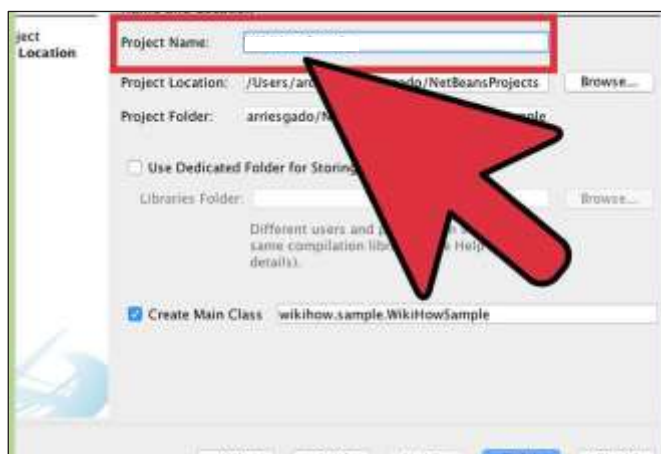
1. एप्लीकेशन फ़ोल्डर से नेटबीन्स खोलें। यह जावा के लिए डेवलपमेंट एनवायरनमेंट है, और आपको आसानी से कोड लिखने और संकलित करने की अनुमति देगा।



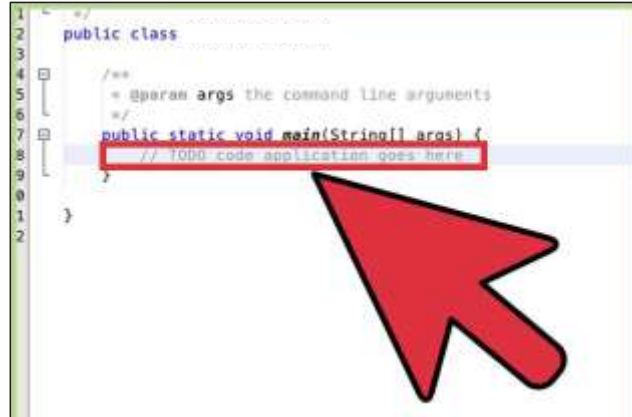
2. फ़ाइल पर क्लिक करें और "न्यू प्रोजेक्ट" सिलेक्ट करें। यह नेटबीन्स में एक "न्यू प्रोजेक्ट शुरू करेगा।



3. "जावा" श्रेणी और "जावा एप्लिकेशन" प्रोजेक्ट का चयन करें। यह नेटबीन्स को इस प्रोजेक्ट के लिए जावा फ़ाइलें बनाने के लिए सेट करेगा।



4. प्रोजेक्ट को एक नाम दें और "समाप्त करें" पर क्लिक करें। इस उदाहरण के लिए, इसे "हैलोवर्ल्ड" कहें। यह प्रोजेक्ट बनने के बाद कोड एडिटर खोलेगा।

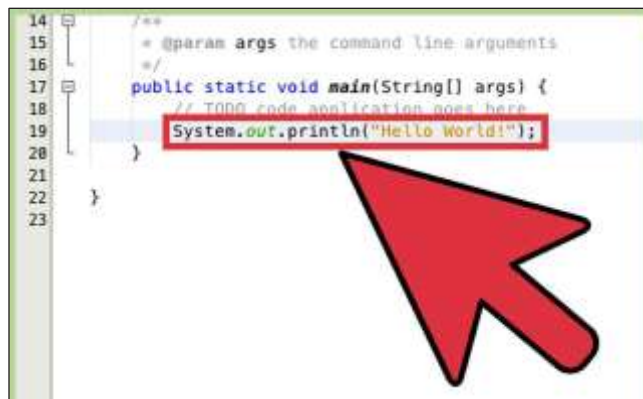


```

1  public class
2
3
4  /**
5   * @param args the command line arguments
6   */
7  public static void main(String[] args) {
8      // TODO code application goes here
9  }
10
11
12

```

5. "// TODO code application goes here" लाइन खोजें। आपका प्रोग्राम कोड इस लाइन के नीचे जाएगा।



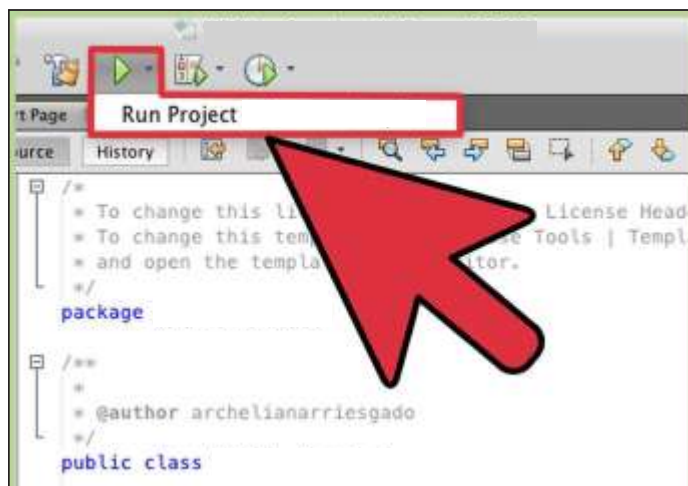
```

14  /**
15   * @param args the command line arguments
16   */
17  public static void main(String[] args) {
18      // TODO code application goes here
19      System.out.println("Hello World!");
20  }
21
22
23

```

6. एक नई लाइन पर अपना प्रोग्राम कोड दर्ज करें। उसी इंडेंटेशन के साथ एक नई लाइन बनाने के लिए "// TODO code application goes here" लाइन के बाद ↵ रिटर्न दबाएं। निम्नलिखित कोड टाइप करें:

```
System.out.println("Hello World!");
```



7. "रन प्रोजेक्ट" बटन पर क्लिक करें। यह हरे रंग के प्ले बटन की तरह दिखता है, और टूलबार में पाया जा सकता है।

```

13
14
15 // > @param args the command line arguments
16 //
17 public static void main(String[] args) {
18     // TODO code application goes here
19     System.out.println("Hello World!");
20 }
21
22
23

```

Output -

Hello World!
BUILD SUCCESSFUL (total time: ...)

8. अपने प्रोजेक्ट को एक्शन (क्रिया) में देखने के लिए आउटपुट टैब देखें। प्रोजेक्ट चलाने के बाद यह फ्रेम आपकी स्क्रीन के नीचे दिखाई देगा।

```

13
14
15 // > @param args the command line arguments
16 //
17 public static void main(String[] args) {
18     // TODO code application goes here
19     System.out.println("Hello World!");
20 }
21
22
23

```

Output -

Hello World!
BUILD SUCCESSFUL (total time: 12 seconds)

9. किसी भी त्रुटि को ठीक करें। यदि प्रोजेक्ट में कोई त्रुटि नहीं है, तो आपको "Hello World!" (हेल्लो वर्ल्ड) और आउटपुट टैब में "BUILD SUCCESSFUL" (बिल्ड सक्सेसफुल) दिखाई देगा। यदि त्रुटियां हैं, तो आप देखेंगे कि वे किन पंक्तियों में होती हैं ताकि आप वापस जा सकें और उन्हें ठीक कर सकें।



10. जावा सीखना जारी रखें। अब जब आपके पास जेडीके इंस्टॉल हो गया है और काम कर रहा है, तो आप जावा में प्रोग्राम करना सीखना जारी रख सकते हैं।

जावा में कंडीशनल स्टेटमेंट, लूप्स, मेथड्स और एरेज़

किसी निष्कर्ष पर पहुंचने के लिए किसी भी आदेश को निष्पादित और समाप्त करने के लिए सशर्त बयान और लूप का उपयोग किया जाता है। कोड का एक ब्लॉक जिसे केवल तभी निष्पादित किया जाता है जब उसे कॉल किया जाता है उसे एक विधि कहा जाता है। एक एरे का उपयोग कई मानों को संग्रहित करने के लिए किया जाता है। जावा में सशर्त बयानों, लूपों, विधियों और एरेज़के विविध अनुप्रयोगों पर इस अध्याय में गहन चर्चा की गई है।

सशर्त विवरण का उपयोग कैसे करें

कंडीशनल स्टेटमेंट एक ऐसा स्टेटमेंट है जो कंप्यूटर प्रोग्रामिंग लैंग्वेज का इस्तेमाल यह तय करने के लिए करता है कि सही कंडीशन पूरी होने पर कौन सा कोड चलाना है या सही कंडीशन पूरी नहीं होने पर कौन सा कोड नहीं चलाना है।

इफ़ स्टेटमेंट्स (if statements) का उपयोग कैसे करें

जावा इफ़ (If - then) स्टेटमेंट

जावा में इफ़-देन स्टेटमेंट का सिंटैक्स है:

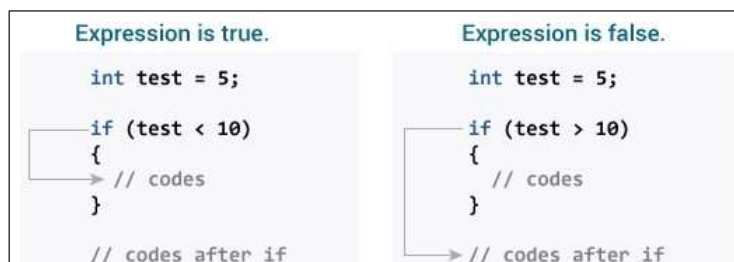
```
If (expression) {
    // statements
}
```

यहां expression एक बूलियन expression है (या तो true या false है) ।

यदि अभिव्यक्ति का मूल्यांकन सत्य पर किया जाता है, तो इफ़ (कोष्ठक के अंदर कथन) के शरीर के अंदर कथन निष्पादित किए जाते हैं।

यदि व्यंजक का मूल्यांकन असत्य पर किया जाता है, तो इफ़ के शरीर के अंदर कथनों को निष्पादन से छोड़ दिया जाता है।

इफ़ स्टेटमेंट कैसे काम करता है?



उदाहरण: Java if Statement

```
class If Statement {
    public static void main(String[] args) {

        int number = 10;

        if (number > 0) {
            System.out.println("Number is positive.");
        }

        System.out.println("This statement is always executed.");
    }
}
```

जब आप प्रोग्राम चलाते हैं, तो आउटपुट होगा:

```
Number is positive.
This statement is always executed.
```

जब संख्या 10 होती है, तो परीक्षण व्यंजक संख्या >0 का मूल्यांकन सत्य पर किया जाता है। इसलिए, इफ़ स्टेटमेंट के शरीर के अंदर कोड निष्पादित किए जाते हैं।

अब, संख्या के मान को ऋणात्मक पूर्णांक में बदलें। बता दें -5. इस मामले में आउटपुट होगा: यह कथन हमेशा निष्पादित होता है।

जब संख्या -5 होती है, तो परीक्षण व्यंजक संख्या > 0 का मूल्यांकन असत्य होता है। इसलिए, जावा कंपाइलर इफ़ स्टेटमेंट के बाँडी के निष्पादन को छोड़ देता है।

if ..else statements (इफ़..एल्स स्टेटमेंट) का कैसे उपयोग करें

एक इफ़ स्टेटमेंट के बाद एक वैकल्पिक अन्य स्टेटमेंट हो सकता है, जो बूलियन एक्सप्रेसन के गलत होने पर निष्पादित होता है।

सिंटैक्स

इफ़...एल्स स्टेटमेंट के निम्नलिखित सिंटैक्स है -

```
if(Boolean_expression) {
    // Executes when the Boolean expression is true
}else {
    // Executes when the Boolean expression is false
```

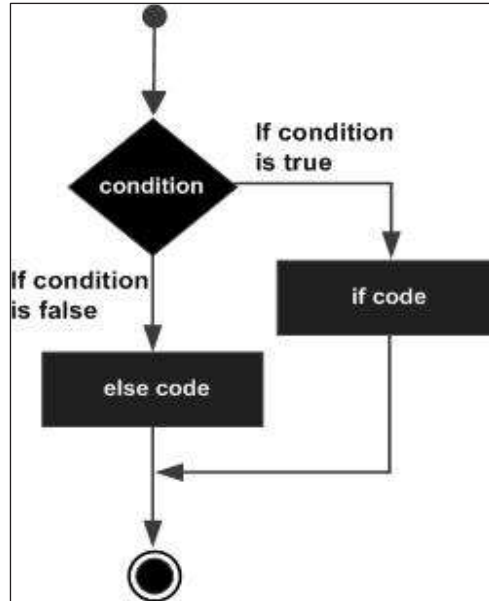


```
}

```

यदि बूलियन अभिव्यक्ति सत्य का मूल्यांकन करती है, तो कोड के इफ ब्लॉक को निष्पादित किया जाएगा, अन्यथा कोड के ब्लॉक को निष्पादित किया जाएगा।

प्रवाह आरेख



उदाहरण:

```

public class Test {
    public static void main(String args[]) {
        int x = 30;

        if( x < 20 ) {
            System.out.print("This is if statement");
        }else {
            System.out.print("This is else statement");
        }
    }
}

```

यह निम्नलिखित परिणाम देगा -

आउटपुट:

This is else statement

if..else..if statement (इफ़.. एल्स.. इफ़ स्टेटमेंट) का उपयोग कैसे करें

इफ-एल्स-इफ स्टेटमेंट का उपयोग तब किया जाता है जब हमें कई स्थितियों की जाँच करने की आवश्यकता होती है। इस कथन में हमारे पास केवल एक "if" (इफ) और एक "एल्स" है, हालांकि हमारे पास कई "else if" (एल्स इफ) हो सकते हैं। इसे सीढ़ी के रूप में भी जाना जाता है। यह इस तरह दिखता है:

```
if(condition_1) {
    /*if condition_1 is true execute this*/
    statement(s);
}
else if(condition_2) {
    /* execute this if condition_1 is not met and
    * condition_2 is met
    */
    statement(s);
}
else if(condition_3) {
    /* execute this if condition_1 & condition_2 are
    * not met and condition_3 is met
    */
    statement(s);
}
.
.
.
else {
    /* if none of the condition is true
    * then these statements gets executed
    */
    statement(s);
}
```

नोट: यहां ध्यान देने वाली सबसे महत्वपूर्ण बात यह है कि इफ-एल्स-इफ स्टेटमेंट में, जैसे ही शर्त पूरी होती है, स्टेटमेंट के संबंधित सेट को निष्पादित किया जाता है, बाकी को अनदेखा कर दिया जाता है। यदि कोई भी शर्त पूरी नहीं होती है तो "एल्स" के अंदर के बयान निष्पादित हो जाते हैं।

if-else-if (इफ-एल्स-इफ) का उदाहरण

```
public class IfElseIfExample {

    public static void main(String args[]){

        int num=1234;

        if(num <100 && num>=1) {

            System.out.println("Its a two digit number");

        }

        else if(num <1000 && num>=100) {

            System.out.println("Its a three digit number");

        }

        else if(num <10000 && num>=1000) {

            System.out.println("Its a four digit number");

        }

        else if(num <100000 && num>=10000) {

            System.out.println("Its a five digit number");

        }

        else {

            System.out.println("number is not between 1 & 99999");

        }

    }

}
```

आउटपुट:

यह चार अंकों की संख्या है

नेस्टेड इफ स्टेटमेंट का उपयोग कैसे करें

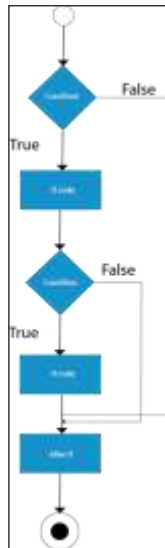
नेस्टेड इफ स्टेटमेंट दूसरे ब्लॉक के भीतर इफ ब्लॉक का प्रतिनिधित्व करता है। यहां, आंतरिक इफ ब्लॉक की स्थिति केवल तभी निष्पादित होती है जब बाहरी इफ ब्लॉक की स्थिति सही होती है।

सिंटैक्स:

```

if(condition){
    //code to be executed
    if(condition){
        //code to be executed
    }
}

```



उदाहरण:

```

//Java Program to demonstrate the use of Nested If Statement.
public class JavaNestedIfExample {
public static void main(String[] args) {
    //Creating two variables for age and weight
    int age=20;
    int weight=80;
    //applying condition on age and weight
    if(age>=18){
        if(weight>50){
            System.out.println("You are eligible to donate blood");
        }
    }
}

```

```
    }
  }}

```

आउटपुट:

You are eligible to donate blood

उदाहरण:

```
//Java Program to demonstrate the use of Nested If Statement.
public class JavaNestedIfExample2 {
    public static void main(String[] args) {
        //Creating two variables for age and weight
        int age=25;
        int weight=48;
        //applying condition on age and weight
        if(age>=18){
            if(weight>50){
                System.out.println("You are eligible to donate blood");
            } else{
                System.out.println("You are not eligible to donate blood");
            }
        } else{
            System.out.println("Age must be greater than 18");
        }
    }
}
```

आउटपुट:

You are not eligible to donate blood

स्विच स्टेटमेंट का उपयोग कैसे करें

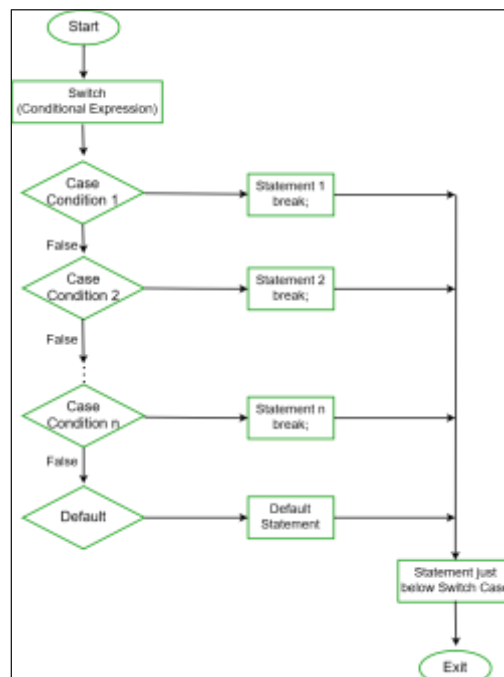
स्विच स्टेटमेंट एक मल्टी-वे ब्रांच स्टेटमेंट है। यह अभिव्यक्ति के मूल्य के आधार पर कोड के विभिन्न भागों में निष्पादन को भेजने का एक आसान तरीका प्रदान करता है। मूल रूप से, अभिव्यक्ति बाइट, शॉर्ट, चार और इंटर आदिम डेटा प्रकार हो सकती है। जेडीके 7 से शुरू होकर, यह प्रगणित प्रकारों (जावा में एनम्स), स्ट्रिंग क्लास और रैपर क्लासेस के साथ भी काम करता है।

स्विच-केस का सिंटैक्स:

```
// switch statement
switch(expression)
```

```
{  
    // case statements  
    // values must be of same type of expression  
    case value1 :  
        // Statements  
        break; // break is optional  
  
    case value2 :  
        // Statements  
        break; // break is optional  
  
    // We can have any number of case statements  
    // below is default statement, used when none of the cases is true.  
    // No break is needed in the default case.  
    default :  
        // Statements  
}
```

स्विच-केस का प्रवाह आरेख:



स्विच स्टेटमेंट के लिए कुछ महत्वपूर्ण नियम:

- डुप्लिकेट केस वेल्ज (मानों) की अनुमति नहीं है।
- किसी मामले के लिए मान उसी डेटा प्रकार का होना चाहिए जो स्विच में चर के रूप में है।
- किसी मामले का मान स्थिर या शाब्दिक होना चाहिए। चर की अनुमति नहीं है।
- स्टेटमेंट अनुक्रम को समाप्त करने के लिए स्विच के अंदर ब्रेक स्टेटमेंट का उपयोग किया जाता है।
- ब्रेक स्टेटमेंट वैकल्पिक है। यदि छोड़ दिया जाता है, तो निष्पादन अगले मामले में जारी रहेगा।
- डिफ़ॉल्ट विवरण वैकल्पिक है और स्विच ब्लॉक के अंदर कहीं भी प्रकट हो सकता है। मामले में, यदि यह अंत में नहीं है, तो अगले केस स्टेटमेंट के निष्पादन को छोड़ने के लिए डिफ़ॉल्ट स्टेटमेंट के बाद एक ब्रेक स्टेटमेंट रखा जाना चाहिए।

उदाहरण:

निम्नलिखित जावा प्रोग्राम पर विचार करें, यह एक इंट नामित दिन घोषित करता है जिसका मूल्य एक दिन (1- 7) का प्रतिनिधित्व करता है। कोड स्विच स्टेटमेंट का उपयोग करके दिन के मूल्य के आधार पर दिन का नाम प्रदर्शित करता है।

```
// Java program to demonstrate switch case
// with primitive(int) data type
public class Test {
    public static void main(String[] args)
    {
        int day = 5;
        String dayString;

        // switch statement with int data type
        switch (day) {
            case 1:
                dayString = "Monday";
                break;
            case 2:
                dayString = "Tuesday";
                break;
            case 3:
                dayString = "Wednesday";
                break;
```

```
    case 4:
        dayString = "Thursday";
        break;
    case 5:
        dayString = "Friday";
        break;
    case 6:
        dayString = "Saturday";
        break;
    case 7:
        dayString = "Sunday";
        break;
    default:
        dayString = "Invalid day";
        break;
}
System.out.println(dayString);
}
```

आउटपुट:

Friday

लूपिंग स्टेटमेंट का उपयोग कैसे करें

प्रोग्रामिंग भाषाओं में लूपिंग एक ऐसी सुविधा है जो निर्देशों/कार्यों के एक सेट को बार-बार निष्पादित करने की सुविधा प्रदान करती है जबकि कुछ शर्त सत्य का मूल्यांकन करती है। जावा लूप को निष्पादित करने के तीन तरीके प्रदान करता है। जबकि सभी तरीके समान बुनियादी कार्यक्षमता प्रदान करते हैं, वे अपने सिंटैक्स और स्थिति की जाँच के समय में भिन्न होते हैं।

लूप के लिए कैसे उपयोग करें

लूप के लिए सिंटैक्स:

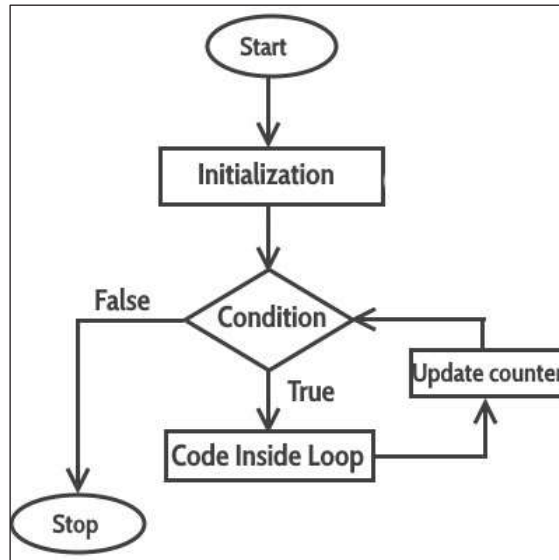
```
for(initialization; condition ; increment/decrement)
{
```



```
statement (s) ;
}
```

फॉर लूप (for loop) का निष्पादन का प्रवाह

जैसे ही एक प्रोग्राम निष्पादित होता है, दुभाषिया हमेशा इस बात पर नज़र रखता है कि कौन सा स्टेटमेंट निष्पादित होने वाला है। इसे हम नियंत्रण प्रवाह या प्रोग्राम के निष्पादन का प्रवाह कहते हैं।



पहला चरण: लूप के लिए, इनिशियलाइज़ेशन पहले और केवल एक बार होता है, जिसका अर्थ है कि लूप का इनिशियलाइज़ेशन भाग केवल एक बार निष्पादित होता है।

दूसरा चरण: लूप के लिए स्थिति का मूल्यांकन प्रत्येक पुनरावृत्ति पर किया जाता है, यदि स्थिति सत्य है तो लूप बाँडी के अंदर के कथन निष्पादित हो जाते हैं। एक बार जब स्थिति झूठी हो जाती है, तो लूप के लिए स्टेटमेंट निष्पादित नहीं होता है और नियंत्रण लूप के बाद प्रोग्राम में अगले स्टेटमेंट में स्थानांतरित हो जाता है।

तीसरा चरण: लूप के शरीर के प्रत्येक निष्पादन के बाद, लूप के लिए वृद्धि/कमी भाग लूप काउंटर को अद्यतन करता है।

चौथा चरण: तीसरे चरण के बाद, नियंत्रण दूसरे चरण में चला जाता है और स्थिति का पुनर्मूल्यांकन किया जाता है।

सिम्पल फॉर लूप (Simple For Loop) का उदाहरण

```
class ForLoopExample {
    public static void main(String args[]) {
        for(int i=10; i>1; i--){
            System.out.println("The value of i is: "+i);
        }
    }
}
```

```
}
```

आउटपुट:

```
The value of i is: 10
The value of i is: 9
The value of i is: 8
The value of i is: 7
The value of i is: 6
The value of i is: 5
The value of i is: 4
The value of i is: 3
The value of i is: 2
```

उपरोक्त कार्यक्रम में:

इंट $i = 1$ इनिशियलाइजेशन एक्सप्रेशन है

$i > 1$ कंडीशन है (बूलियन एक्सप्रेशन)

i - डिक्रीमेंट ऑपरेशन

इंफिनाइट फॉर लूप (Infinite for loop)

बूलियन एक्सप्रेशन और इंक्रीमेंट/डिक्रीमेंट ऑपरेशन को ऑर्डिनेशन का महत्व:

```
class ForLoopExample2 {
    public static void main(String args[]) {
        for(int i=1; i>=1; i++){
            System.out.println("The value of i is: "+i);
        }
    }
}
```

यह एक अनंत लूप है क्योंकि स्थिति कभी भी झूठी नहीं होगी। इनिशियलाइजेशन स्टेप वैरिएबल i के मान को 1 पर सेट कर रहा है, क्योंकि हम i के मान को बढ़ा रहे हैं; यह हमेशा 1 (बूलियन एक्सप्रेशन: $i > 1$) से अधिक होगा, इसलिए यह कभी भी गलत नहीं होगा। यह अंततः अनंत लूप स्थिति की ओर ले जाएगा। इस प्रकार यह निर्धारित करने के लिए कि लूप किसी समय समाप्त होगा या नहीं, यह निर्धारित करने के लिए बूलियन अभिव्यक्ति और वृद्धि/कमी संचालन के बीच समन्वय देखना महत्वपूर्ण है।

लूप के लिए अनंत का एक और उदाहरण यहां दिया गया है:

```
// infinite loop
```

```
for ( ; ; ) {
    // statement(s)
}
```

व्हाइल लूप (while Loop) का उपयोग कैसे करें

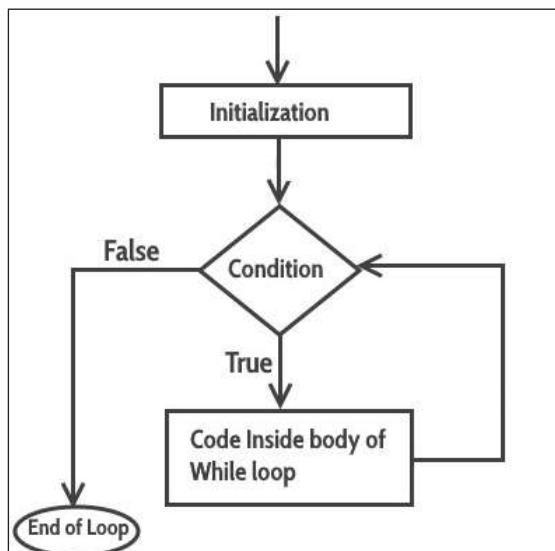
व्हाइल लूप (while Loop) के सिंटैक्स

```
while(condition)
{
    statement(s);
}
```

व्हाइल लूप (while Loop) कैसे काम करता है?

व्हाइल लूप में, स्थिति का मूल्यांकन पहले किया जाता है और यदि यह सही होता है तो व्हाइल लूप के अंदर के स्टेटमेंट निष्पादित होते हैं। जब स्थिति गलत होती है, तो नियंत्रण लूप से बाहर आता है और लूप के बाद अगले स्टेटमेंट पर चला जाता है।

नोट: व्हाइल लूप का उपयोग करते समय ध्यान देने योग्य महत्वपूर्ण बिंदु यह है कि हमें लूप के अंदर इंक्रीमेंट या डिक््रीमेंट स्टेटमेंट का उपयोग करने की आवश्यकता होती है ताकि लूप बेरिण्डल प्रत्येक पुनरावृत्ति पर बदल जाए, और किसी बिंदु पर स्थिति गलत हो। इस तरह हम लूप के निष्पादन को समाप्त कर सकते हैं अन्यथा लूप अनिश्चित काल तक निष्पादित होगा।



सिम्पल व्हाइल लूप (Simple while Loop) का उदाहरण

व्हाइल लूप वर्ग का उदाहरण {

```
public static void main(String args[]) {
    int i=10;
```

```
        while(i>1){
            System.out.println(i);
            i--;
        }
    }
}
```

आउटपुट:

```
10
9
8
7
6
5
4
3
2
```

इंफिनाइट व्हाइल लूप (Infinite while loop)

```
class WhileLoopExample2 {
    public static void main(String args[]){
        int i=10;
        while(i>1)
        {
            System.out.println(i);
            i++;
        }
    }
}
```

यह लूप कभी खत्म नहीं होगा, यह एक अनंत व्हाइल लूप है। ऐसा इसलिए है क्योंकि स्थिति $i > 1$ है जो हमेशा सही होगी क्योंकि हम लूप के दौरान i के मान को अंदर बढ़ा रहे हैं।

लूप के दौरान अनंत का एक और उदाहरण यहां दिया गया है:

```
while (true){
```

```

statement (s) ;
}

```

डू-व्हाइल लूप (do-while Loop) का उपयोग कैसे करें

डू-व्हाइल लूप व्हाइल लूप के समान है, हालांकि उनके बीच एक अंतर है: व्हाइल लूप, लूप बाँडी के निष्पादन से पहले स्थिति का मूल्यांकन किया जाता है, लेकिन लूप के शरीर के निष्पादन के बाद लूप की स्थिति का मूल्यांकन किया जाता है।

डू-व्हाइल लूप का सिंटैक्स:

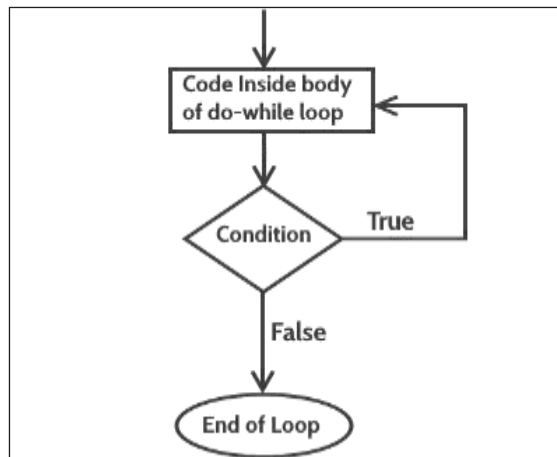
```

do
{
    statement (s) ;
} while (condition) ;

```

डू-व्हाइल लूप (do-while Loop) कैसे काम करता है?

सबसे पहले, लूप के अंदर के स्टेटमेंट को निष्पादित किया जाता है और फिर स्थिति का मूल्यांकन किया जाता है, यदि स्थिति सही हो जाती है तो नियंत्रण "डू" में स्थानांतरित हो जाता है अन्यथा यह डू-टाइम के बाद अगले कथन पर कूद जाता है।



डू-व्हाइल लूप (do-while Loop) का उदाहरण

```

class DoWhileLoopExample {
    public static void main(String args[]){
        int i=10;
        do{
            System.out.println(i);

```

```
        i--;  
    }while(i>1);  
}  
}
```

आउटपुट:

```
10  
9  
8  
7  
6  
5  
4  
3  
2
```

ब्रेक स्टेटमेंट (Break Statement) का उपयोग कैसे करें

जब लूप के अंदर एक ब्रेक स्टेटमेंट का सामना करना पड़ता है, तो लूप को तुरंत समाप्त कर दिया जाता है और लूप के बाद अगले स्टेटमेंट पर प्रोग्राम कंट्रोल फिर से शुरू हो जाता है।

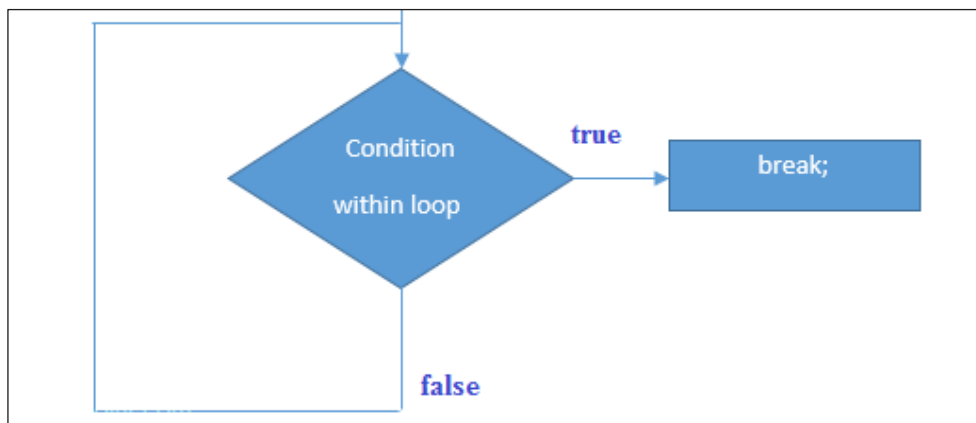
जावा ब्रेक का उपयोग लूप या स्विच स्टेटमेंट को तोड़ने के लिए किया जाता है। यह निर्दिष्ट स्थिति में कार्यक्रम के वर्तमान प्रवाह को तोड़ता है। आंतरिक लूप के मामले में, यह केवल आंतरिक लूप को तोड़ता है।

हम सभी प्रकार के लूप में जावा ब्रेक स्टेटमेंट का उपयोग कर सकते हैं जैसे लूप, व्हाइल लूप और डू- व्हाइल लूप।

सिंटैक्स:

```
jump-statement.
```

```
break.
```



लूप के साथ जावा ब्रेक स्टेटमेंट (Java break Statement with Loop)

उदाहरण:

```
//Java Program to demonstrate the use of break statement
//inside the for loop.
public class BreakExample {
public static void main(String[] args) {
    //using for loop
    for(int i=1;i<=10;i++){
        if(i==5){
            //breaking the loop
            break;
        }
        System.out.println(i);
    }
}
```

आउटपुट:

```
1
2
3
4
```

इनर लूप के साथ जावा ब्रेक स्टेटमेंट (Java Break Statement with Inner Loop)

यह इनर लूप को तभी तोड़ता है जब आप आंतरिक लूप के अंदर ब्रेक स्टेटमेंट का उपयोग करते हैं।

उदाहरण:

```
//Java Program to illustrate the use of break statement
//inside an inner loop
public class BreakExample2 {
public static void main(String[] args) {
    //outer loop
    for(int i=1;i<=3;i++){
```

```

//inner loop
for(int j=1;j<=3;j++){
    if(i==2&&j==2){
        //using break statement inside the inner loop
        break;
    }
    System.out.println(i+" "+j);
}
}
}

```

आउटपुट:

```

1 1
1 2
1 3
2 1
3 1
3 2
3 3

```

लेबलड फॉर लूप के साथ जावा ब्रेक स्टेटमेंट (Java Break Statement with Labeled For Loop)

हम एक लेबल के साथ ब्रेक स्टेटमेंट का उपयोग कर सकते हैं। यह सुविधा जेडीके 1.5 के बाद से पेश की गई है। इसलिए, हम अब जावा में किसी भी लूप को तोड़ सकते हैं चाहे वह बाहरी लूप हो या आंतरिक।

उदाहरण:

```

//Java Program to illustrate the use of continue statement
//with label inside an inner loop to break outer loop
public class BreakExample3 {
public static void main(String[] args) {
    aa:
    for(int i=1;i<=3;i++){
        bb:

```



```

        for(int j=1;j<=3;j++){
            if(i==2&&j==2){
                //using break statement with label
                break aa;
            }
            System.out.println(i+" "+j);
        }
    }
}
}

```

आउटपुट:

```

1 1
1 2
1 3
2 1

```

व्हाइल लूप (while Loop) में जावा ब्रेक स्टेटमेंट (Java Break Statement)

उदाहरण:

```

//Java Program to demonstrate the use of break statement
//inside the while loop.
public class BreakWhileExample {
public static void main(String[] args) {
    //while loop
    int i=1;
    while(i<=10){
        if(i==5){
            //using break statement
            i++;
            break;//it will break the loop
        }
        System.out.println(i);
    }
}
}

```

```
        i++;
    }
}
}
```

आउटपुट:

```
1
2
3
4
```

डू-व्हाइल लूप (do-while Loop) में जावा ब्रेक स्टेटमेंट (Java Break Statement)

उदाहरण:

```
//Java Program to demonstrate the use of break statement
```

```
//inside the Java do-while loop.
```

```
public class BreakDoWhileExample {
public static void main(String[] args) {
    //declaring variable
    int i=1;
    //do-while loop
    do{
        if(i==5){
            //using break statement
            i++;
            break;//it will break the loop
        }
        System.out.println(i);
        i++;
    }while(i<=10);
}
}
```

आउटपुट:

```
1
```

2
3
4

कंटिन्यू स्टेटमेंट (Continue Statement) का उपयोग कैसे करें

कंटिन्यू स्टेटमेंट ज्यादातर लूप के अंदर प्रयोग किया जाता है। जब भी लूप के अंदर इसका सामना किया जाता है, तो नियंत्रण सीधे अगले पुनरावृत्ति के लिए लूप की शुरुआत में कूद जाता है, वर्तमान पुनरावृत्ति के लिए लूप के शरीर के अंदर स्टेटमेंट के निष्पादन को छोड़ देता है। यह विशेष रूप से तब उपयोगी होता है जब आप लूप को जारी रखना चाहते हैं, लेकिन लूप बॉडी में बाकी स्टेटमेंट्स (स्टेटमेंट जारी रखने के बाद) को उस विशेष पुनरावृत्ति के लिए निष्पादित नहीं करना चाहते हैं।

सिंटैक्स:

सेमी कोलन के बाद शब्द जारी रखें।

```
continue;
```

उदाहरण: फॉर लूप के अन्दर कंटिन्यू स्टेटमेंट (Continue Statement Inside for Loop)

```
public class ContinueExample {
    public static void main(String args[]) {
        for (int j=0; j<=6; j++)
        {
            if (j==4)
            {
                continue;
            }

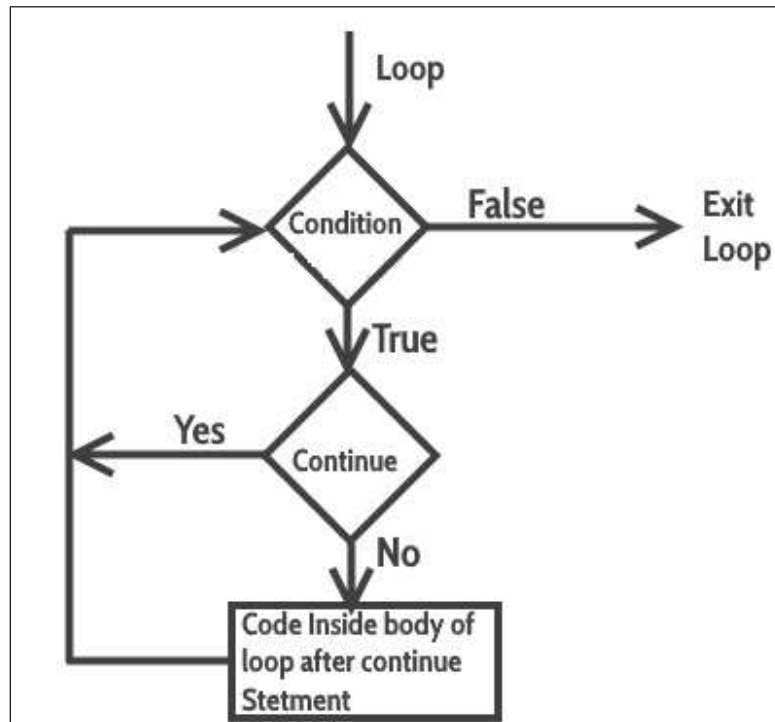
            System.out.print(j+" ");
        }
    }
}
```

आउटपुट:

0 1 2 3 5 6

जैसा कि आपने देखा होगा, आउटपुट में मान 4 लुप्त है, क्यों? क्योंकि जब वैरिएबल `j` का मान 4 होता है, तो प्रोग्राम को एक कंटिन्यू स्टेटमेंट का सामना करना पड़ता है, जो इसे अगले पुनरावृत्ति के लिए लूप की शुरुआत में कूदने के लिए बनाता है, वर्तमान पुनरावृत्ति के लिए स्टेटमेंट को छोड़ देता है (यही कारण है कि प्रिंट-इन निष्पादित नहीं हुआ जब `j` का मान 4 था)।

कंटिन्यू स्टेटमेंट (Continue statement) का फ्लो डायग्राम (Flow Diagram)



उदाहरण: व्हाइल लूप (while Loop) में कंटिन्यू (Continue) का उपयोग

वही चीज़ आप यहां देख सकते हैं। हम काउंटर वैल्यू के लिए इस लूप को 10 से 0 तक पुनरावृत्त कर रहे हैं और जब काउंटर वैल्यू 7 है तो लूप ने प्रिंट स्टेटमेंट को छोड़ दिया और लूप के अगले पुनरावृत्ति को शुरू कर दिया।

```

public class ContinueExample2 {

    public static void main(String args[]){

        int counter=10;

        while (counter >=0)

        {

            if (counter==7)

            {

                counter--;

                continue;

            }

        }

    }

}
  
```

```

        System.out.print(counter+" ");
        counter--;
    }
}
}

```

आउटपुट:

10 9 8 6 5 4 3 2 1 0

डू-व्हाइल लूप (do-while Loop) में कंटिन्यू (Continue) का उदाहरण

```

public class ContinueExample3 {

    public static void main(String args[]){

        int j=0;

        do

        {

            if (j==7)

            {

                j++;

                continue;

            }

            System.out.print(j+ " ");

            j++;

        }while(j<10);

    }

}

```

आउटपुट:

0 1 2 3 4 5 6 8 9

जावा में अभिकथन का उपयोग कैसे करें

एक अभिकथन कार्यक्रम में की गई किसी भी धारणा की शुद्धता का परीक्षण करने की अनुमति देता है।

जावा में एसर्ट स्टेटमेंट का उपयोग करके अभिकथन प्राप्त किया जाता है। स्टेटमेंट को क्रियान्वित करते समय इसे सही माना जाता है। यदि यह विफल हो जाता है, तो जेवीएम एसरशन एरर नाम की एक त्रुटि दिखाता है। यह मुख्य रूप से विकास के दौरान परीक्षण उद्देश्यों के लिए उपयोग किया जाता है।

एसर्ट स्टेटमेंट का उपयोग बूलियन अभिव्यक्ति के साथ किया जाता है और इसे दो अलग-अलग तरीकों से लिखा जा सकता है।

पहला रास्ता :

```
assert expression;
```

दूसरा तरीका:

```
assert expression1 : expression2;
```

एसरशन का उदाहरण :-

```
// Java program to demonstrate syntax of assertion
import java.util.Scanner;

class Test
{
    public static void main( String args[] )
    {
        int value = 15;
        assert value >= 20 : " Underweight";
        System.out.println("value is "+value);
    }
}
```

आउटपुट:

```
value is 15
```

अभिकथन सक्षम करने के बाद आउटपुट:

```
Exception in thread "main" java.lang.AssertionError: Underweight
```

अभिकथन (Assertions) सक्षम करना

डिफॉल्ट रूप से, अभिकथन अक्षम हैं। हमें दिए गए कोड को चलाने की जरूरत है। जावा स्रोत कोड में अभिकथन स्टेटमेंट को सक्षम करने के लिए वाक्य रचना है:

```
java -ea Test
```

या

```
java -enableassertions Test
```

यहां, टेस्ट फ़ाइल का नाम है।

अभिकथन (Assertions) अक्षम करना

जावा में अभिकथन अक्षम करने के लिए वाक्य रचना हैं:

```
java -da Test
```

या

```
java -disableassertions Test
```

यहां, टेस्ट फ़ाइल का नाम है।

अभिकथन का उपयोग क्यों करें

जहाँ भी एक प्रोग्रामर देखना चाहता है कि उसकी धारणाएँ गलत हैं या नहीं।

- यह सुनिश्चित करने के लिए कि एक पहुंच से बाहर दिखने वाला कोड वास्तव में पहुंच योग्य नहीं है।
- यह सुनिश्चित करने के लिए कि टिप्पणियों में लिखी गई धारणाएँ सही हैं।

```
if ((x & 1) == 1)
    { }
    else // x must be even
    { assert (x % 2 == 0); }
```

- यह सुनिश्चित करने के लिए कि डिफ़ॉल्ट स्विच केस तक नहीं पहुंचा है।
- वस्तु की स्थिति की जाँच करने के लिए।
- विधि की शुरुआत में
- विधि आह्वान के बाद।

अभिकथन बनाम सामान्य एक्सेप्शन हैंडलिंग (Assertion Vs Normal Exception Handling)

मुख्य रूप से तार्किक रूप से असंभव स्थितियों की जाँच के लिए अभिकथन का उपयोग किया जाता है। उदाहरण के लिए, उनका उपयोग उस स्थिति की जांच करने के लिए किया जा सकता है, जिसके चलने से पहले एक कोड अपेक्षित है या उसके चलने के बाद स्थिति। सामान्य अपवाद/त्रुटि प्रबंधन के विपरीत, अभिकथन आमतौर पर रन-टाइम पर अक्षम होते हैं।

अभिकथन(Assertions) का प्रयोग कहाँ करें

- निजी तरीकों के लिए, तर्क। निजी तर्क केवल डेवलपर के कोड द्वारा प्रदान किए जाते हैं और डेवलपर तर्कों के बारे में अपनी धारणाओं की जांच करना चाह सकता है।
- सशर्त मामले।
- किसी भी विधि की शुरुआत में शर्तें।

अभिकथन (Assertion) का उपयोग कहां नहीं करना

- त्रुटि संदेशों को बदलने के लिए अभिकथन का उपयोग नहीं किया जाना चाहिए
- सार्वजनिक तरीकों में तर्कों की जाँच के लिए अभिकथनों का उपयोग नहीं किया जाना चाहिए क्योंकि वे उपयोगकर्ता द्वारा प्रदान किए जा सकते हैं। उपयोगकर्ता द्वारा प्रदान की गई त्रुटियों को संभालने के लिए त्रुटि प्रबंधन का उपयोग किया जाना चाहिए।
- कमांड लाइन तर्कों पर अभिकथन का उपयोग नहीं किया जाना चाहिए।

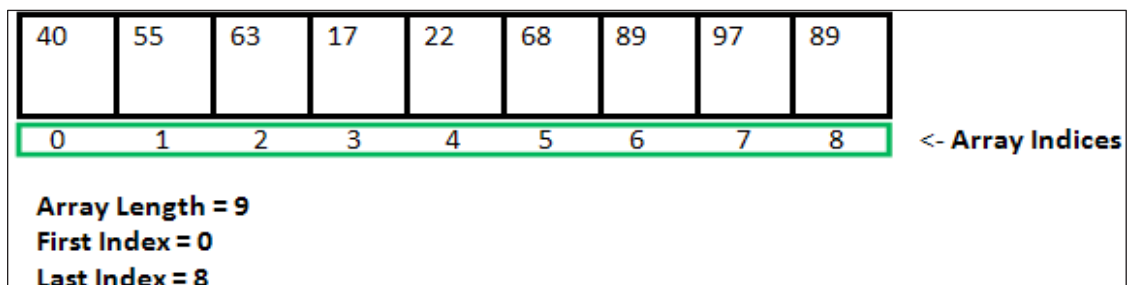
जावा में ऐरेज़ का उपयोग कैसे करें

एक ऐरे(Array) कैसे घोषित करें

एक ऐरे समान-टाइप किए गए चर का एक समूह है जिसे एक सामान्य नाम से संदर्भित किया जाता है। जावा में सारणियां वे C/C++ में अलग तरीके से काम करते हैं। जावा में ऐरेज़ के बारे में कुछ महत्वपूर्ण बिंदु निम्नलिखित हैं।

- जावा में सभी ऐरेज़को गतिशील रूप से आवंटित किया जाता है।
- चूंकि जावा में सरणियाँ ऑब्जेक्ट हैं, हम सदस्य लंबाई का उपयोग करके उनकी लंबाई पा सकते हैं। यह C/C++ से भिन्न है जहां हम sizeof का उपयोग करके लंबाई ज्ञात करते हैं।
- डेटा प्रकार के बाद [] के साथ अन्य चर की तरह एक जावा ऐरे चर भी घोषित किया जा सकता है।
- ऐरे में चर का आदेश दिया जाता है और प्रत्येक का सूचकांक 0 से शुरू होता है।
- जावा ऐरे का उपयोग स्थिर क्षेत्र, स्थानीय चर या विधि पैरामीटर के रूप में भी किया जा सकता है।
- एक ऐरे का आकार एक int(इंट) मान द्वारा निर्दिष्ट किया जाना चाहिए और लंबा या छोटा नहीं होना चाहिए।
- किसी ऐरे प्रकार का प्रत्यक्ष सुपरक्लास ऑब्जेक्ट है।

ऐरे में ऐरे की परिभाषा के आधार पर प्राइमेटिव डेटा प्रकारों के साथ-साथ एक वर्ग की वस्तुएं भी हो सकती हैं। आदिम डेटा प्रकारों के मामले में, वास्तविक मान सन्निहित स्मृति स्थानों में संग्रहित किए जाते हैं। एक वर्ग की वस्तुओं के मामले में, वास्तविक वस्तुओं को ढेर खंड में संग्रहित किया जाता है।



एक ऐरे बनाना, आरंभ करना और एक्सेस करना।

एक आयामी सारणियाँ:

एक आयामी ऐरे डीक्लरेशन का सामान्य रूप है

```
type var-name [ ] ;
```


OR

```
type[] var-name;
```

एक ऐरे डीक्लियरेशन में दो घटक होते हैं: द टाइप और द नेमा। द टाइप ऐरे के तत्व टाइप की डीक्लियरेशन करता है। टाइप तत्व प्रत्येक तत्व के डेटा प्रकार को निर्धारित करता है जिसमें ऐरे शामिल है। इंटर प्रकार की ऐरे की तरह, हम अन्य आदिम डेटा प्रकारों जैसे चार, फ्लोट, डबल..आदि या उपयोगकर्ता परिभाषित डेटा प्रकार (एक वर्ग की वस्तुएं) की एक ऐरे भी बना सकते हैं। इस प्रकार, ऐरे के लिए तत्व प्रकार निर्धारित करता है कि क्या डेटा का प्रकार ऐरे धारण करेगा।

उदाहरण:

```
// both are valid declarations
int intArray[];
or int[] intArray;

byte byteArray[];
short shortsArray[];
boolean booleanArray[];
long longArray[];
float floatArray[];
double doubleArray[];
char charArray[];

// an array of references to objects of
// the class MyClass (a class created by
// user)
MyClass myClassArray[];

Object[] ao,          // array of Object
Collection[] ca;    // array of Collection
                    // of unknown type
```

हालांकि उपरोक्त पहली डीक्लियरेशन इस तथ्य को स्थापित करती है कि इंटर ऐरे एक ऐरे चर है, वास्तव में कोई ऐरे मौजूद नहीं है। यह केवल संकलक को बताता है कि यह (इंटर ऐरे) चर पूर्णांक प्रकार की एक ऐरे रखेगा। पूर्णांकों के वास्तविक, भौतिक ऐरे के साथ इंटर ऐरे को जोड़ने के लिए, आपको नए का उपयोग करके एक आवंटित करना होगा और इसे इंटर ऐरे को असाइन करना होगा।

जावा में एक ऐरे को इंस्टेंट करना

जब एक ऐरे घोषित की जाती है, तो केवल ऐरे का संदर्भ बनाया जाता है। वास्तव में स्मृति बनाने या देने के लिए सरणी,

आप इस तरह एक ऐरे बनाते हैं: नया का सामान्य रूप जैसा कि यह एक-आयामी ऐरे पर लागू होता है, निम्नानुसार प्रकट होता है:

```
var-name = new type [size];
```

यहां, टाइप आवंटित किए जा रहे डेटा के प्रकार को निर्दिष्ट करता है, आकार ऐरे में तत्वों की संख्या निर्दिष्ट करता है, और वार-नेम ऐरे से जुड़े ऐरे चर का नाम है। अर्थात्, किसी ऐरे को आवंटित करने के लिए नए का उपयोग करने के लिए, आपको आवंटित किए जाने वाले तत्वों के टाइप और संख्या को निर्दिष्ट करना होगा।

उदाहरण:

```
int intArray[];    //declaring array
intArray = new int[20]; // allocating memory to array
```

या

```
int[] intArray = new int[20]; // combining both statements in one
```

नोट:

1. नए द्वारा आवंटित ऐरे में तत्व स्वचालित रूप से शून्य (संख्यात्मक प्रकारों के लिए), झूठी (बूलियन के लिए), या शून्य (संदर्भ प्रकारों के लिए) में प्रारंभ हो जाएंगे। जावा में डिफॉल्ट ऐरे मान देखें
2. एक ऐरे प्राप्त करना एक दो-चरणीय प्रक्रिया है। सबसे पहले, आपको वांछित ऐरे प्रकार का एक चर घोषित करना होगा। दूसरा, आपको उस मेमोरी को आवंटित करना होगा जो नए का उपयोग करके ऐरे को धारण करेगा, और इसे ऐरे चर को असाइन करेगा। इस प्रकार, जावा में सभी ऐरेज़ को गतिशील रूप से आवंटित किया जाता है।

शाब्दिक ऐरे (Array Literal)

ऐसी स्थिति में, जहां ऐरे के आकार और ऐरे के चर पहले से ही ज्ञात हैं, ऐरे अक्षर का उपयोग किया जा सकता है।

```
int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };
// Declaring array literal
```

- इस ऐरे की लंबाई निर्मित ऐरे की लंबाई निर्धारित करती है।
- जावा के नवीनतम संस्करणों में नया इंट [] भाग लिखने की कोई आवश्यकता नहीं है

लूप के लिए जावा ऐरे तत्वों उपयोग करना

प्रत्येक तत्व ऐरे में इसकी अनुक्रमणिका के माध्यम से पहुँचा जाता है। सूचकांक 0 से शुरू होता है और -1 (कुल ऐरे का आकार) समाप्त होता है। जावा फॉर लूप का उपयोग करके ऐरे के सभी तत्वों तक पहुँचा जा सकता है।

```
// accessing the elements of the specified array
for (int i = 0; i < arr.length; i++)
    System.out.println("Element at index " + i +
                        " : "+ arr[i]);
```

कार्यान्वयन:

```
// Java program to illustrate creating an array
// of integers, puts some values in the array,
// and prints each value to standard output.

class GFG
{
    public static void main (String[] args)
    {
        // declares an Array of integers.
        int[] arr;

        // allocating memory for 5 integers.
        arr = new int[5];

        // initialize the first elements of the array
        arr[0] = 10;

        // initialize the second elements of the array
        arr[1] = 20;

        //so on...
        arr[2] = 30;
        arr[3] = 40;
        arr[4] = 50;

        // accessing the elements of the specified array
        for (int i = 0; i < arr.length; i++)
            System.out.println("Element at index " + i +
                               " : "+ arr[i]);
    }
}
```

```

    }
}

```

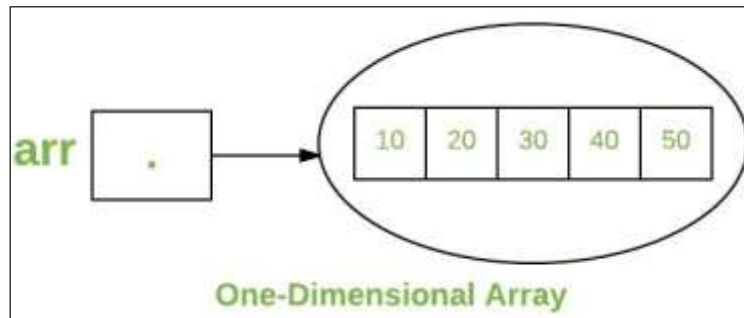
आउटपुट:"

```

Element at index 0 : 10
Element at index 1 : 20
Element at index 2 : 30
Element at index 3 : 40
Element at index 4 : 50

```

आप फ़ोरैच लूप का उपयोग करके जावा ऐरेज़ तक भी पहुँच सकते हैं



ओब्जेक्ट्स के ऐरेज़ (Arrays of Objects)

वस्तुओं की एक ऐरे निम्न प्रकार से साधारण प्रकार के डेटा आइटम की एक ऐरे की तरह ही बनाई जाती है।

```
Student[] arr = new Student[7]; //student is a user-defined class
```

स्टूडेंट ऐरे में छात्र वर्ग के आकार के सात मेमोरी स्पेस होते हैं जिसमें सात छात्र वस्तुओं का पता संग्रहित किया जा सकता है। छात्र वस्तुओं को छात्र वर्ग के निर्माता का उपयोग करके तत्काल किया जाना चाहिए और उनके संदर्भों को ऐरे तत्वों को सौंपा जाना चाहिए इसके अनुसार।

```
Student[] arr = new Student[5];
```

```
// Java program to illustrate creating an array of
// objects
```

```
class Student
```

```
{
```

```
    public int roll_no;
```

```
public String name;
Student(int roll_no, String name)
{
    this.roll_no = roll_no;
    this.name = name;
}
}

// Elements of array are objects of a class Student.
public class GFG
{
    public static void main (String[] args)
    {
        // declares an Array of integers.
        Student[] arr;

        // allocating memory for 5 objects of type Student.
        arr = new Student[5];

        // initialize the first elements of the array
        arr[0] = new Student(1,"aman");

        // initialize the second elements of the array
        arr[1] = new Student(2,"vaibhav");

        // so on...
        arr[2] = new Student(3,"shikar");
        arr[3] = new Student(4,"dharmesh");
        arr[4] = new Student(5,"mohit");
    }
}
```

```
// accessing the elements of the specified array
for (int i = 0; i < arr.length; i++)
    System.out.println("Element at " + i + " : " +
        arr[i].roll_no +" "+ arr[i].name);
}
```

आउटपुट:

```
Element at 0 : 1 aman
Element at 1 : 2 vaibhav
Element at 2 : 3 shikar
Element at 3 : 4 dharmesh
Element at 4 : 5 mohit
```

यदि हम ऐरे साइज़ के बाहर तत्व तक पहुंचने का प्रयास करते हैं तो क्या होता है?

जेवीएम यह इंगित करने के लिए अरे इंडेक्स आउट ऑफ बाउंड एक्सेप्शन (`ArrayIndexOutOfBoundsException`) दिखाता है कि ऐरे को अवैध सूचकांक के साथ एक्सेस किया गया है। सूचकांक या तो ऋणात्मक है या ऐरे के आकार से बड़ा या उसके बराबर है।

```
class GFG
{
    public static void main (String[] args)
    {
        int[] arr = new int[2];
        arr[0] = 10;
        arr[1] = 20;

        for (int i = 0; i <= arr.length; i++)
            System.out.println(arr[i]);
    }
}
```

रनटाइम त्रुटि

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 2
    at GFG.main(File.java:12)
```

आउटपुट:

10

20

बहुआयामी ऐरेज़ (Multidimensional Arrays)

बहुआयामी ऐरे अन्य ऐरे के संदर्भ वाले ऐरे के प्रत्येक तत्व के साथ ऐरे के ऐरे हैं। इन्हें जेगड ऐरेज़ के रूप में भी जाना जाता है। प्रति आयाम वर्ग कोष्ठक ([]) के एक सेट को लंबित करके एक बहुआयामी ऐरे बनाई जाती है। उदाहरण:

```
int[][] intArray = new int[10][20]; //a 2D array or matrix
int[][][] intArray = new int[10][20][10]; //a 3D array
class multiDimensional
{
    public static void main(String args[])
    {
        // declaring and initializing 2D array
        int arr[][] = { {2,7,9},{3,6,1},{7,4,2} };

        // printing 2D array
        for (int i=0; i< 3 ; i++)
        {
            for (int j=0; j < 3 ; j++)
                System.out.print(arr[i][j] + " ");

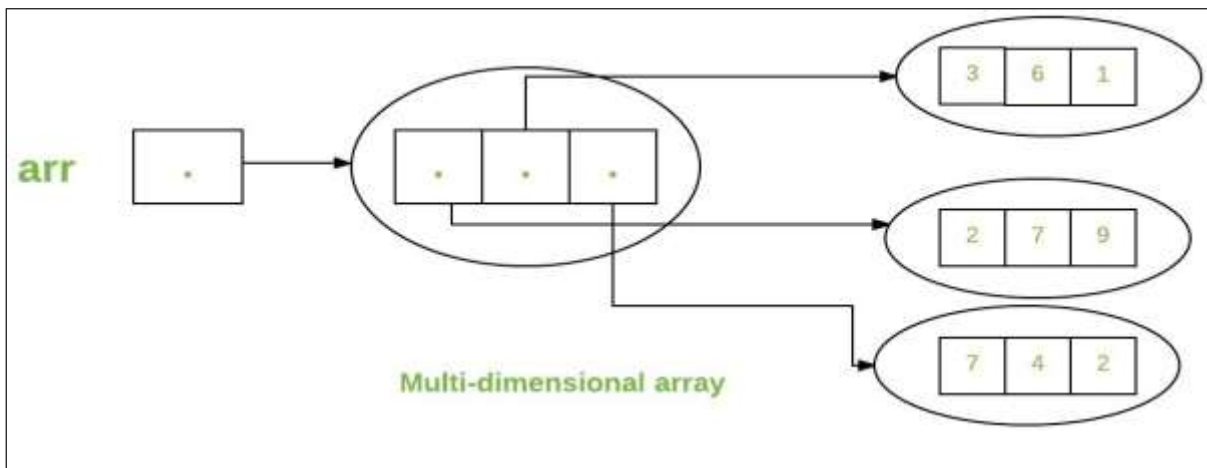
            System.out.println();
        }
    }
}
```

आउटपुट:

2 7 9

3 6 1

7 4 2



विधियों के लिए ऐरे पास करना

वेरिएबल्स की तरह, हम भी विधियों के लिए ऐरेज़ को पारित कर सकते हैं। उदाहरण के लिए, ऐरे के मूल्यों के योग की गणना के लिए प्रोग्राम पास ऐरे योग (sum) विधि से नीचे है।

```
// Java program to demonstrate
// passing of array to method

class Test
{
    // Driver method
    public static void main(String args[])
    {
        int arr[] = {3, 1, 2, 5, 4};

        // passing array to method m1
        sum(arr);
    }

    public static void sum(int[] arr)
    {
        // getting sum of array values
```



```
int sum = 0;

for (int i = 0; i < arr.length; i++)
    sum+=arr[i];

System.out.println("sum of array values : " + sum);
}
}
```

आउटपुट:

```
sum of array values : 15
```

मेथड्स से रिटर्निंग ऐरे (Returning Arrays from Methods)

हमेशा की तरह, एक विधि एक ऐरे भी लौटा सकती है। उदाहरण के लिए, नीचे प्रोग्राम विधि एम1 (m1) से एक ऐरे देता है।

```
// Java program to demonstrate
// return of array from method

class Test
{
    // Driver method
    public static void main(String args[])
    {
        int arr[] = m1();

        for (int i = 0; i < arr.length; i++)
            System.out.print(arr[i]+" ");

    }

    public static int[] m1()
    {
```

```

        // returning array
        return new int[]{1,2,3};
    }
}

```

आउटपुट:

1 2 3

ऐरेज़ के लिए क्लास ऑब्जेक्ट्स

प्रत्येक ऐरे में एक संबद्ध क्लास ऑब्जेक्ट होता है, जो समान घटक प्रकार वाले अन्य सभी ऐरेज़ के साथ साझा किया जाता है।

```

// Java program to demonstrate
// Class Objects for Arrays

class Test
{
    public static void main(String args[])
    {
        int intArray[] = new int[3];
        byte byteArray[] = new byte[3];
        short shortsArray[] = new short[3];

        // array of Strings
        String[] strArray = new String[3];

        System.out.println(intArray.getClass());
        System.out.println(intArray.getClass().getSuperclass());
        System.out.println(byteArray.getClass());
        System.out.println(shortsArray.getClass());
        System.out.println(strArray.getClass());
    }
}

```

आउटपुट:

class [I

```
class java.lang.Object
class [B
class [S
class [Ljava.lang.String;
```

व्याख्या :

- स्ट्रिंग "[I" क्लास ऑब्जेक्ट के लिए रन-टाइम टाइप सिग्नेचर है "घटक के साथ सरणी" इंट टाइप करें"।
- किसी भी ऐरे प्रकार का एकमात्र प्रत्यक्ष सुपरक्लास `java.lang.Object` है।
- स्ट्रिंग "[B" क्लास ऑब्जेक्ट के लिए रन-टाइम टाइप सिग्नेचर है "घटक के साथ सरणी" बाइट टाइप करें"।
- स्ट्रिंग "[S" क्लास ऑब्जेक्ट के लिए रन-टाइम टाइप सिग्नेचर है "घटक के साथ सरणी" शॉर्ट टाइप करें"।
- स्ट्रिंग "[L" क्लास ऑब्जेक्ट के लिए रन-टाइम टाइप सिग्नेचर है "ऐरे विद कंपोनेंट टाइप ऑफ़ ए क्लास"। इसके बाद क्लास का नाम फॉलो किया जाता है।

ऐरे मेम्बर्स (Array Members)

अभी जैसा कि आप जानते हैं कि सारणियाँ एक वर्ग की वस्तु हैं और ऐरेज़ का प्रत्यक्ष सुपरक्लास वर्ग वस्तु है। एक ऐरे प्रकार के सदस्य निम्नलिखित में से सभी हैं:

- सार्वजनिक अंतिम फ़ील्ड लंबाई, जिसमें ऐरे के घटकों की संख्या होती है। लंबाई सकारात्मक या शून्य हो सकता है।
- सभी सदस्य वर्ग वस्तु से विरासत में मिले हैं; ऑब्जेक्ट की एकमात्र विधि जो विरासत में नहीं मिली है, वह है इसकी क्लोन विधि।
- सार्वजनिक विधि क्लोन (), जो क्लास ऑब्जेक्ट में क्लोन विधि को ओवरराइड करती है और कोई चेक अपवाद नहीं फेंकती है।

ऐरेज़ की क्लोनिंग

- जब आप क्लोन करते हैं एक एकल आयामी ऐरे, जैसे ऑब्जेक्ट [], एक "डीप कॉपी" के साथ किया जाता है संदर्भों के विपरीत मूल ऐरे के तत्वों की प्रतियां युक्त नई सरणी।

- `// Java program to demonstrate`
- `// cloning of one-dimensional arrays`

```
class Test
{
    public static void main(String args[])
    {
```

```

int intArray[] = {1,2,3};

int cloneArray[] = intArray.clone();

// will print false as deep copy is created
// for one-dimensional array
System.out.println(intArray == cloneArray);

for (int i = 0; i < cloneArray.length; i++) {
    System.out.print(cloneArray[i]+" ");
}
}
}

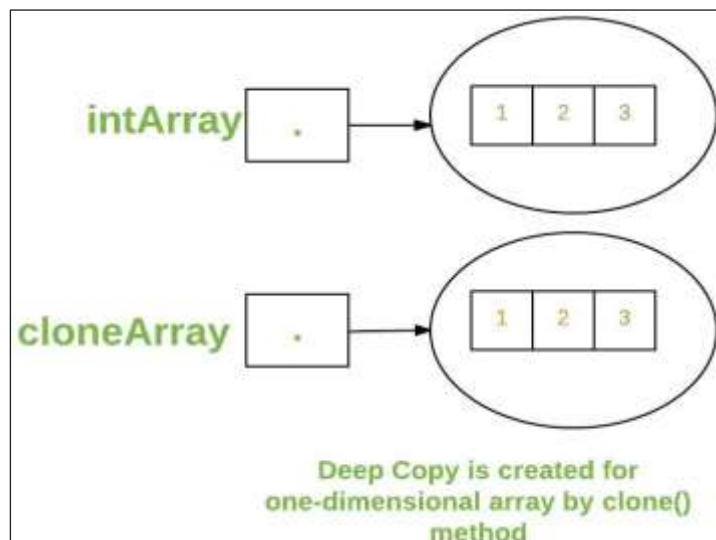
```

आउटपुट:

false

1 2 3

एक बहुआयामी ऐरे का एक क्लोन (जैसे ऑब्जेक्ट [] []) एक "शेलो कॉपी" है, हालांकि, यह कहना है कि यह प्रत्येक तत्व ऐरे के साथ केवल एक नया ऐरे बनाता है जो मूल तत्व ऐरे का संदर्भ देता है लेकिन उप-ऐरे साझा की जाती है।



```

// Java program to demonstrate
// cloning of multi-dimensional arrays

```

```

class Test
{
    public static void main(String args[])
    {
        int intArray[][] = {{1,2,3},{4,5}};

        int cloneArray[][] = intArray.clone();

        // will print false
        System.out.println(intArray == cloneArray);

        // will print true as shallow copy is created
        // i.e. sub-arrays are shared
        System.out.println(intArray[0] == cloneArray[0]);
        System.out.println(intArray[1] == cloneArray[1]);

    }
}

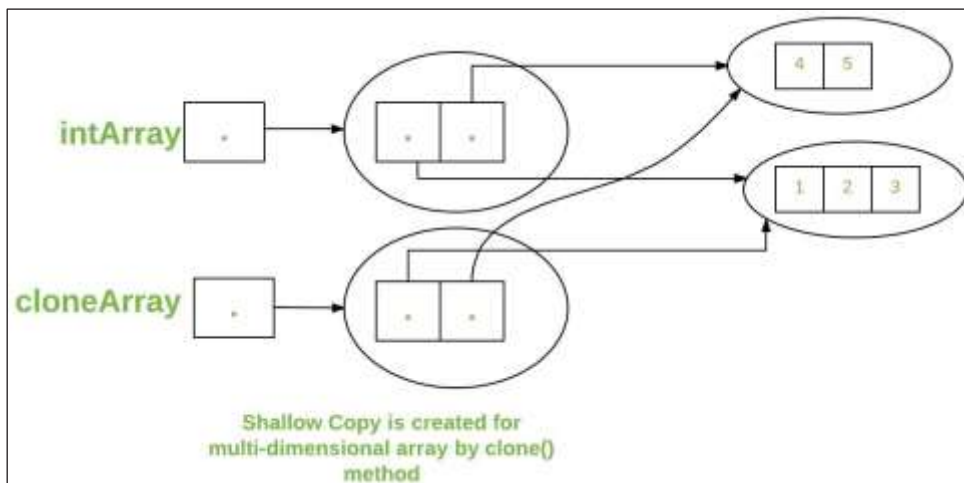
```

आउटपुट:

false

true

true



जावा में एक ऐरे(Array) का प्रारंभ कैसे करें

```

1 public class JavaArrayInitialization {
2
3     // examples for initializing an array in java
4     public static void main(String[] args) {
5         // initialize primitive one dimensional array
6         int[] arrInt = new int[5];
7
8         // initialize Object one dimensional array
9         String[] strArr; // declaration
10
11        strArr = new String[4]; // initialization
12
13        // initialize multidimensional array
14        int[][] twoArrInt = new int[4][5];
15
16        // multidimensional array initialization with only leftmost dimension
17        int[][] twoIntArr = new int[2][];
18        // complete initialization is required before we use the array
19        twoIntArr[0] = new int[2];
20        twoIntArr[1] = new int[3];
21
22        // array initialization using shortcut syntax
23        int[] arr1 = { 1, 2, 3 };
24        int[][] arr2 = { { 1, 2 }, { 1, 2, 3 } };
25
26        int[] twoArrInt1[] = new int[4][5];
27
28        int twoIntArr2[][] = new int[5][];
29    }
30
31 }

```

एक ऐरे एक आयामी हो सकती है या यह बहुआयामी भी हो सकती है। जब हम किसी ऐरे की लंबाई को लागू करते हैं, तो यह ऐरे में पंक्तियों की संख्या या सबसे बाएं आयाम का मान लौटाता है।

हम नए कीवर्ड का उपयोग करके या शॉर्टकट सिंटैक्स का उपयोग करके एक ऐरे को प्रारंभ कर सकते हैं जो एक ही समय में ऐरे बनाता और आरंभ करता है।

जब हम नए ऑपरेटर का उपयोग करके एक ऐरे बनाते हैं, तो हमें इसके आयाम प्रदान करने की आवश्यकता होती है। बहुआयामी ऐरे के लिए, हम ऐरे के सभी आयाम या केवल सबसे बाएं आयाम प्रदान कर सकते हैं।

जावा में एक ऐरे शुरू करना – प्राइमिटिव टाइप (Primitive Type)

```

//initialize primitive one dimensional array
int[] arrInt = new int[5];

```

जावा में एक ऐरे शुरू करना - ऑब्जेक्ट टाइप (Object Type)

```

//initialize Object one dimensional array
String[] strArr; //declaration

strArr = new String[4]; //initialization

```

जावा में एक बहुआयामी ऐरे(Multidimensional Array) प्रारंभ करना

```
//initialize multidimensional array
int[][] twoArrInt = new int[4][5];

//multidimensional array initialization with only leftmost dimension
int[][] twoIntArr = new int[2][];
twoIntArr[0] = new int[2];
twoIntArr[1] = new int[3]; //complete initialization is required before we use
the array
```

शॉर्टकट सिंटैक्स का उपयोग करके जावा में एक ऐरे को कैसे प्रारंभ करें

```
//array initialization using shortcut syntax
int[] arrI = {1,2,3};
int[][] arrI2 = {{1,2}, {1,2,3}};
```

यदि आप ऊपर नोटिस करते हैं, तो द्वि-आयामी ऐरे एआरआरआई 2 एक सममित मैट्रिक्स नहीं है। ऐसा इसलिए है क्योंकि जावा में एक बहुआयामी ऐरे वास्तव में ऐरे की एक ऐरे है।

जावा में एक ऐरे को प्रारंभ करने के अमान्य तरीके

किसी ऐरे को प्रारंभ करने के कुछ अमान्य तरीके यहां दिए गए हैं।

```
//invalid because dimension is not provided
int[] a = new int[];

//invalid because leftmost dimension value is not provided
int[][] aa = new int[][5];
```

यहाँ जावा में ऐरेज़ को घोषित करने के कुछ अन्य रूप हैं लेकिन भ्रम से बचने के लिए उन्हें दृढ़ता से हतोत्साहित किया जाता है:

```
int[] twoArrInt[] = new int[4][5];

int twoIntArr[][] = new int[5][];
```

जावा में एक ऐरे घोषित करने और आरंभ करने के लिए बस इतना ही।

ऐरे तत्वों तक कैसे पहुँचें

एक बार ऐरे बन जाने के बाद, आप ऐरे के नाम का उपयोग करके एक ऐरे तत्व तक पहुंच सकते हैं और उसके वर्ग कोष्ठक की एक जोड़ी के बीच संलग्न एक सूचकांक का उपयोग कर सकते हैं। किसी ऐरे का सूचकांक या सबस्क्रिप्ट ऐरे में किसी तत्व की स्थिति को इंगित करता है।

ऐरे में पहले तत्व का सूचकांक हमेशा 0 (शून्य) होता है, दूसरे तत्व का सूचकांक 1 होता है और इसी तरह। अंतिम तत्व का सूचकांक हमेशा ऐरे में तत्वों की संख्या से कम होता है। एक ऐरे तत्व तक पहुँचने के लिए वाक्य रचना है:

```
arrayRefVar[index];
```

यहां, अनुक्रमणिका एक शाब्दिक पूर्णांक या एक व्यंजक है जो एक का मूल्यांकन करता है।

हमारे उदाहरण में, ऐरे का पहला तत्व num[0] है, दूसरा तत्व num[1] है और अंतिम तत्व num[4] है।

```
public class AccessingArrayElements
{
    public static void main (String [] args)
    {
        int[] num = new int[5];
        num[0] = 5; //Assigning value 5 to element at index 0
        num[1] = 15;
        num[2] = 25;
        num[3] = 30;
        num[4] = 50;

        System.out.println("Array Element num[0] : " + num[0]);
        System.out.println("Array Element num[1] : " + num[1]);
        System.out.println("Array Element num[2] : " + num[2]);
        System.out.println("Array Element num[3] : " + num[3]);
        System.out.println("Array Element num[4] : " + num[4]);
    }
}
```



```

C:\Java> javac AccessingArrayElements.java
C:\Java> java AccessingArrayElements
Array Element num[0] : 5
Array Element num[1] : 15
Array Element num[2] : 25
Array Element num[3] : 30
Array Element num[4] : 50
C:\Java>

```

आइए अब विचार करें एक अन्य उदाहरण जो किसी ऐरे के तत्वों को इनपुट करता है और उसकी सामग्री को प्रिंट करता है।

```
import java.util.Scanner; //program uses Scanner class
```



```

public class ArrayElements
{
    public static void main(String[] args)
    {
        int[] numArray = new int[5];
        int i;

        Scanner input=new Scanner(System.in);

        System.out.print("Enter the 5 Array Elements : ");

        for(i=0; i<5; i++)
            numArray[i]=input.nextInt(); //Read number

        for(i=0; i<numArray.length; i++)
            System.out.println("Array element[" + i + "]" : " +numArray[i]);
    }
}

```



```

F:\Java>javac ArrayElements.java
F:\Java>java ArrayElements
Enter the 5 Array Elements : 1 2 3 4 5
Array element[0] : 1
Array element[1] : 2
Array element[2] : 3
Array element[3] : 4
Array element[4] : 5
F:\Java>_

```

यह उदाहरण एक ऐरे इनपुट करता है और इसकी सामग्री को प्रिंट करता है।

स्टेटमेंट,

```
int[] numArray = new int[5];
```

बनाता है एक numArray ऐरे चर जो 5 तत्वों वाले एक इंट-ऐरे को संदर्भित करता है। हम कीबोर्ड से इनपुट प्राप्त करने के लिए एक स्कैनर ऑब्जेक्ट बनाते हैं। अंत में, जब सभी इनपुट किए जाते हैं तो इसे प्रिंट किया जाता है।

फॉर-ईच लूप (for-each Loop) का कैसे उपयोग करें

फॉर-ईच लूप के लिए जावा या लूप के लिए एन्हांसड को जे2एसई 5.0 (J2SE 5.0) के बाद से पेश किया गया है। यह जावा में ऐरे या संग्रह को पार करने के लिए एक वैकल्पिक दृष्टिकोण प्रदान करता है। यह मुख्य रूप से ऐरे या संग्रह तत्वों को पार करने के लिए उपयोग किया जाता है। ईच लूप के लिए लाभ यह है कि यह बग की संभावना को समाप्त करता है और कोड को अधिक पठनीय बनाता है। इसे प्रत्येक लूप के लिए जाना जाता है क्योंकि यह प्रत्येक तत्व को एक-एक करके पार करता है।

लूप के लिए एन्हांसड का दोष यह है कि यह तत्वों को उल्टे क्रम में पार नहीं कर सकता है।

यहां, आपके पास किसी भी तत्व को छोड़ने का विकल्प नहीं है क्योंकि यह इंडेक्स के आधार पर काम नहीं करता है। इसके अलावा, आप केवल विषम या सम तत्वों को पार नहीं कर सकते।

लेकिन, ऐरे और संग्रह के तत्वों को पार करने के लिए प्रत्येक लूप के लिए जावा का उपयोग करने की अनुशंसा की जाती है क्योंकि यह कोड को पठनीय बनाता है।

लाभ

- यह कोड को और अधिक पठनीय बनाता है।
- यह प्रोग्रामिंग त्रुटियों की संभावना को समाप्त करता है।

सिंटैक्स

प्रत्येक लूप के लिए जावा के सिंटैक्स में एक कोलन (:) के बाद वेरिएबल के साथ डेटा_टाइप होता है, फिर ऐरे या संग्रह।

```
for(data_type variable : array | collection){
//body of for-each loop
}
```

यह कैसे काम करता है?

जावा फॉर-ईच लूप के लिए अंतिम तत्व तक ऐरे या संग्रह को पार करता है। ईच एलिमेंट के लिए, यह तत्व को चर में संग्रहित करता है और प्रत्येक लूप के लिए शरीर को निष्पादित करता है।=

फॉर ईच लूप (for-each loop) का उदाहरण: ऐरे तत्वों को ट्रेवर्स करना

```
//An example of Java for-each loop
class ForEachExample1{
    public static void main(String args[]){
        //declaring an array int
        arr[]={12,13,14,44};
        //traversing the array with for-each loop
        for(int i:arr){
            System.out.println(i);
        }
    }
}
```

आउटपुट:

12

12

14

44

आइए प्रत्येक लूप के लिए जावा का एक और देखें जहां हम तत्वों को कुल करने जा रहे हैं।

```
class ForEachExample1{
    public static void main(String args[]){
        int arr[]={12,13,14,44};
        int total=0;
        for(int i:arr){
            total=total+i;
        }
        System.out.println("Total: "+total);
    }
}
```

आउटपुट:

Total: 83

प्रत्येक लूप के लिए उदाहरण: संग्रह तत्वों को पार करना

```
import java.util.*;
class ForEachExample2{
    public static void main(String args[]){
        //Creating a list of elements
        ArrayList<String> list=new ArrayList<String>();
        list.add("vimal");
        list.add("sonoo");
        list.add("ratan");
        //traversing the list of elements using for-each loop
        for(String s:list){
            System.out.println(s);
        }
    }
}
```

आउटपुट:

```
vimal
sonoo
ratan
```

बहुआयामी ऐरेज़ का उपयोग कैसे करें

बहुआयामी ऐरेज़ को सरल शब्दों में ऐरेज़के ऐरे के रूप में परिभाषित किया जा सकता है। बहुआयामी ऐरेज़ में डेटा ऐरेबद्ध रूप में (पंक्ति प्रमुख क्रम में) संग्रहित किया जाता है।

सिंटैक्स:

```
data_type[1st dimension][2nd dimension][..[Nth dimension] array_name = new
data_type[size1][size2]...[sizeN];
```

जहां:

- डेटा_टाइप: ऐरे में संग्रहित किए जाने वाले डेटा का प्रकार। उदाहरण के लिए: int, char, आदि।
- आयाम: बनाई गई ऐरे का आयाम। उदाहरण के लिए: 1डी, 2डी, आदि।
- ऐरे_नेम: ऐरे का नाम
- आकार1, आकार2,..., आकार एन: क्रमशः आयामों के आकार।

उदाहरण:

Two dimensional array:

```
int[][] twoD_arr = new int[10][20];
```

Three dimensional array:

```
int[][][] threeD_arr = new int[10][20][30];
```

बहुआयामी ऐरेज़ का आकार: एक बहुआयामी ऐरे में संग्रहित किए जा सकने वाले तत्वों की कुल संख्या की गणना सभी आयामों के आकार को गुणा करके की जा सकती है।

उदाहरण के लिए:

array int [][] x = new int [10] [20] कुल (10 * 20) = 200 तत्वों को संग्रहित कर सकता है। इसी तरह, array int [] [] [] x = new int [5] [10] [20] कुल (5 * 10 * 20) = 1000 तत्वों को संग्रहित कर सकता है।

द्वि - आयामी ऐरे (2डी- ऐरे)

द्वि-आयामी ऐरे बहुआयामी ऐरे का सबसे सरल रूप है। एक द्वि-आयामी ऐरे को आसानी से समझने के लिए एक-आयामी ऐरे की एक ऐरे के रूप में देखा जा सकता है।

डीक्लियरेशन की अप्रत्यक्ष विधि:

- डीक्लियरेशन - सिंटैक्स:

```
data_type[][] array_name = new data_type[x][y];
```

For example: `int[][] arr = new int[10][20];`

आरंभीकरण - सिंटैक्स:

```
array_name[row_index][column_index] = value;
```

For example: `arr[0][0] = 1;`

उदाहरण:

```
class GFG {
    public static void main(String[] args)
    {

        int[][] arr = new int[10][20];
        arr[0][0] = 1;

        System.out.println("arr[0][0] = " + arr[0][0]);
    }
}
```

आउटपुट:

```
arr[0][0] = 1
```

डीक्लियरेशन की प्रत्यक्ष विधि:

सिंटैक्स:

```
data_type[][] array_name = {
                                {valueR1C1, valueR1C2, ....},
                                {valueR2C1, valueR2C2, ....}
                                };
```

For example: `int[][] arr = {{1, 2}, {3, 4}};`

उदाहरण:

```
class GFG {
    public static void main(String[] args)
    {
        int[][] arr = { { 1, 2 }, { 3, 4 } };

        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                System.out.println("arr[" + i + "][" + j + "] = "
                    + arr[i][j]);
    }
}
```

आउटपुट:

```
arr[0][0] = 1
arr[0][1] = 2
arr[1][0] = 3
arr[1][1] = 4
```

द्वि-आयामी ऐरेज़ के तत्वों तक पहुंचना

द्वि-आयामी ऐरेज़ में तत्वों को आमतौर पर `x[i][j]` द्वारा संदर्भित किया जाता है जहां 'i' पंक्ति संख्या है और 'j' स्तंभ संख्या है।

सिंटैक्स:

```
x[row_index][column_index]
```

उदाहरण के लिए:

```
int[][] arr = new int[10][20];
arr[0][0] = 1;
```

उपरोक्त उदाहरण पहली पंक्ति और पहले कॉलम में मौजूद तत्व का प्रतिनिधित्व करता है।

नोट: ऐरेज़ में यदि ऐरे का आकार N (एन) है। इसका सूचकांक 0 से N-1 तक होगा। इसलिए, रॉ_सूचकांक 2 के लिए, वास्तविक पंक्ति संख्या $2+1 = 3$ है।

उदाहरण:

```
class GFG {
    public static void main(String[] args)
```

```

{
    int[][] arr = { { 1, 2 }, { 3, 4 } };

    System.out.println("arr[0][0] = " + arr[0][0]);
}
}

```

आउटपुट:

```
arr[0][0] = 1
```

सारणीबद्ध प्रारूप में 2डी ऐरे का प्रतिनिधित्व: एक द्वि-आयामी ऐरे को 'x' पंक्तियों और 'y' स्तंभों वाली तालिका के रूप में देखा जा सकता है जहां पंक्ति संख्या 0 से (x-1) तक होती है और स्तंभ संख्या 0 से (वाई-1)। 3 पंक्तियों और 3 स्तंभों के साथ एक द्वि-आयामी ऐरे 'x' नीचे दिखाया गया है:

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

सारणीबद्ध प्रारूप में 2डी ऐरे प्रिंट करें:

दो-आयामी ऐरे के सभी तत्वों को आउटपुट करने के लिए, लूप के लिए नेस्टेड का उपयोग करें। इसके लिए दो लूप की आवश्यकता होती है, एक पंक्तियों को पार करने के लिए और दूसरा स्तंभों को पार करने के लिए।

उदाहरण:

```

class GFG {
    public static void main(String[] args)
    {

        int[][] arr = { { 1, 2 }, { 3, 4 } };

        for (int i = 0; i < 2; i++) {

```

```

        for (int j = 0; j < 2; j++) {
            System.out.print(arr[i][j] + " ");
        }

        System.out.println();
    }
}

```

आउटपुट:

```

1 2
3 4

```

त्रि-आयामी ऐरे (3डी-ऐरे)

त्रि-आयामी ऐरे एक बहुआयामी ऐरे का एक जटिल रूप है। एक त्रि-आयामी ऐरे को आसान समझने के लिए द्वि-आयामी ऐरे की एक ऐरे के रूप में देखा जा सकता है।

डीक्लियरेशन की अप्रत्यक्ष विधि:

- डीक्लियरेशन - सिंटैक्स:

```
data_type[][][] array_name = new data_type[x][y][z];
```

For example: `int[][][] arr = new int[10][20][30];`

- आरंभीकरण - सिंटैक्स:

```
array_name[array_index][row_index][column_index] = value;
```

For example: `arr[0][0][0] = 1;`

उदाहरण:

```

class GFG {
    public static void main(String[] args)
    {

        int[][][] arr = new int[10][20][30];
        arr[0][0][0] = 1;
    }
}

```



```

        System.out.println("arr[0][0][0] = " + arr[0][0][0]);
    }
}

```

आउटपुट:

```
arr[0][0][0] = 1
```

डीक्लयरेशन की प्रत्यक्ष विधि:

सिंटैक्स:

```

data_type[][][] array_name = {
    {
        {valueA1R1C1, valueA1R1C2, ....},
        {valueA1R2C1, valueA1R2C2, ....}
    },
    {
        {valueA2R1C1, valueA2R1C2, ....},
        {valueA2R2C1, valueA2R2C2, ....}
    }
};

```

For example: `int[][][] arr = { {{1, 2}, {3, 4}}, {{5, 6}, {7, 8}} };`

उदाहरण:

```

class GFG {
    public static void main(String[] args)
    {

        int[][][] arr = { { { 1, 2 } }, { 3, 4 } }, { { 5, 6 }, { 7, 8 } } };

        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                for (int z = 0; z < 2; z++)

```

के लिये (इंट जेड = 0; जेड < 2; जेड ++)

```

        System.out.println("arr[" + i
                            + «] [«
                            + j + «] ["
                            + z + «] = "
                            + arr[i][j][z]);
    }
}

```

आउटपुट:

```

arr[0][0][0] = 1
arr[0][0][1] = 2
arr[0][1][0] = 3
arr[0][1][1] = 4
arr[1][0][0] = 5
arr[1][0][1] = 6
arr[1][1][0] = 7
arr[1][1][1] = 8

```

त्रि-आयामी ऐरेज़ का तत्वों तक पहुंच

तत्वों त्रि-आयामी ऐरेज़में आमतौर पर `x[i][j][k]` द्वारा संदर्भित किया जाता है जहां 'i' ऐरे संख्या है, 'j' पंक्ति संख्या है और 'k' स्तंभ संख्या है।
सिंटैक्स:

```
x[array_index][row_index][column_index]
```

उदाहरण के लिए:

```

int[][][] arr = new int[10][20][30];
arr[0][0][0] = 1;

```

उपरोक्त उदाहरण पहली पंक्ति में मौजूद तत्व और घोषित 3डी ऐरे में पहली ऐरे के पहले कॉलम का प्रतिनिधित्व करता है।

नोट: ऐरेज़में यदि ऐरे का आकार N है। इसका सूचकांक 0 से N-1 तक होगा। इसलिए, राँ सूचकांक 2 के लिए, वास्तविक पंक्ति संख्या $2+1 = 3$ है।

उदाहरण:

```
class GFG {
```

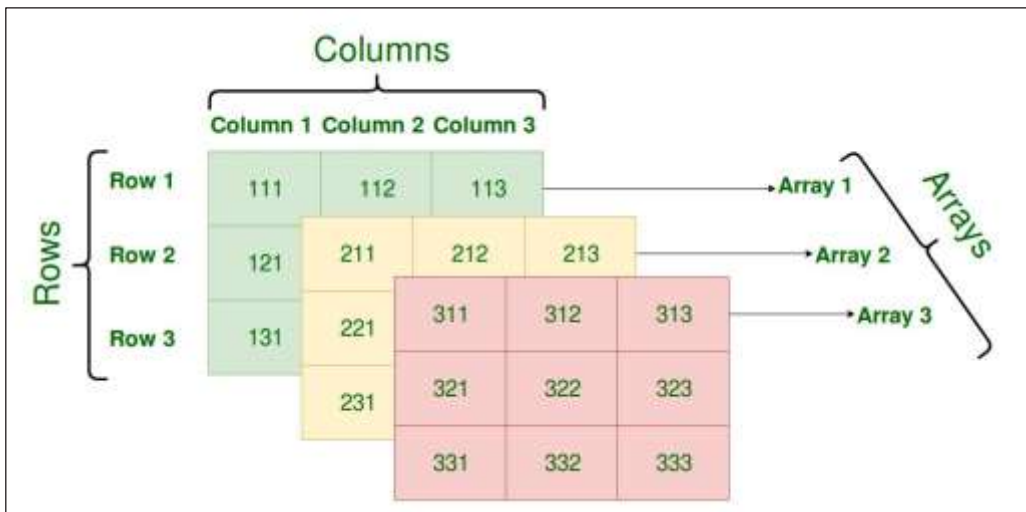
```
public static void main(String[] args)
{
    int[][][] arr = { { { 1, 2 }, { 3, 4 } }, { { 5, 6 }, { 7, 8 } } };

    System.out.println("arr[0][0][0] = " + arr[0][0][0]);
}
```

} Output:

```
arr[0][0][0] = 1
```

सारणीबद्ध प्रारूप में 3डी ऐरे का प्रतिनिधित्व: एक त्रि-आयामी ऐरे को 'x' पंक्तियों और 'y' स्तंभों के साथ ऐरेज़ की तालिका के रूप में देखा जा सकता है जहां पंक्ति संख्या 0 से (x-1) तक होती है और स्तंभ संख्या 0 से (y-1) करने के लिए होती है। 3 पंक्तियों और 3 स्तंभों वाली 3 ऐरे वाली एक त्रि-आयामी ऐरे नीचे दिखाई गई है:



सारणीबद्ध प्रारूप में 3डी ऐरे प्रिंट करें:

त्रि-आयामी ऐरे के सभी तत्वों को आउटपुट करने के लिए, लूप के लिए नेस्टेड का उपयोग करें। इसके लिए तीन लूप की आवश्यकता होती है, पहली ऐरेज़ को पार करने के लिए, दूसरी पंक्तियों को पार करने के लिए और दूसरी स्तंभों को पार करने के लिए।

उदाहरण:

```
class GFG {
    public static void main(String[] args)
    {
```

```
int[][][] arr = { { { 1, 2 }, { 3, 4 } },
                  { { 5, 6 }, { 7, 8 } } };

for (int i = 0; i < 2; i++) {

    for (int j = 0; j < 2; j++) {

        for (int k = 0; k < 2; k++) {

            System.out.print(arr[i][j][k] + " ");

        }

        System.out.println();

    }

    System.out.println();

}

}
```

आउटपुट:

1 2

3 4

5 6

7 8

जावा में विधियों का उपयोग कैसे करें

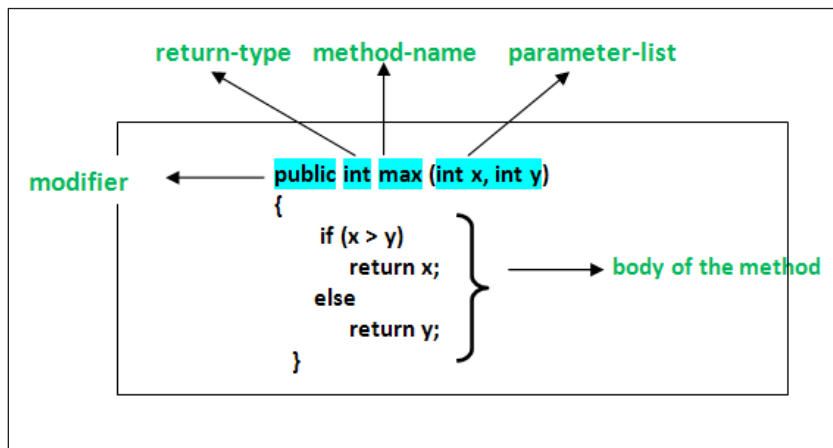
एक विधि स्टेटमेंट का एक संग्रह है जो कुछ विशिष्ट कार्य करता है और कॉलर को परिणाम लौटाता है। एक विधि कुछ भी लौटाए बिना कुछ विशिष्ट कार्य कर सकती है। तरीके हमें कोड को दोबारा टाइप किए बिना कोड का पुनः उपयोग करने की अनुमति देते हैं। जावा में, प्रत्येक विधि किसी न किसी वर्ग का हिस्सा होनी चाहिए जो C, C++ और पायथन जैसी भाषाओं से अलग है।

तरीकें समय बचाने वाले हैं और कोड को दोबारा टाइप किए बिना कोड का पुनः उपयोग करने में हमारी सहायता करते हैं।

एक विधि को कैसे परिभाषित करें

सामान्य तौर पर, विधि डीक्लियरेशनओं में छह घटक होते हैं:

- संशोधक:- विधि के एक्सेस टाइप को परिभाषित करता है यानी जहां से इसे आपके एप्लिकेशन में एक्सेस किया जा सकता है। जावा में, 4 प्रकार के एक्सेस स्पेसिफायर हैं।
 - सार्वजनिक: आपके आवेदन में सभी वर्गों में सुलभ।
 - संरक्षित: उस वर्ग के भीतर पहुंच योग्य जिसमें इसे परिभाषित किया गया है और इसके उपवर्गों में ।
 - निजी: केवल उस वर्ग के भीतर पहुँचा जा सकता है जिसमें इसे परिभाषित किया गया है।
 - डिफॉल्ट (किसी भी संशोधक का उपयोग किए बिना घोषित/परिभाषित): उसी वर्ग और पैकेज के भीतर पहुंच योग्य जिसके भीतर इसकी कक्षा परिभाषित की गई है।
- रीटर्न टाइप: विधि या शून्य द्वारा लौटाए गए मान का डेटा टाइप यदि कोई मान वापस नहीं करता है।
- विधि के नाम: फ़ील्ड नामों के नियम विधि नामों पर भी लागू होते हैं, लेकिन परंपरा थोड़ी अलग है।
- पैरामीटर सूची : इनपुट पैरामीटर की अल्पविराम से अलग की गई सूची को उनके डेटा प्रकार के साथ संलग्न कोष्ठक के भीतर परिभाषित किया गया है। यदि कोई पैरामीटर नहीं है, तो आपको खाली कोष्ठक () का उपयोग करना चाहिए।
- एक्सेप्शन सूची : आप विधि द्वारा अपेक्षित अपवादों को फेंक सकते हैं, आप इन अपवादों को निर्दिष्ट कर सकते हैं।
- मेथड बॉडी: यह ब्रेसिज़ के बीच संलग्न है। अपने इच्छित कार्यों को करने के लिए आपको जिस कोड को निष्पादित करने की आवश्यकता है।



सिग्नेचर विधि: इसमें विधि का नाम और पैरामीटर सूची (पैरामीटर की संख्या, पैरामीटर का प्रकार और पैरामीटर का क्रम) शामिल है। रिटर्न टाइप और अपवादों को इसके हिस्से के रूप में नहीं माना जाता है।

उपरोक्त फंक्शन की सिग्नेचर विधि :

```
max(int x, int y)
```

एक विधि (Method) को नाम कैसे दें?: एक विधि का नाम आमतौर पर एक एकल शब्द होता है जो लोअर केस या बहु-शब्द में एक क्रिया होना चाहिए, जो कि लोअरकेस में एक क्रिया से शुरू होता है, उसके बाद विशेषण, संज्ञा... .. पहले शब्द के बाद, पहले प्रत्येक शब्द का अक्षर बड़ा होना चाहिए। उदाहरण के लिए, फाईंडसम, कंप्यूटमैक्स, सेटएक्स और गेटएक्स ।

आमतौर पर, एक मेथड का उस वर्ग के भीतर एक अद्वितीय नाम होता है जिसमें इसे परिभाषित किया जाता है, लेकिन कभी-कभी एक विधि का नाम उसी वर्ग के भीतर अन्य विधि नामों के समान हो सकता है क्योंकि जावा में विधि ओवर-लोडिंग की अनुमति है ।

एक विधि (Method) को कैसे अपनाएं

जावा में प्रोग्रामिंग शुरू करते समय, सीखने के लिए कई नई अवधारणाएँ होती हैं। कक्षाएं, विधियां, एक्सेप्शन, निर्माता, चर, और बहुत कुछ हैं, और यह भारी हो सकता है।

चरण

```

1 package com.test.testing;
2
3 public class Test{
4
5     public static void methodExample(){
6
7     }
8
9 }
10

```

1. एक विधि (Method) c जैसी भाषाओं में एक फंक्शन के बराबर है जो कोड के पुनः उपयोग में मदद करती है। स्टेटमेंट का एक सेट एक विधि (Method) बनाता है, और इस पद्धति को अन्य स्टेटमेंट के माध्यम से लागू किया जा सकता है। जब लागू किया जाता है (कहा जाता है), तो सभी स्टेटमेंट जो विधि(Method) का एक हिस्सा हैं, निष्पादित किए जाएंगे। उदाहरण के लिए, इस विधि को देखें: "सार्वजनिक स्थैतिक शून्य विधि (Method) उदाहरण () {}"। इसमें वर्तमान में कोई कोड नहीं है, लेकिन विधि नाम से पहले तीन कीवर्ड हैं। सार्वजनिक, स्थिर और शून्य है।

```

1 package com.test.testing;
2
3 public class Test{
4
5     private String format = "";
6
7     public static void methodExample(){
8
9     }
10
11     private static void main(String[] args){
12         System.out.println("Hello world!");
13     }
14
15     protected String getformat(){
16         return this.format;
17     }
18
19     public void setformat(String format){
20         this.format = format;
21     }
22
23 }
24

```

2. मेथड के नाम से पहले पब्लिक शब्द का अर्थ है कि जब तक आप क्लास इंपोर्ट करते हैं, तब तक मेथड को किसी भी जगह से कॉल किया जा सकता है, जिसमें अन्य क्लासेस भी शामिल हैं, यहां तक कि अलग-अलग पैकेज (फाइल्स) से भी। तीन अन्य शब्द हैं जो जनता की जगह ले सकते हैं। वे संरक्षित और प्राइवेट हैं। यदि कोई विधि सुरक्षित है, तो केवल यह वर्ग और उपवर्ग (वर्ग जो इसे आधार के रूप में उपयोग करते हैं) विधि को कॉल कर सकते हैं। यदि कोई विधि निजी है, तो विधि को केवल कक्षा के अंदर ही बुलाया जा सकता है। अंतिम कीवर्ड वास्तव में एक शब्द भी नहीं है। यह तब है जब आपके पास सार्वजनिक, संरक्षित या निजी के स्थान पर कुछ भी नहीं था। इसे डिफॉल्ट, या पैकेज-प्राइवेट कहा जाता है। इसका मतलब है कि केवल एक ही पैकेज में कक्षाएं ही विधि(Method) को अपना सकते हैं।

```

1 package com.test.testing;
2
3 class ExampleClass{
4
5     static int j = 0;
6
7     static void methodExample(){
8         j++;
9         System.out.println("Value of static variable j is " + j);
10    }
11 }
12
13
14 class TestStatic{
15     public static void main(String[] args){
16         ExampleClass.methodExample();
17     }
18 }
19

```

3. दूसरा कीवर्ड, स्टैटिक का अर्थ है कि विधि (method) वर्ग से संबंधित है न कि वर्ग (ऑब्जेक्ट) के किसी भी उदाहरण से कक्षा के नाम का उपयोग करके स्टैटिक विधियों को बुलाया जाना चाहिए "ExampleClass.methodExample()"। हालाँकि, यदि कीवर्ड स्टैटिक नहीं था, तो विधि को केवल किसी ऑब्जेक्ट के माध्यम से लागू किया जा सकता है। उदाहरण के लिए, यदि वर्ग को ExampleObject कहा जाता है और इसमें एक कंस्ट्रक्टर (ऑब्जेक्ट बनाने के लिए) होता है, तो हम ExampleObject obj = new ExampleObject();, टाइप करके एक नई वस्तु बना सकते हैं और "obj.methodExample();" के साथ विधि को अपनाएं।

```

1 package com.test.testing;
2
3 class ExampleClass{
4
5     static int j = 0;
6
7     static void methodExample(){
8         j++;
9         System.out.println("Value of static variable j is " + j);
10    }
11 }
12
13
14 class TestStatic{
15     public static void main(String[] args){
16         ExampleClass.methodExample();
17     }
18 }
19

```

4. विधि(नाम से पहले अंतिम शब्द वॉइड है। वॉइड शब्द का अर्थ है कि विधि कुछ भी नहीं वापस करती है (जब आप विधि चलाते हैं तो यह कुछ भी वापस नहीं करता है)। यदि आप कुछ वापस करने के लिए एक विधि चाहते हैं, तो वॉइड शब्द को उस ऑब्जेक्ट (या प्राइमिटिव टाइप) के डेटा टाइप (प्राइमिटिव या रेफरेंस टाइप) से बदलें, जिसे आप वापस करना चाहते हैं। फिर विधि के कोड के अंत में कहीं न कहीं उस प्रकार की एक ऑब्जेक्ट के साथ वापसी जोड़ें।

```

1 package com.test.testing;
2
3 public class TestMax{
4
5     public static void main(String[] args){
6         int i = 5;
7         int j = 2;
8         int k = max(i, j);
9         System.out.println("The maximum between " + i + " and " + j + " is " + k);
10    }
11
12    public static int max(int num1, int num2){
13        int result;
14        if(num1 > num2)
15            result = num1;
16        else
17            result = num2;
18        return result;
19    }
20
21 }
22

```

5. किसी ऐसी विधि को कॉल करते समय जो कुछ लौटाती है, जो यह लौटाते हैं आप उसका उपयोग कर सकते हैं। उदाहरण के लिए, यदि कोई `someMethod()` एक पूर्णांक देता है, तो आप `int a = someMethod();` के साथ एक पूर्णांक सेट कर सकते हैं जो वह दिखाता है।

```

1 package com.test.testing;
2
3 public class TestInt{
4
5     public static void main(String[] args){
6         int a = 5;
7         int s = computeSquare(a);
8         System.out.println("Square root is " + s);
9     }
10
11    public static int computeSquare(int x){
12        int sq;
13        sq = (int) Math.sqrt(x);
14        return sq;
15    }
16
17 }
18

```

6. कुछ विधियों के लिए एक पैरामीटर की आवश्यकता होती है। एक विधि जिसके लिए एक पूर्णांक के पैरामीटर की आवश्यकता होती है, वह कुछ विधि (int a) जैसा दिखता है, इस तरह की विधि का उपयोग करते समय, आप विधि(method) का नाम लिखेंगे, और फिर कोष्ठक में एक पूर्णांक लिखेंगे: `someMethod(5)` या `someMethod(n)` if n एक पूर्णांक है।


```

1 package com.test.testing;
2
3 public class TestInt{
4
5     public static void main(String[] args){
6         int w = 5;
7         int h = 2;
8         int a = computeArea(w, h);
9         System.out.println("Area is " + a);
10    }
11
12    public static int computeArea(int width, int height){
13        int area;
14        area = width * height;
15        return area;
16    }
17
18 }
19

```

7. विधियों में कई पैरामीटर भी हो सकते हैं, जिन्हें केवल अल्पविराम द्वारा अलग किया जाता है। यदि विधि सम-मैथड के लिए दो पैरामीटर की आवश्यकता होती है, `int a` और `Object obj`, तो यह `someMethod(int a, Object obj)` "जैसा दिखेगा। इस नई विधि का उपयोग करने के लिए, इसे विधि नाम से एक पूर्णांक और ऑब्जेक्ट को कोष्ठक में बुलाया जाएगा: `someMethod(4, बात)` जहां चीज़ एक वस्तु है।

मैथड विदिन मैथड (नेस्टिंग) को कैसे अपनाएं

जावा "प्रत्यक्ष" नेस्टेड विधियों का समर्थन नहीं करता है। कई कार्यात्मक प्रोग्रामिंग भाषाएं विधि के भीतर विधि का समर्थन करती हैं। लेकिन आप जावा 7 या पुराने संस्करण में स्थानीय कक्षाओं, वर्ग को विधि के भीतर परिभाषित करके नेस्टेड विधि कार्यक्षमता प्राप्त कर सकते हैं, इसलिए यह संकलन करता है। और जावा 8 और नए संस्करण में आप इसे लैम्ब्डा अभिव्यक्ति द्वारा प्राप्त करते हैं।

विधि (Method) 1 (एनोनिमस उपवर्गों का उपयोग करना)

यह बिना नाम का एक आंतरिक वर्ग है और जिसके लिए केवल एक ही वस्तु बनाई जाती है। एक एनोनिमस आंतरिक वर्ग उपयोगी हो सकता है जब किसी वस्तु का उदाहरण कुछ "अतिरिक्त" जैसे कि किसी वर्ग या इंटरफ़ेस के ओवर-लोडिंग विधियों के साथ, वास्तव में किसी वर्ग को उपवर्ग किए बिना है।

```

// Java program implements method inside method
public class GFG {

    // create a local interface with one abstract
    // method run()
    interface myInterface {
        void run();
    }
}

```

```

// function have implements another function run()
static void Foo()
{
    // implement run method inside Foo() function
    myInterface r = new myInterface() {
        public void run()
        {
            System.out.println("geeksforgeeks");
        };
    };
    r.run();
}
public static void main(String[] args)
{
    Foo();
}
}

```

आउटपुट:

geeksforgeeks

विधि (Method) 2 (लोकल क्लासेज़ का उपयोग करना)

आप लोकल क्लास के अंदर एक विधि भी लागू कर सकते हैं। एक विधि के अंदर बनाए गए वर्ग को स्थानीय आंतरिक वर्ग कहा जाता है। यदि आप लोकल इनर क्लास के तरीकों को लागू करना चाहते हैं, तो आपको इस क्लास को विधि के अंदर तुरंत चालू करना होगा।

```

// Java program implements method inside method
public class GFG {

    // function have implementation of another
    // function inside local class
    static void Foo()
    {

```

```

// local class
class Local {
    void fun()
    {
        System.out.println("geeksforgeeks");
    }
}
new Local().fun();
}
public static void main(String[] args)
{
    Foo();
}
}

```

आउटपुट:

geeksforgeeks

विधि (Method) 3 (लैम्ब्डा एक्सप्रेसन का उपयोग करना)

लैम्ब्डा एक्सप्रेसन मूल रूप से कार्यात्मक इंटरफेस के उदाहरण व्यक्त करते हैं (एकल सार विधि के साथ एक इंटरफेस को कार्यात्मक इंटरफेस कहा जाता है। एक उदाहरण `java.lang.Runnable`)। लैम्ब्डा एक्सप्रेसन केवल अमूर्त फ़ंक्शन को लागू करते हैं और इसलिए कार्यात्मक इंटरफेस को लागू करते हैं। अभिव्यक्ति के बारे में अधिक जानकारी के लिए यहां क्लिक करें।

```

// Java program implements method inside method
public class GFG {
    interface myInterface {
        void run();
    }

    // function have implements another function
    // run() using Lambda expression
    static void Foo()
    {

```

```
// Lambda expression
myInterface r = () ->
{
    System.out.println("geeksforgeeks");
};
r.run();
}
public static void main(String[] args)
{
    Foo();
}
}
```

आउटपुट:

geeksforgeeks

विधि(Method) को ओवरलोड कैसे करें

यदि किसी वर्ग में समान नाम वाली कई विधियाँ हैं लेकिन मापदंडों में भिन्न हैं, तो इसे मेथड ओवरलोडिंग के रूप में जाना जाता है।

यदि हमें केवल एक ही ऑपरेशन करना है, तो विधियों के समान नाम रखने से प्रोग्राम की पठनीयता बढ़ जाती है।

मान लीजिए कि आपको दी गई संख्याओं का योग करना है, लेकिन कई तर्क हो सकते हैं, यदि आप दो मापदंडों के लिए `a(int,int)` और तीन मापदंडों के लिए `b(int,int,int)` जैसी विधि लिखते हैं। तो आपके और अन्य प्रोग्रामर के लिए विधि के व्यवहार को समझना मुश्किल हो सकता है क्योंकि इसका नाम अलग है।

इसलिए, हम प्रोग्राम को जल्दी से समझने के लिए मेथड ओवरलोडिंग करते हैं।



विधि (Method) ओवरलोडिंग के लाभ

मेथड ओवरलोडिंग से प्रोग्राम की पठनीयता बढ़ जाती है।

विधि (Method) को ओवरलोडिंग करने के विभिन्न तरीके

जावा में विधि को ओवरलोडिंग करने के दो तरीके हैं

- तर्कों की संख्या बदलने से
- डेटा टाइप बदलकर

जावा में, केवल विधि के रिटर्न टाइप को बदलकर मेथड ओवरलोडिंग संभव नहीं है।

1. ओवरलोडिंग मेथड: आर्ग्यूमेंट की संख्या को बदलना।

इस उदाहरण में, हमने दो विधियाँ बनाई हैं, फर्स्ट ऐड () विधि(method) दो संख्याओं का योग करती है और दूसरी ऐड विधि तीन संख्याओं का योग करती है।

इस उदाहरण में, हम स्थैतिक विधियाँ बना रहे हैं ताकि हमें कॉलिंग विधियों के लिए उदाहरण बनाने की आवश्यकता न पड़े।

```
class Adder{
    static int add(int a,int b){return a+b;}
    static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
    public static void main(String[] args){
        System.out.println(Adder.add(11,11));
        System.out.println(Adder.add(11,11,11));
    }
}
```

आउटपुट:

22

33

2. ओवरलोडिंग विधि: आर्ग्यूमेंट के डेटा टाइप को बदलना

इस उदाहरण में, हमने दो विधियाँ बनाई हैं जो डेटा टाइप में भिन्न हैं। पहली ऐड विधि(first add method) दो पूर्णांक तर्क प्राप्त करती है और दूसरी ऐड विधि(second add method) दो दोहरे तर्क प्राप्त करती है।

```
class Adder{
    static int add(int a, int b){return a+b;}
    static double add(double a, double b){return a+b;}
}
```

```

}
class TestOverloading2{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6));
}}

```

आउटपुट:

22

24.9

प्रश्न) केवल रिटर्न टाइप को बदलने से मेथड ओवरलोडिंग संभव क्यों नहीं है?

जावा में, केवल अस्पष्टता के कारण मेथड के रिटर्न टाइप को बदलकर मेथड ओवरलोडिंग संभव नहीं है। आइए देखें कि अस्पष्टता कैसे हो सकती है:

```

class Adder{
static int add(int a,int b){return
a+b;} static double add(int a,int
b){return a+b;}
}
class TestOverloading3{
public static void main(String[] args){
System.out.println(Adder.add(11,11)); //a
mbiguity
}}

```

आउटपुट:

Compile Time Error: method add(int,int) is already defined in class Adder

System.out.println (Adder.add(11,11)); // यहां, जावा कैसे निर्धारित कर सकता है कि किस योग () विधि को बुलाया जाना चाहिए?

कंपाइल टाइम एरर रन टाइम एरर से बेहतर है। तो, जावा कंपाइलर कंपाइलर टाइम एरर प्रस्तुत करता है यदि आप समान पैरामीटर वाले समान विधि की डीक्लयरेशन करते हैं।

क्या हम जावा मुख्य () विधि(Java Main() Method) ओवरलोड कर सकते हैं?

ओवरलोड विधि द्वारा कक्षा में आपके पास कई मुख्य विधियां हो सकती हैं। लेकिन जेवीएम मुख्य () विधि को कॉल करता है जो स्ट्रिंग एरे को बल तर्क के रूप में प्राप्त करता है। आइए सरल उदाहरण देखें:

```

class TestOverloading4{
public static void main(String[] args){System.out.println("main with String[]");}
}

```

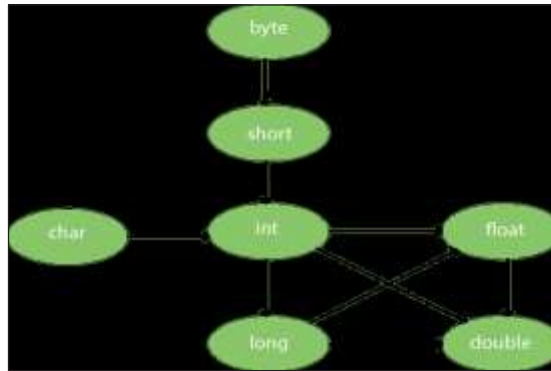
```
public static void main(String args){System.out.println("main with String");}
public static void main(){System.out.println("main without args");}
}
```

आउटपुट:

main with String[]

ओवरलोडिंग विधि (Overloading Method) और टाइप प्रमोशन (Type Promotion)

यदि कोई मेल खाने वाला डेटाटाइप नहीं मिलता है तो एक प्रकार को दूसरे प्रकार से प्रचारित किया जाता है। आइए नीचे दिए गए चित्र द्वारा अवधारणा को समझते हैं:



जैसा कि ऊपर दिए गए डायग्राम में दिखाया गया है, बाइट को शॉर्ट, इंटर, लॉन्ग, फ्लोट या डबल में प्रमोट किया जा सकता है। शॉर्ट डेटाटाइप को इंटर, लॉन्ग, फ्लोट या डबल में प्रमोट किया जा सकता है। चार डेटाटाइप को इंटर, लॉन्ग, फ्लोट या डबल वगैरह में बढ़ावा दिया जा सकता है।

टाइपप्रमोशन के साथ विधि ओवरलोडिंग(Method Overloading) का उदाहरण

```
class OverloadingCalculation1{
    void sum(int a,long b){System.out.println(a+b);}
    void sum(int a,int b,int c){System.out.println(a+b+c);}

    public static void main(String args[]){
        OverloadingCalculation1 obj=new OverloadingCalculation1();
        obj.sum(20,20);//now second int literal will be promoted to long
        obj.sum(20,20,20);
    }
}
```

}

आउटपुट: 40

60

ओवरलोडिंग विधि का टाइप प्रमोशन के साथ इफ मैचिंग(if Matching) मिलने का उदाहरण

यदि विधि में मैचिंग टाइप आर्ग्यूमेंट हैं, तो टाइप प्रमोशन प्रदर्शित नहीं किया जाता है।

```
class OverloadingCalculation2{
    void sum(int a,int b){System.out.println("int arg method invoked");}
    void sum(long a,long b){System.out.println("long arg method invoked");}

    public static void main(String args[]){
        OverloadingCalculation2 obj=new OverloadingCalculation2();
        obj.sum(20,20);//now int arg sum() method gets invoked
    }
}
```

आउटपुट:

int arg method invoked

अस्पष्टता के मामले में टाइप प्रमोशन के साथ ओवरलोडिंग मेथड का उदाहरण

यदि विधि में कोई मैचिंग टाइप के आर्ग्यूमेंट नहीं हैं, और प्रत्येक विधि समान संख्या में आर्ग्यूमेंट को बढ़ावा देती है, तो अस्पष्टता होगी।

```
class OverloadingCalculation3{
    void sum(int a,long b){System.out.println("a method invoked");}
    void sum(long a,int b){System.out.println("b method invoked");}

    public static void main(String args[]){
        OverloadingCalculation3 obj=new OverloadingCalculation3();
        obj.sum(20,20);//now ambiguity
    }
}
```

आउटपुट:

Compile Time Error

एक टाइप अव्यक्त रूप से डी-प्रमोटेड नहीं है उदाहरण के लिए डबल को किसी भी प्रकार से अव्यक्त रूप से डिप्रमोटेड नहीं किया जा सकता है।

ऑब्जेक्ट ओरिएंटेड जावा प्रोग्रामिंग

एक प्रकार की प्रोग्रामिंग जो प्रोग्रामर को डेटा संरचना के डेटा प्रकार को परिभाषित करने के साथ-साथ उस पर लागू होने वाले कार्यों को लिखने में सक्षम बनाती है, ऑब्जेक्ट ओरिएंटेड जावा प्रोग्रामिंग कहलाती है। इसमें कक्षाओं और वस्तुओं का उपयोग, बहुरूपता, वंशानुक्रम, अपवाद प्रबंधन आदि शामिल हैं। इस अध्याय में विस्तृत विषय ऑब्जेक्ट ओरिएंटेड जावा प्रोग्रामिंग के बारे में बेहतर परिप्रेक्ष्य प्राप्त करने में मदद करेंगे।

क्लासेज़ और ऑब्जेक्ट्स

क्लासेज़ और ऑब्जेक्ट्स ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग की बुनियादी अवधारणाएँ हैं जो वास्तविक जीवन संस्थाओं के इर्द-गिर्द घूमती हैं।

क्लास(Class)

एक क्लास एक उपयोगकर्ता परिभाषित ब्लूप्रिंट या प्रोटोटाइप है जिससे ऑब्जेक्ट बनाए जाते हैं। यह उन गुणों या विधियों के समूह का प्रतिनिधित्व करता है जो एक प्रकार की सभी वस्तुओं के लिए सामान्य हैं। सामान्य तौर पर, क्लास की घोषणाओं में इन घटकों को क्रम में शामिल किया जा सकता है:

- संशोधक: एक वर्ग सार्वजनिक हो सकता है या उसकी डिफ़ॉल्ट पहुंच हो सकती है।
- क्लास का नाम: नाम एक प्रारंभिक अक्षर (सम्मेलन द्वारा बड़े अक्षरों में) से शुरू होना चाहिए।
- सुपरक्लास (यदि कोई हो): कीवर्ड से पहले वर्ग के माता-पिता (सुपरक्लास) का नाम, यदि कोई हो, विस्तारित होता है। एक वर्ग केवल एक माता-पिता का विस्तार (उपवर्ग) कर सकता है।
- इंटरफेस (यदि कोई हो): वर्ग द्वारा कार्यान्वित इंटरफेस की अल्पविराम से अलग सूची, यदि कोई हो, कीवर्ड इम्प्लीमेंट से पहले। एक वर्ग एक से अधिक इंटरफेस को लागू कर सकता है।
- बाँडी: ब्रेसिज़ से घिरा क्लास बाँडी, { }।

कंस्ट्रक्टर्स का उपयोग नई वस्तुओं को प्रारंभ करने के लिए किया जाता है। फ़ील्ड वेरिएबल हैं जो कि स्थिति प्रदान करते हैं क्लास और इसके ऑब्जेक्ट्स, और विधियों का उपयोग वर्ग और उसकी वस्तुओं के व्यवहार को लागू करने के लिए किया जाता है।

ऑब्जेक्ट

यह ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग की एक बुनियादी इकाई है और वास्तविक जीवन संस्थाओं का प्रतिनिधित्व करती है। एक विशिष्ट जावा प्रोग्राम कई ऑब्जेक्ट बनाता है, जैसा कि आप जानते हैं, विधियों को लागू करके बातचीत करते हैं। एक वस्तु से मिलकर बनता है:

- अवस्था(State): यह किसी वस्तु की विशेषताओं द्वारा दर्शाया जाता है। यह किसी वस्तु के गुणों को भी दर्शाता है।
- व्यवहार(Behavior): इसे किसी वस्तु की विधियों द्वारा दर्शाया जाता है। यह किसी ऑब्जेक्ट की अन्य ऑब्जेक्ट के साथ प्रतिक्रिया को भी दर्शाता है।

- पहचान(Identity): यह एक ऑब्जेक्ट को एक अनूठा नाम देता है और एक वस्तु को अन्य वस्तुओं के साथ वातचीत करने में सक्षम बनाता है।

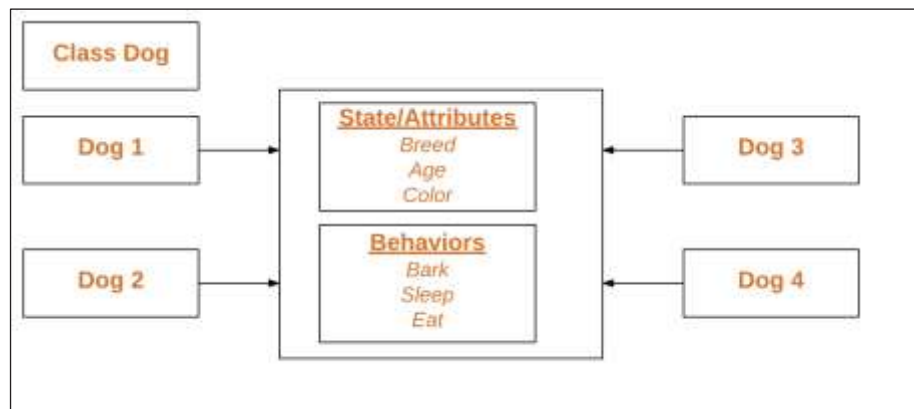
एक ऑब्जेक्ट का उदाहरण: कुत्ता(dog)।



वस्तुएं वास्तविक दुनिया में मिली चीजों के अनुरूप हैं। उदाहरण के लिए, एक ग्राफिक्स प्रोग्राम में "सर्कल", "स्क्वायर", "मेनू" जैसी वस्तुएं हो सकती हैं। एक ऑनलाइन शॉपिंग सिस्टम में "शॉपिंग कार्ट", "ग्राहक" और "उत्पाद" जैसी वस्तुएं हो सकती हैं।

ऑब्जेक्ट्स की घोषणा करना

जब किसी वर्ग का कोई ऑब्जेक्ट बनाया जाता है, तो उस वर्ग को तत्काल कहा जाता है। सभी उदाहरण वर्ग की विशेषताओं और व्यवहार को साझा करते हैं। लेकिन उन विशेषताओं के मूल्य, यानी राज्य प्रत्येक वस्तु के लिए अद्वितीय हैं। एक एकल वर्ग में कई उदाहरण हो सकते हैं।



जैसा कि हम चर घोषित करते हैं जैसे (टाइप नेम);। यह कंपाइलर को सूचित करता है कि हम नाम का उपयोग उस डेटा को संदर्भित करने के लिए करेंगे जिसका टाइप टाइप है। एक आरंभिक चर के साथ, यह घोषणा चर के लिए उचित मात्रा में स्मृति भी सुरक्षित रखती है। तो रेफरेन्स चर के लिए, टाइप दृढ़ता से एक ठोस वर्ग का नाम होना चाहिए। सामान्य तौर पर, हम एक अमूर्त वर्ग या एक इंटरफ़ेस की वस्तुएँ नहीं बना सकते हैं।

```
Dog tuffy;
```

यदि हम इस तरह से रेफरेन्स चर (टफी) घोषित करते हैं, तो इसका मान तब तक अनिर्धारित (शून्य) रहेगा जब तक कि कोई वस्तु वास्तव में बनाई और उसे असाइन नहीं की जाती। केवल एक रेफरेन्स चर घोषित करने से कोई वस्तु नहीं बनती है।

एक ऑब्जेक्ट को प्रारंभ करना

नया ऑपरेटर एक नए ऑब्जेक्ट के लिए मेमोरी आवंटित करके और उस मेमोरी के संदर्भ को वापस करके एक वर्ग को इंस्टेंट करता है। नया ऑपरेटर क्लास कंस्ट्रक्टर को भी आमंत्रित करता है।

```
// Class Declaration
public class Dog
{
    // Instance Variables
    String name;
    String breed;
    int age;
    String color;
    // Constructor Declaration of Class
    public Dog(String name, String breed,
               int age, String color)
    {
        this.name = name;
        this.breed = breed;
        this.age = age;
        this.color = color;
    }
    // method 1
    public String getName()
    {
        return name;
    }
    // method 2
    public String getBreed()
    {
        return breed;
    }
    // method 3
    public int getAge()
    {
        return age;
    }
}
```

```

    }

    // method 4
    public String getColor()
    {
        return color;
    }

    @Override
    public String toString()
    {
        return («Hi my name is «+ this.getName()+
            «.\nMy breed,age and color are " +
            this.getBreed()+", " + this.getAge()+
            «,»+ this.getColor());
    }

    public static void main(String[] args)
    {
        Dog tuffy = new Dog("tuffy","papillon", 5, "white");
        System.out.println(tuffy.toString());
    }
}

```

आउटपुट:

Hi my name is tuffy.

My breed,age and color are papillon,5,white

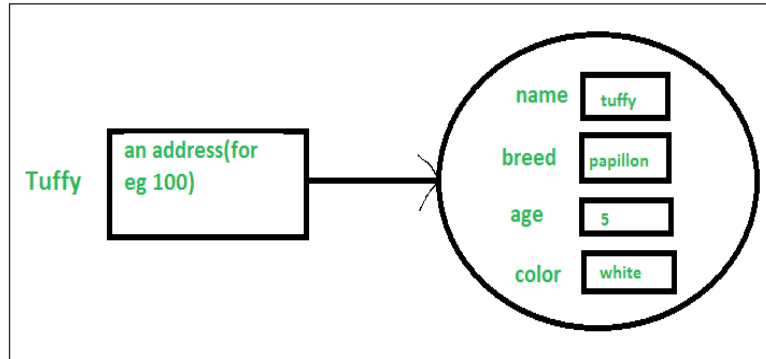
- इस वर्ग में एक ही कंस्ट्रक्टर होता है। हम एक कंस्ट्रक्टर को पहचान सकते हैं क्योंकि इसकी घोषणा वर्ग के समान नाम का उपयोग करती है और इसका कोई रिटर्न टाइप नहीं है। जावा कंपाइलर कंस्ट्रक्टर्स को संख्या और आर्ग्यूमेंट के टाइप के आधार पर अलग करता है। डॉग क्लास में कंस्ट्रक्टर चार आर्ग्यूमेंट लेता है। निम्नलिखित कथन "टफी", "पैपिलॉन", 5, "व्हाइट" को उन तर्कों के मूल्यों के रूप में प्रदान करता है:

```
Dog tuffy = new Dog("tuffy","papillon",5, "white");
```

इस कथन को क्रियान्वित करने के परिणाम को इस प्रकार दर्शाया जा सकता है:

सभी वर्गों में कम से कम एक कंस्ट्रक्टर होता है। यदि कोई वर्ग स्पष्ट रूप से किसी को घोषित नहीं करता है, तो जावा कंपाइलर स्वचालित रूप से एक बिना आर्ग्यूमेंट वाला कंस्ट्रक्टर प्रदान करता है, जिसे डिफॉल्ट कंस्ट्रक्टर भी कहा जाता है। यह डिफॉल्ट कंस्ट्रक्टर क्लास पेरेंट के नो-ऑर्गमेंट कंस्ट्रक्टर को कॉल

करता है (क्योंकि इसमें केवल एक स्टेटमेंट होता है यानी सुपर ());, या ऑब्जेक्ट क्लास कंस्ट्रक्टर यदि क्लास में कोई अन्य पैरेंट नहीं है (क्योंकि ऑब्जेक्ट क्लास प्रत्यक्ष या अप्रत्यक्ष रूप से सभी वर्गों का जनक है)।



एक क्लास के ऑब्जेक्ट को बनाने के तरीके

जावा में ऑब्जेक्ट बनाने के चार तरीके हैं। दृढ़ता से बोलना केवल एक ही तरीका है (नए कीवर्ड का उपयोग करके), और बाकी आंतरिक रूप से नए कीवर्ड का उपयोग करते हैं।

- नये कीवर्ड का उपयोग करना: जावा में ऑब्जेक्ट बनाने का यह सबसे आम और सामान्य तरीका है। उदाहरण:

```
// creating object of class Test
Test t = new Test();
```

- क्लास डॉट फॉर नेम (स्ट्रिंग क्लास नेम) विधि का उपयोग करना: `java.lang` पैकेज में एक पूर्व-परिभाषित वर्ग है जिसका नाम `Class` है। फॉरनाम (स्ट्रिंग क्लासनाम) विधि दिए गए स्ट्रिंग नाम के साथ क्लास से जुड़ी क्लास ऑब्जेक्ट लौटाती है। हमें क्लास के लिए पूरी तरह से योग्य नाम देना होगा। इस क्लास ऑब्जेक्ट पर `newInstance()` विधि को कॉल करने पर दिए गए स्ट्रिंग नाम के साथ क्लास का नया इंस्टेंस लौटाता है।

```
// creating object of public class Test
// consider class Test present in com.pl package
Test obj = (Test)Class.forName("com.pl.Test").newInstance();
```

- क्लोन () विधि का उपयोग करना: क्लोन () विधि ऑब्जेक्ट क्लास में मौजूद है। यह ऑब्जेक्ट की प्रतिलिपि बनाता और वापस करता है।

```
// creating object of class Test
Test t1 = new Test();
// creating clone of above object
Test t2 = (Test)t1.clone();
```

- डिसेरिएलाइज़ेशन: डी-सीरियलाइज़ेशन किसी फ़ाइल में सेव की गई स्थिति से किसी ऑब्जेक्ट को पढ़ने की तकनीक है।

```
FileInputStream file = new FileInputStream(filename);
```

```
ObjectInputStream in = new ObjectInputStream(file);
Object obj = in.readObject();
```

केवल वन टाइप द्वारा मल्टिपल ओब्जेक्ट्स बनाना

- वास्तविक समय में, हमें अलग-अलग तरीकों से एक वर्ग की विभिन्न वस्तुओं की आवश्यकता होती है। उन्हें संग्रहित करने के लिए कई रेफरेंस बनाना एक अच्छा अभ्यास नहीं है और इसलिए हम एक स्थिर रेफरेंस चर घोषित करते हैं और जब भी आवश्यकता होती है इसका उपयोग करते हैं। ऐसे में मेमोरी की बर्बादी कम होती है। जिन वस्तुओं को अब संदर्भित नहीं किया जाता है उन्हें जावा के कचरा कलेक्टर द्वारा नष्ट कर दिया जाएगा। उदाहरण:

```
Test test = new Test();
test = new Test();
```

- इनहेरिटेन्स प्रणाली में, हम उप-वर्ग ऑब्जेक्ट को स्टोर करने के लिए पैरेंट क्लास रेफरेंस वेरिएबल का उपयोग करते हैं। इस मामले में, हम एक ही संदर्भित चर का उपयोग करके विभिन्न उपवर्ग वस्तुओं में स्विच कर सकते हैं। उदाहरण:

```
class Animal {}
class Dog extends Animal {}
class Cat extends Animal {}
public class Test
{
    // using Dog object
    Animal obj = new Dog();
    // using Cat object
    obj = new Cat();
}
```

अस्पष्ट ऑब्जेक्ट्स

अस्पष्ट ऑब्जेक्ट्स वे ओब्जेक्ट्स होते हैं जिन्हें तत्काल किया जाता है लेकिन रेफरेंस चर में संग्रहित नहीं किया जाता है।

- उनका उपयोग तत्काल विधि कॉलिंग के लिए किया जाता है।
- विधि कॉलिंग के बाद उन्हें नष्ट कर दिया जाएगा।
- वे विभिन्न पुस्तकालयों में व्यापक रूप से उपयोग किए जाते हैं। उदाहरण के लिए, एडब्ल्यूटी पुस्तकालयों में, उनका उपयोग किसी घटना को कैप्चर करने पर कुछ क्रिया करने के लिए किया जाता है (उदाहरण के लिए एक कुंजी प्रेस)।
- नीचे दिए गए उदाहरण में, जब एक कुंजी बटन (बीटीएन द्वारा संदर्भित) दबाया जाता है, तो हम केवल कॉलिंग हैंडलर विधि के लिए इवेंट हैंडलर वर्ग की अस्पष्ट ऑब्जेक्ट बना रहे हैं।

```
btn.setOnAction(new EventHandler()
{
```

```
public void handle(ActionEvent event)
{
    System.out.println("Hello World!");
}
});
```

एक क्लास कैसे डीक्लेयर करें

जावा प्रोग्रामिंग में, एक क्लास को क्लास डीक्लेयरेशन द्वारा परिभाषित किया जाता है, जो कि कोड का एक टुकड़ा है जो इस मूल रूप का अनुसरण करता है:

```
[public] class ClassName {class-body}
```

सार्वजनिक कीवर्ड इंगित करता है कि यह वर्ग अन्य वर्गों द्वारा उपयोग के लिए उपलब्ध है। हालांकि यह वैकल्पिक है, आप आमतौर पर इसे अपनी क्लास डीक्लेयरेशन में शामिल करते हैं ताकि अन्य वर्ग उस वर्ग से ऑब्जेक्ट बना सकें जिसे आप परिभाषित कर रहे हैं।

क्लास नेम क्लास के लिए नाम प्रदान करता है। आप किसी भी पहचानकर्ता का उपयोग कर सकते हैं जिसे आप नाम देना चाहते हैं, लेकिन निम्नलिखित तीन दिशानिर्देश आपके जीवन को सरल बना सकते हैं:

- क्लास के नाम की शुरुआत बड़े अक्षर से करें। यदि कक्षा के नाम में एक से अधिक शब्द हैं, तो प्रत्येक शब्द को बड़ा करें: उदाहरण के लिए, बॉल, रिटेल कस्टमर और गेसिंगोमा।
- जब भी संभव हो, अपने वर्ग के नामों के लिए संज्ञाओं का प्रयोग करें। कक्षाएं वस्तुओं का निर्माण करती हैं, और संज्ञाएं हैं आपके द्वारा उपयोग किए जाने वाले शब्द वस्तुओं की पहचान करना। इस प्रकार, अधिकांश वर्ग के नाम संज्ञा होनी चाहिए।
- जावा एपीआई वर्ग के नाम का उपयोग करने से बचें। कोई नियम नहीं कहता है कि आपको बिल्कुल करना है, लेकिन यदि आप एक ऐसा वर्ग बनाते हैं जिसका नाम जावा एपीआई वर्ग के समान है, तो आपको अपनी क्लास को एपीआई से अलग बताने के लिए पूरी तरह से योग्य नामों (जैसे `java.util.Scanner`) का उपयोग करना होगा। एक ही नाम के साथ वर्ग।

एक वर्ग का वर्ग निकाय वह सब कुछ है जो वर्ग घोषणा के अंत में ब्रेसिज़ के भीतर जाता है, जिसमें निम्नलिखित तत्व हो सकते हैं:

- फ़िल्ड: परिवर्तनीय घोषणाएँ एक वर्ग के सार्वजनिक या निजी क्षेत्रों को परिभाषित करती हैं।
- विधियाँ: विधि घोषणाएँ एक वर्ग के तरीकों को परिभाषित करती हैं।
- कंस्ट्रक्टर्स: एक कंस्ट्रक्टर कोड का एक ब्लॉक होता है जो एक विधि के समान होता है, लेकिन एक इंस्टेंस बनाए जाने पर किसी ऑब्जेक्ट को प्रारंभ करने के लिए चलाया जाता है। एक कंस्ट्रक्टर का नाम उसी वर्ग के समान होना चाहिए, और हालांकि यह एक विधि जैसा दिखता है, लेकिन इसमें रिटर्न टाइप नहीं होता है।
- इनिशियलाइज़र: कोड के ये स्टैंड-अलोन ब्लॉक केवल एक बार चलाए जाते हैं, जब क्लास को प्रारंभ किया जाता है। दूसरा टाइप स्थिर प्रारंभकर्ता और उदाहरण प्रारंभकर्ता हैं।
- अन्य वर्ग: एक वर्ग में एक और वर्ग शामिल हो सकता है, जिसे बाद में एक आंतरिक वर्ग या एक नेस्टेड वर्ग कहा जाता है।

एक सार्वजनिक वर्ग को एक स्रोत फ़ाइल में लिखा जाना चाहिए जिसका नाम वर्ग के समान है, `extension.java` के साथ। उदाहरण के लिए, ग्रीटर नाम की एक सार्वजनिक कक्षा को `Greeter.java` नाम की फ़ाइल में रखा जाना चाहिए।

आप एक ही फ़ाइल में दो सार्वजनिक कक्षाएं नहीं रख सकते। उदाहरण के लिए, आपके पास एक स्रोत फ़ाइल नहीं हो सकती है इस तरह दिखता है:

```
public class Class1
{
    // class body for Class1 goes here
}
public class Class2
{
    // class body for Class2 goes here
}
```

कंपाइलर एक त्रुटि संदेश उत्पन्न करेगा जो दर्शाता है कि `Class2` एक सार्वजनिक वर्ग है और इसे `Class2.java` नामक फ़ाइल में घोषित किया जाना चाहिए। दूसरे शब्दों में, क्लास 1 और क्लास 2 को अलग-अलग फ़ाइलों में परिभाषित किया जाना चाहिए।

मेम्बर वेरियेबल कैसे घोषित करें

विविध प्रकार के चर हैं:

- एक वर्ग में सदस्य चर: इन्हें फ़ील्ड कहा जाता है।
- एक विधि या कोड के ब्लॉक में चर: इन्हें स्थानीय चर कहा जाता है।
- विधि घोषणाओं में चर: इन्हें पैरामीटर कहा जाता है।

साइकिल वर्ग अपने क्षेत्रों को परिभाषित करने के लिए कोड की निम्नलिखित पंक्तियों का उपयोग करता है:

```
public int cadence;
public int gear;
public int speed;
```

फील्ड डिक्लेरेशन क्रम में तीन घटकों से बना है:

- शून्य या अधिक संशोधक, जैसे `public` या `private` ।
- फ़ील्ड के प्रकार ।
- फ़ील्ड के नाम।

`Bicycle` के क्षेत्र `cadence`, `gear`, तथा `speed` नामित हैं और सभी डेटा टाइप पूर्णांक हैं (`int`) ।

`public` कीवर्ड इन क्षेत्रों को सार्वजनिक सदस्यों के रूप में पहचानता है, किसी भी वस्तु द्वारा पहुँचा जा सकता है जो कक्षा तक पहुँच सकता है।

एक्सेस मोडीफायर (Access Modifiers)

उपयोग किया गया पहला (left-most) मोडीफायर आपको यह नियंत्रित करने देता है कि अन्य वर्गों के पास सदस्य फ़ील्ड तक क्या पहुँच है। फिलहाल, केवल `public` और `private` विचार करें।

- `public` मोडीफायर: फ़ील्ड सभी वर्गों से सुलभ है।
- `private` मोडीफायर: फ़ील्ड केवल अपनी कक्षा के भीतर ही पहुँच योग्य है।

इनकैप्सुलेशन की भावना में, फ़ील्ड को निजी बनाना आम बात है। इसका मतलब है कि उन्हें केवल साइकिल वर्ग से सीधे पहुँचा जा सकता है। हालाँकि, हमें अभी भी इन मूल्यों तक पहुँच की आवश्यकता है। यह परोक्ष रूप से हमारे लिए फ़ील्ड मान प्राप्त करने वाली सार्वजनिक विधियों को जोड़कर किया जा सकता है:

```
public class Bicycle {
    private int cadence;
    private int gear;
    private int speed;
    public Bicycle(int startCadence, int startSpeed, int startGear) {
        gear = startGear;
        cadence = startCadence;
        speed = startSpeed;
    }
    public int getCadence() {
        return cadence;
    }
    public void setCadence(int newValue) {
        cadence = newValue;
    }
    public int getGear() {
        return gear;
    }
    public void setGear(int newValue) {
        gear = newValue;
    }
}
```

```

public int getSpeed() {
    return speed;
}

public void applyBrake(int decrement) {
    speed -= decrement;
}

public void speedUp(int increment) {
    speed += increment;
}
}

```

प्रकार

सभी चरों में एक प्रकार होना चाहिए। आप प्राथमिक प्रकारों का उपयोग कर सकते हैं जैसे कि `int`, `float`, `boolean`, आदि। या आप संदर्भ प्रकारों का उपयोग कर सकते हैं, जैसे कि स्ट्रिंग्स, सरणियाँ, या ऑब्जेक्ट।

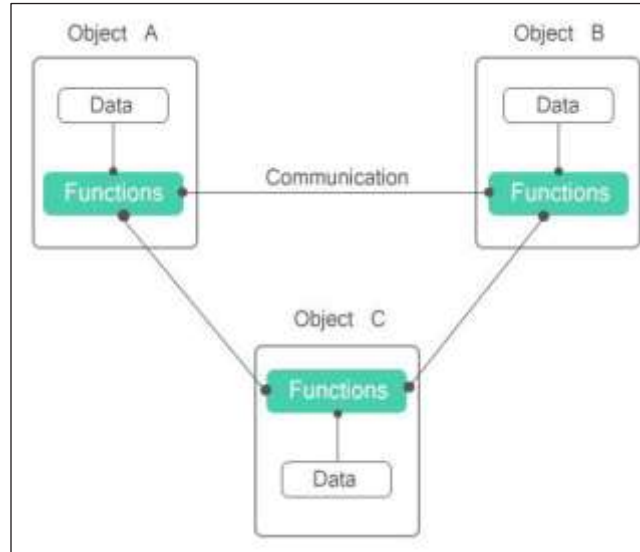
ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग पेरैडिज़म

- ऑब्जेक्ट-ओरिएंटेड दृष्टिकोण के आविष्कार में प्रमुख प्रेरक कारक प्रक्रियात्मक दृष्टिकोण में सामने आई खामियों को दूर करना है।
- ओओपी प्रोग्राम के विकास में डेटा को एक महत्वपूर्ण तत्व के रूप में मानता है और इसे सिस्टम के चारों ओर स्वतंत्र रूप से प्रवाहित करने की अनुमति नहीं देता है।
- यह डेटा को उस पर काम करने वाले कार्यों से अधिक निकटता से जोड़ता है, और इसे बाहरी कार्यों से आकस्मिक संशोधन से बचाता है।
- ओओपी एक समस्या के कई इकाइयों में अपघटन की अनुमति देता है जिसे ऑब्जेक्ट कहा जाता है और फिर इन ऑब्जेक्ट्स के आसपास डेटा और फंक्शन बनाता है
- किसी ऑब्जेक्ट के डेटा को केवल उस ऑब्जेक्ट से जुड़े फंक्शन द्वारा ही एक्सेस किया जा सकता है।
- हालाँकि, एक वस्तु के कार्य अन्य वस्तुओं के कार्यों तक पहुँच सकते हैं।

ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग की कुछ खास विशेषताएँ हैं:

- प्रक्रिया के बजाय डेटा पर जोर है।
- कार्यक्रमों को वस्तुओं के रूप में जाना जाता है में विभाजित किया जाता है।
- डेटा संरचनाओं को इस तरह डिज़ाइन किया गया है कि वे वस्तुओं की विशेषता बताते हैं।
- डेटा छिपा हुआ है और बाहरी कार्यों द्वारा पहुँचा नहीं जा सकता है।

- ओब्जेक्ट्स कार्यों के माध्यम से एक दूसरे के साथ संचार कर सकते हैं।
- नए आंकड़े और कार्यों को जब भी आवश्यक हो आसानी से जोड़ा जा सकता है।
- प्रोग्राम डिजाइन में बॉटम-अप अप्रोच का अनुसरण करता है।



बहुरूपता(Polymorphism) का प्रयोग कैसे करें

बहुरूपता(Polymorphism) एक वस्तु की कई रूपों को लेने की क्षमता है। ओओपी में बहुरूपता का सबसे आम उपयोग तब होता है जब एक अभिभावक वर्ग रिफ्रेन्स का उपयोग चाइल्ड क्लास ऑब्जेक्ट को संदर्भित करने के लिए किया जाता है।

कोई भी जावा ऑब्जेक्ट जो एक से अधिक आइएस-ए परीक्षण पास कर सकता है उसे बहुरूपी माना जाता है। जावा में, सभी जावा ऑब्जेक्ट बहुरूपी होते हैं क्योंकि कोई भी ऑब्जेक्ट अपने स्वयं के टाइप के लिए और क्लास ऑब्जेक्ट के लिए आइएस-ए परीक्षण पास करेगा।

यह जानना महत्वपूर्ण है कि किसी वस्तु तक पहुँचने का एकमात्र संभव तरीका संदर्भ चर के माध्यम से है। एक संदर्भ चर केवल एक प्रकार का हो सकता है। एक बार घोषित होने के बाद, रिफ्रेन्स वेरियेबल के टाइप को बदला नहीं जा सकता है।

रिफ्रेन्स वेरियेबल को अन्य वस्तुओं को फिर से सौंपा जा सकता है बशर्ते कि इसे अंतिम घोषित नहीं किया गया हो। एनएस के टाइप रिफ्रेन्स वेरियेबल उन विधियों को निर्धारित करेगा जिन्हें वह ऑब्जेक्ट पर लागू कर सकता है।

एक रिफ्रेन्स वेरियेबल अपने घोषित टाइप की किसी भी ऑब्जेक्ट या उसके घोषित प्रकार के किसी भी उप प्रकार को संदर्भित कर सकता है। एक संदर्भ चर को एक वर्ग या इंटरफ़ेस प्रकार के रूप में घोषित किया जा सकता है।

उदाहरण:

```
public interface Vegetarian{}
public class Animal{}
public class Deer extends Animal implements Vegetarian{}
```

अब डीयर वर्ग को बहुरूपी माना जाता है क्योंकि इसमें कई वंशानुक्रम होते हैं। उपरोक्त उदाहरणों के लिए निम्नलिखित सत्य हैं:

- एक डीयर आईएस-ए पशु,
- एक डीयर आईएस-ए शाकाहारी,
- एक डीयर आईएस-ए डीयर,
- एक डीयर आईएस-ए ऑब्जेक्ट।

जब हम रिफ्रेन्स वेरियेबल तथ्यों को डीयर ऑब्जेक्ट संदर्भ में लागू करते हैं, तो निम्नलिखित घोषणाएं कानूनी होती हैं:

उदाहरण:

```
Deer d = new Deer();
Animal a = d;
Vegetarian v = d;
Object o = d;
```

सभी रिफ्रेन्स वेरियेबल डी, ए, वी, ओ ढेर में एक ही हिरण वस्तु को संदर्भित करते हैं।

वर्चुअल विधि(Virtual Method)

उदाहरण:

```
/* File name : Employee.java */
public class Employee {
    private String name;
    private String address;
    private int number;
    public Employee(String name, String address, int number) {
        System.out.println("Constructing an Employee");
        this.name = name;
        this.address = address;
        this.number = number;
    }
    public void mailCheck() {
        System.out.println("Mailing a check to " + this.name + " " + this.address);
    }
}
```

```

public String toString() {
    return name + " " + address + " " + number;
}
public String getName() {
    return name;
}
public String getAddress() {
    return address;
}
public void setAddress(String newAddress) {
    address = newAddress;
}
public int getNumber() {
    return number;
}
}

```

अब मान लीजिए हम निम्नानुसार कर्मचारी वर्ग का विस्तार करते हैं:

```

/* File name : Salary.java */
public class Salary extends Employee {
    private double salary; // Annual salary
    public Salary(String name, String address, int number, double salary) {
        super(name, address, number);
        setSalary(salary);
    }
    public void mailCheck() {
        System.out.println("Within mailCheck of Salary class ");
        System.out.println("Mailing check to " + getName()
            + " with salary " + salary);
    }
    public double getSalary() {

```

```

        return salary;
    }

    public void setSalary(double newSalary) {
        if(newSalary >= 0.0) {
            salary = newSalary;
        }
    }

    public double computePay() {
        System.out.println("Computing salary pay for " + getName());
        return salary/52;
    }
}

```

अब, आप निम्नलिखित कार्यक्रम का ध्यानपूर्वक अध्ययन करें और इसके आउटपुट को निर्धारित करने का प्रयास करें:

```

/* File name : VirtualDemo.java */
public class VirtualDemo {
    public static void main(String [] args) {
        Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00);
        Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00);
        System.out.println("Call mailCheck using Salary reference --");
        s.mailCheck();
        System.out.println("\n Call mailCheck using Employee reference--");
        e.mailCheck();
    }
}

```

यह निम्नलिखित परिणाम देगा:

आउटपुट:

Constructing an Employee

Constructing an Employee

Call mailCheck using Salary reference --

```

Within mailCheck of Salary class
Mailing check to Mohd Mohtashim with salary 3600.0

Call mailCheck using Employee reference--
Within mailCheck of Salary class
Mailing check to John Adams with salary 2400.0

```

यहां, हम दो सेलरी ऑब्जेक्ट को तत्काल करते हैं। एक वेतन संदर्भ का उपयोग कर रहा है, और दूसरा कर्मचारी संदर्भ ई का उपयोग कर रहा है।

S.mailCheck () को लागू करते समय, कंपाइलर संकलन समय पर वेतन वर्ग में मेलचेक () देखता है, और जेवीएम रन टाइम पर वेतन वर्ग में मेलचेक () को आमंत्रित करता है।

ई(e) पर मेलचेक(mailcheck) () काफी अलग है क्योंकि ई एक कर्मचारी संदर्भ है। जब कंपाइलर e.mailCheck () देखता है, तो कंपाइलर कर्मचारी वर्ग में मेलचेक () विधि देखता है।

यहां, कंपाइलर टाइम पर, कंपाइलर ने इस कथन को मान्य करने के लिए कर्मचारी में मेलचेक () का उपयोग किया। रन टाइम पर, हालांकि, जेवीएम वेतन वर्ग में मेलचेक () को आमंत्रित करता है।

इस व्यवहार को वर्चुअल मेथड इनवोकेशन कहा जाता है, और इन मेथड्स को वर्चुअल मेथड कहा जाता है। एक ओवरराइड विधि को रन टाइम पर लागू किया जाता है, इससे कोई फर्क नहीं पड़ता कि डेटा टाइप का रिफ्लेस क्या है जो कंपाइलर टाइम पर स्रोत कोड में उपयोग किया गया था।

इनहेरिटेंस के साथ कैसे डील करें

जावा में वंशानुक्रम एक ऐसा तंत्र है जिसमें एक वस्तु एक मूल वस्तु के सभी गुणों और व्यवहारों को प्राप्त कर लेती है। यह OOPs (ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग सिस्टम) का एक महत्वपूर्ण हिस्सा है।

जावा में विरासत के पीछे का विचार यह है कि आप नए वर्ग बना सकते हैं जो मौजूदा वर्गों पर बनाए गए हैं। जब आप किसी मौजूदा वर्ग से इनहेरिट करते हैं, तो आप मूल वर्ग के तरीकों और क्षेत्रों का पुनः उपयोग कर सकते हैं। इसके अलावा, आप अपनी वर्तमान कक्षा में भी नई विधियों और क्षेत्रों को जोड़ सकते हैं।

वंशानुक्रम आईएस-ए संबंध का प्रतिनिधित्व करता है जिसे माता-पिता-बच्चे के संबंध के रूप में भी जाना जाता है।

जावा में इनहेरिटेंस का उपयोग करें

- मेथड ओवरराइडिंग के लिए (इसलिए रनटाइम बहुरूपता प्राप्त किया जा सकता है)।
- कोड पुनः प्रयोज्य के लिए।

जावा में इनहेरिटेंस

- विरासत।
- वंशानुक्रम के प्रकार।

- कक्षा के मामले में जावा में एकाधिक विरासत क्यों संभव नहीं है?

जावा में वंशानुक्रम एक ऐसा तंत्र है जिसमें एक वस्तु एक मूल वस्तु के सभी गुणों और व्यवहारों को प्राप्त कर लेती है। यह ओओपीएस (ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग सिस्टम) का एक महत्वपूर्ण हिस्सा है।

जावा में विरासत के पीछे का विचार यह है कि आप नए वर्ग बना सकते हैं जो मौजूदा वर्गों पर बनाए गए हैं। जब आप किसी मौजूदा वर्ग से इनहेरिट करते हैं, तो आप मूल वर्ग के तरीकों और क्षेत्रों का पुनः उपयोग कर सकते हैं। इसके अलावा, आप अपनी वर्तमान कक्षा में भी नई विधियों और क्षेत्रों को जोड़ सकते हैं।

वंशानुक्रम आईएस-ए संबंध का प्रतिनिधित्व करता है जिसे माता-पिता-बच्चे के संबंध के रूप में भी जाना जाता है।

जावा में इनहेरिटेंस का उपयोग क्यों करें

- मेथड ओवरराइडिंग के लिए (इसलिए रनटाइम बहुरूपता प्राप्त किया जा सकता है)।
- पुनः प्रयोज्य के लिए कोड।

इनहेरिटेंस में टर्म्स का उपयोग किया जाता है

- वर्ग: एक वर्ग ऑब्जेक्ट्स का एक समूह है जिसमें सामान्य गुण होते हैं। यह एक टेम्प्लेट या ब्लू-प्रिंट है जिससे ऑब्जेक्ट बनाए जाते हैं।
- उप वर्ग/चाइल्ड क्लास: उपवर्ग एक ऐसा वर्ग है जो दूसरे वर्ग को इनहेरिटेंस में मिला है। इसे व्युत्पन्न वर्ग, विस्तारित वर्ग, या बाल वर्ग भी कहा जाता है।
- सुपर क्लास/पैरेंट क्लास: सुपरक्लास वह वर्ग है जहां से एक उपवर्ग को विशेषताएं विरासत में मिलती हैं। इसे बेस क्लास या पैरेंट क्लास भी कहा जाता है।
- पुनःप्रयोग: जैसा कि नाम निर्दिष्ट करता है, पुनःप्रयोग एक तंत्र है जो आपको एक नया वर्ग बनाते समय मौजूदा वर्ग के क्षेत्रों और विधियों का पुनः उपयोग करने की सुविधा देता है। आप उन्हीं फ़ील्ड्स और विधियों का उपयोग कर सकते हैं जो पिछली कक्षा में पहले से ही परिभाषित हैं।

जावा इनहेरिटेंस का सिंटैक्स

```
class Subclass-name extends Superclass-name
{
    //methods and fields
}
```

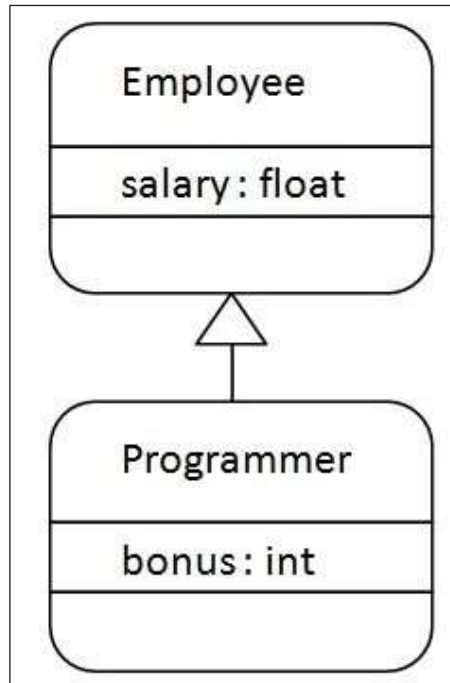
विस्तारित कीवर्ड इंगित करता है कि आप एक नया वर्ग बना रहे हैं जो मौजूदा वर्ग से निकला है। "विस्तारित" का अर्थ कार्यक्षमता को बढ़ाना है।

जावा की शब्दावली में, एक वर्ग जो विरासत में मिला है उसे पैरेंट या सुपरक्लास कहा जाता है, और नए वर्ग को चाइल्ड या सबक्लास कहा जाता है।

जावा इनहेरिटेंस का उदाहरण

जैसा कि नीचे दिए गए चित्र में दिखाया गया है, प्रोग्रामर उपवर्ग है और कर्मचारी सुपरक्लास है।

दो वर्गों के बीच संबंध प्रोग्रामर आईएस-ए कर्मचारी है। इसका मतलब है कि प्रोग्रामर एक प्रकार का कर्मचारी है।



```

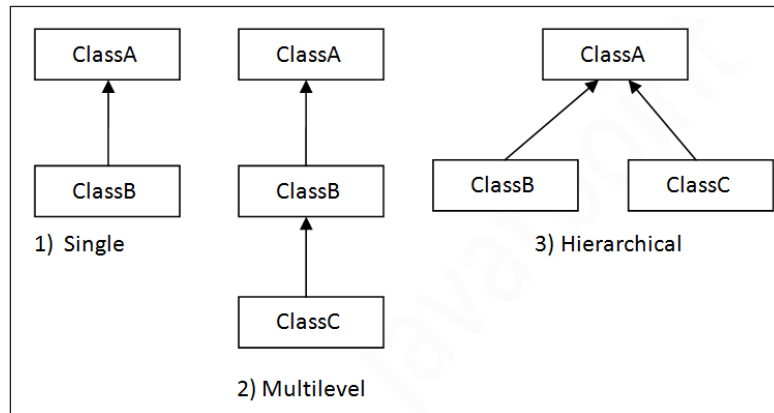
class Employee{
floatsalary=40000;
}
class Programmer extends Employee{
int bonus=10000;
public static void main(String args[]){
Programmer p=new Programmer();
System.out.println("Programmer salary is:"+p.salary);
System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
Programmer salary is:40000.0
Bonus of programmer is:10000
  
```

उपरोक्त उदाहरण में, प्रोग्रामर ऑब्जेक्ट स्वयं के वर्ग के साथ-साथ कर्मचारी वर्ग यानी कोड पुनः प्रयोज्य के क्षेत्र तक पहुंच सकता है।

जावा में इनहेरिटेंस के प्रकार

क्लास के आधार पर जावा में इनहेरिटेंस तीन प्रकार के हो सकते हैं: एकल, बहुस्तरीय और पदानुक्रमित।

जावा प्रोग्रामिंग में, मल्टीपल और हाइब्रिड इनहेरिटेंस केवल इंटरफेस के माध्यम से समर्थित है। हम इंटरफेस के बारे में बाद में सीखेंगे।



जब एक क्लास को कई क्लास इनहेरिटेंस में मिलते हैं, तो इसे एकाधिक इनहेरिटेंस के रूप में जाना जाता है। उदाहरण के लिए:

एकल इनहेरिटेंस का उदाहरण

File: TestInheritance.java

```

class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
  
```

Output:

barking...

eating...

बहुस्तरीय इनहेरिटेन्स का उदाहरण

फ़ाइल: TestInheritance2.java

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

आउटपुट:

```
weeping...
barking...
eating...
```

पदानुक्रमिक इनहेरिटेन्स के उदाहरण

File: TestInheritance3.java

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
```

```

class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}

```

आउटपुट:

```

meowing...
eating...

```

जावा में मल्टिपल इनहेरिटेन्स समर्थित क्यों नहीं है?

जटिलता को कम करने और भाषा को सरल बनाने के लिए, जावा में मल्टिपल इनहेरिटेन्स समर्थित नहीं है।

एक परिदृश्य पर विचार करें जहां ए, बी और सी तीन वर्ग हैं। सी वर्ग को ए और बी वर्ग इनहेरिटेन्स में मिलते हैं। यदि ए और बी क्लासों में एक ही विधि है और आप इसे चाइल्ड क्लास ऑब्जेक्ट से कॉल करते हैं, तो ए या बी क्लास की विधि को कॉल करने के लिए अस्पष्टता होगी।

चूंकि कंपाइल-टाइम एरर(compile-time error) रनटाइम एरर(runtime error) से बेहतर होते हैं, इसलिए यदि आप 2 क्लासेस इनहेरिट करते हैं तो जावा कंपाइल-टाइम एरर (compile-time error) प्रदान करता है। तो क्या आपके पास एक ही विधि या अलग है, कंपाइल टाइम एरर होगी।

```

class A{
void msg(){System.out.println("Hello");}
}
class B{
void msg(){System.out.println("Welcome");}
}
class C extends A,B{//suppose if it were

public static void main(String args[]){
C obj=new C();

```

```
obj.msg();//Now which msg() method would be invoked?
}
}
```

कंस्ट्रक्टर का उपयोग कैसे करें

कंस्ट्रक्टर कोड का एक ब्लॉक है जो नव निर्मित ऑब्जेक्ट को इनिशियलाइज़ करता है। एक कंस्ट्रक्टर जावा में इंस्टेंस विधि जैसा दिखता है लेकिन यह एक विधि नहीं है क्योंकि इसमें रिटर्न टाइप नहीं है।

कंस्ट्रक्टर का वर्ग के समान नाम है और जावा कोड में ऐसा दिखता है।

```
public class MyClass{
    //This is the constructor
    MyClass(){
    }
    ..
}
```

ध्यान दें कि कंस्ट्रक्टर नाम क्लास के नाम से मेल खाता है और इसका कोई रिटर्न टाइप नहीं है।

एक कंस्ट्रक्टर(Constructor) कैसे कार्य करता है

कंस्ट्रक्टर की कार्यप्रणाली को समझने के लिए, एक उदाहरण लेते हैं। मान लें कि हमारे पास कक्षा माई क्लास है। जब हम माई क्लास का ऑब्जेक्ट इस तरह बनाते हैं:

```
MyClass obj = new MyClass()
```

यहां नया कीवर्ड माई क्लास वर्ग का ऑब्जेक्ट बनाता है और कंस्ट्रक्टर को इनिशियलाइज़ करने के लिए इनवाइट करता है

आइए नीचे दिए गए कोड पर एक नज़र डालें:

जावा में एक साधारण कंस्ट्रक्टर प्रोग्राम

यहां हमने हैलो क्लास का ऑब्जेक्ट ओब्ज बनाया है और फिर हमने ऑब्जेक्ट का इंस्टेंस वेरिएबल नाम प्रदर्शित किया है। जैसा कि आप देख सकते हैं कि आउटपुट BeginnersBook.com है, जिसे हमने कंस्ट्रक्टर में इनिशियलाइज़ेशन के दौरान नाम दिया है। इससे पता चलता है कि जब हमने ऑब्जेक्ट ओब्ज बनाया तो कंस्ट्रक्टर को इनवाइट किया गया। इस उदाहरण में हमने इस कीवर्ड का उपयोग किया है, जो इस उदाहरण में वर्तमान ऑब्जेक्ट, ऑब्जेक्ट ओब्ज को संदर्भित करता है।

```
public class Hello {
    String name;
    //Constructor
```

```

Hello() {
    this.name = "BeginnersBook.com";
}

public static void main(String[] args) {
    Hello obj = new Hello();
    System.out.println(obj.name);
}
}

```

आउटपुट: BeginnersBook..com.

```

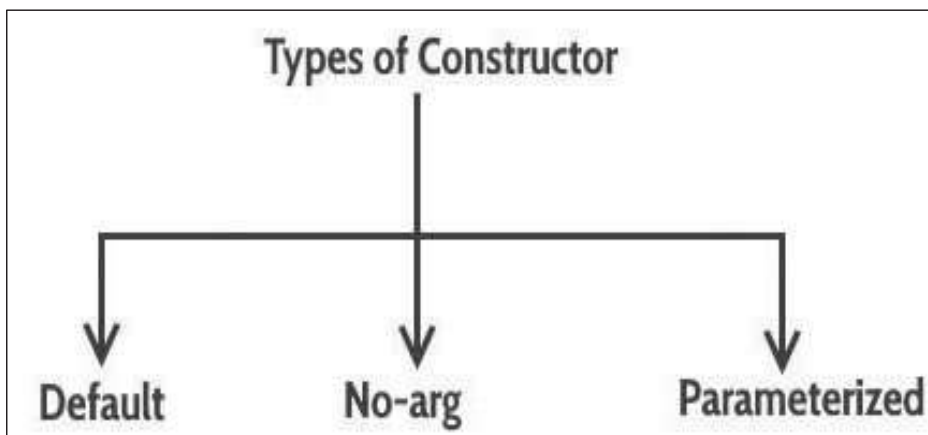
public class MyClass{
    // Constructor
    MyClass(){ ←
        System.out.println("BeginnersBook.com");
    }
    public static void main(String args[]){
        MyClass obj = new MyClass(); —
        ...
    }
}

```

... New keyword creates the object of MyClass & invokes the constructor to initialize the created object.

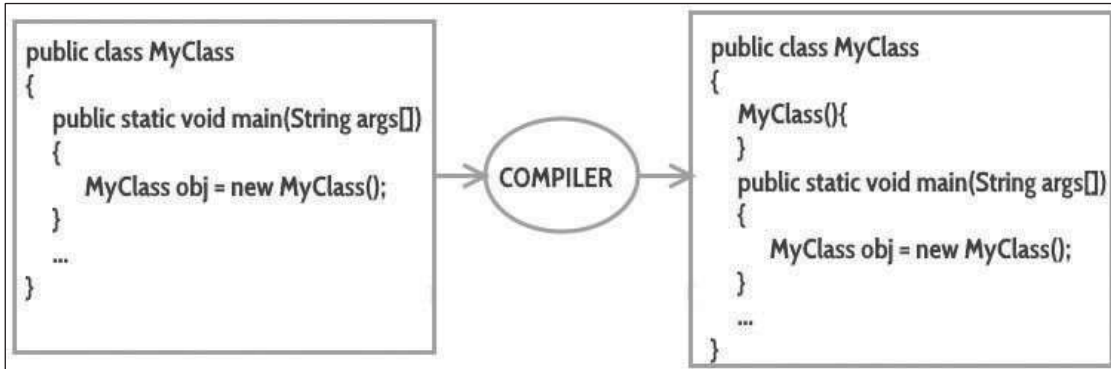
कंस्ट्रक्टर के प्रकार

कंस्ट्रक्टर तीन प्रकार के होते हैं: डिफॉल्ट, नो-आर्ग कंस्ट्रक्टर और पैरामीटरयुक्त।



डिफॉल्ट कंस्ट्रक्टर(Default Constructor)

यदि आप अपनी क्लास में किसी भी कंस्ट्रक्टर को लागू नहीं करते हैं, तो जावा कंपाइलर आपकी ओर से आपके कोड में एक डिफॉल्ट कंस्ट्रक्टर सम्मिलित करता है। इस कंस्ट्रक्टर को डिफॉल्ट कंस्ट्रक्टर के रूप में जाना जाता है। आप इसे अपने स्रोत कोड (जावा फ़ाइल) में नहीं पाएंगे क्योंकि इसे संकलन के दौरान कोड में डाला जाएगा और डॉट क्लास फ़ाइल में मौजूद है। इस प्रक्रिया को नीचे दिए गए चित्र में दिखाया गया है:



यदि आप किसी कंस्ट्रक्टर को लागू करते हैं तो आपको जावा कंपाइलर से डिफॉल्ट कंस्ट्रक्टर प्राप्त नहीं होगा।

नो-आर्ग कंस्ट्रक्टर(No-arg Constructor)

बिना तर्क वाले कंस्ट्रक्टर को नो-आर्ग कंस्ट्रक्टर के रूप में जाना जाता है। हस्ताक्षर डिफॉल्ट निर्माता के समान है; हालाँकि बाँडी में डिफॉल्ट कंस्ट्रक्टर के विपरीत कोई भी कोड हो सकता है जहाँ कंस्ट्रक्टर का शरीर खाली होता है।

हालाँकि आप देख सकते हैं कि कुछ लोग दावा करते हैं कि डिफॉल्ट और नो-आर्ग कंस्ट्रक्टर समान हैं, लेकिन वास्तव में वे नहीं हैं, भले ही आप अपनी क्लास के डेमो में पब्लिक डेमो () {} लिखते हैं, इसे डिफॉल्ट कंस्ट्रक्टर नहीं कहा जा सकता है क्योंकि आपने इसका कोड लिखा है।

उदाहरण: नो-आर्ग कंस्ट्रक्टर

```

{
    public Demo()
    {
        System.out.println("This is a no argument constructor");
    }
    public static void main(String args[]) {
        new Demo();
    }
}

```

आउटपुट:

यह कोई तर्क निर्माता नहीं है।

पैरामीटरयुक्त कंस्ट्रक्टर

तर्क के साथ कंस्ट्रक्टर (या आप पैरामीटर कह सकते हैं) को पैरामीटेराइज्ड कंस्ट्रक्टर के रूप में जाना जाता है।

उदाहरण: पैरामीटरयुक्त कंस्ट्रक्टर

1. इस उदाहरण में हमारे पास दो पैरामीटर आईडी और नाम के साथ एक पैरामीटरयुक्त कंस्ट्रक्टर है। ऑब्जेक्ट बनाते समय ओब्ज 1 और ओब्ज 2 दो तर्क पारित किए गए हैं ताकि यह कंस्ट्रक्टर ओब्ज1 और ओब्ज2 के निर्माण के बाद लागू हो जाए।

```
public class Employee {  
    int empId;  
    String empName;  
  
    //parameterized constructor with two parameters  
    Employee(int id, String name){  
        this.empId = id;  
        this.empName = name;  
    }  
    void info(){  
        System.out.println("Id: "+empId+" Name: "+empName);  
    }  
  
    public static void main(String args[]){  
        Employee obj1 = new Employee(10245,"Chaitanya");  
        Employee obj2 = new Employee(92232,"Negan");  
        obj1.info();  
        obj2.info();  
    }  
}
```

आउटपुट:

Id: 10245 Name: Chaitanya

Id: 92232 Name: Negan

2. इस उदाहरण में, हमारे पास दो कंस्ट्रक्टर हैं, एक डिफॉल्ट कंस्ट्रक्टर और एक पैरामीटराइज्ड कंस्ट्रक्टर। जब हम नए कीवर्ड का उपयोग करके ऑब्जेक्ट बनाते समय कोई पैरामीटर पास नहीं करते हैं तो डिफॉल्ट कंस्ट्रक्टर को लागू किया जाता है, हालांकि जब आप एक पैरामीटर पास करते हैं तो पैरामीटरयुक्त कंस्ट्रक्टर जो पास किए गए पैरामीटर सूची से मेल खाता है, लागू हो जाता है।

```
class Example2
{
    private int var;
    //default constructor
    public Example2()
    {
        this.var = 10;
    }
    //parameterized constructor
    public Example2(int num)
    {
        this.var = num;
    }
    public int getValue()
    {
        return var;
    }
    public static void main(String args[])
    {
        Example2 obj = new Example2();
        Example2 obj2 = new Example2(100);
        System.out.println("var is: "+obj.getValue());
        System.out.println("var is: "+obj2.getValue());
    }
}
```

आउटपुट:

```
var is: 10
var is: 100
```

यदि आप कक्षा में केवल पैरामीटरयुक्त कंस्ट्रक्टर लागू करते हैं तो क्या होगा

Class Example3

```
{
    private int var;
    public Example3(int num)
    {
        var=num;
    }
    public int getValue()
    {
        return var;
    }
    public static void main(String args[])
    {
        Example3 myobj = new Example3();
        System.out.println("value of var is: "+myobj.getValue());
    }
}
```

आउटपुट:

यह एक कंपाइलेशन एरर दिखाएगा। इसका कारण यह है कि कथन उदाहरण 3 myobj = नया उदाहरण 3 () एक डिफॉल्ट कंस्ट्रक्टर को लागू कर रहा है जो हमारे कार्यक्रम में नहीं है। जब आप अपनी कक्षा में किसी भी कंस्ट्रक्टर को लागू नहीं करते हैं, तो कंपाइलर आपके कोड में डिफॉल्ट कंस्ट्रक्टर को सम्मिलित करता है, हालाँकि जब आप किसी कंस्ट्रक्टर को लागू करते हैं, तो आपको अपने कोड में कंपाइलर द्वारा डिफॉल्ट कंस्ट्रक्टर प्राप्त नहीं होता है।

यदि हम उपरोक्त कोड से पैरामीटरयुक्त कंस्ट्रक्टर को हटा देते हैं तो प्रोग्राम ठीक चलेगा, क्योंकि तब कंपाइलर आपके कोड में डिफॉल्ट कंस्ट्रक्टर डालेगा।

कंस्ट्रक्टर चेनिंग(Constructor Chaining)

जब एक कंस्ट्रक्टर उसी क्लास के दूसरे कंस्ट्रक्टर को कॉल करता है तो इसे कंस्ट्रक्टर चेनिंग कहा जाता है।

```

public class MyClass{
    ....
    MyClass() { ←
        this("BeginnersBook.com");
    }
    MyClass(String s) { ←
        this(s, 6);
    }
    MyClass(String s, int age) { ←
        this.name =s;
        this.age = age;
    }
    public static void main(String args[]) {
        MyClass obj = new MyClass(); ←
        ....
    }
}

```

सुपर()

जब भी किसी चाइल्ड क्लास कंस्ट्रक्टर को बुलाया जाता है तो यह मूल रूप से पैरेंट क्लास के कंस्ट्रक्टर को इनवाइट करता है। आप यह भी कह सकते हैं कि कंपाइलर एक सुपर () चाइल्ड क्लास कंस्ट्रक्टर की शुरुआत में स्टेटमेंट सम्मिलित करता है।

```

class MyParentClass {
    MyParentClass() {
        System.out.println("MyParentClass Constructor");
    }
}

class MyChildClass extends MyParentClass{
    MyChildClass() {
        System.out.println("MyChildClass Constructor");
    }

    public static void main(String args[]) {
        new MyChildClass();
    }
}

```

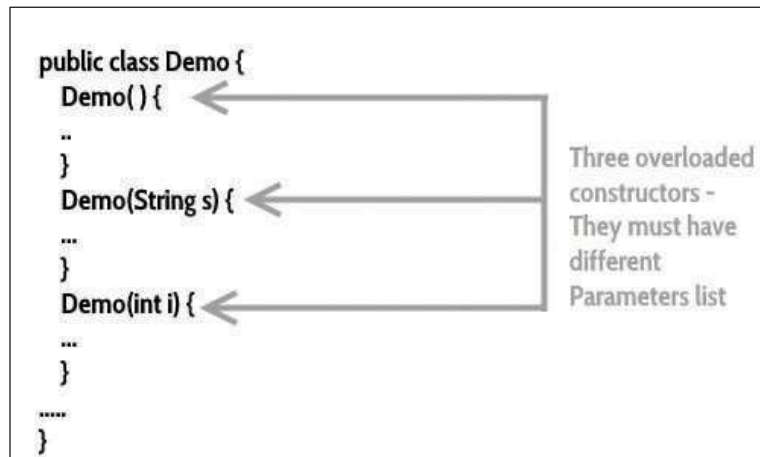
आउटपुट:

MyParentClass Constructor

MyChildClass Constructor

कंस्ट्रक्टर ओवरलोडिंग(Constructor Overloading)

कंस्ट्रक्टर ओवरलोडिंग विभिन्न पैरामीटर सूची के साथ एक से अधिक कंस्ट्रक्टर होने की एक अवधारणा है, इस तरह से कि प्रत्येक कंस्ट्रक्टर एक अलग कार्य करता है।



जावा कॉपी कंस्ट्रक्टर(Java Copy Constructor)

कॉपी कंस्ट्रक्टर का उपयोग एक वस्तु के मूल्यों को दूसरी वस्तु में कॉपी करने के लिए किया जाता है।

```

class JavaExample{
    String web;

    JavaExample(String w) {
        web = w;
    }

    /* This is the Copy Constructor, it
    * copies the values of one object
    * to the another object (the object
    * that invokes this constructor)
    */

    JavaExample(JavaExample je) {
        web = je.web;
    }

    void disp(){
        System.out.println("Website: "+web);
    }
}

```

```

public static void main(String args[]){
    JavaExample obj1 = new JavaExample("BeginnersBook");

    /* Passing the object as an argument to the constructor
    * This will invoke the copy constructor
    */

    JavaExample obj2 = new JavaExample(obj1);

    obj1.disp();
    obj2.disp();
}
}

```

आउटपुट:

Website: BeginnersBook

Website: BeginnersBook

बेस क्लास कंस्ट्रक्टर को अपनाने के लिए सुपर कीवर्ड का उपयोग कैसे करें

एक व्युत्पन्न जावा वर्ग सुपर कीवर्ड का उपयोग करके अपने बेस क्लास में एक कंस्ट्रक्टर को कॉल कर सकता है। वास्तव में, व्युत्पन्न वर्ग में एक कंस्ट्रक्टर को सुपर के कंस्ट्रक्टर को कॉल करना चाहिए जब तक कि दोनों वर्गों के लिए डिफॉल्ट कंस्ट्रक्टर न हों।

यह समझने के लिए कि बेस क्लास कंस्ट्रक्टर को कॉल करने के लिए सुपर कीवर्ड का उपयोग कैसे करें, इन 10 चरणों का पालन करें।

1. अपना टेक्स्ट एडिटर खोलें और निम्नलिखित जावा स्टेटमेंट टाइप करें:

```

1 // Java bean for Person
2 public class Person {
3     // Private variables
4     private String firstName;
5     private String lastName;
6     // Constructor
7     public Person(String firstName, String lastName) {
8         setFirstName(firstName);
9         setLastName(lastName);
10    }
11    // Getter and setter for variable
12    public String getFirstName() {
13        return firstName;
14    }

```

```

15 public void setFirstName(String firstName) {
16     this.firstName=firstName;
17 }
18 public String getLastName() {
19     return lastName;
20 }
21 public void setLastName (String lastName) {
22     this.lastName=lastName;
23 }
24 }

```

कार्यक्रम दो गुणों को परिभाषित करता है, फ़र्स्ट नेम और लास्ट नेम। ध्यान दें कि प्रोग्राम एक संपत्ति को परिभाषित करने के लिए जावाबीन विनिर्देश का पालन करता है, यानी गेट्टर और सेटर विधियों के साथ एक निजी चर। यह भी ध्यान दें कि कंस्ट्रक्टर को दो मापदंडों की आवश्यकता होती है, फ़र्स्ट नेम और लास्ट नेम।

2. अपनी फ़ाइल को पर्सन डॉट जावा के रूप में सहेजें।
3. एक कमांड प्रॉम्प्ट खोलें और अपने जावा प्रोग्राम वाली निर्देशिका में नेविगेट करें। फिर टाइप स्रोत को संकलित करने के लिए कमांड में और एंटर दबाएं।

```

Command Prompt
c:\JavaStuff\how-tos\use-super-keyword-to-call-base-constructor>javac Person.java
c:\JavaStuff\how-tos\use-super-keyword-to-call-base-constructor>

```

4. अपने टेक्स्ट एडिटर में एक नई फाइल बनाएं। निम्नलिखित जावा स्टेटमेंट में टाइप करें:

```

1 // Java bean for Employee
2 public class Employee extends Person {
3     // Private variable
4     private int empId;
5     // Constructor
6     public Employee(String firstName, String lastName, int empId) {
7         super(firstName, lastName);
8         setEmpId(empId);
9     }
10    // Getter and setter for variable
11    public int getEmpId() {
12        return empId;
13    }
14    public void setEmpId (int empId) {
15        this.empId=empId;
16    }
17 }

```

कर्मचारी वर्ग एक संपत्ति, एम्प आईडी को परिभाषित करता है। इस संपत्ति में कर्मचारी आईडी होगी। ध्यान दें कि इस क्लास में कंस्ट्रक्टर सुपर क्लास (पर्सन) में कंस्ट्रक्टर को कॉल करता है। फ़र्स्ट नेम और लास्ट नेम पैरामीटर कंस्ट्रक्टर को पास किया जाता है।

5. अपनी फ़ाइल को एम्प्लोयी डॉट जावा के रूप में सहेजें।
6. कमांड प्रॉम्प्ट पर लौटें। फिर स्रोत को संकलित करने के लिए कमांड टाइप करें और एंटर दबाएं।

```

c:\JavaStuff\how-tos\use-super-keyword-to-call-base-constructor>javac Employee.java
c:\JavaStuff\how-tos\use-super-keyword-to-call-base-constructor>

```

7. अब आप अपनी व्युत्पन्न क्लास का परीक्षण करने के लिए एक आवेदन तैयार करेंगे। अपने टेक्स्ट एडिटर में एक नई फाइल बनाएं और निम्नलिखित जावा स्टेटमेंट टाइप करें:

```

1 public class UsesuperKeywordToCallBaseConstructor {
2     public static void main (String[] args) {
3         Employee employee=new Employee("Stephen", "Withrow", 1234);
4         System.out.println(employee.getFirstName() + " " + employee.getLastName());
5         System.out.println(employee.getEmpId());
6     }
7 }

```

प्रोग्राम एक कर्मचारी ऑब्जेक्ट को इंस्टेंट करता है और फिर गेट फ़र्स्ट नेम और गेट लास्ट नेम विधियों को कॉल करता है। फ़र्स्ट नेम और लास्ट नेम पैरामीटर व्युत्पन्न क्लास कंस्ट्रक्टर द्वारा बेस कंस्ट्रक्टर को पास कर दिया गया है। प्रोग्राम कर्मचारी ऑब्जेक्ट में गेटएम्प आईडी विधि को भी कॉल करता है।

8. अपनी फ़ाइल को "UsesuperKeywordToCallBaseConstructor.java" के रूप में सहेजें।
9. कमांड प्रॉम्प्ट पर लौटें। फिर स्रोत को संकलित करने के लिए कमांड टाइप करें और एंटर दबाएं।

```

c:\JavaStuff\how-tos\use-super-keyword-to-call-base-constructor>javac UsesuperKey
ywordToCallBaseConstructor.java
c:\JavaStuff\how-tos\use-super-keyword-to-call-base-constructor>

```

10. अब आप अपने कार्यक्रम का परीक्षण करेंगे। जावा रनटाइम लॉन्चर चलाने के लिए कमांड टाइप करें और फिर एंटर दबाएं।

```

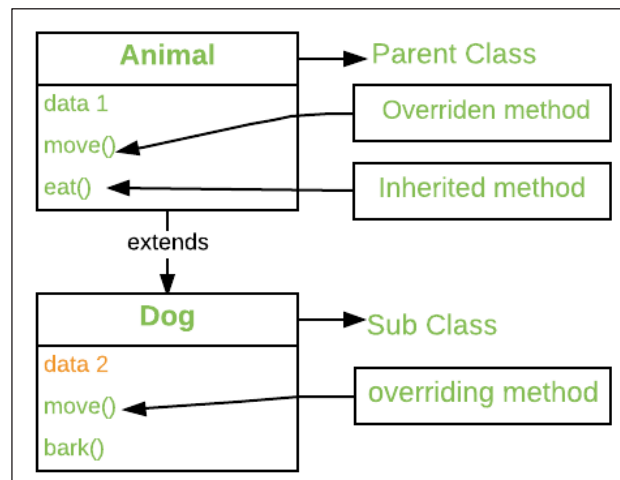
c:\JavaStuff\how-tos\use-super-keyword-to-call-base-constructor>java UsesuperKey
ywordToCallBaseConstructor
Stephen Withrow
1234
c:\JavaStuff\how-tos\use-super-keyword-to-call-base-constructor>

```

ध्यान दें कि प्रोग्राम का आउटपुट पुष्टि करता है कि फ़र्स्ट नेम और लास्ट नेम कर्मचारी कंस्ट्रक्टर द्वारा बेस क्लास कंस्ट्रक्टर को सफलतापूर्वक पास किया गया था ।

ओवरराइड कैसे करें

किसी भी ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग भाषा में, ओवरराइडिंग एक ऐसी सुविधा है जो एक उपवर्ग या चाइल्ड क्लास को एक ऐसी विधि का विशिष्ट कार्यान्वयन प्रदान करने की अनुमति देती है जो पहले से ही इसके सुपर-क्लास या पैरेंट क्लास में से एक द्वारा प्रदान की जाती है। जब एक उपवर्ग में एक विधि का एक ही नाम, समान पैरामीटर या हस्ताक्षर और एक ही रिटर्न टाइप (या उप-प्रकार) होता है, तो उपवर्ग में विधि को सुपर-क्लास में विधि को ओवरराइड करने के लिए कहा जाता है।



मैथड ओवरराइडिंग(Method overriding) उन तरीकों में से एक है जिसके द्वारा जावा रन टाइम पॉलीमॉर्फिज्म(Run Time Polymorphism) प्राप्त करता है। निष्पादित विधि का संस्करण उस ऑब्जेक्ट द्वारा निर्धारित किया जाएगा जिसका उपयोग इसे लागू करने के लिए किया जाता है। यदि विधि को लागू करने के लिए मूल वर्ग की किसी वस्तु का उपयोग किया जाता है, तो मूल वर्ग के संस्करण को निष्पादित किया जाएगा, लेकिन यदि उपवर्ग की किसी वस्तु का उपयोग विधि को लागू करने के लिए किया जाता है, तो चाइल्ड क्लास में संस्करण निष्पादित किया जाएगा। दूसरे शब्दों में, यह उस वस्तु का प्रकार है जिसे संदर्भित किया जा रहा है (संदर्भ चर का प्रकार नहीं) जो निर्धारित करता है कि ओवरराइड विधि का कौन सा संस्करण निष्पादित किया जाएगा।

```

// A Simple Java program to demonstrate
// method overriding in java
// Base Class
class Parent {
    void show()
    {
        System.out.println("Parent's show()");
    }
}
// Inherited class
class Child extends Parent {
    // This method overrides show() of Parent
  
```



```

@Override
void show()
{
    System.out.println("Child's show()");
}
}
// Driver class
class Main {
    public static void main(String[] args)
    {
        // If a Parent type reference refers
        // to a Parent object, then Parent's
        // show is called
        Parent obj1 = new Parent();
        obj1.show();
        // If a Parent type reference refers
        // to a Child object Child's show()
        // is called. This is called RUN TIME
        // POLYMORPHISM.
        Parent obj2 = new Child();
        obj2.show();
    }
}

```

आउटपुट:

Parent's show()

Child's show()

मैथड ओवरराइडिंग(Method Overriding) के लिए नियम

1. ओवरराइडिंग और एक्सेस-मोडीफायर: ओवरराइडिंग मैथड के लिए एक्सेस मोडीफायर ओवरराइड विधि की तुलना में अधिक, लेकिन कम नहीं, एक्सेस की अनुमति दे सकता है। उदाहरण के लिए, सुपर-क्लास में एक संरक्षित इंस्टेंस विधि को सार्वजनिक किया जा सकता है, लेकिन उपवर्ग में निजी नहीं। ऐसा करने से, कंपाइल-टाइम एरर उत्पन्न होगी।

```
// A Simple Java program to demonstrate
// Overriding and Access-Modifiers
class Parent {
    // private methods are not overridden
    private void m1()
    {
        System.out.println("From parent m1()");
    }
    protected void m2()
    {
        System.out.println("From parent m2()");
    }
}
class Child extends Parent {
    // new m1() method
    // unique to Child class
    private void m1()
    {
        System.out.println("From child m1()");
    }
    // overriding method
    // with more accessibility
    @Override
    public void m2()
    {
        System.out.println("From child m2()");
    }
}
// Driver class
class Main {
    public static void main(String[] args)
```

```

{
    Parent obj1 = new Parent();
    obj1.m2();
    Parent obj2 = new Child();
    obj2.m2();
}

```

}
आउटपुट:

From parent m2()

From child m2()

2. अंतिम विधियों को ओवरराइड नहीं किया जा सकता: यदि हम किसी विधि को ओवरराइड नहीं करना चाहते हैं, तो हम इसे अंतिम के रूप में घोषित करते हैं।

```
// final methods cannot be overridden
```

```

class Parent {
    // Can't be overridden
    final void show() {}
}
class Child extends Parent {
    // This would produce error
    void show() {}
}

```

आउटपुट:

13: error: show() in Child cannot override show() in Parent

```
void show() { }
```

^

overridden method is final

3. स्टैटिक विधियों को ओवरराइड नहीं किया जा सकता है (ओवरराइडिंग मेथड बनाम हाइडिंग मेथड): जब आप बेस क्लास में स्टैटिक मेथड के समान सिग्नेचर वाली स्टैटिक मेथड को परिभाषित करते हैं तो इसे हाइडिंग मेथड के रूप में जाना जाता है।

निम्न तालिका सारांशित करती है कि जब आप एक सुपर-क्लास में एक विधि के रूप में एक ही सिग्नेचर के साथ एक विधि को परिभाषित करते हैं तो क्या होता है।

	सुपरक्लास इन्स्टेन्स विधि	सुपरक्लास स्टेटिक विधि
उपवर्ग इन्स्टेन्स विधि	ओवरराइड	एक कंपाइल-टाइम एरर उत्पन्न करता है
उपवर्ग स्टेटिक विधि	एक कंपाइल-टाइम एरर उत्पन्न करता है	हाइड्स

```
// Java program to show that
// if the static method is redefined by
// a derived class, then it is not
// overriding, it is hiding
class Parent {
    // Static method in base class
    // which will be hidden in subclass
    static void m1()
    {
        System.out.println("From parent "
            + «static m1()");
    }
    // Non-static method which will
    // be overridden in derived class
    void m2()
    {
        System.out.println("From parent "
            + «non-static(instance) m2()");
    }
}
class Child extends Parent {
    // This method hides m1() in Parent
    static void m1()
    {
        System.out.println("From child static m1()");
    }
    // This method overrides m2() in Parent
```

```

@Override

public void m2()
{
    System.out.println("From child "
        + «non-static(instance) m2()");
}
}
// Driver class
class Main {
    public static void main(String[] args)
    {
        Parent obj1 = new Child();
        // As per overriding rules this
        // should call to class Child static
        // overridden method. Since static
        // method can not be overridden, it
        // calls Parent's m1()
        obj1.m1();
        // Here overriding works
        // and Child's m2() is called
        obj1.m2();
    }
}

```

आउटपुट:

```

From parent static m1()
From child non-static(instance) m2()

```

4. निजी विधियों को ओवरराइड नहीं किया जा सकता: निजी विधियों को ओवरराइड नहीं किया जा सकता क्योंकि वे कंपाइल टाइम के दौरान बॉडेड होते हैं। इसलिए हम उपवर्ग में निजी विधियों को ओवरराइड भी नहीं कर सकते हैं।
5. ओवरराइडिंग विधि में समान रिटर्न टाइप (या उपप्रकार) होना चाहिए: जावा 5.0 से आगे चाइल्ड क्लास में ओवरराइडिंग विधि के लिए अलग रिटर्न टाइप होना संभव है, लेकिन चाइल्ड का रिटर्न टाइप पैरेंट के रिटर्न टाइप का उप-प्रकार होना चाहिए। इस घटना को सहसंयोजक वापसी प्रकार के रूप में जाना जाता है।

6. उप-वर्ग से ओवरराइड विधि का लागू होना : हम सुपर कीवर्ड का उपयोग करके ओवरराइडिंग विधि में मूल वर्ग विधि को कॉल कर सकते हैं।

```
// A Java program to demonstrate that overridden
// method can be called from sub-class
// Base Class
class Parent {
    void show()
    {
        System.out.println("Parent's show()");
    }
}
// Inherited class
class Child extends Parent {
    // This method overrides show() of Parent
    @Override
    void show()
    {
        super.show();
        System.out.println("Child's show()");
    }
}
// Driver class
class Main {
    public static void main(String[] args)
    {
        Parent obj = new Child();
        obj.show();
    }
}
```

आउटपुट:

Parent's show()

Child's show()

7. ओवरराइडिंग और कंस्ट्रक्टर: हम कंस्ट्रक्टर को ओवरराइड नहीं कर सकते क्योंकि पैरेंट और चाइल्ड क्लास कभी भी एक ही नाम का कंस्ट्रक्टर नहीं हो सकता है (कंस्ट्रक्टर का नाम हमेशा क्लास के नाम के समान होना चाहिए)।
8. ओवरराइडिंग और एक्सेप्शन-हैंडलिंग: अपवाद-हैंडलिंग से संबंधित तरीकों को ओवरराइड करते समय ध्यान देने योग्य दो नियम नीचे दिए गए हैं।
 - नियम 1: यदि सुपर-क्लास ओवरराइड विधि अपवाद नहीं देती है, तो उप-वर्ग ओवरराइड विधि केवल अनियंत्रित अपवाद दे सकती है, चेक किए गए अपवाद को देने से कंपाइल-टाइम एरर होगी।

```
/* Java program to demonstrate overriding when
superclass method does not declare an exception
*/
```

```
class Parent {
    void m1()
    {
        System.out.println("From parent m1()");
    }
    void m2()
    {
        System.out.println("From parent m2()");
    }
}
class Child extends Parent {
    @Override
    // no issue while throwing unchecked exception
    void m1() throws ArithmeticException
    {
        System.out.println("From child m1()");
    }
    @Override
    // compile-time error
    // issue while throwin checked exception
    void m2() throws Exception
```

```

    {
        System.out.println("From child m2");
    }
}

```

आउटपुट:

error: m2() in Child cannot override m2() in Parent

```

    void m2() throws Exception{ System.out.println("From child m2");}
    ^

```

overridden method does not throw Exception

- नियम 2: यदि सुपर-क्लास ओवरराइड विधि एक्सेप्शन दिखाता है, तो उप-वर्ग ओवरराइडिंग विधि केवल वही, उप-वर्ग अपवाद दिखा सकती है। अपवाद पदानुक्रम में पैरेंट अपवाद को देने से कंपाइल टाइम एरर हो जाएगी। इसके अलावा, कोई समस्या नहीं है अगर उपवर्ग ओवरराइड विधि कोई अपवाद नहीं दे रही है।

```

// Java program to demonstrate overriding when
// superclass method does declare an exception
class Parent {
    void m1() throws RuntimeException
    {
        System.out.println("From parent m1()");
    }
}
class Child1 extends Parent {
    @Override
    // no issue while throwing same exception
    void m1() throws RuntimeException
    {
        System.out.println("From child1 m1()");
    }
}
class Child2 extends Parent {
    @Override
    // no issue while throwing subclass exception

```



```

void m1() throws ArithmeticException
{
    System.out.println("From child2 m1()");
}
}
class Child3 extends Parent {
    @Override
    // no issue while not throwing any exception
    void m1()
    {
        System.out.println("From child3 m1()");
    }
}
class Child4 extends Parent {
    @Override
    // compile-time error
    // issue while throwing parent exception
    void m1() throws Exception
    {
        System.out.println("From child4 m1()");
    }
}

```

आउटपुट:

```
error: m1() in Child4 cannot override m1() in Parent
```

```
void m1() throws Exception
```

```
^
```

```
overridden method does not throw Exception
```

9. ओवरराइडिंग और एब्स्ट्रैक्ट मेथड: एक इंटरफेस या एब्स्ट्रैक्ट क्लास में एब्स्ट्रैक्ट मेथड्स को व्युत्पन्न कंक्रिट क्लासेस में ओवरराइड करना होता है अन्यथा एक कंपाइल-टाइम एरर दे दिया जाएगा।

10. ओवरराइडिंग और सिंक्रोनाइज़्ड/स्ट्रिक्ट एफपी मेथड: मेथड के साथ सिंक्रोनाइज़्ड/स्ट्रिक्ट एफपी मॉडिफायर की मौजूदगी का ओवरराइडिंग के नियमों पर कोई प्रभाव नहीं पड़ता है, यानी यह संभव है कि एक सिंक्रोनाइज़्ड/स्ट्रिक्ट एफपी मेथड एक नॉन-सिंक्रोनाइज़्ड/स्ट्रिक्ट एफपी एक को ओवरराइड कर सकती है और इसके विपरीत।

- C++ में, हमें ओवरराइडिंग या रन टाइम पॉलीमॉर्फिज्म प्राप्त करने के लिए वर्चुअल कीवर्ड की आवश्यकता होती है। जावा में, डिफॉल्ट विधियां रूप से वर्चुअल होती हैं।
- हमारे पास बहुस्तरीय विधि-ओवरराइडिंग हो सकती है।

```
// A Java program to demonstrate
// multi-level overriding
// Base Class
class Parent {
    void show()
    {
        System.out.println("Parent's show()");
    }
}
// Inherited class
class Child extends Parent {
    // This method overrides show() of Parent
    void show() { System.out.println("Child's show()"); }
}
// Inherited class
class GrandChild extends Child {
    // This method overrides show() of Parent
    void show()
    {
        System.out.println("GrandChild's show()");
    }
}
// Driver class
class Main {
    public static void main(String[] args)
    {
        Parent obj1 = new GrandChild();
        obj1.show();
    }
}
```

```

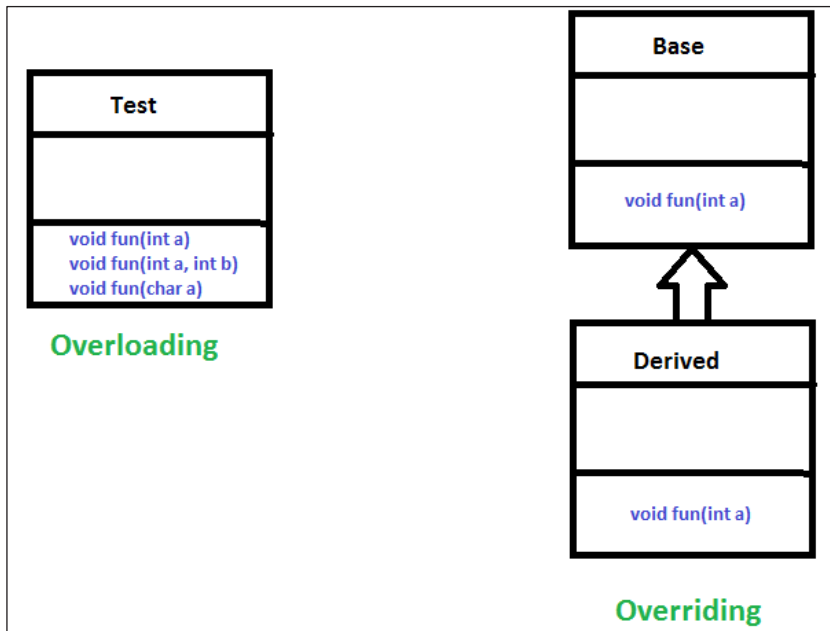
    }
}

```

आउटपुट:

GrandChild's show()

- ओवरराइडिंग बनाम ओवरलोडिंग:
 - ओवरलोडिंग एक ही विधि के बारे में अलग-अलग सिग्नेचर हैं। ओवरराइडिंग एक ही विधि, एक ही सिग्नेचर लेकिन इनहेरिटेंस के माध्यम से जुड़े विभिन्न वर्गों के बारे में है।



- ओवरलोडिंग कम्पाइलर –टाइम बहुरूपता का एक उदाहरण है और ओवरराइडिंग रन टाइम बहुरूपता का एक उदाहरण है।

ओवरराइडिंग विधि क्यों?

ओवरराइड विधियाँ जावा को रन-टाइम बहुरूपता का समर्थन करने की अनुमति देती हैं। एक कारण से ऑब्जेक्ट-ओरिएंटेड प्रोग्रामिंग के लिए बहुरूपता आवश्यक है: यह एक सामान्य वर्ग को उन तरीकों को निर्दिष्ट करने की अनुमति देता है जो उपवर्गों को कुछ या सभी विधियों के विशिष्ट कार्यान्वयन को परिभाषित करने की अनुमति देते हुए इसके सभी डेरिवेटिव के लिए सामान्य होंगे। ओवरराइड विधियाँ एक और तरीका है जिससे जावा बहुरूपता के "एक इंटरफ़ेस, एकाधिक विधियाँ" पहलू को लागू करता है।

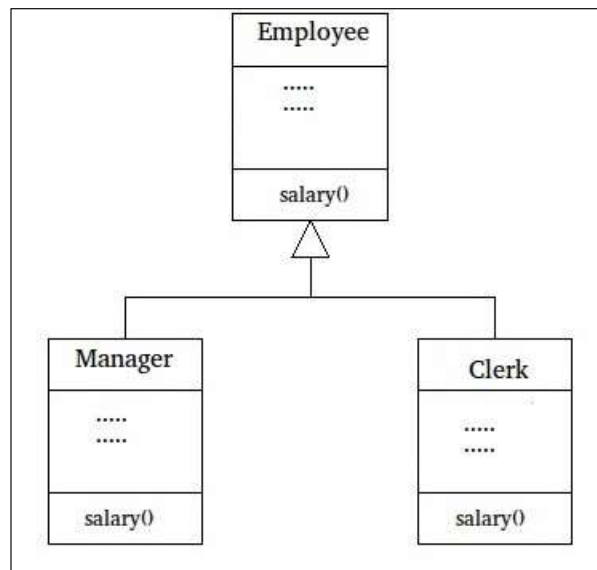
डायनामिक मेथड डिस्पैच सबसे शक्तिशाली तंत्रों में से एक है जो ऑब्जेक्ट-ओरिएंटेड डिज़ाइन कोड के पुनः उपयोग और मजबूती को सहन करता है। एक स्वच्छ सार इंटरफ़ेस को बनाए रखते हुए पुनः संकलित किए बिना नए वर्गों के उदाहरणों पर कॉल करने के लिए कोड लाइब्रेरी मौजूद होने की क्षमता एक बहुत शक्तिशाली उपकरण है।

ओवरराइड विधियाँ हमें व्युत्पन्न वर्ग वस्तु के प्रकार को जाने बिना किसी भी व्युत्पन्न वर्ग के तरीकों को कॉल करने की अनुमति देती हैं।

ओवरराइडिंग विधि कब प्रयोग करें?

ओवरराइडिंग और इनहेरिटेंस : बहुरूपता को सफलतापूर्वक लागू करने की कुंजी का एक हिस्सा यह समझना है कि सुपरक्लास और उपवर्ग एक पदानुक्रम बनाते हैं जो कम से अधिक विशेषज्ञता की ओर बढ़ता है। सही ढंग से उपयोग किया जाता है, सुपरक्लास उन सभी तत्वों को प्रदान करता है जो एक उपवर्ग सीधे उपयोग कर सकता है। यह उन विधियों को भी परिभाषित करता है जिन्हें व्युत्पन्न वर्ग को स्वयं लागू करना चाहिए। यह उपवर्ग को अपनी विधियों को परिभाषित करने के लिए लचीलेपन की अनुमति देता है, फिर भी एक सुसंगत इंटरफ़ेस को लागू करता है। इस प्रकार, इनहेरिटेंस को ओवरराइड विधियों के साथ जोड़कर, एक सुपरक्लास उन विधियों के सामान्य रूप को परिभाषित कर सकता है जिनका उपयोग इसके सभी उपवर्गों द्वारा किया जाएगा।

आइए एक अधिक व्यावहारिक उदाहरण देखें जो ओवरराइडिंग विधि का उपयोग करता है। एक संगठन के लिए एक कर्मचारी प्रबंधन सॉफ्टवेयर पर विचार करें, कोड में एक साधारण आधार वर्ग कर्मचारी है, कक्षा में रेज़ सेलरी (), ट्रांसफर (), प्रमोट (), ... आदि जैसे तरीके हैं। विभिन्न प्रकार के कर्मचारी जैसे मैनेजर, इंजीनियर, ..आदि के पास बेस क्लास कर्मचारी में मौजूद विधियों का कार्यान्वयन हो सकता है। हमारे पूरे सॉफ्टवेयर में, हमें बस हर जगह कर्मचारियों की एक सूची पास करने की जरूरत है और कर्मचारी के प्रकार को जाने बिना भी उचित तरीकों को कॉल करने की जरूरत है। उदाहरण के लिए, हम कर्मचारियों की सूची के माध्यम से पुनरावृत्ति करके सभी कर्मचारियों का वेतन आसानी से बढ़ा सकते हैं। प्रत्येक प्रकार के कर्मचारी की अपनी कक्षा में तर्क हो सकता है, हमें चिंता करने की आवश्यकता नहीं है क्योंकि यदि किसी विशिष्ट कर्मचारी प्रकार के लिए रेज़ सेलरी () मौजूद है, तो केवल उस विधि को बुलाया जाएगा।



```

// A Simple Java program to demonstrate application
// of overriding in Java
// Base Class
class Employee {
    public static int base = 10000;
    int salary()
    {

```

```
        return base;
    }
}
// Inherited class
class Manager extends Employee {
    // This method overrides salary() of Parent
    int salary()
    {
        return base + 20000;
    }
}
// Inherited class
class Clerk extends Employee {
    // This method overrides salary() of Parent
    int salary()
    {
        return base + 10000;
    }
}
// Driver class
class Main {
    // This method can be used to print the salary of
    // any type of employee using base class reference
    static void printSalary(Employee e)
    {
        System.out.println(e.salary());
    }
    public static void main(String[] args)
    {
        Employee obj1 = new Manager();
    }
}
```

```

        // We could also get type of employee using
        // one more overridden method.loke getType()
        System.out.print("Manager's salary : ");
        printSalary(obj1);
        Employee obj2 = new Clerk();
        System.out.print("Clerk's salary : ");
        printSalary(obj2);
    }
}

```

आउटपुट:

Manager's salary : 30000

Clerk's salary : 20000

एक्सेप्शन हैंडलिंग (Exception Handling)

जावा में एक्सेप्शन हैंडलिंग रनटाइम एरर को संभालने के लिए शक्तिशाली तंत्र में से एक है, इसलिए आवेदन के सामान्य प्रवाह को बनाए रखा जा सकता है।

जावा में एक्सेप्शन क्या है

एक्सेप्शन एक असामान्य स्थिति है। जावा में, एक एक्सेप्शन एक ऐसी घटना है जो प्रोग्राम के सामान्य प्रवाह को बाधित करती है । यह एक वस्तु है जिसे रनटाइम पर दिया जाता है ।

एक्सेप्शन हैंडलिंग क्या है

एक्सेप्शन हैंडलिंग रनटाइम वृटियों जैसे क्लास नॉट फाउंड, आईओ एक्सेप्शन, एसक्यूएल एक्सेप्शन, रीमोटएक्सेप्शन, आदि को संभालने के लिए एक तंत्र है।

एक्सेप्शन हैंडलिंग के लाभ

एक्सेप्शन से निपटने का मुख्य लाभ एप्लिकेशन के सामान्य प्रवाह को बनाए रखना है। एक एक्सेप्शन आमतौर पर एप्लिकेशन के सामान्य प्रवाह को बाधित करता है, इसलिए हम एक्सेप्शन हैंडलिंग का उपयोग करते हैं। आइए एक परिदृश्य लें:

```

statement 1;
statement 2;
statement 3;
statement 4;

```

```

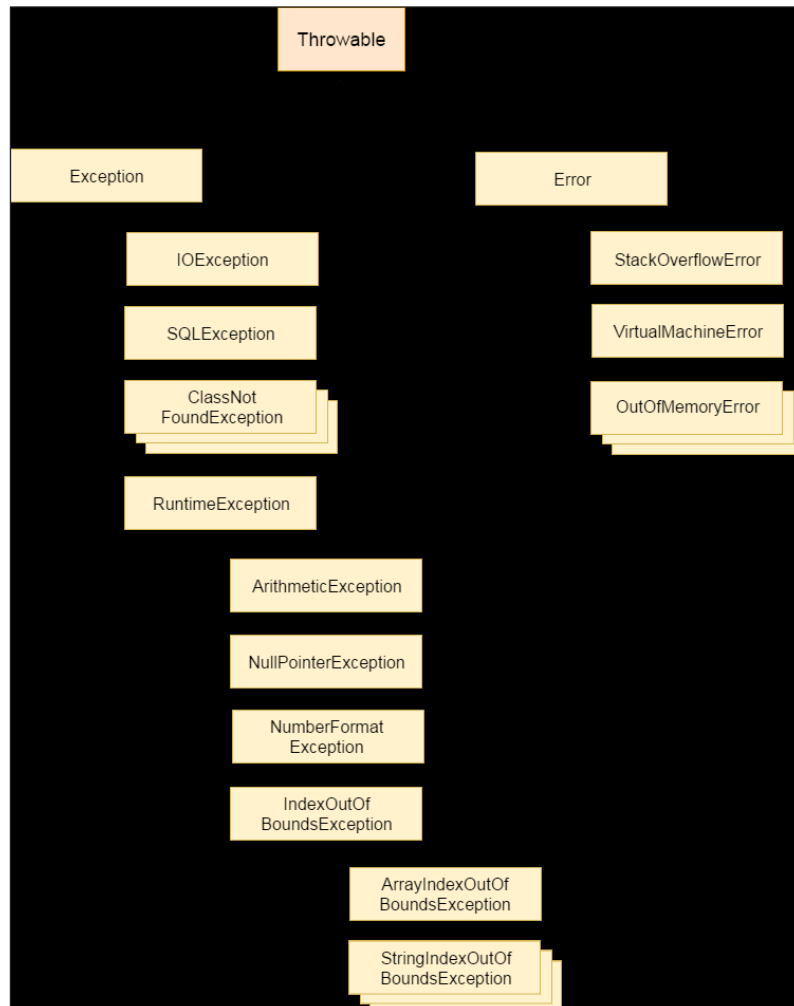
statement 5;//exception occurs
statement 6;
statement 7;
statement 8;
statement 9;
statement 10;

```

मान लीजिए आपके प्रोग्राम में 10 स्टेटमेंट हैं और स्टेटमेंट 5 में एक एक्सेप्शन होता है, बाकी कोड को निष्पादित नहीं किया जाएगा यानी स्टेटमेंट 6 से 10 को निष्पादित नहीं किया जाएगा। यदि हम एक्सेप्शन हैंडलिंग करते हैं, तो शेष कथन निष्पादित किया जाएगा। इसलिए हम जावा में एक्सेप्शन हैंडलिंग का उपयोग करते हैं।

जावा एक्सेप्शन क्लासेस(Java Exception Classes) के पदानुक्रम

जावा डॉट लैंग(java.lang.) । थ्रोएबल क्लास जावा एक्सेप्शन पदानुक्रम का मूल वर्ग है जो दो उपवर्गों द्वारा इनहेरिटेन्स में मिला है: एक्सेप्शन(Exception) और एरर(Error) । जावा एक्सेप्शन वर्गों का एक पदानुक्रम नीचे दिया गया है:



जावा एक्सेप्शन(Java Exception) के प्रकार

मुख्य रूप से दो प्रकार के एक्सेप्शन हैं: चेक और अनचेक। यहां, एक त्रुटि को अनचेक एक्सेप्शन माना जाता है। ओरेकल के अनुसार, तीन प्रकार के एक्सेप्शन हैं:

- चेक एक्सेप्शन।
- अनचेक एक्सेप्शन।
- एरर।



चेकड(Checked) और अनचेकड(Unchecked) एक्सेप्शन(Exception) के बीच अंतर

- चेक एक्सेप्शन: रनटाइम एक्सेप्शन और एरर को छोड़कर सीधे थ्रोएवल क्लास को इनहेरिट करने वाली क्लास को चेक किए गए एक्सेप्शन के रूप में जाना जाता है जैसे आईओ एक्सेप्शन, एसक्यूएल एक्सेप्शन आदि। चेक किए गए एक्सेप्शन को संकलन-समय पर चेक किया जाता है।
- अनचेक एक्सेप्शन: रनटाइम एक्सेप्शन को इनहेरिट करने वाले वर्गों को अनचेक अपवादों के रूप में जाना जाता है जैसे कि अर्थमेटिकएक्सेप्शन, नलपॉइंटर एक्सेप्शन, अरेइंडेक्सआउटऑफ-बाउंड्स एक्सेप्शन आदि। अनचेक एक्सेप्शन को कंपाइल-टाइम पर चेक नहीं किया जाता है, लेकिन उन्हें रनटाइम पर चेक किया जाता है।
- एरर: एरर अपरिवर्तनीय है जैसे आउटऑफमेमोरीएरर, वर्चुअलमेकेनिकएरर, एसरशनएरर आदि।

जावा एक्सेप्शन कीवर्ड(Java Exception Keywords)

जावा में एक्सेप्शन को संभालने के लिए 5 कीवर्ड का उपयोग किया जाता है।

कीवर्ड	विवरण
ट्राइ	"ट्राइ" कीवर्ड का उपयोग उस ब्लॉक को निर्दिष्ट करने के लिए किया जाता है जहां हमें एक्सेप्शन कोड रखना चाहिए। ट्राइ ब्लॉक का पालन या तो कैच या अंत में किया जाना चाहिए। इसका मतलब है, हम अकेले ट्राइ ब्लॉक का उपयोग नहीं कर सकते।

कैच	एक्सेप्शन को संभालने के लिए "कैच" ब्लॉक का उपयोग किया जाता है। इसके पहले ट्राइ ब्लॉक होना चाहिए जिसका अर्थ है कि हम अकेले कैच ब्लॉक का उपयोग नहीं कर सकते हैं। इसके बाद अंत में बाद में ब्लॉक किया जा सकता है।
फाइनली	प्रोग्राम के महत्वपूर्ण कोड को निष्पादित करने के लिए " फाइनली " ब्लॉक का उपयोग किया जाता है। यह निष्पादित किया जाता है कि एक एक्सेप्शन को संभाला जाता है या नहीं।
थ्रो	एक्सेप्शन देने के लिए " थ्रो " कीवर्ड का उपयोग किया जाता है।
थ्रोज़	एक्सेप्शन घोषित करने के लिए " थ्रोज़ " कीवर्ड का उपयोग किया जाता है। यह एक एक्सेप्शन नहीं देता है। यह निर्दिष्ट करता है कि एक्सेप्शन में कोई अपवाद हो सकता है। यह हमेशा सिग्रेचर विधि के साथ प्रयोग किया जाता है।

जावा एक्सेप्शन हैंडलिंग(Java Exception Handling) के उदाहरण

आइए जावा एक्सेप्शन हैंडलिंग का एक उदाहरण देखें जहां हम हैंडल करने के लिए ट्राइ-कैच स्टेटमेंट का उपयोग करते हैं।

```
public class JavaExceptionExample{
    public static void main(String args[]){
        try{
            //code that may raise exception
            int data=100/0;
        }catch(ArithmeticException e){System.out.println(e);}
        //rest code of the program
        System.out.println("rest of the code...");
    }
}
```

आउटपुट:

```
Exception in thread main java.lang.ArithmeticException:/ by zero
rest of the code...
```

उपरोक्त उदाहरण में, 100/0 एक अंकगणित अपवाद उठाता है जिसे एक कोशिश-पकड़ ब्लॉक द्वारा नियंत्रित किया जाता है।

जावा एक्सेप्शन के सामान्य परिदृश्य

ऐसे कुछ परिदृश्य दिए गए हैं जहां अनचेक अपवाद हो सकते हैं। वे इस प्रकार हैं:

1) एक परिदृश्य जहां अंकगणित अपवाद होता है: यदि हम किसी संख्या को शून्य से विभाजित करते हैं, तो अर्थमेटिक एक्सेप्शन होता है।

```
int a=50/0;// ArithmeticException
```

2) एक परिदृश्य जहां नलपोइंटर एक्सेप्शन होता है: यदि हमारे पास किसी भी वेरिएबल में शून्य मान है, तो वेरिएबल पर किसी भी ऑपरेशन को निष्पादित करने से नलपोइंटर एक्सेप्शन फेंकता है।

```
String s=null;

System.out.println(s.length()); //NullPointerException
```

- 3) एक परिदृश्य जहां नंबर फॉर्मेट एक्सेप्शन होता है: किसी भी मान का गलत स्वरूपण नंबर फॉर्मेट एक्सेप्शन हो सकता है। मान लीजिए कि हमारे पास एक स्ट्रिंग वेरिएबल है जिसमें वर्ण हैं, इस वेरिएबल को अंकों में परिवर्तित करने से नंबर फॉर्मेट एक्सेप्शन होगा।

```
String s="abc";

int i=Integer.parseInt(s); //NumberFormatException
```

- 4) एक परिदृश्य जहां ऐरे इंडेक्स आउटऑफबाउंड्सएक्सेप्शन होता है: यदि आप गलत इंडेक्स में कोई मान डाल रहे हैं, तो इसका परिणाम ऐरे इंडेक्स आउटऑफबाउंड्सएक्सेप्शन जैसा कि नीचे दिखाया गया है:

```
int a[]=new int[5];

a[10]=50; //ArrayIndexOutOfBoundsException
```

उपयोगकर्ता द्वारा परिभाषित कस्टम एक्सेप्शन को कैसे हैंडल करें

जावा हमें अपने स्वयं के एक्सेप्शन बनाने की सुविधा प्रदान करता है जो मूल रूप से एक्सेप्शन के व्युत्पन्न वर्ग हैं। उदाहरण के लिए नीचे दिए गए कोड में माईएक्सेप्शन एक्सेप्शन वर्ग का विस्तार करता है।

हम सुपर क्लास के कंस्ट्रक्टर को स्ट्रिंग पास करते हैं- एक्सेप्शन जो बनाई गई वस्तु पर "गेट- मेसेज()" फ़ंक्शन का उपयोग करके प्राप्त किया जाता है।

```
// A Class that represents use-defined exception
class MyException extends Exception
{
    public MyException(String s)
    {
        // Call constructor of parent Exception
        super(s);
    }
}

// A Class that uses above MyException
public class Main
{
    // Driver Program
    public static void main(String args[])
```

```

    {
        try
        {
            // Throw an object of user defined exception
            throw new MyException("GeeksGeeks");
        }
        catch (MyException ex)
        {
            System.out.println("Caught");
            // Print the message from MyException object
            System.out.println(ex.getMessage());
        }
    }
}

```

आउटपुट:

Caught

GeeksGeeks

पिछले कोड में, माईएक्सेप्शन के निर्माता को इसके तर्क के रूप में एक स्ट्रिंग की आवश्यकता होती है। सुपर () का उपयोग करके स्ट्रिंग को पैरेंट क्लास एक्सेप्शन के कंस्ट्रक्टर को पास किया जाता है। एक्सेप्शन वर्ग के निर्माता को पैरामीटर के बिना भी बुलाया जा सकता है और सुपर को कॉल करना अनिवार्य नहीं है।

// A Class that represents use-defined exception

```
class MyException extends Exception
```

```
{
```

```
}
```

// A Class that uses above MyException

```
public class setText
```

```
{
```

```
    // Driver Program
```

```
    public static void main(String args[])
```

```
{
```

```
try
{
    // Throw an object of user defined exception
    throw new MyException();
}
catch (MyException ex)
{
    System.out.println("Caught");
    System.out.println(ex.getMessage());
}
}
```

आउटपुट:

Caught

null

जावा में कंटेनर, सॉर्टिंग और श्रेड्स

एक कंटेनर जावा के अन्य घटकों को रखता है। सॉर्टिंग एक परिभाषित क्रम में डेटा के एक जटिल सेट को व्यवस्थित करने की प्रक्रिया को संदर्भित करता है, या तो संख्यात्मक या वर्णानुक्रम में। किसी प्रोग्राम को निष्पादित करते समय उसके बाद के पथ को श्रेड कहा जाता है। यह अध्याय विषय की व्यापक समझ प्रदान करने के लिए जावा में कंटेनरों, छंटाई और श्रेड्स की बारीकी से जाँच करता है।

स्ट्रिंग का उपयोग कैसे करें

जावा स्ट्रिंग सबसे व्यापक रूप से उपयोग की जाने वाली कक्षाओं में से एक है। जावा स्ट्रिंग क्लास को जावा डॉट लैंग पैकेज में परिभाषित किया गया है।

जावा स्ट्रिंग(Java String)

- मूल रूप से, स्ट्रिंग वर्णों का एक क्रम है लेकिन यह एक प्राइमिटिव टाइप नहीं है।
- जब हम जावा में एक स्ट्रिंग बनाते हैं, तो यह वास्तव में स्ट्रिंग प्रकार का ऑब्जेक्ट बनाता है।
- स्ट्रिंग अपरिवर्तनीय वस्तु है जिसका अर्थ है कि इसे बनाने के बाद इसे बदला नहीं जा सकता है।
- स्ट्रिंग एकमात्र वर्ग है जहाँ जावा में ऑपरेटर ओवरलोडिंग का समर्थन किया जाता है। हम दो तार का उपयोग कर जोड़ सकते हैं + ऑपरेटर। उदाहरण के लिए "ए" + "बी" = "एबी"।
- जावा स्ट्रिंग हेरफेर के लिए दो उपयोगी वर्ग प्रदान करता है - स्ट्रिंगबफर और स्ट्रिंगबिल्डर।

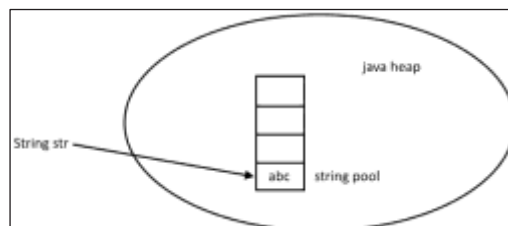
स्ट्रिंग(String) बनाने के विभिन्न तरीके

जावा में स्ट्रिंग ऑब्जेक्ट बनाने के कई तरीके हैं, उनमें से कुछ लोकप्रिय नीचे दिए गए हैं।

शाब्दिक स्ट्रिंग का उपयोग करते हुए

यह स्ट्रिंग बनाने का सबसे आम तरीका है। इस मामले में एक स्ट्रिंग अक्षर दोहरे उद्धरण चिह्नों के साथ संलग्न है।

```
String str = "abc";
```



जब हम दोहरे उद्धरण चिह्नों का उपयोग करके एक स्ट्रिंग बनाते हैं, तो जेवीएम स्ट्रिंग पूल में देखता है कि क्या कोई अन्य है।

स्ट्रिंग को समान मान के साथ संग्रहित किया जाता है। यदि पाया जाता है, तो यह केवल उस स्ट्रिंग ऑब्जेक्ट का संदर्भ देता है अन्यथा यह दिए गए मान के साथ एक नया स्ट्रिंग ऑब्जेक्ट बनाता है और इसे स्ट्रिंग पूल में संग्रहित करता है।

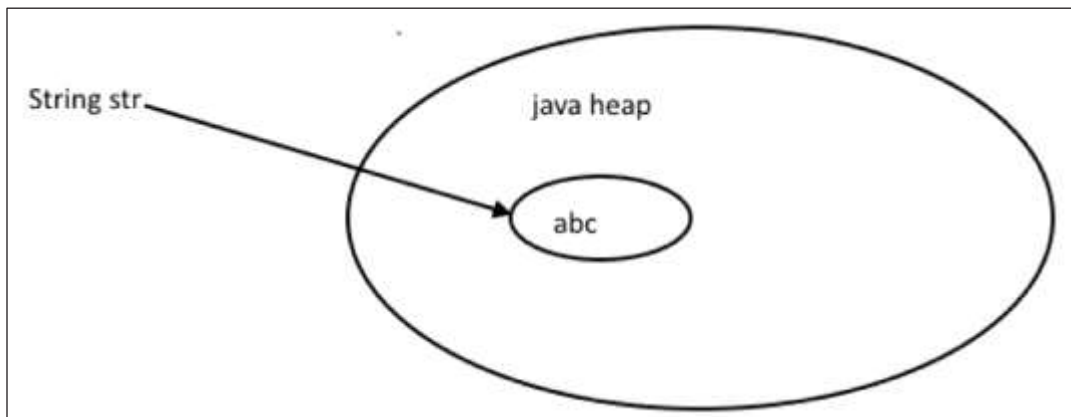
नए कीवर्ड(New Keyword) का उपयोग करना

हम किसी भी सामान्य जावा वर्ग की तरह, नए ऑपरेटर का उपयोग करके स्ट्रिंग ऑब्जेक्ट बना सकते हैं। स्ट्रिंग क्लास में चार ऐरे, बाइट ऐरे, स्ट्रिंग बफर और स्ट्रिंग बिल्डर से स्ट्रिंग प्राप्त करने के लिए कई कंस्ट्रक्टर उपलब्ध हैं।

```
String str = new String("abc");
```

```
char[] a = {'a', 'b', 'c'};
```

```
String str2 = new String(a);
```



जावा स्ट्रिंग कंपेयर(Java String Compare)

स्ट्रिंग क्लास दो स्ट्रिंग्स की तुलना करने के लिए `equals()` और `equalsIgnoreCase()` तरीके प्रदान करता है। ये विधियाँ स्ट्रिंग के मान की तुलना यह जाँचने के लिए करती हैं कि दो तार बराबर हैं या नहीं। यदि दो तार बराबर हैं और यदि नहीं तो यह सही है।

```
package com.journaldev.string.examples;
```

```
/**
```

```
 * Java String Example
```

```
 *
```

```
 * @author pankaj
```

```
 *
```

```
 */
```

```
public class StringEqualExample {
```

```
    public static void main(String[] args) {
```

```
        //creating two string object
```

```

String s1 = "abc";

String s2 = "abc";

String s3 = "def";

String s4 = "ABC";

System.out.println(s1.equals(s2)); //true

System.out.println(s2.equals(s3)); //false

System.out.println(s1.equals(s4)); //false;

System.out.println(s1.equalsIgnoreCase(s4)); //true
    }
}

```

उपरोक्त कार्यक्रम का आउटपुट

true

false

false

true

स्ट्रिंग क्लास तुलनात्मक इंटरफेस लागू करता है, जो `compareTo()` तथा `compareToIgnoreCase()` विधियां प्रदान करता है और यह दो स्ट्रिंग्स की लेक्सिकोग्राफिक रूप से तुलना करता है।

दोनों स्ट्रिंग्स तुलना के लिए यूनिकोड मान में परिवर्तित हो जाते हैं और एक पूर्णांक मान लौटाते हैं जो शून्य से अधिक, कम या बराबर हो सकता है। यदि तार बराबर हैं तो यह शून्य लौटाता है या फिर यह शून्य से अधिक या कम देता है।

```

package com.journaldev.examples;

/**
 * Java String compareTo Example
 *
 * @author pankaj
 *
 */
public class StringCompareToExample {

```

```
public static void main(String[] args) {  
  
    String a1 = "abc";  
    String a2 = "abc";  
    String a3 = "def";  
    String a4 = "ABC";  
  
    System.out.println(a1.compareTo(a2)); //0  
    System.out.println(a2.compareTo(a3)); //less than 0  
    System.out.println(a1.compareTo(a4)); //greater than 0  
    System.out.println(a1.compareToIgnoreCase(a4)); //0  
}  
}
```

उपरोक्त कार्यक्रम का आउटपुट

```
-3  
32  
0
```

जावा स्ट्रिंग मेथड्स(Java String Methods)

आइए एक उदाहरण प्रोग्राम के साथ कुछ लोकप्रिय स्ट्रिंग क्लास विधियों पर एक नज़र डालें।

स्प्लिट (Split) ()

जावा स्ट्रिंग स्प्लिट () मेथडका उपयोग दिए गए एक्सप्रेशन का उपयोग करके स्ट्रिंग को विभाजित करने के लिए किया जाता है। स्प्लिट () मेथडके दो प्रकार हैं।

- `split(String regex)` : यह मेथडदी गई रेगेक्स अभिव्यक्ति का उपयोग करके स्ट्रिंग को विभाजित करती है और स्ट्रिंग की सरणी लौटाती है।
- `split(String regex, int limit)` : यह मेथडदिए गए रेगेक्स एक्सप्रेशन और स्ट्रिंग के रिटर्न एरे का उपयोग करके स्ट्रिंग को विभाजित करती है लेकिन एरे का तत्व निर्दिष्ट सीमा तक सीमित है। यदि निर्दिष्ट सीमा 2 है, तो एक एरे का मेथड रिटर्न आकार 2 होता है।
- `package com.journaldev.examples;`


```
/**
 * Java String split example
 *
 * @author pankaj
 *
 */
public class StringSplitExample {

    public static void main(String[] args) {

        String s = "a/b/c/d";
        String[] a1 = s.split("/");
        System.out.println("split string using only regex:");
        for (String string : a1) {
            System.out.println(string);
        }
        System.out.println("split string using regex with limit:");
        String[] a2 = s.split("/", 2);
        for (String string : a2) {
            System.out.println(string);
        }
    }
}
```

उपरोक्त कार्यक्रम का आउटपुट

Split string using only regex:

a
b
c
d

split string using regex with limit:

a

b/c/d

कंटेन्स (चार सीक्वेंस) (Contains (CharSequences))

जावा स्ट्रिंग कंटेन्स() विधियाँ इफ स्ट्रिंग में करेक्टर का निर्दिष्ट अनुक्रम जाँचती है या नहीं। यदि स्ट्रिंग में करेक्टर का निर्दिष्ट क्रम है, तो यह मेथडसही है, अन्यथा गलत है।

```
package com.journaldev.examples;

/**
 * Java String contains() Example
 *
 * @author pankaj
 *
 */
public class StringContainsExample {

    public static void main(String[] args) {

        String s = "Hello World";

        System.out.println(s.contains("W")); //true
        System.out.println(s.contains("X")); //false
    }

}
```

उपरोक्त कार्यक्रम का आउटपुट

true

false

लेंथ(Length) ()

जावा स्ट्रिंग लेंथ() मेथडस्ट्रिंग की लंबाई रिटर्न करती है।

```

package com.journaldev.examples;

/**
 * Java String length
 *
 * @author pankaj
 *
 */
public class StringLengthExample {

    public static void main(String[] args) {

        String s1 = "abc";
        String s2 = "abcdef";
        String s3 = "abcdefghi";

        System.out.println(s1.length()); //3
        System.out.println(s2.length()); //6
        System.out.println(s3.length()); //9

    }

}

```

रिप्लेस(Replays) ()

जावा स्ट्रिंग रिप्लेस () मेथडका उपयोग स्ट्रिंग के एक विशिष्ट भाग को अन्य स्ट्रिंग से बदलने के लिए किया जाता है। वहां चार प्रतिस्थापन () मेथडके वेरिएंट हैं।

- `replace(char oldChar, char newChar)` : यह मेथडपुराने-चार की सभी घटनाओं को स्ट्रिंग में न्यूचार के साथ प्रतिस्थापित करती है।
- `replace(CharSequence target, CharSequence replacement)` : यह मेथडप्रत्येक लक्ष्य अक्षर को स्ट्रिंग में प्रतिस्थापन शाब्दिक अशुद्धि के साथ प्रतिस्थापित करती है।
- `replaceAll(String regex, String replacement)`: यह मेथडसभी घटित को प्रतिस्थापित करती है- स्ट्रिंग में निर्दिष्ट प्रतिस्थापन के साथ निर्दिष्ट रेगेक्स के साथ सबस्ट्रिंग मेल खाता है।

- `replaceFirst(String regex, String replacement)` : यह मैथड स्ट्रिंग में निर्दिष्ट प्रतिस्थापन के साथ निर्दिष्ट रेगेक्स के साथ मेल खाने वाले सबस्ट्रिंग की पहली घटना को प्रतिस्थापित करती है।
- `package com.journaldev.examples;`

```
/**
 * Java String replace
 *
 * @author pankaj
 *
 */
public class StringReplaceExample {

    public static void main(String[] args) {

        //replace(char oldChar, char newChar)
        String s = "Hello World";
        s = s.replace('l', 'm');
        System.out.println("After Replacing l with m :");
        System.out.println(s);

        //replaceAll(String regex, String replacement)
        String s1 = "Hello journaldev, Hello pankaj";
        s1 = s1.replaceAll("Hello", "Hi");
        System.out.println("After Replacing :");
        System.out.println(s1);

        //replaceFirst(String regex, String replacement)
        String s2 = "Hello guys, Hello world";
        s2 = s2.replaceFirst("Hello", "Hi");
        System.out.println("After Replacing :");
```

```

        System.out.println(s2);
    }
}

```

उपरोक्त कार्यक्रम का आउटपुट

After Replacing l with m :

Hemmo Wormd

After Replacing :

Hi journaldev, Hi pankaj

After Replacing :

Hi guys, Hello world

फॉर्मेट(Format) ()

- जावा स्ट्रिंग फॉर्मेट(Java Sting format) () मेथड (method) का उपयोग स्ट्रिंग को प्रारूपित करने के लिए किया जाता है। जावा स्ट्रिंग प्रारूप () मेथड के दो प्रकार हैं।
- `format(Locale l, String format, Object... args)`: यह मेथड स्ट्रिंग को प्रारूपित करती है निर्दिष्ट लोकल, स्ट्रिंग फॉर्मेट और तर्कों का उपयोग करना।
- `format(String format, Object... args)`: यह मेथड निर्दिष्ट स्ट्रिंग फॉर्मेट और तर्कों का उपयोग करके स्ट्रिंग को प्रारूपित करती है।

```
package com.journaldev.examples;
```

```
import java.util.Locale;
```

```
/**
```

```
 * Java String format
```

```
 *
```

```
 * @author pankaj
```

```
 *
```

```
 */
```

```
public class StringFormatExample {

    public static void main(String[] args) {

        String s = "journaldev.com";
        // %s is used to append the string
        System.out.println(String.format("This is %s", s));

        //using locale as Locale.US
        System.out.println(String.format(Locale.US, "%f", 3.14));
    }
}
```

उपरोक्त कार्यक्रम का आउटपुट

```
This is journaldev.com
3.140000
```

सबस्ट्रिंग ()

यह विधि(method) निर्दिष्ट अनुक्रमणिका के आधार पर स्ट्रिंग का एक भाग रिटर्न करती है।

```
package com.journaldev.examples;

/**
 * Java String substring
 *
 */
public class StringSubStringExample {

    public static void main(String[] args) {

        String s = "This is journaldev.com";
        s = s.substring(8,18);
```

```
        System.out.println(s);
    }
}
```

स्ट्रिंग कॉन्सटेनेशन(String Concatenation)

जावा में स्ट्रिंग कॉन्सटेनेशन बहुत ही बुनियादी ऑपरेशन है। स्ट्रिंग को "+" ऑपरेटर का उपयोग करके या कॉन्कैट () मेथडका उपयोग करके जोड़ा जा सकता है।

```
package com.journaldev.examples;

/**
 * Java String concatenation
 *
 * @author pankaj
 *
 */
public class StringConcatExample {

    public static void main(String[] args) {

        String s1 = "Hello";
        String s2 = "World";
        String s3 = s1 + s2;
        //using + operator
        System.out.println("Using + operator: ");
        System.out.println(s3);

        //using concat method
        System.out.println("Using concat method: ");
        System.out.println(s1.concat(s2));

    }

}
```

उपरोक्त कार्यक्रम का आउटपुट

Using + operator:

```
HelloWorld
```

Using concat method:

```
HelloWorld
```

जावा स्ट्रिंग पूल (Java String Pool)

मेमोरी मैनेजमेंट किसी भी प्रोग्रामिंग भाषा का सबसे महत्वपूर्ण पहलू है। जावा में स्ट्रिंग के मामले में मेमोरी मैनेजमेंट किसी भी अन्य वर्ग की तुलना में थोड़ा अलग है। जावा को अधिक मेमोरी कुशल बनाने के लिए, जेवीएम ने स्ट्रिंग कॉन्स्टेंट पूल नामक स्ट्रिंग के लिए एक विशेष मेमोरी क्षेत्र पेश किया।

जब हम एक स्ट्रिंग शाब्दिक बनाते हैं तो यह जांचता है कि स्ट्रिंग पूल में समान स्ट्रिंग पहले से मौजूद है या नहीं। यदि यह वहां है तो यह स्ट्रिंग पूल की मौजूदा स्ट्रिंग का संदर्भ वापस कर देगा।

आइए नीचे दिए गए उदाहरण कार्यक्रम पर एक नजर डालते हैं।

```
package com.journaldev.examples;
```

```
/**
```

```
 * Java String Pool Example
```

```
 *
```

```
 */
```

```
public class StringPoolExample {
```

```
    public static void main(String[] args) {
```

```
        String a = "abc";
```

```
        String b = "abc";
```

```
        String c = "def";
```

```
        //same reference
```

```
        if (a==b) {
```

```
            System.out.println("Both string refer to the same object");
```

```
        }
```



```

//different reference
if (a==c) {
    System.out.println("Both strings refer to the same object");
}else {
    System.out.println("Both strings refer to the different ob-
ject");
}
}
}
}

```

उपरोक्त कार्यक्रम का आउटपुट

Both string refer to the same object

Both strings refer to the different object

स्ट्रिंग इंटर्न () विधि (String Intern () Method)

जब हम स्ट्रिंग शाब्दिक(string literal) का उपयोग करके एक स्ट्रिंग बनाते हैं, तो यह स्ट्रिंग पूल में बनाया जाएगा, लेकिन क्या होगा यदि हम नए कीवर्ड का उपयोग करके उसी मान के साथ एक स्ट्रिंग बनाते हैं जो स्ट्रिंग पूल में मौजूद है? क्या हम स्ट्रिंग को हीप मेमोरी से स्ट्रिंग पूल में ले जा सकते हैं?

इसके लिए इंटर्न () मेथड का उपयोग किया जाता है और यह स्ट्रिंग ऑब्जेक्ट का एक विहित प्रतिनिधित्व देता है। जब हम नए कीवर्ड का उपयोग करके बनाई गई स्ट्रिंग ऑब्जेक्ट पर इंटर्न () मेथडको कॉल करते हैं, तो यह जांचता है कि क्या पूल में समान मान के साथ पहले से ही एक स्ट्रिंग है?

यदि हां, तो यह पूल से उस स्ट्रिंग ऑब्जेक्ट का संदर्भ देता है। यदि नहीं, तो यह पूल में समान सामग्री के साथ एक नया स्ट्रिंग बनाता है और रिफ्रेंस देता है।

```
package com.journaldev.examples;
```

```

/**
 * Java String intern
 *
 * @author pankaj
 *
 */

```

```
public class StringInternExample {  
  
    public static void main(String[] args) {  
  
        String s1 = "pankaj";  
        String s2 = "pankaj";  
        String s3 = new String("pankaj");  
  
        System.out.println(s1==s2);//true  
        System.out.println(s2==s3);//false  
  
        String s4 = s3.intern();  
        System.out.println(s1==s4);//true  
  
    }  
  
}
```

स्ट्रिंग अपरिवर्तनीयता का लाभ

स्ट्रिंग के अपरिवर्तनीय वर्ग होने के कुछ लाभ हैं:

- स्ट्रिंग कॉन्स्टेंट पूल, इसलिए मेमोरी बचाता है।
- सुरक्षा के रूप में इसे बदला नहीं जा सकता।
- थ्रेड सेफ
- क्लास लोडिंग सीक्योरिटी

जावा 8 स्ट्रिंग जॉइन() (Java 8 String join ())

जावा 8 में स्ट्रिंग क्लास में एक नया स्टैटिक मेथड जॉइन () जोड़ा गया है। यह मेथड एक नया स्ट्रिंग लौटाता है, जो कि निर्दिष्ट सीमांकक की एक कॉपी के साथ जुड़े हुए चार-सीक्वेंस एलिमेंट्स की कॉपी से बना होता है। आइए इसे आसानी से समझने के लिए एक उदाहरण देखें।

```
List<String> words = Arrays.asList(new String[]{"Hello", "World", "2019"});  
String msg = String.join(" ", words);  
System.out.println(msg);
```

आउटपुट: Hello World 2019

जावा 9 स्ट्रिंग विधि (Java 9 String Method)

जावा 9 रिलीज में स्ट्रिंग क्लास में दो तरीके जोड़े गए हैं। वे हैं - कोडपॉइंट्स () और चार्स ()। ये दोनों विधियाँ IntStream ऑब्जेक्ट लौटाती हैं, जिस पर हम कुछ ऑपरेशन कर सकते हैं ।

आइए इन विधियों पर एक त्वरित नज़र डालें।

```
String s = "abc";
s.codePoints().forEach(x -> System.out.println(x));
s.chars().forEach(x -> System.out.println(x));
```

उपरोक्त कार्यक्रम का आउटपुट

```
97
98
99
97
98
99
```

जावा 11 स्ट्रिंग क्लास न्यू मेथड्स (Java 11 String Class New Methods)

जावा 11 रिलीज में स्ट्रिंग क्लास में कई नए तरीके जोड़े गए हैं।

- इसब्लॉक () - यदि स्ट्रिंग खाली है या केवल सफेद स्थान कोडपॉइंट हैं, तो सही है, अन्यथा झूठा।
- लाइंस () - इस स्ट्रिंग से निकाली गई लाइनों की एक धारा लौटाता है, जिसे लाइन टर्मिनेटर द्वारा अलग किया जाता है।
- स्ट्रिप (), स्ट्रिपलीडिंग (), स्ट्रिपट्रेलिंग () - स्ट्रिंग से अग्रणी और अनुगामी सफेद रिक्त स्थान को अलग करने के लिए ।
- रिपीट () - एक स्ट्रिंग लौटाता है जिसका मान इस स्ट्रिंग का संयोजन है जिसे बार-बार दोहराया जाता है।

आइए इन विधियों के लिए एक उदाहरण प्रोग्राम देखें।

```
package com.journaldev.strings;

import java.util.List;
import java.util.stream.Collectors;

/**
```

```
* JDK 11 New Functions in String class
*
* @author pankaj
*
*/
public class JDK11StringFunctions {

    public static void main(String[] args) {

        // isBlank()
        String s = "abc";
        System.out.println(s.isBlank());
        s = "";
        System.out.println(s.isBlank());

// lines()
        String s1 = "Hi\nHello\rHowdy";
        System.out.println(s1);
        List lines = s1.lines().collect(Collectors.toList());
        System.out.println(lines);

        // strip(), stripLeading(), stripTrailing()
        String s2 = "  Java,  \tPython\t ";
        System.out.println("#" + s2 + "#");
        System.out.println("#" + s2.strip() + "#");
        System.out.println("#" + s2.stripLeading() + "#");
        System.out.println("#" + s2.stripTrailing() + "#");

        // repeat()
        String s3 = "Hello\n";
        System.out.println(s3.repeat(3));
        s3 = "Co";
```

```

        System.out.println(s3.repeat(2));
    }
}

```

उपरोक्त कार्यक्रम का आउटपुट

```

false
true
Hi
Hello
Howdy
[Hi, Hello, Howdy]
# Java, Python #
#Java, Python#
#Java, Python #
# Java, Python#
Hello
Hello
Hello

CoCo

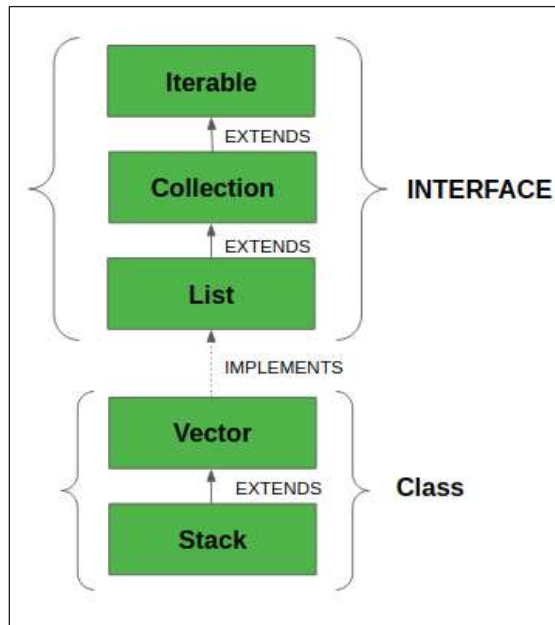
```

जावा स्ट्रिंग क्लास, इसकी मेथड और स्ट्रिंग मैनिपुलेशन उदाहरणों के बारे में यह सब कुछ है।

स्टैक(Stack) का उपयोग कैसे करें

जावा संग्रह ढांचा एक स्टैक क्लास प्रदान करता है जो स्टैक डेटा संरचना को मॉडल और लागू करता है। क्लास लास्ट-इन-फर्स्ट-आउट के मूल सिद्धांत पर आधारित है। बुनियादी पुश और पॉप संचालन के अलावा, वर्ग खाली, खोज और झांकने के तीन और कार्य प्रदान करता है। वर्ग को वेक्टर का विस्तार करने के लिए भी कहा जा सकता है और कक्षा को क्लास उल्लिखित कार्यों के साथ एक ढेर के रूप में मानता है। क्लास को वेक्टर के उपवर्ग के रूप में भी संदर्भित किया जा सकता है।

यह आरेख स्टैक क्लास के पदानुक्रम को दर्शाता है:



क्लास एक डिफॉल्ट कंस्ट्रक्टर (default constructor) स्टैक () का समर्थन करता है जिसका उपयोग एंप्टी स्टैक (empty stack) बनाने के लिए किया जाता है। नीचे कार्यक्रम स्टैक क्लास द्वारा प्रदान किए गए कुछ बुनियादी संचालन दिखाता है:

```
// Java code for stack implementation

import java.io.*;
import java.util.*;

class Test
{
    // Pushing element on the top of the stack
    static void stack_push(Stack<Integer> stack)
    {
        for(int i = 0; i < 5; i++)
        {
            stack.push(i);
        }
    }

    // Popping element from the top of the stack
```

```
static void stack_pop(Stack<Integer> stack)
{
    System.out.println("Pop :");

    for(int i = 0; i < 5; i++)
    {
        Integer y = (Integer) stack.pop();
        System.out.println(y);
    }
}

// Displaying element on the top of the stack
static void stack_peek(Stack<Integer> stack)
{
    Integer element = (Integer) stack.peek();
    System.out.println("Element on stack top : " + element);
}

// Searching element in the stack
static void stack_search(Stack<Integer> stack, int element)
{
    Integer pos = (Integer) stack.search(element);

    if(pos == -1)
        System.out.println("Element not found");
    else
        System.out.println("Element is found at position " + pos);
}

public static void main (String[] args)
{
```

```

Stack<Integer> stack = new Stack<Integer>();

stack_push(stack);
stack_pop(stack);
stack_push(stack);
stack_peek(stack);
stack_search(stack, 2);
stack_search(stack, 6);
}
}

```

उपरोक्त कार्यक्रम का आउटपुट

पॉप :

4
3
2
1
0

स्टैक टॉप पर तत्व: 4

तत्व स्थिति 3 पर पाया जाता है

तत्व नहीं पाया गया

स्टैक क्लास(Stack Class) में तरीकें(Methods)

- ऑब्जेक्ट पुश (ऑब्जेक्ट एलिमेंट) : स्टैक के शीर्ष पर एक तत्व को धक्का देता है।
- ऑब्जेक्ट पॉप () : स्टैक के शीर्ष तत्व को हटाता है और वापस करता है। एक 'EmptyStackException' अपवाद को फेंक दिया जाता है यदि हम पॉप () को कॉल करते हैं जब इनवोकिंग स्टैक खाली होता है।
- ऑब्जेक्ट पीक () : स्टैक के शीर्ष पर तत्व लौटाता है, लेकिन इसे हटाता नहीं है।
- बूलियन एम्प्टी () : यदि स्टैक के शीर्ष पर कुछ भी नहीं है तो यह सत्य हो जाता है। अन्यथा, झूठी वापसी।
- इंट सर्च(ऑब्जेक्ट एलिमेंट): यह निर्धारित करता है कि कोई वस्तु स्टैक में मौजूद है या नहीं। यदि तत्व पाया जाता है, तो यह स्टैक के शीर्ष से तत्व की स्थिति लौटाता है। अन्यथा, यह -1 रिटर्न करता है।

क्यू(Queue) का उपयोग कैसे करें

जावा क्यू जावा डॉट यूटिल पैकेज में उपलब्ध एक इंटरफ़ेस है और जावा डॉट यूटिल डॉट कलेक्शन इंटरफ़ेस का विस्तार करता है। जावा लिस्ट की तरह ही, जावा क्यू ऑर्डर किए गए तत्वों (या ओब्जेक्ट्स) का संग्रह है, लेकिन यह अलग-अलग संचालन को सम्मिलित करता है और हटा देता है। हम उन तत्वों को संसाधित करने से पहले तत्वों को संग्रहित करने के लिए पंक्ति का उपयोग कर सकते हैं।

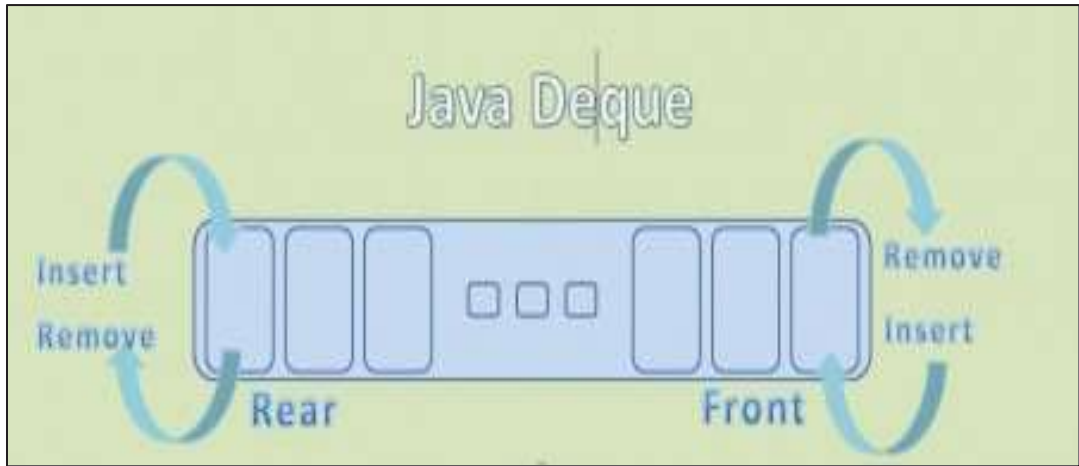


जावा क्यू (Java Queue)

हम जावा क्यू के बारे में कुछ महत्वपूर्ण बिंदुओं पर चर्चा करेंगे:

- `java.util.Queue` इंटरफ़ेस `java.util.Collection` इंटरफ़ेस का एक उपप्रकार है।
- वास्तविक दुनिया की पंक्ति की तरह (उदाहरण के लिए, बैंक या एटीएम में), पंक्ति के अंत में तत्वों को सम्मिलित करती है और पंक्ति की शुरुआत से हटा देती है।
- जावा पंक्ति तत्वों की एक क्रमबद्ध सूची का प्रतिनिधित्व करती है।
- जावा पंक्ति इसके तत्वों को सम्मिलित करने और हटाने के लिए फीफो आदेश का पालन करती है। एफआईएफओ का मतलब फर्स्ट इन फर्स्ट आउट है।
- जावा पंक्ति संग्रह इंटरफ़ेस के सभी तरीकों का समर्थन करती है।
- सबसे अधिक उपयोग की जाने वाली पंक्ति कार्यान्वयन हैं लिंकडलिस्ट, ऐरे ब्लोकिंग क्यू और प्राइमिटिव पंक्ति।
- ब्लोकिंग क्यू अशक्त तत्वों को स्वीकार नहीं करता है। यदि हम कोई अशक्त संबंधित ऑपरेशन करते हैं, तो यह शून्य सूचक का एक्सेप्शन देता है।
- ब्लोकिंग क्यू का उपयोग निर्माता/उपभोक्ता आधारित अनुप्रयोगों को लागू करने के लिए किया जाता है।
- ब्लोकिंग क्यू थ्रेड-सुरक्षित हैं।
- सभी पंक्तियां `java.util` पैकेज में उपलब्ध असंमित कतारें हैं और कतारें जो `java.util.concurrent` पैकेज में उपलब्ध हैं, बाउंडेड क्यू हैं।
- सभी डीक्स के थ्रेड-सुरक्षित नहीं हैं।
- कोनकरेंट लिंकडक्यू लिंकड नोड्स पर आधारित एक असंमित थ्रेड-सुरक्षित पंक्ति है।
- डीक्स को छोड़कर, सभी क्यू पंक्ति के टेल पर सम्मिलन और पंक्ति के टॉप पर हटाने का समर्थन करता है।

- डीक्स पंक्ति हैं लेकिन वे दोनों सिरों पर तत्व सम्मिलन और निष्कासन का समर्थन करते हैं।



जावा क्यू क्लास डायग्राम

जावा क्यू इंटरफ़ेस कलेक्शन इंटरफ़ेस का विस्तार करता है। कलेक्शन इंटरफ़ेस इटरेबल इंटरफ़ेस का विस्तार करता है। लिंकडलिस्ट, प्रायोरिटी क्यू, ऐरे-ब्लॉकिंग क्यू, डिलेक्यू, लिंकडब्लॉकिंग क्यू, प्रायोरिटीब्लॉकिंग क्यू आदि अक्सर उपयोग की जाने वाली क्यू कार्यान्वयन कक्षाओं में से कुछ हैं।

जावा क्यू मेथड

- इंट साइज़ (): सेट में तत्वों की संख्या प्राप्त करने के लिए।
- बूलियन इज़ एम्पटी(): यह जांचने के लिए कि सेट खाली है या नहीं।
- बूलियन कंटेन्स(ऑब्जेक्ट ओ): यदि इस सेट में निर्दिष्ट तत्व है तो सही रिटर्न करता है।
- इटरेटर इटरेटर (): इस सेट में तत्वों पर एक पुनरावर्तक देता है। तत्वों को किसी विशेष क्रम में फिर से चालू नहीं किया जाता है।
- बूलियन रिमूव ऑल (कलेक्शन सी): इस सेट से इसके सभी तत्वों को हटा देता है जो निर्दिष्ट संग्रह (वैकल्पिक संचालन) में निहित हैं।
- बूलियन रिटेनऑल (कलेक्शन सी): इस सेट में केवल उन तत्वों को बनाए रखता है जो निर्दिष्ट संग्रह (वैकल्पिक संचालन) में निहित हैं।
- वॉइड क्लीयर (): सेट से सभी तत्वों को हटा देता है।
- ई रिमूव(): इस पंक्ति के प्रमुख को पुनः प्राप्त करें और हटा दें।
- ई पोल (): इस पंक्ति के प्रमुख को पुनः प्राप्त करें और हटा दें, या यदि यह पंक्ति खाली है तो अशक्त हो जाती है।

- ई पीक (): इस पंक्ति के प्रमुख को प्राप्त करता है, लेकिन हटाता नहीं है, या यदि यह पंक्ति खाली है तो कुछ भी नहीं लौटाता है।
- बूलियन ऑफ़र (E e): यदि क्षमता प्रतिबंधों का उल्लंघन किए बिना तुरंत ऐसा करना संभव हो तो निर्दिष्ट तत्व को इस पंक्ति में सम्मिलित करता है।
- ई तत्व (): इस पंक्ति के प्रमुख को प्राप्त करता है, लेकिन हटाता नहीं है।
- बूलियन ऐड (E e): यदि क्षमता प्रतिबंधों का उल्लंघन किए बिना तुरंत ऐसा करना संभव है, तो निर्दिष्ट तत्व को इस कतार में सम्मिलित करता है, सफलता पर सही लौटाता है और वर्तमान में कोई स्थान उपलब्ध नहीं होने पर एक इलीगल स्टेट एक्सेप्शन दिखाता है।
- ऑब्जेक्ट [] टूएरे (): इस सेट में सभी तत्वों से युक्त एक सारणी देता है। यदि यह सेट कोई गारंटी देता है कि इसके तत्वों को इसके पुनरावर्तक द्वारा किस क्रम में लौटाया जाता है, तो इस मेथड के तत्वों को उसी क्रम में वापस करना होगा।

जावा क्यू बेसिक्स

चूंकि जावा क्यू जावा संग्रह का विस्तार करता है, यह सभी संग्रह इंटरफ़ेस संचालन का भी समर्थन करता है। आइये निम्नलिखित उदाहरण में कुछ सरल कार्यों का पता लगाएं:

```
package com.journaldev.queue;
```

```
import java.util.*;
```

```
public class QueueExample {
```

```
    public static void main(String[] args) {
```

```
        Queue<String> queue = new LinkedList<>();
```

```
        queue.add("one");
```

```
        queue.add("two");
```

```
        queue.add("three");
```

```
        queue.add("four");
```

```
        System.out.println(queue);
```

```
        queue.remove("three"); System.out.println(queue);
```

```
        System.out.println("Queue Size: " + queue.size());
```

```
        System.out.println("Queue Contains element 'two' or not? : " + queue.contains("two"));
```

```

        // To empty the queue
        queue.clear();
    }
}

```

उपरोक्त कार्यक्रम का आउटपुट

```

[one, two, three, four]
[one, two, four]
Queue Size: 3
Queue Contains element 'two' or not? : true

```

जावा ऐरे टू क्यू

यहां हम एक साधारण उदाहरण के साथ "Collections.addAll ()" मेथड का उपयोग करके जावा सारणी को पंक्ति में परिवर्तित करने का तरीका जान सकते हैं।

```

import java.util.*;

public class ArrayToQueue {
    public static void main(String[] args) {

        String nums[] = {"one", "two", "three", "four", "five"};
        Queue<String> queue = new LinkedList<>();
        Collections.addAll(queue, nums);
        System.out.println(queue);
    }
}

```

उपरोक्त कार्यक्रम का आउटपुट

जब हम उपरोक्त प्रोग्राम चलाते हैं, तो हमें निम्न आउटपुट मिलेगा:

```

[one, two, three, four, five]

```

जावा क्यू टू ऐरे

यहां हम एक सरल उदाहरण के साथ "toArray ()" का उपयोग करके जावा पंक्ति को जावा ऐरे में बदलने का तरीका तलाशेंगे।

```

import java.util.*;

```

```

public class QueueToArray {
    public static void main(String[] args) {

        Queue<String> queue = new LinkedList<>();
        queue.add("one");
        queue.add("two");
        queue.add("three");
        queue.add("four");
        queue.add("five");

        String strArray[] = queue.toArray(new String[queue.size()]);
        System.out.println(Arrays.toString(strArray));
    }
}

```

उपरोक्त कार्यक्रम का आउटपुट

जब हम उपरोक्त प्रोग्राम चलाते हैं, तो हमें निम्न आउटपुट मिलेगा:

```
[one, two, three, four, five]
```

जावा क्यू कॉमन ऑपरेशन

जावा क्यू कलेक्शन इंटरफ़ेस और कुछ और संचालन द्वारा समर्थित सभी कार्यों का समर्थन करती है। यह लगभग सभी कार्यों को दो रूपों में सपोर्ट करता है।

- ऑपरेशन का एक सेट ऑपरेशन विफल होने पर एक्सेप्शन देता है।
- यदि ऑपरेशन विफल हो जाता है, तो ऑपरेशन का दूसरा सेट एक विशेष मान देता है।

निम्न तालिका ऑल क्यू कॉमन ऑपरेशन को संक्षेप में बताती है।

कार्यवाही	अपवाद फेंकता है	विशेष मूल्य
डालने	जोड़ें (ई)	प्रस्ताव (ई)
हटाना	हटाना()	मतदान ()
की जांच	तत्व ()	झांकना ()

जावा क्यू इन्सर्ट ऑपरेशन

यदि यह ऑपरेशन सफलतापूर्वक चलता है, तो यह "सही" मान देता है। जैसा कि हम जानते हैं, क्यू दो रूपों में इन्सर्ट ऑपरेशन का समर्थन करता है:

- `Queue.add (e)`:

यदि ऑपरेशन विफल हो जाता है तो यह अपवाद फेंकता है।

- `Queue.offer (e)`:

ऑपरेशन विफल होने पर यह एक विशेष मान देता है।

नोट:- यहां विशेष मान या तो "गलत" या "शून्य" हो सकता है

क्यू एड () ऑपरेशन

एड () ऑपरेशन का उपयोग कतार में नया तत्व डालने के लिए किया जाता है। यदि यह सम्मिलित ऑपरेशन सफलतापूर्वक करता है, तो यह "सत्य" मान देता है। अन्यथा यह `java.lang.IllegalStateException` फेंकता है।

आइए इस कार्यक्षमता को प्रदर्शित करने के लिए एक सरल उदाहरण विकसित करें।

```
import java.util.concurrent.*;

public class QueueAddOperation {
    public static void main(String[] args) {

        BlockingQueue<String> queue = new ArrayBlockingQueue<>(2);

        System.out.println(queue.add("one"));
        System.out.println(queue.add("two"));
        System.out.println(queue);
        System.out.println(queue.add("three"));
        System.out.println(queue);
    }
}
```

उपरोक्त कार्यक्रम का आउटपुट

जब हम उपरोक्त प्रोग्राम चलाते हैं, तो हमें निम्न आउटपुट मिलेगा:

```
true
```

```
true
```

```
[one, two]
```

```
Exception in thread "main" java.lang.IllegalStateException: Queue full
```

चूंकि हमारी कतार दो तत्वों तक सीमित है, जब हम `BlockingQueue` का उपयोग करके तीसरे तत्व को जोड़ने का प्रयास करते हैं। जोड़ें (), यह एक अपवाद फेंकता है।

क्यू ऑफर () ऑपरेशन

ऑफर () ऑपरेशन का उपयोग पंक्ति में नए तत्व को सम्मिलित करने के लिए किया जाता है। यदि यह सम्मिलित ऑपरेशन सफलतापूर्वक करता है, तो यह "सत्य" मान देता है। अन्यथा यह "गलत" मान देता है।

आइए इस कार्यक्षमता को प्रदर्शित करने के लिए एक सरल उदाहरण विकसित करें।

```
import java.util.concurrent.*;

public class QueueOfferOperation {
    public static void main(String[] args) {

        BlockingQueue<String> queue = new ArrayBlockingQueue<>(2);

        System.out.println(queue.offer("one"));
        System.out.println(queue.offer("two"));
        System.out.println(queue);
        System.out.println(queue.offer("three"));
        System.out.println(queue);
    }
}
```

उपरोक्त कार्यक्रम का आउटपुट

जब हम उपरोक्त प्रोग्राम चलाते हैं, तो हमें निम्न आउटपुट मिलेगा:

```
true
true
[one, two]
false
[one, two]
```

चूंकि हमारी कतार दो तत्वों तक सीमित है, जब हम ब्लॉकिंग क्यू का उपयोग करके तीसरे तत्व को जोड़ने का प्रयास करते हैं। ऑफर () ऑपरेशन, यह "गलत" मान देता है।

जावा क्यू डिलीट ऑपरेशन

यदि यह सफलतापूर्वक निष्पादित होता है, तो डिलीट ऑपरेशन कतार के मुख्य तत्व को लौटाता है। जैसा कि हम जानते हैं, कतार दो रूपों में डिलीट ऑपरेशन का समर्थन करती है:

- `Queue.remove ()`:

यदि ऑपरेशन विफल हो जाता है तो यह अपवाद फेंकता है।

- `Queue.poll ()`:

ऑपरेशन विफल होने पर यह एक विशेष मान देता है।

नोट:- यहां विशेष मान या तो "गलत" या "शून्य" हो सकता है

क्यू रिमूव () ऑपरेशन

रिमूव () ऑपरेशन का उपयोग पंक्ति के प्रमुख से किसी तत्व को हटाने के लिए किया जाता है। यदि यह सफलतापूर्वक डिलीट ऑपरेशन करता है, तो यह पंक्ति का मुख्य तत्व रिटर्न करता है। अन्यथा यह जावा दिखाता है। युटिलिटी नो सच एलिमेंट एक्सेप्शन।

आइए इस कार्यक्षमता को प्रदर्शित करने के लिए एक सरल उदाहरण विकसित करें।

```
import java.util.*;

public class QueueRemoveOperation
{
    public static void main(String[] args)
    {
        Queue<String> queue = new LinkedList<>();
        queue.offer("one");
        queue.offer("two");
        System.out.println(queue);
        System.out.println(queue.remove());
        System.out.println(queue.remove());
        System.out.println(queue.remove());
    }
}
```

उपरोक्त कार्यक्रम का आउटपुट

जब हम उपरोक्त प्रोग्राम चलाते हैं।

हमें निम्नलिखित आउटपुट मिलेगा:

```
[one, two]
```

```
one
```

```
two
```

```
Exception in thread "main" java.util.NoSuchElementException
```

चूंकि हमारी पंक्ति में केवल दो तत्व हैं, जब हम तीसरी बार रिमूव () मेथड को कॉल करने का प्रयास करते हैं, तो यह एक एक्सेप्शन दिखाता है।

Queue.remove (element) का उपयोग पंक्ति से एक निर्दिष्ट तत्व को हटाने के लिए किया जाता है। अगर यह सफलतापूर्वक डिलीट ऑपरेशन करता है, तो यह "सही" मान देता है। अन्यथा यह "गलत" मान देता है।

क्यू पोल () ऑपरेशन

पोल () ऑपरेशन का उपयोग पंक्ति के प्रमुख से किसी तत्व को हटाने के लिए किया जाता है। यदि यह सफलतापूर्वक डिलीट ऑपरेशन करता है, तो यह पंक्ति का मुख्य तत्व लौटाता है। अन्यथा यह "शून्य" मान देता है।

आइए इस कार्यक्षमता को प्रदर्शित करने के लिए एक सरल उदाहरण विकसित करें।

```
import java.util.*;

public class QueuePollOperation
{
    public static void main(String[] args)
    {
        Queue<String> queue = new LinkedList<>();
        queue.offer("one");
        queue.offer("two");
        System.out.println(queue);
        System.out.println(queue.poll());
        System.out.println(queue.poll());
        System.out.println(queue.poll());
    }
}
```

उपरोक्त कार्यक्रम का आउटपुट

जब हम उपरोक्त प्रोग्राम चलाते हैं, तो हमें निम्न आउटपुट मिलेगा:

```
[one, two]
one
two
null
```

चूंकि हमारी पंक्ति में केवल दो तत्व हैं, जब हम तीसरी बार `पोल ()` मेथड को कॉल करने का प्रयास करते हैं, तो यह शून्य मान वापस आ जाता है।

जावा क्यू एग्जेमाइन ऑपरेशन

यदि यह ऑपरेशन सफलतापूर्वक करता है, तो यह कतार के मुख्य तत्व को हटाए बिना वापस कर देता है। जैसा कि हम जानते हैं, पंक्ति दो रूपों में जांच ऑपरेशन का समर्थन करती है:

- `Queue.element ()`:

यदि ऑपरेशन विफल हो जाता है तो यह एक्सेप्शन दिखाता है।

- `Queue.peek ()`:

ऑपरेशन विफल होने पर यह एक विशेष मान देता है।

नोट:- यहां विशेष मान या तो "गलत" या "शून्य" हो सकता है

क्यू एलिमेंट () ऑपरेशन

एलिमेंट () ऑपरेशन का उपयोग पंक्ति के टॉप से एक एलिमेंट को पुनः प्राप्त करने के लिए किया जाता है। यदि यह सफलतापूर्वक जांच संचालन करता है, तो यह पंक्ति का मुख्य तत्व लौटाता है। अन्यथा यह `java.util.NoSuchElementException` दिखाता है।

आइए इस कार्यक्षमता को प्रदर्शित करने के लिए एक सरल उदाहरण विकसित करें।

```
import java.util.*;

public class QueueElementOperation {
    public static void main(String[] args) {

        Queue<String> queue = new LinkedList<>();
        queue.add("one");

        System.out.println(queue.element());
        System.out.println(queue);
        queue.clear();
    }
}
```

```

        System.out.println(queue.element());
    }
}

```

उपरोक्त कार्यक्रम का आउटपुट

जब हम उपरोक्त प्रोग्राम चलाते हैं, तो हमें निम्न आउटपुट मिलेगा:

```

one
[one]
Exception in thread "main" java.util.NoSuchElementException

```

यदि हम खाली कतार पर तत्व () मेथडको कॉल करने का प्रयास करते हैं, तो यह एक अपवाद फेंकता है।

क्यू पीक () ऑपरेशन

पीक () ऑपरेशन का उपयोग पंक्ति के प्रमुख से किसी तत्व को हटाए बिना उसे पुनः प्राप्त करने के लिए किया जाता है। यदि यह सफलतापूर्वक जांच संचालन करता है, तो यह पंक्ति का मुख्य तत्व लौटाता है। अन्यथा यह शून्य मान देता है।

आइए इस कार्यक्षमता को प्रदर्शित करने के लिए एक सरल उदाहरण विकसित करें।

```

import java.util.*;

public class QueuePeekOperation {
    public static void main(String[] args) {

        Queue<String> queue = new LinkedList<>();
        queue.add("one");

        System.out.println(queue.peek());
        System.out.println(queue);
        queue.clear();
        System.out.println(queue.peek());
    }
}

```

उपरोक्त कार्यक्रम का आउटपुट

जब हम उपरोक्त प्रोग्राम चलाते हैं, तो हमें निम्न आउटपुट मिलेगा:

one

[one]

null

यदि हम खाली कतार पर पीक () मेथडको कॉल करने का प्रयास करते हैं, तो यह शून्य मान देता है, लेकिन एक पूर्व-धारणा

जावा क्यू कटेग्रीज़

जावा में, हम कई कतार कार्यान्वयन पा सकते हैं। डब्ल्यू मोटे तौर पर उन्हें निम्नलिखित दो प्रकारों में वर्गीकृत कर सकता है:

- बाउंडेड क्यूज़
- अनबाउंडेड क्यूज़

बंधी हुई पंक्तियां वे पंक्तियां होती हैं जो क्षमता से बंधी होती हैं, जिसका अर्थ है कि निर्माण के समय हमें पंक्ति का अधिकतम आकार प्रदान करने की आवश्यकता होती है। उदाहरण के लिए ऐरे ब्लॉकिंग पंक्ति।

असीमित पंक्ति ऐसी कतारें हैं जो क्षमता से बंधी नहीं हैं, जिसका अर्थ है कि हमें पंक्ति का आकार प्रदान नहीं करना चाहिए। उदाहरण के लिए लिंकड लिस्ट।

सभी क्यूज़ जो `java.util` पैकेज में उपलब्ध हैं, वे असीमित कतारें और क्यूज़ हैं जो हैं `java.util.concurrent` पैकेज में उपलब्ध बाउंडेड क्यूज़ हैं।

अन्य तरीकों से, डब्ल्यू उन्हें मोटे तौर पर निम्नलिखित दो प्रकारों में वर्गीकृत कर सकता है:

- ब्लॉकिंग क्यूज़
- नॉन-ब्लॉकिंग क्यूज़

ब्लॉकिंग क्यूज़ इंटरफ़ेस को लागू करने वाली सभी कतारें ब्लॉकिंग क्यू हैं और बाकी गैर-अवरुद्ध कतारें हैं।

ब्लॉकिंग क्यूज़ तब तक ब्लॉक करता है जब तक कि वह अपना काम पूरा नहीं कर लेता या टाइम आउट नहीं हो जाता, लेकिन नॉन-ब्लॉकिंग क्यू नहीं करता।

कुछ क्यू डेक्स हैं और कुछ क्यू प्रायोरिटी क्यू हैं।

ब्लॉकिंग क्यू ऑपरेशन

क्यू के दो प्रकार के ऑपरेशन के अलावा, ब्लॉकिंगक्यूज़ दो और रूपों का समर्थन करता है जैसा कि नीचे दिखाया गया है।

ऑपरेशन	थ्रोज़ एक्सेप्शन	स्पेशल वैल्यू	ब्लॉक्स	टाइम्स आउट
इन्सर्ट	ऐड (ई)	ऑफ़र (ई)	पुट (ई)	ऑफ़र (ई, टाइम, यूनिट)
रिमूव	रिमूव ()	पोल ()	टेक ()	पोल (टाइम, यूनिट)
एग्ज़ेमाइन	एलीमेंट ()	पीक ()	एन/ए	एन/ए

कुछ ऑपरेशन तब तक अवरुद्ध हैं जब तक कि यह अपना काम पूरा नहीं कर लेता है और अन्य समय समाप्त होने तक अवरुद्ध हो जाते हैं।

यह जावा में क्यू पर एक त्वरित राउंडअप है। हमें उम्मीद है कि ये जावा क्यू उदाहरण आपको क्यू संग्रह प्रोग्रामिंग के साथ आरंभ करने में मदद करेंगे।

ऐरे का उपयोग कैसे करें

आमतौर पर, एक ऐरे समान प्रकार के तत्वों का एक संग्रह है जिसमें एक सन्निहित स्मृति स्थान होता है।

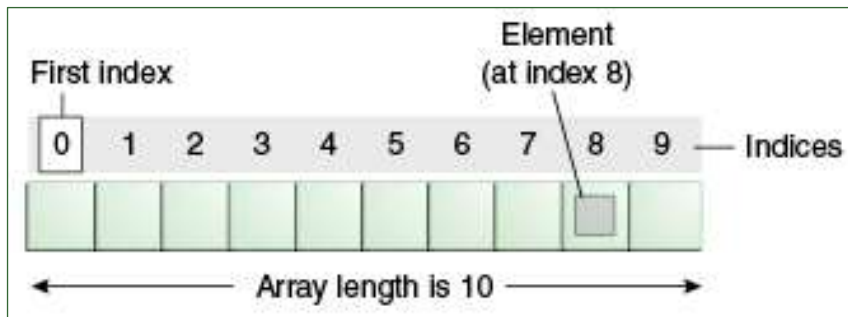
जावा ऐरे एक ऐसी वस्तु है जिसमें समान डेटा प्रकार के तत्व होते हैं। इसके अतिरिक्त, किसी ऐरे के तत्वों को एक सन्निहित स्मृति स्थान में संग्रहित किया जाता है। यह एक डेटा संरचना है जहां हम समान तत्वों को संग्रहित करते हैं। हम जावा ऐरे में केवल तत्वों का एक निश्चित सेट स्टोर कर सकते हैं।

जावा में ऐरे इंडेक्स-आधारित है, ऐरे का पहला तत्व 0 वें इंडेक्स, दूसरा तत्व पर संग्रहित है, दूसरा तत्व इंडेक्स 1 पर संग्रहित है और इसी तरह।

C/C++ के विपरीत, हम लेंथ मेंबर का उपयोग करके ऐरे की लंबाई प्राप्त कर सकते हैं। C/C++ में, हमें साइज़ ऑफ़ करने की आवश्यकता है ऑपरेटर का उपयोग करने की आवश्यकता है।

जावा में, ऐरे एक गतिशील रूप से उत्पन्न वर्ग का एक वस्तु है। जावा ऐरे ऑब्जेक्ट क्लास को इनहेरिट करती है, और सीरियल और साथ ही क्लोनेबल इंटरफेस को लागू करती है। हम जावा में एक ऐरे में प्राइमिटिव वैल्यू या ऑब्जेक्ट को संग्रहित कर सकते हैं। C/C++ की तरह, हम जावा में सिंगल डायमेंशनल या मल्टीडायमेंशनल ऐरेज़ भी बना सकते हैं।

इसके अलावा, जावा नामरहित ऐरे की सुविधा प्रदान करता है जो C/C++ में उपलब्ध नहीं है।



लाभ:

- कोड अनुकूलन: यह कोड को अनुकूलित बनाता है, हम डेटा को कुशलता से पुनर्प्राप्त या सॉर्ट कर सकते हैं।
- रैंडम एक्सेस: हम इंडेक्स पोजीशन पर स्थित कोई भी डेटा प्राप्त कर सकते हैं।

हानि:

- आकार सीमा: हम ऐरे में केवल निश्चित आकार के तत्वों को संग्रहित कर सकते हैं। यह रनटाइम पर अपना आकार नहीं बढ़ाता है। इस समस्या को हल करने के लिए, जावा में संग्रह ढांचे का उपयोग किया जाता है जो स्वचालित रूप से बढ़ता है।

जावा में ऐरे के प्रकार

ऐरे दो प्रकार के होते हैं।

- सिंगल डाइमेंशनल ऐरे
- मल्टीडाइमेंशनल ऐरे

जावा में सिंगल डाइमेंशनल ऐरे

जावा में ऐरे डिक्लेयर करने के लिए सिंटैक्स

```
dataType[] arr; (or)
```

```
dataType []arr; (or)
```

```
dataType arr[];
```

जावा में एक ऐरे का इंस्टेंशिएशन

```
arrayRefVar=new datatype[size];
```

जावा ऐरे का उदाहरण

आइए जावा ऐरे का सरल उदाहरण देखें, जहां हम ऐरे को घोषित करने, इंस्टेंट करने, इनिशियलाइज़ करने और ट्रैवर्स करने जा रहे हैं।

```
//Java Program to illustrate how to declare, instantiate, initialize
//and traverse the Java array.
```

```
class Testarray{
public static void main(String args[]){
int a[]=new int[5];//declaration and instantiation
a[0]=10;//initialization
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;
//traversing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

उपरोक्त कार्यक्रम का आउटपुट

```
20
70
40
50
```

जावा ऐरे की डिक्लेयरेशन, इनटेनशियेशन और इनिशिएलाइजेशन

हम जावा ऐरे को एक साथ डिक्लेयरेशन, इनटेनशियेशन और इनिशिएलाइज कर सकते हैं:

```
int a[]={33,3,4,5};//declaration, instantiation and initialization
```

आइए इस ऐरे को प्रिंट करने के लिए सरल उदाहरण देखें।

```
//Java Program to illustrate the use of declaration, instantiation
//and initialization of Java array in a single line
class Testarray1{
public static void main(String args[]){
int a[]={33,3,4,5};//declaration, instantiation and initialization
//printing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

उपरोक्त कार्यक्रम का आउटपुट

```
33
3
4
5
```

जावा ऐरे के लिए फॉर ईच - लूप

हम ईच लूप के लिए जावा ऐरे को प्रिंट भी कर सकते हैं। ईच लूप के लिए जावा ऐरे तत्वों को एक-एक करके प्रिंट करता है। यह एक चर में एक ऐरे तत्व रखता है, फिर लूप के शरीर को निष्पादित करता है।

ईच लूप के लिए सिंटैक्स नीचे दिया गया है:

```
for(data_type variable:array){
//body of the loop
}
```

आइए ईच लूप के लिए जावा ऐरे के तत्वों को प्रिंट करने का उदाहरण देखें।

```
//Java Program to print the array elements using for-each loop
class Testarray1{
public static void main(String args[]){
int arr[]={33,3,4,5};
//printing array using for-each loop
for(int i:arr)
System.out.println(i);
}}
```

उपरोक्त कार्यक्रम का आउटपुट

```
33
3
4
5
```

पासिंग ऐरे को जावा में एक विधि

हम जावा सरणी को मेथडमें पास कर सकते हैं ताकि हम किसी भी सरणी पर उसी तर्क का पुनः उपयोग कर सकें।

आइए एक मेथडका उपयोग करके किसी सरणी की न्यूनतम संख्या प्राप्त करने के लिए सरल उदाहरण देखें।

```
//Java Program to demonstrate the way of passing an array //to method.
class Testarray2{
//creating a method which receives an array as a parameter
static void min(int arr[]){
int min=arr[0];
for(int i=1;i<arr.length;i++)
if(min>arr[i])
min=arr[i];

System.out.println(min);
}
```



```
public static void main(String args[]){
int a[]={33,3,4,5};//declaring and initializing an array
min(a);//passing array to method
}}
```

उपरोक्त कार्यक्रम का आउटपुट

3

जावा में एनोनिमस ऐरे

जावा एक एनोनिमस ऐरे की सुविधा का समर्थन करता है, इसलिए आपको किसी ऐरे को मेथड में पास करते समय ऐरे डिक्लेयर करने की आवश्यकता नहीं है।

```
//Java Program to demonstrate the way of passing an anonymous array
//to method.
public class TestAnonymousArray{
//creating a method which receives an array as a
parameter static void printArray(int arr[]){
for(int i=0;i<arr.length;i++)
System.out.println(arr[i]);
}
public static void main(String args[]){
printArray(new int[]{10,22,44,66});//passing anonymous array to method
}}
```

उपरोक्त कार्यक्रम का आउटपुट

10
22
44
66

मेथड से रिटर्निंग ऐरे

हम जावा में मेथड से एक ऐरे भी रिटर्न कर सकते हैं।

```
//Java Program to return an array from the method
```

```
class TestReturnArray{
//creating method which returns an array
static int[] get(){
return new int[]{10,30,50,90,60};
}

public static void main(String args[]){
//calling method which returns an array
int arr[]=get();
//printing the values of an array
for(int i=0;i<arr.length;i++)
System.out.println(arr[i]);
}}
```

उपरोक्त कार्यक्रम का आउटपुट

```
10
30
50
90
60
```

ऐरे इंडेक्स आउट ऑफ बाउंड एक्सेप्शन

यदि ऐरे को नेगेटिव में, ऐरे साइज़ के बराबर या ऐरे को पार करते समय सरणी आकार से बड़ा है तो जावा वर्चुअल मशीन (जेवीएम) एक ऐरे इंडेक्स आउट ऑफ बाउंड एक्सेप्शन दिखाता है।

```
//Java Program to demonstrate the case of
//ArrayIndexOutOfBoundsException in a Java Array.
public class TestArrayException{
public static void main(String args[]){
int arr[]={50,60,70,80};
for(int i=0;i<=arr.length;i++){
System.out.println(arr[i]);
}
}}
```

उपरोक्त कार्यक्रम का आउटपुट

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
    at TestArrayException.main (TestArrayException.java:5)
50
60
70
80
```

जावा में मल्टीडाईमेंशनल ऐरे

ऐसे मामले में, डेटा को पंक्ति और स्तंभ आधारित इंडेक्स (मैट्रिक्स रूप के रूप में भी जाना जाता है) में संग्रहित किया जाता है।

जावा में मल्टीडाईमेंशनल ऐरे डिक्लेयर करने के लिए सिंटैक्स

```
dataType[][] arrayRefVar; (or)
dataType [][]arrayRefVar; (or)
dataType arrayRefVar[][]; (or)
dataType []arrayRefVar[];
```

जावा में इन्टेनशियेट मल्टीडाईमेंशनल ऐरे के लिए उदाहरण

```
int[][] arr=new int[3][3]; //3 row and 3 column
```

जावा में इनिशियलाइज़ मल्टीडाईमेंशनल ऐरे के लिए उदाहरण

```
arr[0][0]=1;
arr[0][1]=2;
arr[0][2]=3;
arr[1][0]=4;
arr[1][1]=5;
arr[1][2]=6;
arr[2][0]=7;
arr[2][1]=8;
arr[2][2]=9;
```

जावा में मल्टीडाईमेंशनल का उदाहरण

आइए 2 डाईमेंशनल ऐरे को डिक्लेयर करने, इन्टेनशियेट करने, इनिशियलाइज़ करने और प्रिंट करने के लिए सरल उदाहरण देखें।

```
//Java Program to illustrate the use of multidimensional array
```

```

class Testarray3{
public static void main(String args[]){
//declaring and initializing 2D array
int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
//printing 2D array
for(int i=0;i<3;i++){
    for(int j=0;j<3;j++){
        System.out.print(arr[i][j]+" ");
    }
    System.out.println();
}
}}

```

उपरोक्त प्रोग्राम का आउटपुट

```

1 2 3
2 4 5
4 4 5

```

जावा में जैगड ऐरे(Jagged Array in Java)

यदि हम 2 डी ऐरे में विषम संख्या में स्तंभ बना रहे हैं, तो इसे जैगड ऐरे के रूप में जाना जाता है। दूसरे शब्दों में, यह विभिन्न स्तंभों के साथ सारणियों की एक सारणी है।

```

//Java Program to illustrate the jagged array
class TestJaggedArray{
    public static void main(String[] args){
        //declaring a 2D array with odd columns
        int arr[][] = new int[3][];
        arr[0] = new int[3];
        arr[1] = new int[4];
        arr[2] = new int[2];
        //initializing a jagged array
        int count = 0;
        for (int i=0; i<arr.length; i++)

```

```

        for(int j=0; j<arr[i].length; j++)
            arr[i][j] = count++;

//printing the data of a jagged array
for (int i=0; i<arr.length; i++){
    for (int j=0; j<arr[i].length; j++){
        System.out.print(arr[i][j]+" ");
    }
    System.out.println();//new line
}
}
}

```

उपरोक्त कार्यक्रम का आउटपुट

```

0 1 2
3 4 5 6
7 8

```

जावा ऐरे का वर्ग नाम क्या है?

जावा में, एक ऐरे एक ऑब्जेक्ट है। ऐरे ऑब्जेक्ट के लिए, एक प्रॉक्सी क्लास बनाया जाता है जिसका नाम गेटक्लास () गेटनेम () मेथड द्वारा वस्तु पर प्राप्त किया जा सकता है।

```

//Java Program to get the class name of array in
Java class Testarray4{
public static void main(String args[]){
//declaration and initialization of array
int arr[]={4,4,5};
//getting the class name of Java array
Class c=arr.getClass();
String name=c.getName();
//printing the class name of Java array
System.out.println(name);
}}

```

उपरोक्त कार्यक्रम का आउटपुट

I

एक जावा ऐरे का प्रतिलिपि बनाया जाना

हम सिस्टम क्लास के ऐरेकॉपी () मेथड द्वारा एक ऐरे को दूसरे में कॉपी कर सकते हैं।

ऐरेकॉपी मेथड का सिंटैक्स

```
public static void arraycopy(
Object src, int srcPos, Object dest, int destPos, int length
)
```

जावा में एक ऐरे की प्रतिलिपि बनाने का उदाहरण

```
//Java Program to copy a source array into a destination array in
Java class TestArrayCopyDemo {
    public static void main(String[] args) {
        //declaring a source array
        char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e',
            'i', 'n', 'a', 't', 'e', 'd' };
        //declaring a destination array
        char[] copyTo = new char[7];
        //copying array using System.arraycopy() method
        System.arraycopy(copyFrom, 2, copyTo, 0, 7);
        //printing the destination array
        System.out.println(String.valueOf(copyTo));
    }
}
```

उपरोक्त कार्यक्रम का आउटपुट

caffein

जावा में एक ऐरे का क्लोनिंग करना

चूंकि, जावा ऐरे क्लोनेबल इंटरफेस को लागू करता है, हम जावा ऐरे का क्लोन बना सकते हैं। यदि हम सिंगल-डायमेंशनल ऐरे का क्लोन बनाते हैं, तो यह जावा ऐरे की गहरी प्रतिलिपि बनाता है। इसका मतलब है, यह वास्तविक मूल्य की नकल करेगा। लेकिन, अगर हम एक मल्टी-डायमेंशनल ऐरे का क्लोन बनाते हैं, तो यह जावा ऐरे की उथली प्रतिलिपि बनाता है जिसका अर्थ है कि यह संदर्भों की प्रतिलिपि बनाता है।

```
//Java Program to clone the array
class Testarray1{
public static void main(String args[]){
int arr[]={33,3,4,5};
System.out.println("Printing original array:");
for(int i:arr)
System.out.println(i);

System.out.println("Printing clone of the array:");
int carr[]=arr.clone();
for(int i:carr)
System.out.println(i);

System.out.println("Are both equal?");
System.out.println(arr==carr);

}}
```

उपरोक्त कार्यक्रम का आउटपुट

Printing original array:

33

3

4

5

Printing clone of the array:

33

3

4

5

Are both equal?

false

जावा में 2 मैट्रिक्स का योग

आइए एक सरल उदाहरण देखें जो दो मैट्रिक्स जोड़ता है।

```
//Java Program to demonstrate the addition of two matrices in
Java class Testarray5{
public static void main(String args[]){
//creating two matrices
int a[][]={{1,3,4},{3,4,5}};
int b[][]={{1,3,4},{3,4,5}};

//creating another matrix to store the sum of two
matrices int c[][]=new int[2][3];

//adding and printing addition of 2 matrices
for(int i=0;i<2;i++){
for(int j=0;j<3;j++){
c[i][j]=a[i][j]+b[i][j];
System.out.print(c[i][j]+" ");
}
System.out.println();//new line
}
}}
```

उपरोक्त कार्यक्रम का आउटपुट

```
2 6 8
6 8 10
```

जावा में 2 मैट्रिक्स का मल्टिप्लिकेशन

मैट्रिक्स गुणन के मामले में, पहले मैट्रिक्स के एक-पंक्ति तत्व को सभी से गुणा किया जाता है दूसरे के कॉलम मैट्रिक्स जिसे नीचे दिए गए इमेज से समझा जा सकता है।

$$\text{Matrix1} \begin{Bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{Bmatrix} \quad \text{Matrix2} \begin{Bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{Bmatrix}$$

$$\begin{matrix} \text{Matrix1} \\ * \\ \text{Matrix2} \end{matrix} \left\{ \begin{array}{ccc} 1^*1+1^*2+1^*3 & 1^*1+1^*2+1^*3 & 1^*1+1^*2+1^*3 \\ 2^*1+2^*2+2^*3 & 2^*1+2^*2+2^*3 & 2^*1+2^*2+2^*3 \\ 3^*1+3^*2+3^*3 & 3^*1+3^*2+3^*3 & 3^*1+3^*2+3^*3 \end{array} \right\}$$

$$\begin{matrix} \text{Matrix1} \\ * \\ \text{Matrix2} \end{matrix} \left\{ \begin{array}{ccc} 6 & 6 & 6 \\ 12 & 12 & 12 \\ 18 & 18 & 18 \end{array} \right\}$$

आइए 3 पंक्तियों और 3 स्तंभों के दो आव्यूहों को गुणा करने के लिए एक सरल उदाहरण देखें।

```
//Java Program to multiply two matrices
public class MatrixMultiplicationExample{
public static void main(String args[]){
//creating two matrices
int a[][]={{1,1,1},{2,2,2},{3,3,3}};
int b[][]={{1,1,1},{2,2,2},{3,3,3}};

//creating another matrix to store the multiplication of two
matrices int c[][]=new int[3][3]; //3 rows and 3 columns

//multiplying and printing multiplication of 2 matrices
for(int i=0;i<3;i++){
for(int j=0;j<3;j++){
c[i][j]=0;
for(int k=0;k<3;k++)
{
c[i][j]+=a[i][k]*b[k][j];
}
}
}
System.out.print(c[i][j]+" "); //printing matrix element
}
}
System.out.println();//new line
}
}}
```

उपरोक्त कार्यक्रम का आउटपुट

6 6 6

12 12 12

18 18 18

जावा में सॉर्टिंग

डेटा को सॉर्ट करने का अर्थ है इसे एक निश्चित क्रम में व्यवस्थित करना, अक्सर एक सरणी जैसी डेटा संरचना में। आप विभिन्न ऑर्डरिंग मानदंड का उपयोग कर सकते हैं, सामान्य संख्याएं कम से कम सबसे बड़ी या इसके विपरीत, या स्ट्रिंग्स को लेक्सिकोग्राफिक रूप से सॉर्ट करना।

विभिन्न सॉर्टिंग एल्गोरिदम हैं, और वे सभी समान रूप से कुशल नहीं हैं। हम विश्लेषण करेंगे उनके समय उनकी तुलना करने के लिए जटिलता और यह देखने के लिए कि कौन सबसे अच्छा प्रदर्शन करता है।

आप यहां सीखेंगे एल्गोरिदम की सूची किसी भी तरह से संपूर्ण नहीं है, लेकिन हमने उनमें से कुछ को संकलित किया है आरंभ करने में आपकी सहायता करने के लिए सबसे आम और सबसे कुशल:

- बबल सॉर्ट
- सम्मिलन सॉर्ट
- सिलेक्शन सॉर्ट
- मर्ज सॉर्ट
- हीप सॉर्ट
- क्विक सॉर्ट
- जावा में सॉर्टिंग।

सॉर्ट इन्सर्ट कैसे करें

- जावा इंसर्शन सॉर्ट, हम सभी पूर्व तत्वों से किसी भी सूचकांक पर मूल्य की तुलना करते हैं जब तक कि सभी कम मूल्य नहीं मिलते।
- फिर हम मान को उस सूचकांक पर रखते हैं जिसके पहले कोई कम मान नहीं है।
- उपरोक्त दो चरणों को अंतिम अनुक्रमणिका के लिए पुनरावृत्त रूप से किया जाता है, अंत में हमारे पास पूर्णांकों की एक क्रमबद्ध सरणी होती है।

इंसर्शन सॉर्ट एल्गोरिदम का उदाहरण

आइए एक उदाहरण के साथ इंसर्शन सॉर्ट एल्गोरिदम को समझते हैं। मान लें कि हमारे पास एक क्रमबद्ध सरणी है [5, 4, 14, 2, 8]

- पहली अनुक्रमणिका पुनरावृत्ति: पहली अनुक्रमणिका पर मान = 4, जो 5 से कम है, इसलिए सरणी [5, 5, 14, 2, 8] बन जाती है, जैसे ही हम शुरुआत में पहुँचते हैं हम मान को 0 सूचकांक पर रखते हैं और ऐरे [4] बन जाती है। 5, 14, 2, 8]
- दूसरी सूचकांक पुनरावृत्ति: दूसरे सूचकांक पर मान = 14 जो 5 से अधिक है, इसलिए सरणी को वैसे ही छोड़ दें। अब सरणी = [4, 5, 14, 2, 8]
- तीसरी सूचकांक पुनरावृत्ति: तीसरे सूचकांक पर मान = 2 जो 14 से छोटा है, इसलिए सरणी [4, 5, 14, 14, 8] बन जाती है, फिर 2 5 से छोटी होती है, इसलिए सरणी बन जाती है [4, 5, 5, 14, 8]। फिर से 2, 4 से छोटा है, इसलिए सरणी [4, 4, 5, 14, 8] बन जाती है। जैसे ही हम ऐरे की शुरुआत में पहुँचे, हम 2 को 0वें इंडेक्स पर रखते हैं और ऐरे [2, 4, 5, 14, 8] बन जाता है।
- चौथी सूचकांक पुनरावृत्ति: चौथे सूचकांक पर मान = 8, इसलिए सरणी [2, 4, 5, 14, 14] बन जाती है, फिर 8 5 से बड़ा होता है, इसलिए 8 को 14 वें स्थान पर रखें और सरणी बन जाती है [2, 4, 5, 8, 14]। और हमारी सरणी अब क्रमबद्ध है।

जावा इंसर्शन सॉर्ट (Java Insertion Sort)

मुख्य बिंदु जावा प्रोग्राम में इंसर्शन सॉर्ट का कार्यान्वयन लिखते समय याद रखें:

- ऐरे के अंतिम तत्व के लिए दूसरे तत्व से प्रारंभ करें, इसलिए लूप के लिए उपयोग करें।
- जब हम इंडेक्स वैल्यू को बीच में बदलते हैं तो इसे खोने से बचने के लिए वैल्यू को दूसरे वेरिएबल में स्टोर करें।
- हमें मूल्यों को तब तक बदलते रहने की आवश्यकता है जब तक कि हम 0वें सूचकांक पर न हों या हमें पहले का मूल्य अधिक न मिल जाए, इसलिए हम इसके लिए थोड़ी देर के लूप का उपयोग कर सकते हैं।

उपरोक्त उदाहरण और मुख्य बिंदुओं के आधार पर जावा में इंसर्शन सॉर्ट का कार्यान्वयन यहां दिया गया है।

```
package com.journaldev.test;
```

```
import java.util.Arrays;
```

```
public class InsertionSort {
```

```
    public static void main(String[] args) {
        int A[] = new int[10];
        populateArray(A);
        System.out.println("Before Sorting: ");
        printArray(A);
        // sort the array
        insertionSort(A);
        System.out.println("\nAfter Sorting: ");
    }
}
```

```
        printArray(A);
    }

    /**
     * This method will sort the integer array using insertion sort in java
    algorithm
     *
     * @param arr
     */
    private static void insertionSort(int[] arr) {
        for (int i = 1; i < arr.length; i++) {
            int valueToSort = arr[i];
            int j = i;
            while (j > 0 && arr[j - 1] > valueToSort) {
                arr[j] = arr[j - 1];
                j--;
            }
            arr[j] = valueToSort;
        }
    }

    public static void printArray(int[] B) {
        System.out.println(Arrays.toString(B));
    }

    public static void populateArray(int[] B) {
        for (int i = 0; i < B.length; i++) {
            B[i] = (int) (Math.random() * 100);
        }
    }
}
```

यादृच्छिक संख्या जनरेटर कोड का उपयोग किया गया है और उपरोक्त कार्यक्रम के लिए आउटपुट है:

Before Sorting:

```
[57, 90, 80, 48, 35, 91, 1, 83, 32, 53]
```

After Sorting:

```
[1, 32, 35, 48, 53, 57, 80, 83, 90, 91]
```

इंसर्शन सॉर्ट कोम्प्लेक्सिटी (Insertion Sort complexity)

इंसर्शन सॉर्ट की जटिलता सबसे अच्छी स्थिति के लिए $O(n)$ है (पहले से ही क्रमबद्ध सरणी) और सबसे खराब स्थिति के लिए $O(n^2)$ (रिवर्स ऑर्डर में क्रमबद्ध)।

जावा में इंसर्शन सॉर्ट छोटे आकार के ऐरे के लिए एक अच्छा विकल्प है और जब आप जानते हैं कि मान अधिकतर सॉर्ट किए जाएंगे। उदाहरण के लिए, आकार 100 की एक ऐरे को सॉर्ट करना जहाँ आप जानते हैं कि सभी मान 1 से 10 के बीच होंगे।

सॉर्ट मर्ज कैसे करें

सॉर्ट मर्ज सबसे कुशल सॉर्टिंग तकनीकों में से एक है और यह "डिवाइड एंड कांकर" प्रतिमान पर आधारित है।

एल्गोरिदम

सॉर्ट मर्ज एक "डिवाइड एंड कांकर" एल्गोरिदम है जिसमें हम पहले समस्या को उप-समस्याओं में विभाजित करते हैं। जब उप-समस्याओं का समाधान तैयार हो जाता है, तो हम समस्या का अंतिम समाधान प्राप्त करने के लिए उन्हें एक साथ जोड़ते हैं।

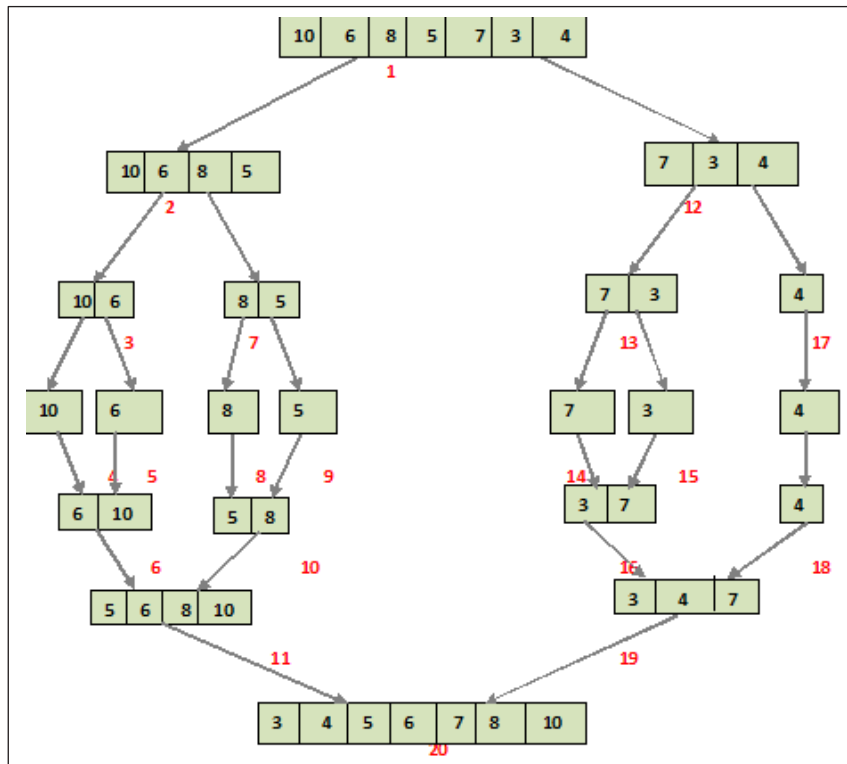
यह एल्गोरिदम में से एक है जिसे रिकर्सन का उपयोग करके आसानी से कार्यान्वित किया जा सकता है क्योंकि हम मुख्य समस्या के बजाय उप-समस्याओं से निपटते हैं।

एल्गोरिदम को निम्नलिखित 2 चरण प्रक्रिया के रूप में वर्णित किया जा सकता है:

- डिवाइड: इस चरण में, हम इनपुट सरणी को 2 हिस्सों में विभाजित करते हैं, पिवट सरणी का मध्य बिंदु होता है। यह चरण सभी आधे सरणियों के लिए पुनरावर्ती रूप से किया जाता है जब तक कि विभाजित करने के लिए और अधिक आधे ऐरे न हों।
- कांकर: इस चरण में, हम विभाजित ऐरे को नीचे से ऊपर तक सॉर्ट और मर्ज करते हैं और क्रमबद्ध ऐरे प्राप्त करते हैं।

निम्न आरेख एक उदाहरण सरणी {10, 6, 8, 5, 7, 3, 4} के लिए संपूर्ण मर्ज सॉर्ट प्रक्रिया दिखाता है।

यदि हम डायग्राम पर करीब से नज़र डालते हैं, तो हम देख सकते हैं कि आकार 1 होने तक सरणी को दो हिस्सों में पुनरावर्ती रूप से विभाजित किया जाता है। एक बार आकार 1 हो जाने के बाद, मर्ज प्रक्रिया क्रिया में आ जाती है और सॉर्ट करते समय ऐरे को वापस मर्ज करना शुरू कर देती है।



कार्यान्वयन

कार्यान्वयन के लिए, हम एक मर्ज सॉर्ट फ़ंक्शन लिखेंगे जो इनपुट ऐरे और इसकी लंबाई को मापदंडों के रूप में लेता है। यह एक पुनरावर्ती कार्य होगा इसलिए हमें आधार और पुनरावर्ती स्थितियों की आवश्यकता है।

बुनियादी स्थिति जांचती है कि क्या ऐरे की लंबाई 1 है और यह बस वापस आ जाएगी। शेष मामलों के लिए, पुनरावर्ती कॉल निष्पादित की जाएगी।

पुनरावर्ती मामले के लिए, हम मध्य सूचकांक प्राप्त करते हैं और दो अस्थायी ऐरे बनाते हैं `l[]` और `r[]`। मर्जसॉर्ट फ़ंक्शन को तब दोनों उप-सारणी के लिए पुनरावर्ती रूप से कहा जाता है:

```
public static void mergeSort(int[] a, int n) {
    if (n < 2) {
        return;
    }
    int mid = n / 2;
    int[] l = new int[mid];
    int[] r = new int[n - mid];

    for (int i = 0; i < mid; i++) {
```

```

        l[i] = a[i];
    }
    for (int i = mid; i < n; i++) {
        r[i - mid] = a[i];
    }
    mergeSort(l, mid);
    mergeSort(r, n - mid);

    merge(a, l, r, mid, n - mid);
}

```

फिर हम मर्ज फ़ंक्शन को कॉल करते हैं जो इनपुट और दोनों उप-सारणी और दोनों उप सारणी के शुरुआती और अंत सूचकांक लेता है।

मर्ज फ़ंक्शन दोनों उप-सारणी के तत्वों की एक-एक करके तुलना करता है और छोटे तत्व को इनपुट सारणी में रखता है।

जब हम किसी एक उप-सारणी के अंत तक पहुँचते हैं, तो दूसरे सारणी के शेष तत्व होते हैं इनपुट सारणी में कॉपी किया गया जिससे हमें अंतिम क्रमबद्ध सारणी मिल गई:

```

public static void merge(
    int[] a, int[] l, int[] r, int left, int right) {
    int i = 0, j = 0, k = 0;
    while (i < left && j < right) {
        if (l[i] <= r[j]) {
            a[k++] = l[i++];
        }
        else {
            a[k++] = r[j++];
        }
    }
    while (i < left) {
        a[k++] = l[i++];
    }
    while (j < right) {

```

```

        a[k++] = r[j++];
    }
}

```

कार्यक्रम के लिए इकाई परीक्षण:

```

@Test
public void positiveTest() {
    int[] actual = { 5, 1, 6, 2, 3, 4 };
    int[] expected = { 1, 2, 3, 4, 5, 6 };
    MergeSort.mergeSort(actual, actual.length);
    assertEquals(expected, actual);
}

```

जटिलता

चूंकि मर्ज सॉर्ट एक पुनरावर्ती एल्गोरिथ्म है, टाइम जटिलता को निम्नलिखित पुनरावर्ती संबंध के रूप में व्यक्त किया जा सकता है:

$$T(n) = 2 T(n/2) + O(n)$$

$2 T(n/2)$ समय से मेल खाता है पूरे सरणी को मर्ज करने के लिए उप-सारणी और ओ (एन) टाइम को सॉर्ट करने के लिए आवश्यक है।

हल होने पर, समय जटिलता $O(n \log n)$ पर आ जाएगी।

यह सबसे खराब, औसत और सर्वोत्तम मामले के लिए सही है क्योंकि यह हमेशा सरणी को दो में विभाजित करेगा और फिर विलय करेगा।

एल्गोरिदम की स्पेस कोम्प्लिसिटी ओ (एन) है क्योंकि हम प्रत्येक रिकर्सिव कॉल में अस्थायी ऐरे बना रहे हैं।

क्विक सॉर्ट कैसे करें

क्विकसॉर्ट एक सॉर्टिंग एल्गोरिदम है, जो डिवाइड-एंड-कॉन्कर सिद्धांत का लाभ उठा रहा है। इसकी औसत $O(n \log n)$ जटिलता है और यह विशेष रूप से बड़े डेटा वॉल्यूम के लिए सबसे अधिक उपयोग किए जाने वाले सॉर्टिंग एल्गोरिदम में से एक है।

यह याद रखना महत्वपूर्ण है कि क्विकसॉर्ट एक स्थिर एल्गोरिथ्म नहीं है। एक स्थिर सॉर्टिंग एल्गोरिथ्म एक एल्गोरिथ्म है जहाँ समान मान वाले तत्व क्रमबद्ध आउटपुट में उसी क्रम में दिखाई देते हैं जैसे वे इनपुट सूची में दिखाई देते हैं।

इनपुट सूची को पिवट नामक तत्व द्वारा दो उप-सूचियों में विभाजित किया गया है; एक उप-सूची जिसमें पिवट से अधिक तत्व नहीं हैं और दूसरा पिवट से अधिक तत्वों वाला है। यह प्रक्रिया प्रत्येक उप-सूची के लिए दोहराई जाती है।

अंत में, सभी सॉर्ट की गई उप-सूचियां अंतिम आउटपुट बनाने के लिए विलीन हो जाती हैं।

एल्गोरिदम चरण

- हम सूची से एक तत्व चुनते हैं, जिसे पिवट कहा जाता है। हम इसका उपयोग सूची को दो उप-सूचियों में विभाजित करने के लिए करेंगे।
- हम धुरी के चारों ओर सभी तत्वों को फिर से व्यवस्थित करते हैं - छोटे मूल्य वाले को इसके आगे रखा जाता है, और इसके बाद धुरी से बड़े सभी तत्वों को रखा जाता है। इस चरण के बाद, धुरी अपनी अंतिम स्थिति में है। यह महत्वपूर्ण विभाजन चरण है।
- हम उपरोक्त चरणों को पिवट के बाएँ और दाएँ दोनों उप-सूचियों पर पुनरावर्ती रूप से लागू करते हैं।

हर डिवाइड एंड कोंक़र एप्रोच की तरह क्विकसोर्ट स्वाभाविक रूप से एक पुनरावर्ती एल्गोरिथ्म है

आइए इस एल्गोरिथ्म को बेहतर ढंग से समझने के लिए एक सरल उदाहरण लेते हैं।

```
Arr[] = {5, 9, 4, 6, 5, 3}
```

- आइए मान लें कि हम सादगी के लिए धुरी के रूप में 5 चुनते हैं
- हम पहले 5 से कम के सभी तत्वों को एरे की पहली स्थिति में रखेंगे: {3, 4, 5, 6, 5, 9}
- फिर हम इसे बाईं उप-सारणी {3,4} के लिए 3 को धुरी के रूप में लेते हुए दोहराएंगे,
- 3 से कम का कोई तत्व नहीं है
- हम पिवट के दाईं ओर उप-सारणी पर क्विकसॉर्ट लागू करते हैं, यानी {4}
- इस उप-सारणी में केवल एक क्रमबद्ध तत्व होता है
- हम मूल सारणी के दाहिने हिस्से के साथ जारी रखते हैं, {6, 5, 9} जब तक हमें अंतिम आदेशित सारणी नहीं मिल जाती

ऑप्टीमल पिवट को चुनना

क्विकसॉर्ट में महत्वपूर्ण बिंदु सबसे अच्छा पिवट चुनना है। बेशक, मध्य तत्व सबसे अच्छा है, क्योंकि यह सूची को दो समान उप-सूचियों में विभाजित करेगा।

लेकिन एक अनियंत्रित सूची से मध्य तत्व को खोजना कठिन और समय लेने वाला है, इसलिए हम पहले तत्व, अंतिम तत्व, माध्यिका या किसी अन्य यादृच्छिक तत्व को धुरी के रूप में लेते हैं।

जावा में कार्यान्वयन

पहली विधि क्विकसॉर्ट () है जो पैरामीटर के रूप में सरणी को क्रमबद्ध करने के लिए लेती है, पहली और अंतिम अनुक्रमणिका। सबसे पहले, हम सूचकांकों की जांच करते हैं और केवल तभी जारी रखते हैं जब क्रमबद्ध करने के लिए अभी भी तत्व हों।

हम सॉर्ट किए गए पिवट का इंडेक्स प्राप्त करते हैं और इसका उपयोग रिकर्सिवली कॉल पार्टिशन () मेथड के साथ क्विकसॉर्ट () मेथड के समान मापदंडों के साथ करते हैं, लेकिन अलग-अलग इंडेक्स के साथ:

```
public void quickSort(int arr[], int begin, int end) {
    if (begin < end) {
        int partitionIndex = partition(arr, begin, end);

        quickSort(arr, begin, partitionIndex-1);
        quickSort(arr, partitionIndex+1, end);
    }
}
```

आइए विभाजन () मेथड के साथ जारी रखें। सादगी के लिए, यह फ़ंक्शन अंतिम तत्व को धुरी के रूप में लेता है। फिर, प्रत्येक तत्व की जांच करता है और यदि उसका मान छोटा है तो उसे पिवट से पहले स्वैप कर देता है।

विभाजन के अंत तक, धुरी से कम के सभी तत्व इसके बाईं ओर होते हैं और धुरी से बड़े सभी तत्व इसके दाईं ओर होते हैं। धुरी अपनी अंतिम क्रमबद्ध स्थिति में है और फ़ंक्शन इस स्थिति को लौटाता है:

```
private int partition(int arr[], int begin, int end) {
    int pivot = arr[end];
    int i = (begin-1);

    for (int j = begin; j < end; j++) {
        if (arr[j] <= pivot) {
            i++;

            int swapTemp = arr[i];
            arr[i] = arr[j];
            arr[j] = swapTemp;
        }
    }

    int swapTemp = arr[i+1];
    arr[i+1] = arr[end];
```

```

arr[end] = swapTemp;

return i+1;
}

```

एल्गोरिथम(Algorithm) विश्लेषण

टाइम कोम्प्लेक्सिटी (Time Complexity)

सर्वोत्तम स्थिति में, एल्गोरिथम सूची को दो समान आकार की उप-सूचियों में विभाजित करेगा। तो, पूर्ण एन-आकार की सूची के पहले पुनरावृत्ति को ओ (एन) की आवश्यकता होती है। शेष दो उप-सूचियों को $n/2$ तत्वों के साथ क्रमबद्ध करने पर प्रत्येक $2 * O(n/2)$ लेता है। परिणामस्वरूप, क्विकसॉर्ट एल्गोरिथम में $O(n \log n)$ की जटिलता है।

सबसे खराब स्थिति में, एल्गोरिथम प्रत्येक पुनरावृत्ति में केवल एक तत्व का चयन करेगा, इसलिए $O(n) + O(n-1) + \dots + O(1)$, कौन O के बराबर है (n^2) ।

औसतन क्विकसॉर्ट में $O(n \log n)$ जटिलता होती है, जो इसे बड़े डेटा वॉल्यूम के लिए उपयुक्त बनाती है।

क्विकसॉर्ट बनाम मर्जसॉर्ट (QuickSort vs MergeSort)

आइए चर्चा करें कि किन मामलों में हमें मर्जसॉर्ट पर क्विकसॉर्ट को चुनना चाहिए।

हालाँकि, क्विकसॉर्ट और मर्जसॉर्ट दोनों में $O(n \log n)$ की औसत समय जटिलता है, Quicksort पसंदीदा एल्गोरिथम है, क्योंकि इसमें $O(\log(n))$ स्थान जटिलता है। दूसरी ओर, मर्जसॉर्ट को $O(n)$ अतिरिक्त संग्रहण की आवश्यकता होती है, जो इसे सरणियों के लिए काफी महंगा बनाता है।

जल्दी से सुलझाएं इसके संचालन के लिए विभिन्न सूचकांकों तक पहुँचने की आवश्यकता होती है, लेकिन यह पहुँच लिंकड सूचियों में सीधे संभव नहीं है, क्योंकि कोई निरंतर ब्लॉक नहीं हैं; इसलिए एक तत्व तक पहुँचने के लिए हमें लिंक की गई सूची की शुरुआत से प्रत्येक नोड के माध्यम से पुनरावृत्ति करनी होगी। साथ ही, लिंकडलिस्ट के लिए अतिरिक्त स्थान के बिना मर्जसॉर्ट लागू किया गया है।

ऐसे मामले में, क्विकसॉर्ट और मर्जसॉर्ट के लिए ओवरहेड वृद्धि को आमतौर पर प्राथमिकता दी जाती है।

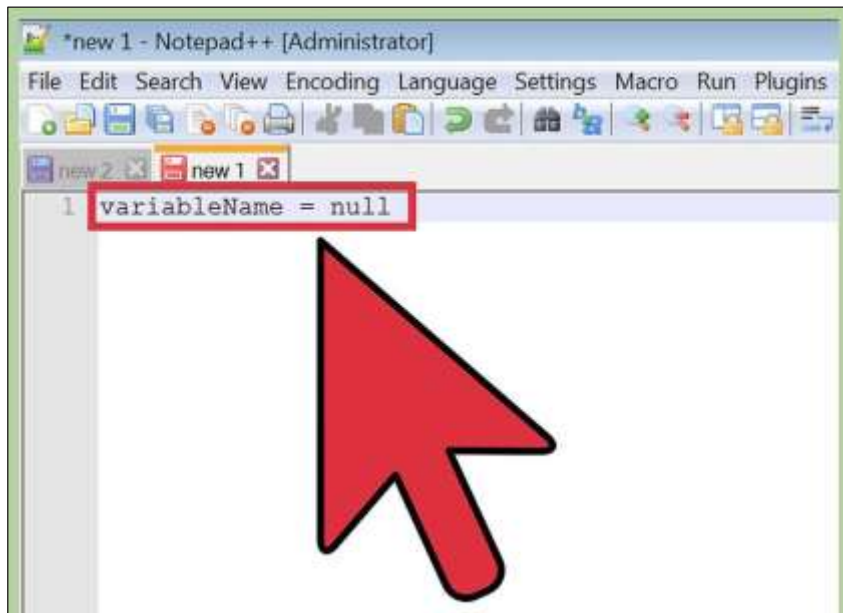
मिसलेनियस एप्लीकेशन्स

जावा के विविध अनुप्रयोग हैं जिनमें प्रिंटिंग ऐरे, चेकिंग नल, कमांड प्रॉम्प्ट का उपयोग करके संकलन करना, प्रतिशत की गणना करना, ऐरे साइज़ ढूंढना, योग निकालना, तिथियों की तुलना करना आदि शामिल हैं। यह अध्याय जावा के विविध अनुप्रयोगों में गहराई से विषय को समझने के लिए है।

जावा में नल(Null) को चेक कैसे करें

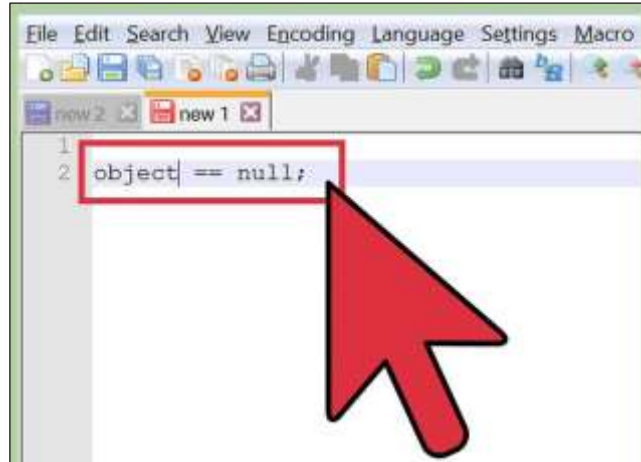
एक नल(Null) इंगित करता है कि एक चर किसी ऑब्जेक्ट को इंगित नहीं करता है और कोई मूल्य नहीं रखता है। कोड के एक टुकड़े में एक नल की जाँच करने के लिए आप एक बुनियादी ' इफ ' स्टेटमेंट का उपयोग कर सकते हैं। नल का उपयोग आमतौर पर किसी चीज के न होने को दर्शाने या सत्यापित करने के लिए किया जाता है। उस संदर्भ में, इसका उपयोग कोड के भीतर अन्य प्रक्रियाओं को शुरू करने या रोकने के लिए एक शर्त के रूप में किया जा सकता है।

भाग 1. जावा में नल(Null) की जाँच करना



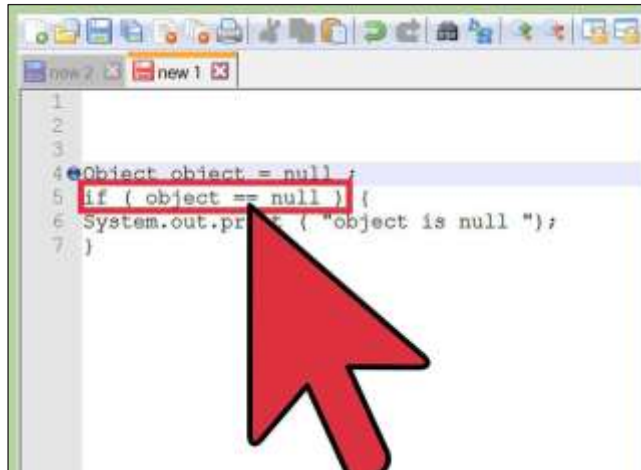
1. एक चर परिभाषित करने के लिए "=" का प्रयोग करें। एक एकल "=" का उपयोग एक चर घोषित करने और उसे एक मान निर्दिष्ट करने के लिए किया जाता है। आप इसे शून्य के लिए एक चर सेट करने के लिए उपयोग कर सकते हैं।

- "0" और नल(null) का मान समान नहीं है और अलग-अलग व्यवहार करेगा।
- वेरिएबल नेम(variableName) = नल (null);



2. एक चर का मान जाँच करने के लिए "==" का प्रयोग करें। एक "==" का उपयोग यह जांचने के लिए किया जाता है कि दोनों तरफ के दो मान बराबर हैं। यदि आप "=" के साथ एक चर को शून्य पर सेट करते हैं, तो जांच कर रहे हैं कि चर शून्य के बराबर है, सत्य वापस आ जाएगा।

- वेरिएबल नेम(variableName) == नल (null);
- आप "!=" का उपयोग यह जांचने के लिए भी कर सकते हैं कि कोई मान बराबर नहीं है।

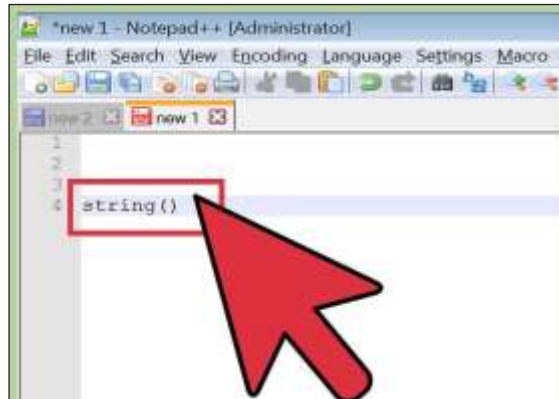


3. शून्य के लिए एक शर्त बनाने के लिए " इफ(if) " कथन का प्रयोग करें। व्यंजक का परिणाम एक बूलियन (सत्य या असत्य) मान होगा। बयान आगे क्या करता है, इसके लिए आप एक शर्त के रूप में बूलियन मान का उपयोग कर सकते हैं।

- उदाहरण के लिए, यदि मान शून्य है, तो "ऑब्जेक्ट शून्य है" टेक्स्ट प्रिंट करें। यदि "==" नहीं मिलता है चर शून्य होने के लिए, तो यह शर्त को छोड़ देगा या एक अलग रास्ता अपना सकता है।

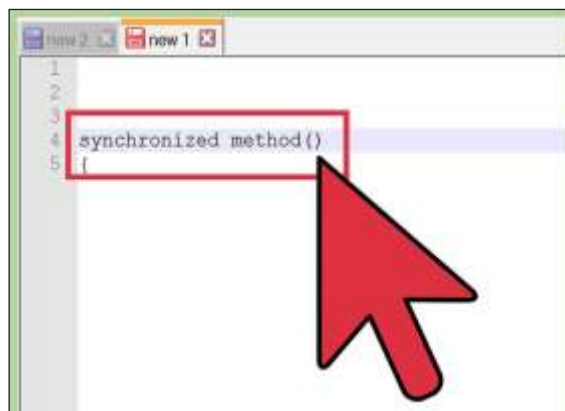
```
Object object = null ;
if ( object == null ) {
System.out.print ( "object is null " );
}
```

भाग 2. नल चेक(Null Check) का उपयोग करना

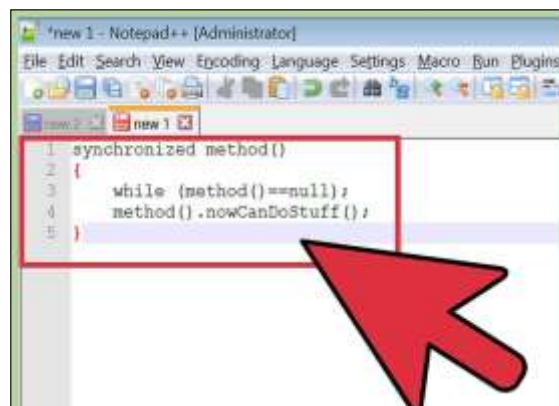


1. अज्ञात मान के रूप में शून्यका उपयोग करें। किसी भी असाइन किए गए मान के बदले में डिफॉल्ट के रूप में नल का उपयोग करना आम बात है।

- स्ट्रिंग() का अर्थ है कि जब तक यह वास्तव में उपयोग नहीं किया जाता है टैब तक मान शून्य है।



2. एक प्रक्रिया को समाप्त करने के लिए एक शर्त के रूप में शून्य का उपयोग करें। एक शून्य मान रिटर्न करने का उपयोग लूप के अंत को ट्रिगर करने या किसी प्रक्रिया को तोड़ने के लिए किया जा सकता है। यह आमतौर पर जब कुछ गलत हो गया हो या एक अवांछित स्थिति हिट हो गई हो तो किसी एरर या एक्सेप्शन को फेंकने के लिए उपयोग किया जाता है।



3. असंचित अवस्था को इंगित करने के लिए अशक्त का प्रयोग करें। इसी तरह, नल का उपयोग ध्वज के रूप में यह दिखाने के लिए किया जा सकता है कि एक प्रक्रिया अभी तक शुरू नहीं हुई है या किसी प्रक्रिया की शुरुआत के रूप में चिह्नित करने के लिए एक शर्त के रूप में इस्तेमाल किया जा सकता है।

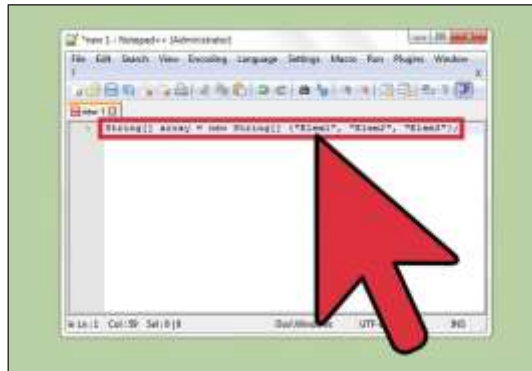
- उदाहरण के लिए: जब तक वस्तु शून्य न हो तब तक कुछ करें या कुछ भी न करें जब तक कि कोई वस्तु शून्य न हो।

```
synchronized method()
{
while (method()==null);
method().nowCanDoStuff();
}
```

जावा में एक ऐरे(Array) कैसे प्रिंट करें

यदि आप जावा पर काम कर रहे हैं और आपके पास बड़ी मात्रा में डेटा के साथ एक ऐरे है, तो आप कुछ तत्वों को आसानी से देखने के लिए उन्हें प्रिंट करना चाह सकते हैं। जावा में आप ऐरे को प्रिंट करने के कई तरीके हैं और नीचे दिए गए उदाहरण आपको प्रक्रिया के बारे में बताएंगे। मान लें कि मुद्रित किए जाने वाले ऐरे का नाम "ऐरे" है और जिन तत्वों को आप प्रिंट करना चाहते हैं उन्हें "एलेम" नाम दिया गया है।

विधि 1. स्ट्रिंग कमांड(string command) का उपयोग करना



1. तत्वों को अपने ऐरे में सेट करना। स्ट्रिंग(String) दर्ज करें [] array = new String [] { "Elem1", "Elem2", "Elem3"} जहां "ElemX" आपके ऐरे में अलग-अलग तत्व हैं।



2. मानक पुस्तकालय स्थिर विधि का प्रयोग करें: Arrays.toString(array)। यह आपको एक आयामी सरणियों का एक स्ट्रिंग प्रतिनिधित्व देगा। दूसरे शब्दों में, क्योंकि यह एक आयामी है, आप डेटा को पंक्तियों या स्तंभों में प्रस्तुत कर सकते हैं। यह विधि डेटा को एक पंक्ति, या स्ट्रिंग में प्रिंट करेगी।

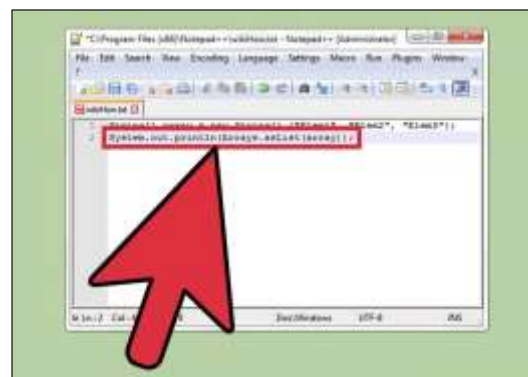


3. रन द प्रोग्राम(Run the program). इस कार्य को पूरा करने के लिए विभिन्न कंपाइलरों के पास अलग-अलग तरीके हैं। आप "फाइल" और फिर "रन" पर जा सकते हैं। आपके पास अपने टूलबार में बस "रन" आइकन पर क्लिक करने का विकल्प भी हो सकता है। आपके एलीमेंट जावा की निचली विंडो में एक स्ट्रिंग में प्रिंट किए जाएंगे।

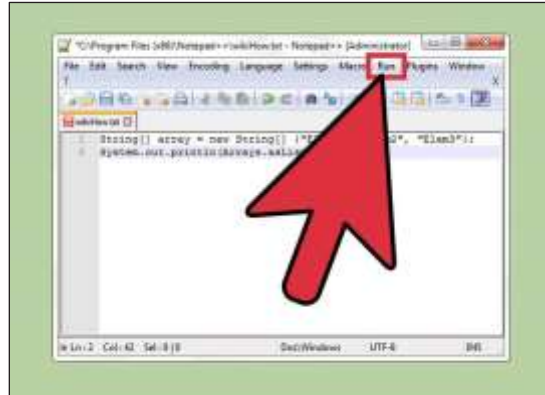
विधि 2. लिस्ट कमांड(List Command) के रूप में उपयोग करना



1. तत्वों को अपनी सरणी में सेट करना। `string[] array = new string [] {"Elem1", "Elem2", "Elem3"}` दर्ज करें, जहां "ElemX" वे अलग-अलग एलिमेंट हैं जिन्हें आप अपने ऐरे में चाहते हैं।

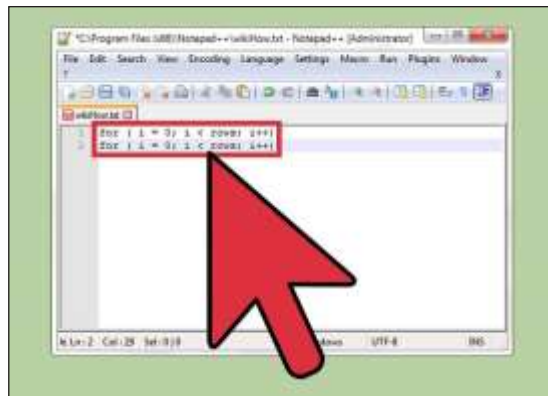


2. मानक पुस्तकालय स्थिर विधि का उपयोग करें: एक आयामी ऐरे के लिए `Arrays.asList()` जो आप प्रिंट एक सूची के रूप में चाहते हैं।

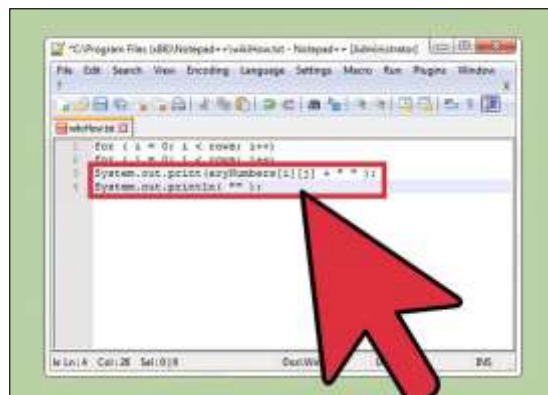


3. प्रोग्राम चलाएँ। इस कार्य को पूरा करने के लिए विभिन्न कंपाइलरों के पास अलग-अलग तरीके हैं। आप "फाइल" और फिर "रन" पर जा सकते हैं। आपके पास अपने टूलबार में बस "रन" आइकन पर क्लिक करने का विकल्प भी हो सकता है। आपके तत्व जावा की निचली विंडो में एक सूची या कॉलम में प्रिंट हो जाएंगे।

विधि 3. बहुआयामी ऐरे को प्रिंट करें



1. तत्वों को अपने में सेट करना। द्वि-आयामी सरणी के लिए, आपके पास दोनों पंक्तियाँ होंगी और कॉलम जिन्हें प्रिंट करने की आवश्यकता है। पंक्तियों के लिए ($i = 0; i < \text{rows}; i++$) दर्ज करें और कॉलम के लिए के लिए ($j = 0; j < \text{column}; j++$) दर्ज करें।



2. मानक पुस्तकालय स्थिर विधि का प्रयोग करें: `System.out.print(arrayNumbers[i][j] + " ");` उसके बाद `System.out.println("");`; एक पंक्ति के रूप में सारणियों और बहुआयामी सारणियों के भीतर सारणियों को प्रिंट करने के लिए।



3. प्रोग्राम चलाएँ। इस कार्य को पूरा करने के लिए विभिन्न कंपाइलरों के पास अलग-अलग तरीके हैं। आप "फाइल" और फिर "रन" पर जा सकते हैं। आपके पास अपने टूलबार में बस "रन" आइकन पर क्लिक करने का विकल्प भी हो सकता है। आपके एलिमेंट जावा की निचली विंडो में एक लाइन या कॉलम में प्रिंट हो जाएंगे।

कमांड प्रॉम्प्ट का उपयोग करके जावा प्रोग्राम संकलित और रन कैसे करें

हालांकि कई प्रोग्रामिंग वातावरण आपको पर्यावरण के भीतर एक प्रोग्राम को संकलित और चलाने की अनुमति देंगे, आप कमांड प्रॉम्प्ट का उपयोग करके भी संकलित कर और चला सकते हैं। विंडोज और मैक दोनों के पास कमांड प्रॉम्प्ट के अपने संस्करण हैं, हालांकि इसे तकनीकी रूप से मैक ओएस पर टर्मिनल कहा जाता है। प्रक्रिया अनिवार्य रूप से विंडोज और मैक के लिए समान है।

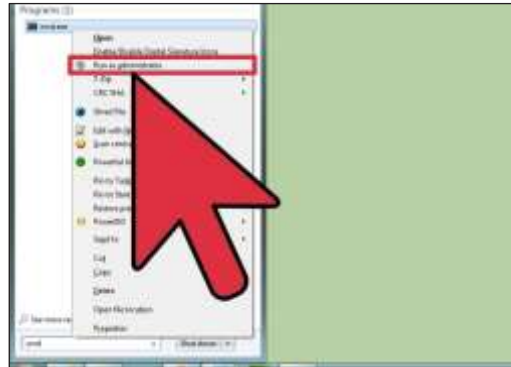
विधि 1. संकलन करना और रन करना



1. प्रोग्राम को सेव करें। अपना जावा प्रोग्राम बनाने के लिए नोटपैड जैसे टेक्स्ट एडिटर का उपयोग करने के बाद, प्रोग्राम को डॉट जावा एक्सटेंशन के साथ सेव करें। बेशक, फ़ाइल का नाम कुछ भी हो सकता है। इस ट्यूटोरियल के प्रयोजनों के लिए, "फ़ाइल नाम" का उपयोग आपके फ़ाइल नाम के लिए प्लेसहोल्डर के रूप में किया जाएगा।

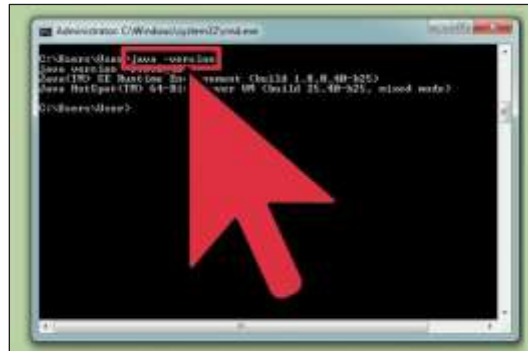
- यह सुनिश्चित करने के लिए कि आपकी फ़ाइल डॉट जावा फ़ाइल के रूप में सहेजी गई है, फ़ाइल नाम के बाद डॉट जावा लिखना सुनिश्चित करें, और ड्रॉपडाउन एक्सटेंशन मेनू से सभी फ़ाइलें चुनते हैं।
- नोट करें कि आपने अपने सिस्टम पर फ़ाइल को कहाँ सहेजा है।

- यदि आप सुनिश्चित नहीं हैं कि जावा में प्रोग्राम कैसे लिखा जाए, तो जावा में प्रोग्राम कैसे करें, इस पर हमारा ट्यूटोरियल देखें। हालाँकि, प्रोग्राम को संकलित करना और चलाना सीखने के लिए, आप किसी भी प्रकार के जावा प्रोग्राम का उपयोग कर सकते हैं।

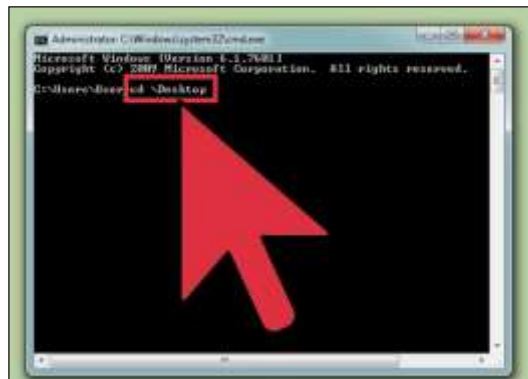


2. ओपन कमांड प्रॉम्प्ट/टर्मिनल। मैक और विन्डोज के लिए कमांड लाइन को एक्सेस करना थोड़ा अलग है।

- विन्डोज: प्रेस होम, फिर सीएमडी टाइप करें। प्रेस कमांड प्रॉम्प्ट खोलने के लिए एंटर करें।
- मैक: फाइंडर में, गो टैब दबाएं, एप्लिकेशन चुनें, यूटिलिटीज चुनें, फिर टर्मिनल चुनें।

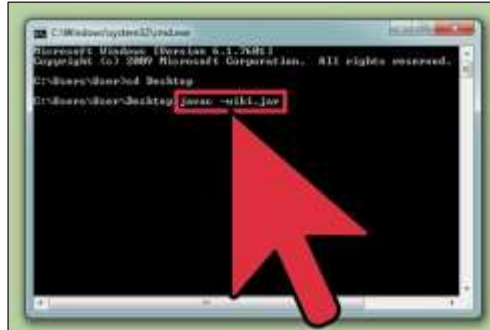


3. चेक करें कि जावा इनस्टॉल है। अपनी कमांड लाइन में जावा -वरज़न टाइप करें। यदि जावा इनस्टॉल है, तो आप देखेंगे कि एक संदेश जो बताता है कि वर्तमान में जावा का कौन सा वर्ज़न इनस्टॉल है।



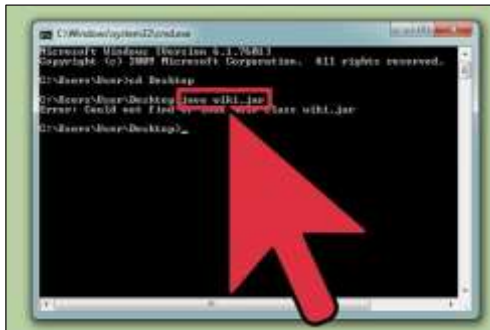
4. सही फ़ोल्डर में नेविगेट करें। आपकी कार्यशील निर्देशिका बदलने के लिए निर्देशिका नाम के बाद सीडी कमांड का उपयोग करें।

- उदाहरण के लिए, यदि आप C:\Users\Bob\Project में काम कर रहे थे और C:\Us- तक पहुंचना चाहते थे। ers\Bob\Project\TitanProject , सीडी टाइटनप्रोजेक्ट में एंटर करें और एंटर दबाएं।
- आप डीआईआर टाइप करके और एंटर दबाकर अपनी वर्तमान निर्देशिका में क्या है, इसकी सूची देख सकते हैं।



5. कार्यक्रम संकलित करें। एक बार जब आप सही निर्देशिका में हों, तो आप कमांड लाइन में javac filename.java टाइप करके और एंटर दबाकर प्रोग्राम को संकलित कर सकते हैं।

- यदि आपके प्रोग्राम में कोई त्रुटि है, या यदि मुश्किल से संकलन हो रहा है, तो कमांड शीघ्र इच्छा आप को सूचित करें।



6. प्रोग्राम चलाएँ। जावा फ़ाइल नाम दर्ज करें और एंटर दबाएँ। बेशक, "फ़ाइल नाम" को से बदलें जो भी आपकी फ़ाइल का नाम वास्तव में है।

- एंटर दबाने के बाद आपका प्रोग्राम चलना चाहिए। यदि आपको कोई त्रुटि मिलती है, या यदि आपका प्रोग्राम किसी भी तरह से चलने में विफल रहता है, तो समस्या निवारण विधि देखें।

विधि 2. ट्रबलशूटिंग(Troubleshooting)



1. अपना रास्ता तय करें। यदि आप एक ही निर्देशिका में सभी फाइलों के साथ एक साधारण प्रोग्राम चला रहे हैं, तो आप शायद ऐसा करने की आवश्यकता नहीं होगी।

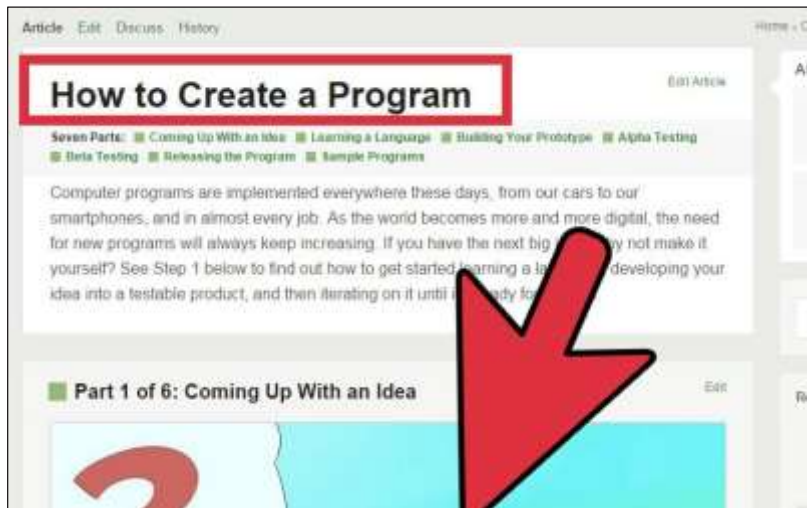
हालाँकि, यदि आप एकाधिक निर्देशिकाओं में फ़ाइलों के साथ अधिक जटिल प्रोग्राम चला रहे हैं, तो आपको कंप्यूटर को यह बताना होगा कि इन फ़ाइलों को कहाँ देखना है।

- विंडोज़: कमांड प्रॉम्प्ट में जावा -वरज़न टाइप करें और एंटर दबाएँ। पहली पंक्ति में प्रदर्शित जावा के संस्करण के आधार पर, कमांड प्रॉम्प्ट पर सेट पथ=%पथ%;सी:\प्रोग्राम फ़ाइलें\जावा\jdk1.5.0_09\bin टाइप करें और ↵ एंटर दबाएं। आपके द्वारा इंस्टॉल किए गए जावा के किसी भी संस्करण के साथ jdk1.5.0_09 को प्रतिस्थापित करें।
- सुनिश्चित करें कि आप अपने जावा प्रोजेक्ट वाली निर्देशिका में रहते हुए इस कमांड में प्रवेश कर रहे हैं।
- मैक: टर्मिनल में /usr/libexec/java_home -v 1.7 टाइप करें और सुनिश्चित करने के लिए ↵ एंटर दबाएं। जावा आपके सिस्टम पर स्थापित है। फिर, इको एक्सपोर्ट टाइप करें "JAVA_HOME=\$(/usr/libexec/ java_home)" >> ~/.bash_profile में टर्मिनल में जाएँ और एंटर दबाएँ। टर्मिनल को बाद में पुनरारंभ करें।

जावा में प्रतिशत कैसे निकाले

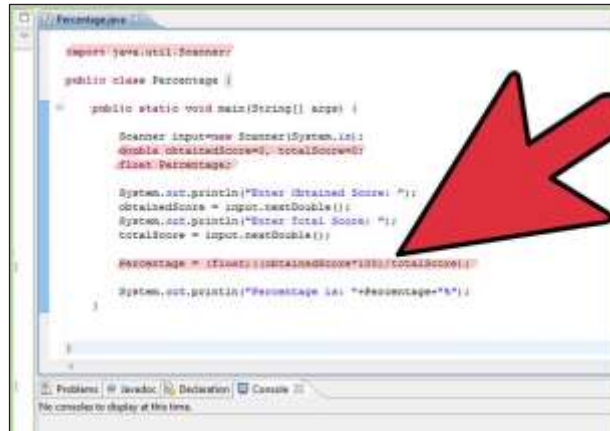
प्रतिशत की गणना करना बहुत मददगार हो सकता है। लेकिन जब संख्याएँ बड़ी हो जाती हैं, तो गणना करने के लिए प्रोग्राम का उपयोग करना बहुत आसान हो जाता है। यहां बताया गया है कि आप जावा में प्रतिशत की गणना के लिए एक प्रोग्राम कैसे बना सकते हैं।

चरण



1. अपने कार्यक्रम की योजना बनाएं। हालाँकि प्रतिशत की गणना करना मुश्किल नहीं है, कोड शुरू करने से पहले अपने कार्यक्रम की योजना बनाने का एक अच्छा अभ्यास हमेशा होता है। निम्नलिखित प्रश्नों के उत्तर खोजने का प्रयास करें:

- क्या आपका प्रोग्राम बड़ी संख्या में हैंडल करने वाला है? यदि हाँ, तो उन तरीकों के बारे में सोचने का प्रयास करें जिनसे आपका प्रोग्राम बड़ी संख्या में संख्याओं को संभाल सकता है। ऐसा करने का एक विधि इंटर के बजाय फ्लोट या लॉन्ग बेरिएबल का उपयोग करना है।



```

import java.util.Scanner;

public class Percentage {

    public static void main(String[] args) {

        Scanner input=new Scanner(System.in);
        double obtainedScore=0, totalScore=0;
        float Percentage;

        System.out.println("Enter Obtained Score: ");
        obtainedScore = input.nextDouble();
        System.out.println("Enter Total Score: ");
        totalScore = input.nextDouble();

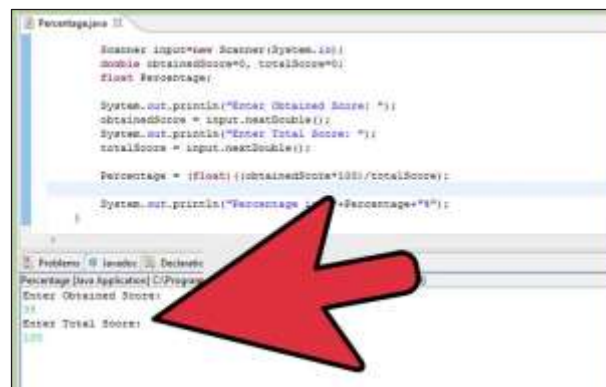
        Percentage = (float)((obtainedScore*100)/totalScore);

        System.out.println("Percentage is: "+Percentage+"%");
    }
}

```

2. कोड लिखें प्रतिशत की गणना करने के लिए, आपको दो मापदंडों की आवश्यकता होगी:

- कुल स्कोर (या अधिकतम संभव स्कोर); तथा,
- प्राप्त अंक जिसका प्रतिशत आप गणना करना चाहते हैं।
 - उदाहरण के लिए: यदि कोई छात्र किसी परीक्षा में 100 में से 30 अंक प्राप्त करता है, और आप छात्र द्वारा प्राप्त प्रतिशत अंकों की गणना करना चाहते हैं, तो 100 कुल अंक (या अधिकतम संभव स्कोर) है। 30 प्राप्त अंक है जिसका प्रतिशत आप गणना करना चाहते हैं।
- प्रतिशत की गणना करने का सूत्र है: प्रतिशत = (प्राप्त अंक x 100) / कुल स्कोर
- उपयोगकर्ता से ये पैरामीटर (इनपुट) प्राप्त करने के लिए, जावा में स्कैनर फ़ंक्शन का उपयोग करने का प्रयास करें।



```

Scanner input=new Scanner(System.in);
double obtainedScore=0, totalScore=0;
float Percentage;

System.out.println("Enter Obtained Score: ");
obtainedScore = input.nextDouble();
System.out.println("Enter Total Score: ");
totalScore = input.nextDouble();

Percentage = (float)((obtainedScore*100)/totalScore);

System.out.println("Percentage is: "+Percentage+"%");

```

Problems | JavaDoc | Declaration | Console |

 Percentage [Java Application] C:\Program

 Enter Obtained Score:

 30

 Enter Total Score:

 100

3. प्रतिशत की गणना करें। प्रतिशत की गणना करने के लिए पिछले चरण में दिए गए सूत्र का उपयोग करें। सुनिश्चित करें कि प्रतिशत के मान को संग्रहित करने के लिए उपयोग किया जाने वाला वेरिएबल फ्लोट टाइप का है। यदि नहीं, तो उत्तर सही नहीं हो सकता है।

- ऐसा इसलिए है, क्योंकि फ्लोट डेटा-प्रकार 32 बिट एकल परिशुद्धता है जो गणितीय गणनाओं में दशमलव पर भी विचार करता है। इस प्रकार, फ्लोट वेरिएबल का उपयोग करते हुए, गणितीय गणना जैसे 5/2 (5 से विभाजित 2) का उत्तर 2.5 . होगा ।
 - यदि समान गणना (5/2) एक अंतर चर का उपयोग करके की जाती है, तो उत्तर 2 होगा ।

- हालाँकि, वे चर जिनमें आपने कुल स्कोर और प्राप्त स्कोर संग्रहित किया है, पूर्णांक हो सकते हैं। प्रतिशत के लिए एक फ्लोट वैरिएबल का उपयोग करने से स्वचालित रूप से इंट को फ्लोट में बदल दिया जाएगा; और कुल गणना इंट के बजाय फ्लोट में की जाएगी।

```

Percentage.java
Scanner input=new Scanner(System.in);
double obtainedScore=0, totalScore=0;
float Percentage;

System.out.println("Enter Obtained Score: ");
obtainedScore = input.nextDouble();
System.out.println("Enter Total Score: ");
totalScore = input.nextDouble();

Percentage = (float) ((obtainedScore*100)/totalScore);

System.out.println("Percentage is: "+Percentage+"%");

```

Problems | Javadoc | Declaration

```

-terminated- Percentage [Java Application]
Enter Obtained Score:
25
Enter Total Score:
100
Percentage is: 25.0%

```

4. उपयोगकर्ता को प्रतिशत प्रदर्शित करें। एक बार प्रोग्राम ने प्रतिशत की गणना कर ली है, इसे उपयोगकर्ता को प्रदर्शित करें। इसके लिए जावा में System.out.print या System.out.println (न्यू लाइन पर प्रिंट करने के लिए) फंक्शन का उपयोग करें।

जावा में ऐरे का साइज़ कैसे खोजें

यह विधि सीधे आगे है लेकिन एक बार जब आप समझ जाते हैं कि यह कैसे काम करता है, तो आप इसे सभी प्रकार के अधिक जटिल संदर्भों में उपयोग कर सकते हैं।

चरण

```

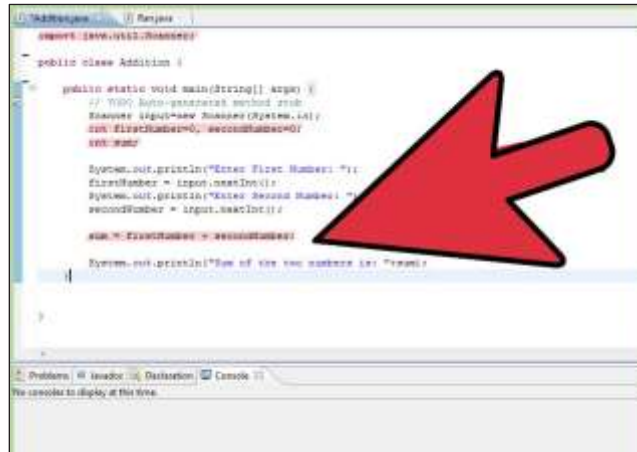
//
// To change this template, choose Tools | Templates
// and open the template in the editor.
//
package javadoc;

//

public class Main {
    public static void main(String args[]){
        int
    }
}

```

1. आईडीई खोलें। अपनी पसंद का आईडीई खोलें, या कोड लिखें और इसे कमांड के माध्यम से तत्पर निष्पादित करें।



```

Address.java | RunJava
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner input=new Scanner(System.in);
        int firstNumber=0, secondNumber=0;
        int sum;

        System.out.println("Enter First Number:");
        firstNumber = input.nextInt();
        System.out.println("Enter Second Number:");
        secondNumber = input.nextInt();

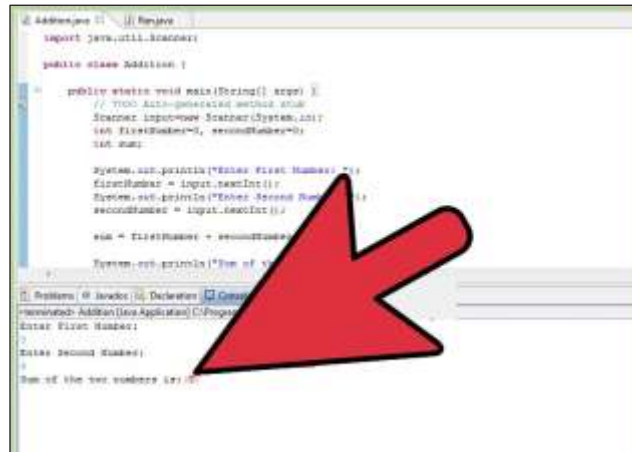
        sum = firstNumber + secondNumber;

        System.out.println("Sum of the two numbers is: "+sum);
    }
}

```

2. कोड लिखें। दो संख्याओं का योग ज्ञात करने का अर्थ है दोनों संख्याओं का साधारण योग।

- योग के मूल्य को संग्रहित करने के लिए एक अलग चर बनाएँ। यह इंट टाइप का हो सकता है।
- योग ज्ञात करने का सूत्र है:
- योग(Sum) = पहली संख्या(First Number) + दूसरी संख्या(Second Number)
- उपयोगकर्ता से ये पैरामीटर (इनपुट) प्राप्त करने के लिए, जावा में स्कैनर फ़ंक्शन का उपयोग करने का प्रयास करें।



```

Address.java | RunJava
import java.util.Scanner;

public class Addition {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner input=new Scanner(System.in);
        int firstNumber=0, secondNumber=0;
        int sum;

        System.out.println("Enter First Number:");
        firstNumber = input.nextInt();
        System.out.println("Enter Second Number:");
        secondNumber = input.nextInt();

        sum = firstNumber + secondNumber;

        System.out.println("Sum of the two numbers is: "+sum);
    }
}

```

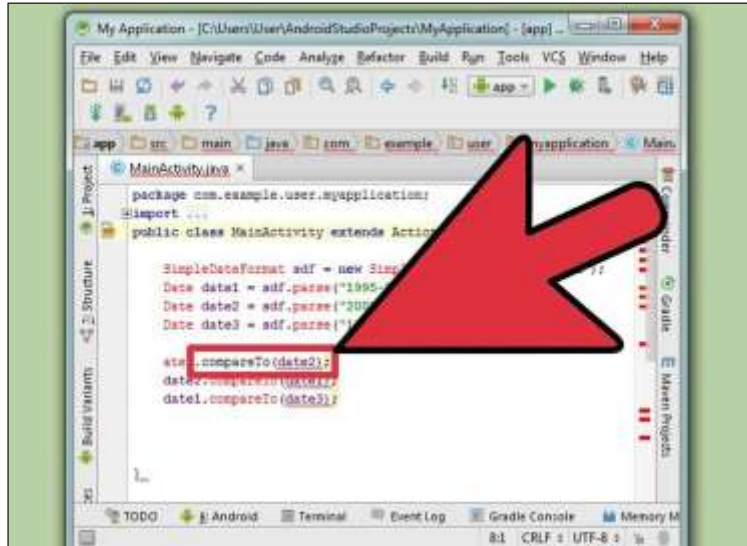
Enter First Number:
Enter Second Number:
Sum of the two numbers is: 10

3. आउटपुट प्रदर्शित करें। एक बार प्रोग्राम ने योग की गणना कर ली है, तो इसे उपयोगकर्ता को प्रदर्शित करें। इसके लिए जावा में System.out.print या System.out.println (नई लाइन पर प्रिंट करने के लिए) फ़ंक्शन का उपयोग करें।

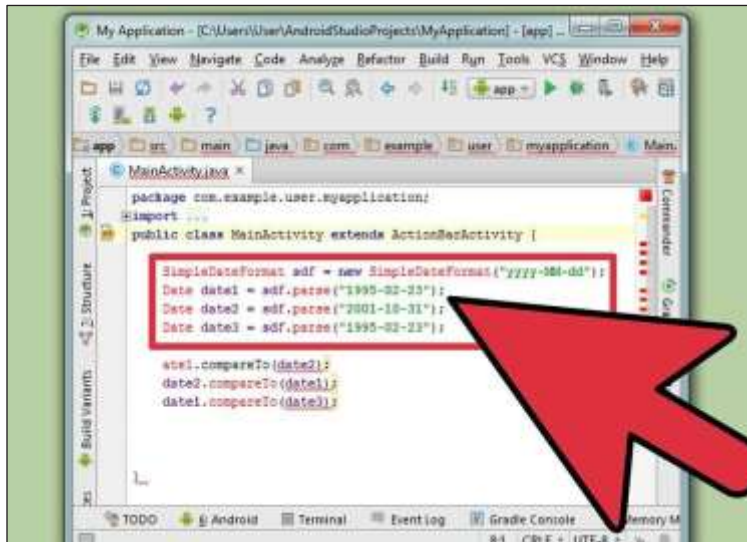
जावा में दो तिथियों की तुलना कैसे करें

जावा में तिथियों की तुलना करने के कई तरीके हैं। आंतरिक रूप से, एक तिथि को समय में एक (लंबे) बिंदु के रूप में दर्शाया जाता है - मिलीसेकंड की संख्या जो 1 जनवरी 1970 से बीत चुकी है। जावा में, दिनांक एक वस्तु है, जिसका अर्थ है कि इसमें तुलना के लिए कई विधियाँ शामिल हैं। दो तिथियों की तुलना करने की कोई भी विधि अनिवार्य रूप से तिथियों के समय की तुलना करेगी।

विधि 1. तुलना(compareTo) का उपयोग करना



1. तुलना का उपयोग करें। दिनांक तुलना करने योग्य <दिनांक> लागू करता है और इसलिए दो तिथियों की तुलना सीधे तुलना विधि से की जा सकती है। यदि दिनांक समान समय के लिए हैं, तो विधि शून्य रिटर्न होती है। यदि दिनांक की तुलना दिनांक आर्ग्यूमेंट से पहले की जाती है, तो शून्य से कम का मान वापस किया जाता है। यदि दिनांक की तुलना दिनांक आर्ग्यूमेंट के बाद की जाती है, तो शून्य से अधिक मान वापस किया जाता है। यदि तिथियाँ समान हैं, तो 0 का मान लौटाया जाता है।



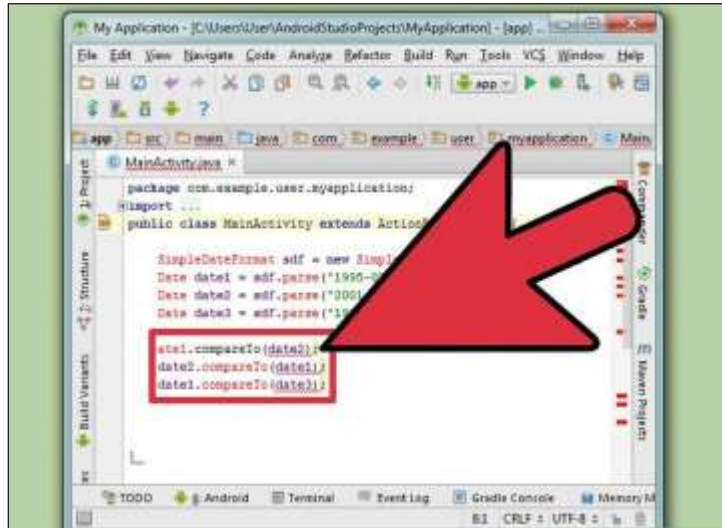
2. डेट ऑब्जेक्ट(date object) बनाएं। इससे पहले कि आप उनकी तुलना करना शुरू कर सकें, आपको प्रत्येक दिनांक वस्तु बनानी होगी। ऐसा करने का एक विधि सिंपल डेट फॉर्मेट वर्ग का उपयोग करना है। यह डेट ऑब्जेक्ट में डेट मानों की आसान प्रविष्टि की अनुमति देता है।

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd"); //For declaring values in new date objects. use same date format when creating dates
```

```
Date date1 = sdf.parse("1995-02-23"); //date1 is February 23, 1995
```

```
Date date2 = sdf.parse("2001-10-31"); //date2 is October 31, 2001
```

```
Date date3 = sdf.parse("1995-02-23"); //date3 is February 23, 1995
```



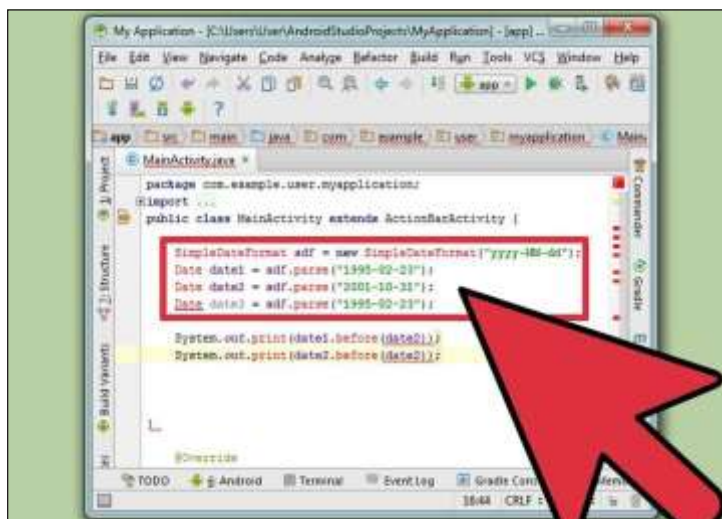
3. डेट ओब्जेक्ट्स की तुलना करें। नीचे दिया गया कोड आपको प्रत्येक मामला दिखाएगा - इससे कम, बराबर, और इससे बड़ा।

```
date1.compareTo(date2); //date1 < date2, returns less than 0
```

```
date2.compareTo(date1); //date2 > date1, returns greater than 0
```

```
date1.compareTo(date3); //date1 = date3, so will print 0 to console
```

विधि 2. आफ्टर(after), बिफोर(before) और इक्वल(equal) का उपयोग करना,



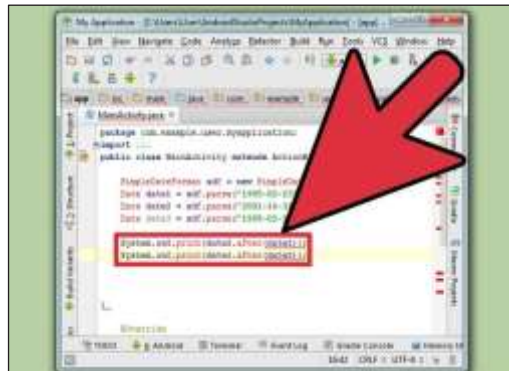
1. बाद में और पहले इक्वल का प्रयोग करें। तिथियों की तुलना विधियों के बाद और पहले के बराबर के साथ की जा सकती है। यदि दो तिथियां एक ही समय बिंदु के लिए हैं, तो इक्वल विधि सही हो जाएगी। उदाहरण कम्पेयर विधि से पहले बनाई गई तिथियों का उपयोग करेंगे।



2. पहले की विधि का उपयोग करके तुलना करें। नीचे दिया गया कोड एक सही और गलत मामला दिखाता है। अगर डेट1 पहले है डेट 2, पहले सही रिटर्न होता है। यदि ऐसा नहीं है, तो झूठे रिटर्न से पहले।

```
System.out.print(date1.before(date2)); //prints true
```

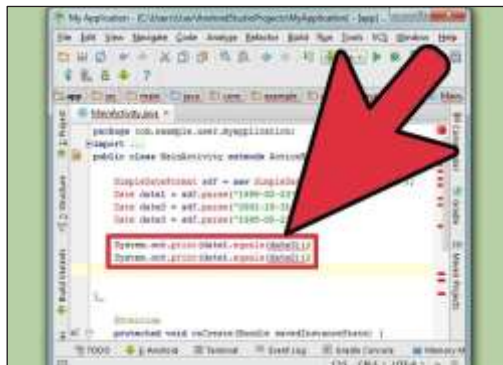
```
System.out.print(date2.before(date2)); //prints false
```



3. बाद की विधि का उपयोग करके तुलना करें। नीचे दिया गया कोड एक सही और गलत मामला दिखाता है। यदि दिनांक 2 के बाद है दिनांक 1, सही रिटर्न होने के बाद यदि ऐसा नहीं है, तो गलत वापसी के बाद।

```
System.out.print(date2.after(date1)); //prints true
```

```
System.out.print(date1.after(date2)); //prints false
```

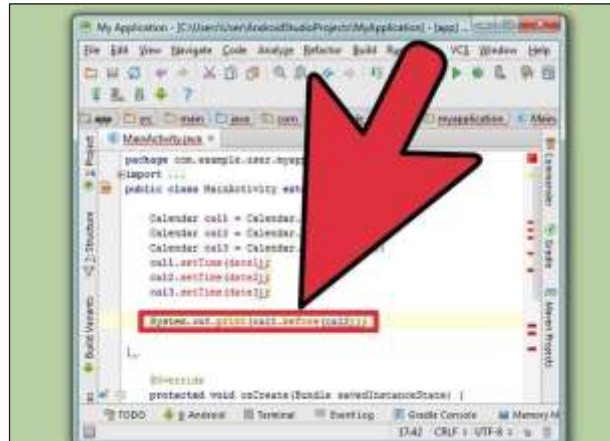


4. इक्वल विधि का उपयोग करके तुलना करें। नीचे दिया गया कोड एक सही और गलत मामला दिखाता है। यदि तिथियां बराबर हैं, इक्वल सही रिटर्न होता है। यदि वे नहीं हैं, तो बराबर रिटर्न झूठा है।

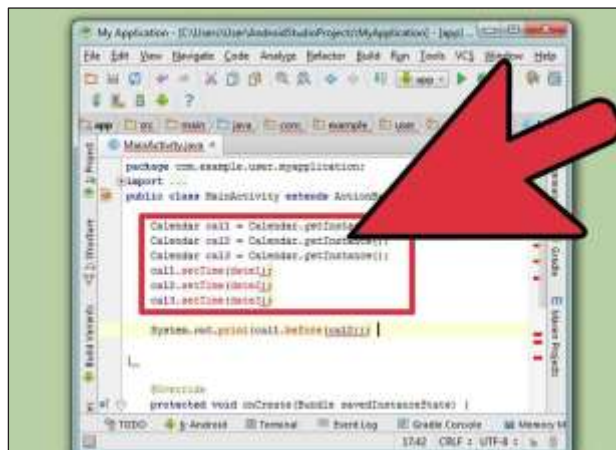
```
System.out.print(date1.equals(date3)); //prints true
```

```
System.out.print(date1.equals(date2)); //prints false
```

विधि 3. कैलेंडर क्लास का उपयोग करना



1. कैलेंडर का प्रयोग करें। कैलेंडर वर्ग में भी तुलना करने के लिए, बराबर, बाद में और पहले के तरीके हैं जो उसी तरह से काम करते हैं जैसे दिनांक वर्ग के लिए ऊपर वर्णित है। इसलिए यदि तारीख की जानकारी किसी कैलेंडर में रखी जा रही है, तो केवल तुलना करने के लिए तारीख निकालने की जरूरत नहीं है।



2. कैलेंडर के उदाहरण बनाएँ। कैलेंडर विधियों का उपयोग करने के लिए, आपको कुछ कैलेंडर उदाहरणों की आवश्यकता होगी। सौभाग्य से, आप पहले से बनाए गए दिनांक उदाहरणों से केवल समय ले सकते हैं।

```
Calendar cal1 = Calendar.getInstance(); //declares cal1
```

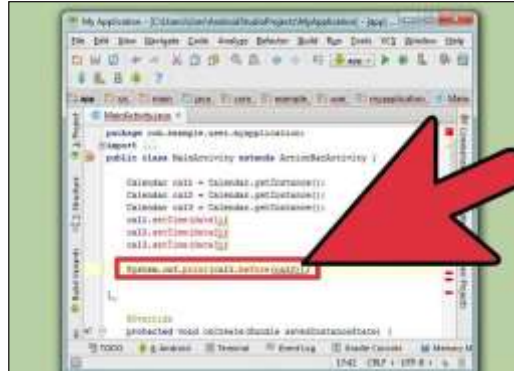
```
Calendar cal2 = Calendar.getInstance(); //declares cal2
```

```
Calendar cal3 = Calendar.getInstance(); //declares cal3
```

```
cal1.setTime(date1); //applies date to cal1
```

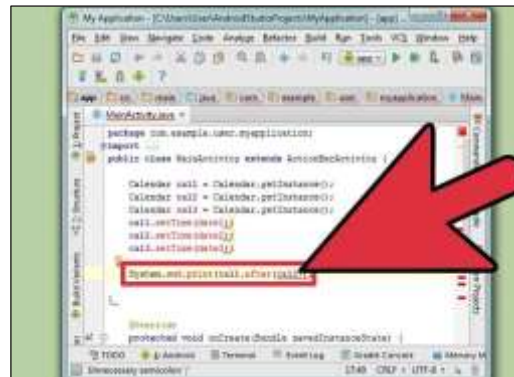
```
cal2.setTime(date2);
```

```
cal3.setTime(date3);
```



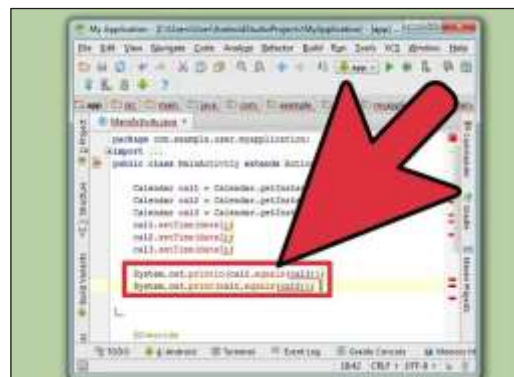
3. पहले का उपयोग करके कॉल 1 और कॉल 2 की तुलना करें। नीचे दिया गया कोड सही प्रिंट होना चाहिए क्योंकि कॉल 1 कॉल 2 से पहले है।

```
System.out.print(cal1.before(cal2)); //will print true
```



4. कॉल 1 और कॉल 2 के बाद का उपयोग करके तुलना करें। नीचे दिए गए कोड को झूठा प्रिंट करना चाहिए क्योंकि कॉल 1 कॉल 2 से पहले है।

```
System.out.print(cal1.after(cal2)); //prints false
```

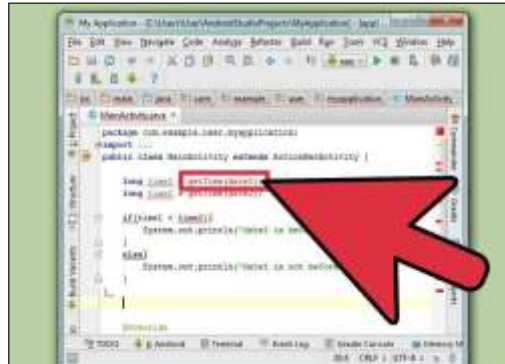


5. बराबर का उपयोग करके cal1 और cal2 की तुलना करें। नीचे दिया गया कोड एक सच्चे और झूठे मामले दोनों का एक उदाहरण दिखाएगा। स्थिति तुलना किए जा रहे कैलेंडर इंस्टेंस पर निर्भर करती है। कोड को अगली पंक्ति में "सच," फिर "गलत" प्रिंट करना चाहिए।

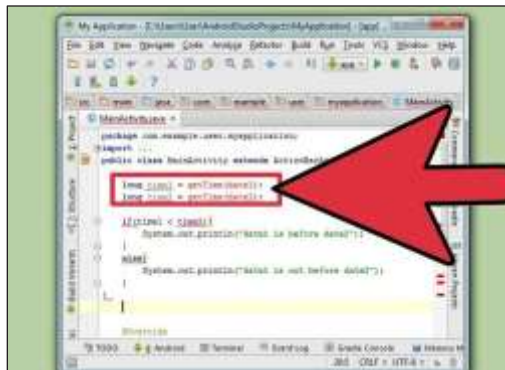
```
System.out.println(cal1.equals(cal3)); //prints true: cal1 == cal3
```

```
System.out.print(cal1.equals(cal2)); //prints false: cal1 != cal2
```

विधि 4. गेट टाइम का उपयोग करना



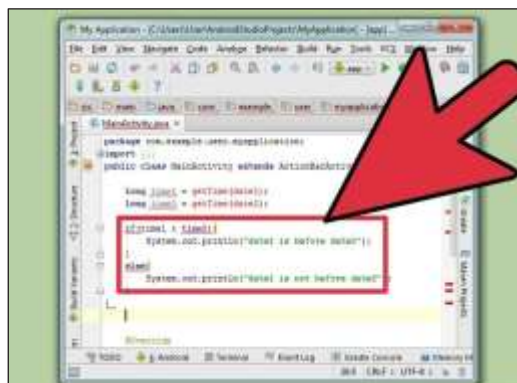
1. गेटटाइम का प्रयोग करें। दो तिथियों के समय बिंदु की सीधे तुलना करना भी संभव है, हालांकि पिछले दृष्टिकोणों में से कोई भी अधिक पठनीय और बेहतर होने की संभावना है। यह दो प्राइमेटिव डेटा प्रकारों की तुलना होगी, इसलिए इसे "<", ">", और "==" के साथ किया जा सकता है।



2. लंबे समय तक वस्तुओं का निर्माण करें। इससे पहले कि आप तिथियों की तुलना कर सकें, आपको पहले बनाए गए दिनांक ऑब्जेक्ट्स के डेटा के साथ लंबे पूर्णांक बनाने होंगे। सौभाग्य से, गेटटाइम () विधि आपके लिए अधिकांश काम करेगी।

```
long time1 = getTime(date1); //declares primitive time1 from date1
```

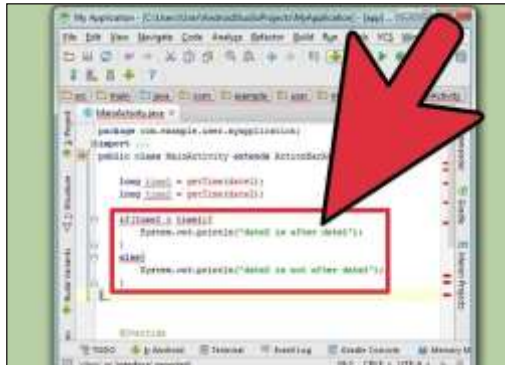
```
long time2 = getTime(date2); //declares primitive time2 from date2
```



3. तुलना से कम करें। इन दो पूर्णांक मानों की तुलना करने के लिए प्रतीक से कम (<) का प्रयोग करें।

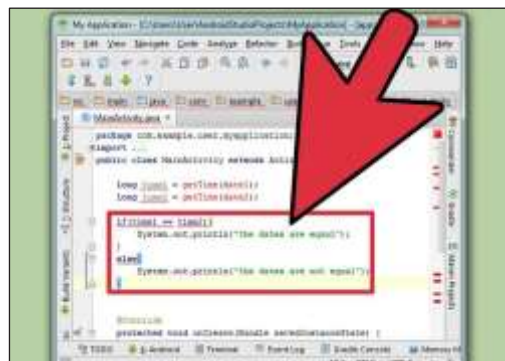
चूँकि समय 1, समय 2 से कम है, इसलिए पहला संदेश प्रिंट होना चाहिए। अन्य कथन उचित वाक्य रचना के लिए शामिल है।

```
if(time1 < time2){
System.out.println("date1 is before date2"); //will print since time1 <time2
}
else{
System.out.println("date1 is not before date2");
}
```



4. तुलना से बड़ा करो। इन दो पूर्णांक मानों की तुलना करने के लिए प्रतीक (>) बड़ा का प्रयोग करें। चूँकि समय 1 समय 2 से बड़ा है, इसलिए पहला संदेश प्रिंट होना चाहिए। अन्य कथन उचित वाक्य रचना के लिए शामिल है।

```
if(time2 > time1){
System.out.println("date2 is after date1"); //will print since time2 > time1
}
else{
System.out.println("date2 is not after date1");
}
```



5. एक समान तुलना करें। समानता के लिए इन दो पूर्णांक मानों की तुलना करने के लिए समानता (==) की जांच के लिए प्रतीक का प्रयोग करें। चूंकि टाइम 1 टाइम 3 के बराबर है, पहला संदेश प्रिंट होना चाहिए। यदि प्रोग्राम अन्य कथन पर पहुँच जाता है, तो इसका अर्थ है कि समय समान नहीं है।

```
if(time1 == time2){
System.out.println("the dates are equal");
}
else{
System.out.println("the dates are not equal"); //will print since time1 != time2
```