1. **Java Tutorial**
2. **What is Java**

* Java là một ngôn ngữ lập trình và là ngôn ngữ nền tảng. Java là ngôn ngữ bậc cao, mạnh mẽ. hướng đối tượng và là ngôn ngữ an toàn.
* Java được phát triển bởi Sun Microsystems (hiện là công ty con của Oracle) vào năm 1995. James Gosling được biết đến là cha đẻ của Java. Trước Java, tên của nó là Oak. Vì Oak đã là một công ty đã đăng ký nên James Gosling và nhóm của ông đã đổi tên từ Oak thành Java.
* Nền tảng : Bất kỳ môi trường phần cứng hoặc phần mềm nào mà chương trình chạy trong đó, được gọi là nền tảng. Vì Java có môi trường thời gian chạy (JRE) và API nên nó được gọi là nền tảng.
* Các loại ứng dụng Java
* Ứng dụng độc lập
* Ứng dụng web
* Ứng dụng doanh nghiệp
* Ứng dụng di động
* Java bao gồm 4 nền tảng như Java SE, Java EE , Java ME, JavaFX

1. **History of Java**

Lịch sử các phiên bản của Java:

* Nhiều phiên bản java đã được phát hành cho đến nay. Bản phát hành ổn định hiện tại của Java là Java SE 10.
* JDK Alpha và Beta (1995)
* JDK 1.0 (ngày 23 tháng 1 năm 1996)
* JDK 1.1 (ngày 19 tháng 2 năm 1997)
* J2SE 1.2 (ngày 8 tháng 12 năm 1998)
* J2SE 1.3 (ngày 8 tháng 5 năm 2000)
* J2SE 1.4 (ngày 6 tháng 2 năm 2002)
* J2SE 5.0 (ngày 30 tháng 9 năm 2004)
* Java SE 6 (ngày 11 tháng 12 năm 2006)
* Java SE 7 (ngày 28 tháng 7 năm 2011)
* Java SE 8 (ngày 18 tháng 3 năm 2014)
* Java SE 9 (ngày 21 tháng 9 năm 2017)
* Java SE 10 (ngày 20 tháng 3 năm 2018)
* Java SE 11 (tháng 9 năm 2018)
* Java SE 12 (tháng 3 năm 2019)
* Java SE 13 (tháng 9 năm 2019)
* Java SE 14 (Tháng 3 năm 2020)
* Java SE 15 (tháng 9 năm 2020)
* Java SE 16 (tháng 3 năm 2021)
* Java SE 17 (tháng 9 năm 2021)
* Java SE 18 (sẽ được phát hành vào tháng 3 năm 2022)

1. **Features of java**

* Một số tính năng quan trọng của java như : Simple, Object-Oriented, Portable, Platform independent, Secured, Robust, Architecture neutral, Interpreted, High Performance, Multithreaded, Distributed, Dynamic

1. **C++ vs Java**

*I.4.1: Bảng so sánh các tính chất của C++ và Java*

|  |  |  |
| --- | --- | --- |
| Chỉ số so sánh | C++ | Java |
| Nền tảng độc lập | C ++ phụ thuộc vào nền tảng. | Java không phụ thuộc vào nền tảng. |
| Chủ yếu được sử dụng cho | C ++ chủ yếu được sử dụng để lập trình hệ thống. | Java chủ yếu được sử dụng để lập trình ứng dụng. Nó được sử dụng rộng rãi trong các ứng dụng dựa trên Windows, dựa trên web, doanh nghiệp và di động. |
| Mục tiêu thiết kế | C++ được thiết kế để lập trình hệ thống và ứng dụng. Đó là một phần mở rộng của ngôn ngữ [lập trình C.](https://www.javatpoint.com/c-programming-language-tutorial) | Java được thiết kế và tạo ra như một trình thông dịch cho các hệ thống in nhưng sau đó được mở rộng như một máy tính mạng hỗ trợ. Nó được thiết kế để dễ sử dụng và dễ tiếp cận với nhiều đối tượng hơn. |
| Đi đến | C++ hỗ trợ câu lệnh [goto](https://www.javatpoint.com/cpp-goto-statement) . | Java không hỗ trợ câu lệnh goto. |
| đa thừa kế | C++ hỗ trợ đa kế thừa. | Java không hỗ trợ đa kế thừa thông qua lớp. Nó có thể đạt được bằng cách sử dụng [các giao diện trong java](https://www.javatpoint.com/interface-in-java) . |
| Quá tải toán tử | C++ hỗ trợ [nạp chồng toán tử](https://www.javatpoint.com/cpp-overloading) . | Java không hỗ trợ nạp chồng toán tử. |
| con trỏ | C++ hỗ trợ con [trỏ](https://www.javatpoint.com/cpp-pointers) . Bạn có thể viết chương trình con trỏ bằng C++. | Java hỗ trợ con trỏ bên trong. Tuy nhiên, bạn không thể viết chương trình con trỏ trong java. Nó có nghĩa là java đã hạn chế hỗ trợ con trỏ trong java. |
| Trình biên dịch và phiên dịch | C++ chỉ sử dụng trình biên dịch. C ++ được biên dịch và chạy bằng trình biên dịch chuyển đổi mã nguồn thành mã máy, do đó, C ++ phụ thuộc vào nền tảng. | Java sử dụng cả trình biên dịch và trình thông dịch. Mã nguồn Java được chuyển đổi thành mã byte tại thời điểm biên dịch. Trình thông dịch thực thi mã byte này trong thời gian chạy và tạo đầu ra. Java được giải thích đó là lý do tại sao nó độc lập với nền tảng. |
| Gọi theo giá trị và gọi theo tham chiếu | C++ hỗ trợ cả gọi theo giá trị và gọi theo tham chiếu. | Java chỉ hỗ trợ gọi theo giá trị. Không có cuộc gọi theo tham chiếu trong java. |
| Cấu trúc và Liên minh | C++ hỗ trợ các cấu trúc và liên kết. | Java không hỗ trợ cấu trúc và liên kết. |
| Hỗ trợ chủ đề | C++ không có hỗ trợ tích hợp sẵn cho các chủ đề. Nó dựa vào các thư viện của bên thứ ba để hỗ trợ luồng. | [Java có hỗ trợ luồng](https://www.javatpoint.com/multithreading-in-java) tích hợp . |
| nhận xét tài liệu | C++ không hỗ trợ nhận xét tài liệu. | Java hỗ trợ chú thích tài liệu (/\*\* ... \*/) để tạo tài liệu cho mã nguồn java. |
| Từ khóa ảo | C++ hỗ trợ từ khóa ảo để chúng ta có thể quyết định có ghi đè chức năng hay không. | Java không có từ khóa ảo. Theo mặc định, chúng tôi có thể ghi đè tất cả các phương thức không tĩnh. Nói cách khác, các phương thức không tĩnh là ảo theo mặc định. |
| dịch chuyển phải không dấu >>> | C++ không hỗ trợ toán tử >>>. | Java hỗ trợ toán tử dịch chuyển phải >>> không dấu điền số 0 ở đầu cho các số âm. Đối với các số dương, nó hoạt động giống như toán tử >>. |
| Cây thừa kế | C++ luôn tạo một cây thừa kế mới. | Java luôn sử dụng một cây thừa kế duy nhất vì tất cả các lớp đều là con của lớp Object trong Java. Lớp Object là gốc của cây [thừa kế](https://www.javatpoint.com/inheritance-in-java) trong java. |
| Phần cứng | C++ gần với phần cứng hơn. | Java không tương tác với phần cứng. |
| hướng đối tượng | C++ là một ngôn ngữ hướng đối tượng. Tuy nhiên, trong ngôn ngữ C, một hệ thống phân cấp gốc đơn lẻ là không thể. | Java cũng là một ngôn ngữ [hướng đối tượng](https://www.javatpoint.com/java-oops-concepts) . Tuy nhiên, mọi thứ (ngoại trừ các kiểu cơ bản) đều là một đối tượng trong Java. Nó là một hệ thống phân cấp gốc duy nhất vì mọi thứ đều bắt nguồn từ java.lang.Object. |

1. **Hello Java program**

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

*I.5.1: Hình ảnh minh họa chương trình chạy thử bằng ngôn ngữ Java*

* **Class:** từ khóa class được sử dụng để khai báo một lớp trong Java.
* **Public:** từ khóa public là một công cụ sửa đổi truy cập đại diện cho khả năng hiển thị. Nó có nghĩa là nó được hiển thị cho tất cả.
* **Static:** static là một từ khóa. Nếu chúng ta khai báo bất kỳ phương thức nào là tĩnh, thì nó được gọi là phương thức tĩnh.
* **Void:** void là kiểu trả về của phương thức. Nó có nghĩa là nó không trả lại bất kỳ giá trị nào.
* **Main:** main đại diện cho điểm bắt đầu của chương trình.
* **String[] args or String args[]:** được sử dụng cho đối số dòng lệnh. Chúng tôi sẽ thảo luận về nó trong phần tới.
* **System.out.println():** được sử dụng để in câu lệnh.
* Các cách viết chương trình java
* Cách 1: static public void main(String args[])
* Cách 2: public static void main(String[] args)

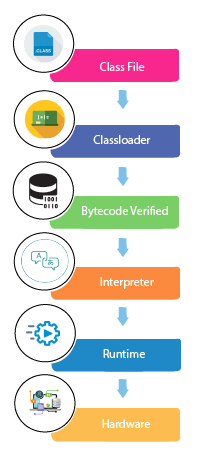
public static void main(String []args)

public static void main(String args[])

* Cách 3: public static void main(String... args) : là kiểu var-args

1. **Program Internal**

* Trong thời gian chạy, các bước sau đây được thực hiện:



*I.6.1: Hình ảnh minh họa các bước thực hiện trong thời gian chạy*

1. **How to set path?**

* Cách cài đặt đường dẫn JDK trong Windows
* Vào ThisPC chuột trái rồi tới Properties
* Click vào advanced tab
* Click vào new của bảng user variables
* Điền đường dẫn JDK sau đó OK

1. **JDK, JRE and JVM**

JVM:

* JVM (Máy ảo Java) là một máy trừu tượng. Nó được gọi là máy ảo vì nó không tồn tại về mặt vật lý. Nó là một đặc điểm kỹ thuật cung cấp môi trường thời gian chạy trong đó mã byte Java có thể được thực thi. Nó cũng có thể chạy các chương trình được viết bằng các ngôn ngữ khác và được biên dịch thành mã byte Java.
* JVM có sẵn cho nhiều nền tảng phần cứng và phần mềm. JVM, JRE và JDK phụ thuộc vào nền tảng vì cấu hình của mỗi [HĐH](https://www.javatpoint.com/os-tutorial) khác nhau. Tuy nhiên, Java là nền tảng độc lập. Có ba khái niệm về JVM: *đặc tả* , *triển khai* và *thể hiện* .
* JVM thực hiện các nhiệm vụ chính sau:
* Tải mã
* Xác minh mã
* Thực thi mã
* Cung cấp môi trường thời gian chạy

JRE:

* JRE là từ viết tắt của Môi trường chạy thi hành Java. Nó cũng được viết là Java RTE. Môi trường chạy thi hành Java là một tập hợp các công cụ phần mềm được sử dụng để phát triển các ứng dụng Java. Nó được sử dụng để cung cấp môi trường thời gian chạy. Đó là việc triển khai JVM. Nó tồn tại về mặt vật lý. Nó chứa một tập hợp các thư viện + các tệp khác mà JVM sử dụng khi chạy.
* Việc triển khai JVM cũng được các công ty khác tích cực phát hành ngoài Sun Micro Systems.

JDK:

* JDK là viết tắt của Java Development Kit. Java Development Kit (JDK) là một môi trường phát triển phần mềm được sử dụng để phát triển các ứng dụng Java và [applet](https://www.javatpoint.com/java-applet) . Nó tồn tại về mặt vật lý. Nó chứa các công cụ phát triển JRE +.
* JDK là một triển khai của bất kỳ một trong các Nền tảng Java được cung cấp dưới đây do Tập đoàn Oracle phát hành:
* Phiên bản tiêu chuẩn Nền tảng Java
* Nền tảng Java phiên bản doanh nghiệp
* Nền tảng Java phiên bản vi mô
* JDK chứa Máy ảo Java (JVM) riêng và một số tài nguyên khác như trình thông dịch/trình tải (java), trình biên dịch (javac), trình lưu trữ (jar), trình tạo tài liệu (Javadoc), v.v. để hoàn thành phát triển của một ứng dụng Java.

1. **JVM: Java Virtual Machine**
2. **Java Variables**

* Cách đặt biến trong java:

<kiểu dữ liệu> <Tên biến>;

<kiểu dữ liệu> <Tên biến> = <giá trị>;

* Có 3 cách đặt biến trong java:
* Biến cục bộ: Một biến được khai báo bên trong phần thân của phương thức được gọi là biến cục bộ. Bạn chỉ có thể sử dụng biến này trong phương thức đó và các phương thức khác trong lớp thậm chí không biết rằng biến đó tồn tại.
* Instance Variable: Một biến được khai báo bên trong lớp nhưng bên ngoài phần thân của phương thức, được gọi là biến thể hiện. Nó không được khai báo là tĩnh.
* Biến tĩnh: Một biến được khai báo là tĩnh được gọi là biến tĩnh. Nó không thể là địa phương. Có thể tạo một bản sao duy nhất của biến tĩnh và chia sẻ nó giữa tất cả các thể hiện của lớp. Cấp phát bộ nhớ cho các biến tĩnh chỉ xảy ra một lần khi lớp được tải vào bộ nhớ.

1. **Java Data Types**

* Các kiểu dữ liệu trong java: boolean, byte,char, short, int, long, float, double

1. **Unicode System**

* Java sử dụng bộ mã hóa ký tự Unicode

1. **Operators**

* Toán tử trong java
* Toán tử một ngôi: a++,a--,++a,--a
* Toán tử hai ngôi: a+b, a-b,a!=b,a%b,a/b,a==b

1. **Keyword**

* **abstract:** Từ khóa trừu tượng Java được sử dụng để khai báo một lớp trừu tượng. Một lớp trừu tượng có thể cung cấp việc triển khai giao diện. Nó có thể có các phương thức trừu tượng và không trừu tượng.
* **boolean:** Từ khóa boolean trong Java được sử dụng để khai báo một biến là kiểu boolean. Nó chỉ có thể giữ các giá trị Đúng và Sai.
* **break:** Từ khóa break trong Java được sử dụng để ngắt vòng lặp hoặc câu lệnh switch. Nó phá vỡ luồng hiện tại của chương trình ở các điều kiện đã chỉ định.
* **byte:** Từ khóa Java byte được sử dụng để khai báo một biến có thể chứa các giá trị dữ liệu 8 bit.
* **case**: Từ khóa case trong Java được sử dụng với các câu lệnh switch để đánh dấu các khối văn bản.
* **catch:** Từ khóa catch trong Java được sử dụng để bắt các ngoại lệ do các câu lệnh try tạo ra. Nó phải được sử dụng chỉ sau khối thử.
* **char:** Từ khóa Java char được sử dụng để khai báo một biến có thể chứa các ký tự Unicode 16-bit không dấu
* **class :** Từ khóa Java class được sử dụng để khai báo một lớp.
* **continue :** Từ khóa continue trong Java được sử dụng để tiếp tục vòng lặp. Nó tiếp tục luồng hiện tại của chương trình và bỏ qua đoạn mã còn lại ở điều kiện đã chỉ định.
* **default**: Từ khóa mặc định của Java được sử dụng để chỉ định khối mã mặc định trong câu lệnh chuyển đổi.
* **do :** Từ khóa Java do được sử dụng trong câu lệnh điều khiển để khai báo một vòng lặp. Nó có thể lặp lại một phần của chương trình nhiều lần.
* **double :** Từ khóa double trong Java được sử dụng để khai báo một biến có thể chứa số dấu phẩy động 64 bit.
* **other :** Từ khóa Java khác được sử dụng để chỉ ra các nhánh thay thế trong câu lệnh if.
* **enum :** Từ khóa enum trong Java được sử dụng để xác định một tập hợp các hằng số cố định. Enum hàm tạo luôn ở chế độ riêng tư hoặc mặc định.
* **extends:** Từ khóa mở rộng Java được sử dụng để chỉ ra rằng một lớp được dẫn xuất từ ​​một lớp hoặc giao diện khác.
* **final :** Từ khóa cuối cùng trong Java được sử dụng để chỉ ra rằng một biến giữ một giá trị không đổi. Nó được sử dụng với một biến. Nó được sử dụng để hạn chế người dùng cập nhật giá trị của biến.
* **finally :** Từ khóa cuối cùng của Java chỉ ra một khối mã trong cấu trúc thử bắt. Khối này luôn được thực thi cho dù có xử lý ngoại lệ hay không.
* **float :** Từ khóa float trong Java được sử dụng để khai báo một biến có thể chứa số dấu phẩy động 32 bit.
* **for :** Từ khóa for trong Java được sử dụng để bắt đầu một vòng lặp for. Nó được sử dụng để thực hiện lặp đi lặp lại một tập lệnh/chức năng khi một số điều kiện trở thành đúng. Nếu số lần lặp là cố định thì nên sử dụng vòng lặp for.
* **if :** Java if từ khóa kiểm tra điều kiện. Nó thực thi khối if nếu điều kiện là đúng.
* **implements :** Java implements từ khóa được sử dụng để triển khai một giao diện.
* **import:** Từ khóa nhập khẩu Java làm cho các lớp và giao diện có sẵn và có thể truy cập được vào mã nguồn hiện tại.
* **instanceof** : Từ khóa instanceof trong Java được sử dụng để kiểm tra xem đối tượng có phải là một thể hiện của lớp đã chỉ định hay thực hiện một giao diện hay không.
* **int :** Từ khóa Java int được sử dụng để khai báo một biến có thể chứa số nguyên có dấu 32 bit.
* **interface :** Từ khóa giao diện Java được sử dụng để khai báo một giao diện. Nó chỉ có thể có các phương thức trừu tượng.
* **long :** Từ khóa long trong Java được sử dụng để khai báo một biến có thể chứa số nguyên 64 bit.
* **native:** Từ khóa bản địa Java được sử dụng để chỉ định rằng một phương thức được triển khai trong mã bản địa bằng cách sử dụng JNI (Giao diện bản địa Java).
* **new :** Java new từ khóa được sử dụng để tạo các đối tượng mới.
* **null :** Từ khóa Java null được sử dụng để chỉ ra rằng một tham chiếu không đề cập đến bất cứ điều gì. Nó loại bỏ giá trị rác.
* **package :** Từ khóa gói Java được sử dụng để khai báo gói Java bao gồm các lớp.
* **private :** Từ khóa private trong Java là một công cụ sửa đổi truy cập. Nó được sử dụng để chỉ ra rằng một phương thức hoặc biến chỉ có thể được truy cập trong lớp mà nó được khai báo.
* **protected :** Từ khóa Java protected là một công cụ sửa đổi truy cập. Nó có thể được truy cập trong gói và bên ngoài gói nhưng chỉ thông qua kế thừa. Nó không thể được áp dụng với lớp học.
* **public :** Từ khóa công khai trong Java là một công cụ sửa đổi truy cập. Nó được sử dụng để chỉ ra rằng một mục có thể truy cập được ở bất cứ đâu. Nó có phạm vi rộng nhất trong số tất cả các công cụ sửa đổi khác.
* **return :** Từ khóa return trong Java được sử dụng để trả về từ một phương thức khi quá trình thực thi của nó hoàn tất.
* **short :** Từ khóa short trong Java được sử dụng để khai báo một biến có thể chứa số nguyên 16 bit.
* **static :** Từ khóa tĩnh trong Java được sử dụng để chỉ ra rằng một biến hoặc phương thức là một phương thức lớp. Từ khóa tĩnh trong Java chủ yếu được sử dụng để quản lý bộ nhớ.
* **strictfp :** Javarictfp được sử dụng để hạn chế các phép tính dấu phẩy động để đảm bảo tính di động.
* **super :** Từ khóa super trong Java là một biến tham chiếu được sử dụng để chỉ các đối tượng của lớp cha. Nó có thể được sử dụng để gọi phương thức của lớp cha ngay lập tức.
* **switch :** Từ khóa switch trong Java chứa câu lệnh switch thực thi mã dựa trên giá trị thử nghiệm. Câu lệnh switch kiểm tra sự bằng nhau của một biến đối với nhiều giá trị.
* [**synchronized**](https://www.javatpoint.com/synchronization-in-java)**:** Từ khóa được đồng bộ hóa của Java được sử dụng để chỉ định các phần hoặc phương thức quan trọng trong mã đa luồng.
* **this :** Java từ khóa này có thể được sử dụng để chỉ đối tượng hiện tại trong một phương thức hoặc hàm tạo.
* **throw :** Từ khóa throw trong Java được sử dụng để ném một ngoại lệ một cách rõ ràng. Từ khóa ném chủ yếu được sử dụng để ném các ngoại lệ tùy chỉnh. Nó được theo sau bởi một ví dụ.
* **throws :** Từ khóa throws trong Java được sử dụng để khai báo một ngoại lệ. Các ngoại lệ đã kiểm tra có thể được truyền bá bằng các lần ném.
* **transient:** Từ khóa tạm thời của Java được sử dụng trong tuần tự hóa. Nếu bạn xác định bất kỳ thành viên dữ liệu nào là nhất thời, nó sẽ không được đánh số thứ tự.
* **try :** Java try từ khóa được sử dụng để bắt đầu một khối mã sẽ được kiểm tra ngoại lệ. Khối thử phải được theo sau bởi khối bắt hoặc khối cuối cùng.
* **void:** Từ khóa void trong Java được sử dụng để xác định rằng một phương thức không có giá trị trả về.
* **volatile :** Từ khóa dễ bay hơi trong Java được sử dụng để chỉ ra rằng một biến có thể thay đổi không đồng bộ.
* **while :** Từ khóa while trong Java được sử dụng để bắt đầu một vòng lặp while. Vòng lặp này lặp đi lặp lại một phần của chương trình nhiều lần. Nếu số lần lặp không cố định thì nên sử dụng vòng lặp while.

1. **Control Statements**
2. **Java Control Statements**

* Trình biên dịch Java thực thi mã từ trên xuống dưới. Các câu lệnh trong mã được thực hiện theo thứ tự mà chúng xuất hiện. Tuy nhiên, Java cung cấp các câu lệnh có thể được sử dụng để kiểm soát dòng mã Java. Các câu lệnh như vậy được gọi là các câu lệnh luồng điều khiển. Đây là một trong những tính năng cơ bản của Java, cung cấp một dòng chương trình trôi chảy.
* Java cung cấp ba loại câu lệnh luồng điều khiển.
* Decision Making statements
* if statements
* switch statement
* Loop statements
* do while loop
* while loop
* for loop
* for-each loop
* Jump statements
* break statement
* continue statement

1. **Java If-else**

* if statements:sử dụng để kiểm tra điều kiện
* cú pháp:

if(condition){

//code

}

* if else statements:sử dụng để kiểm tra điều kiện. Tuy nhiên nếu điều kiện if không đúng thì xuống kiểm tra điều kiện của else.
* Cú pháp:

if(condition){

//code if condition is true

}else{

//code if condition is false

}

1. **Java Switch**

* Switch statement: dùng để thực thi câu lệnh từ nhiều điều kiện. Các điều cần nhớ khi sử dụng switch:
* Có thể có một hoặc N số giá trị chữ hoa chữ thường cho một biểu thức chuyển đổi.
* Giá trị trường hợp phải chỉ thuộc loại biểu thức chuyển đổi. Giá trị trường hợp phải là chữ hoặc hằng số. Nó không cho phép các biến.
* Các giá trị trường hợp phải là duy nhất. Trong trường hợp giá trị trùng lặp, nó sẽ hiển thị lỗi thời gian biên dịch.
* Biểu thức chuyển đổi Java phải là byte, short, int, long (với loại Wrapper của nó), enums và string.
* Mỗi câu lệnh tình huống có thể có một câu lệnh ngắt là tùy chọn. Khi điều khiển đạt đến câu lệnh ngắt, nó sẽ nhảy điều khiển sau biểu thức chuyển đổi. Nếu không tìm thấy câu lệnh break, nó sẽ thực hiện trường hợp tiếp theo.
* Giá trị trường hợp có thể có nhãn mặc định là tùy chọn.
* Cú pháp:

switch(expression){

case value1:

//code to be executed;

break; //optional

case value2:

//code to be executed;

break; //optional

default:

code to be executed if all cases are not matched;

}

1. **Java For Loop**

* Vòng lặp Java for được sử dụng để lặp lại một phần của chương trình nhiều lần. Nếu số lần lặp là cố định thì nên sử dụng vòng lặp for.
* Có ba loại vòng lặp for trong Java.
* Simple for Loop
* For-each or Enhanced for Loop
* Labeled for Loop
* Cú pháp:

for(data\_type variable : array\_name){

//code to be executed

}

1. **Java While Loop**

* while loop: dùng để lặp đi lặp lại nhiều lần điều kiện của chương trình. Ở while trước khi chạy sẽ kiểm tra điều kiện trước rồi mới chạy câu lệnh bên trong vòng while. Về while số lần lặp lại không biết trước
* Cú pháp:

while (condition){

//code to be executed

I ncrement / decrement statement

}

1. **Java Do While Loop**

* do while loop: dùng để lặp đi lặp lại điều kiện của chương trình. Tuy nhiên ở do while thì chạy trước ở do trước rồi kiểm tra điều kiện ở while vì thế do while luôn được chạy ít nhất 1 lần
* Cú pháp:

do{

//code to be executed / loop body

//update statement

}while (condition);

1. **Java Break**

* Break statement: Khi gặp một câu lệnh ngắt bên trong một vòng lặp, vòng lặp đó sẽ ngay lập tức kết thúc và điều khiển chương trình tiếp tục ở câu lệnh tiếp theo sau vòng lặp. Thường hay dùng trong switch case và các vòng lặp
* Cú pháp:

jump-statement;

break;

1. **Java Continue**

* Continue statement: Câu lệnh continue được sử dụng trong cấu trúc điều khiển vòng lặp khi bạn cần chuyển sang bước lặp tiếp theo của vòng lặp ngay lập tức. Nó có thể được sử dụng với vòng lặp for hoặc vòng lặp while.
* Cú pháp:

jump-statement;

continue;

1. **Java Comments**

* Có ba loại comments trong Java:
* Single Line Comment

//This is single line comment

* Multi Line Comment

/\*

This

is

multi line

comment

\*/

* Documentation Comment

/\*\*

\*

\*We can use various tags to depict the parameter

\*or heading or author name

\*We can also use HTML tags

\*

\*/

1. **Java Programs**

* Các chương trình Java thường được hỏi trong cuộc phỏng vấn. Các chương trình này có thể được hỏi từ câu lệnh điều khiển, mảng , chuỗi , oops, v.v. Các chương trình cơ bản của Java như chuỗi fibonacci, số nguyên tố, số giai thừa và số đối xứng thường được hỏi trong các cuộc phỏng vấn và bài kiểm tra. Tất cả các chương trình này được đưa ra với các ví dụ và đầu ra tối đa. Nếu bạn chưa quen với lập trình Java, chúng tôi khuyên bạn nên đọc hướng dẫn Java của chúng tôi trước. Hãy xem danh sách các chương trình Java.
* Các chương trình cơ bản Java
* Chương trình số Java
* Chương trình mảng Java
* Chương trình ma trận Java
* Chương trình chuỗi Java
* Các chương trình tìm kiếm và sắp xếp Java
* Chương trình chuyển đổi Java
* Chương trình mẫu
* Chương trình danh sách liên kết đơn
* Chương trình danh sách liên kết vòng
* Chương trình danh sách liên kết đôi
* Chương trình cây

1. **Tài liệu tham khảo**

*[1]**Java Tutorial | Learn Java Programming - javatpoint*. (n.d.). [www.javatpoint.com](http://www.javatpoint.com).

https://www.javatpoint.com/java-tutorial