# LỜI MỞ ĐẦU

Xin chào, mình là Dương Văn Toàn, đây là những dòng đầu tiên trong chương khởi động của mình về việc học tập ngôn ngữ lập trình Java.

Rất mong các bạn sẽ theo dõi mình nhé !!!

# OOP Basics - For basic concepts (such as Java syntax, data types, variables, and operators).

## 1.Giới thiệu về Java

### Java là gì, là một ngôn ngữ lập trình phổ biến được tạo ra vào năm 1995. Hệ điều hành này thuộc sở hữu của Oracle và có hơn 3 tỉ thiết bị chạy Java.

Nó được sử dụng cho:

* Ứng dụng di động (đặc biệt là ứng dụng Android)
* Ứng dụng máy tính để bàn
* Ứng dụng web
* Máy chủ web và máy chủ ứng dụng
* Trò chơi
* Kết nối cơ sở dữ liệu
* Và còn nhiều hơn thế nữa!

### Tại sao nên sử dụng Java?

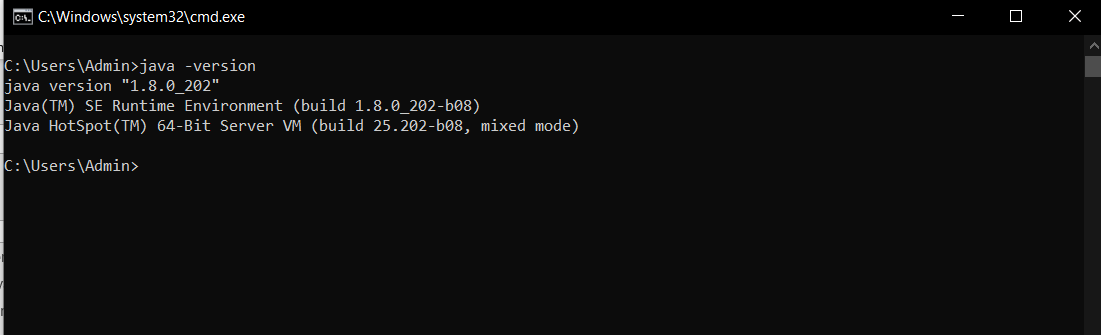
* Java hoạt động trên nhiều nền tảng khác nhau (Windows, Mac, Linux, Raspberry Pi, v.v.)
* Đây là một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới
* Nó có nhu cầu lớn trong thị trường việc làm hiện nay
* Nó dễ học và dễ sử dụng
* Nó là mã nguồn mở và miễn phí
* Nó an toàn, nhanh chóng và mạnh mẽ
* Nó có sự hỗ trợ của cộng đồng lớn (hàng chục triệu nhà phát triển)
* Java là ngôn ngữ hướng đối tượng cung cấp cấu trúc rõ ràng cho các chương trình và cho phép tái sử dụng mã, giúp giảm chi phí phát triển
* Vì Java gần với C++ và C# nên các lập trình viên có thể dễ dàng chuyển sang Java hoặc ngược lại

## 2.Cài đặt Java

Tiếp theo chúng ta sẽ đến với các bước Cài đặt Java nhé

Một số máy tính có thể đã cài sẵn Java.

Để kiểm tra xem bạn đã cài đặt Java trên PC Windows hay chưa, nhấn tổ hợp phím Window + R, gõ cmd , Terminal mở ra gõ lệnh Java – version



Đây là của máy tính của mình. Lệnh cls để xoá sạch trên cmd

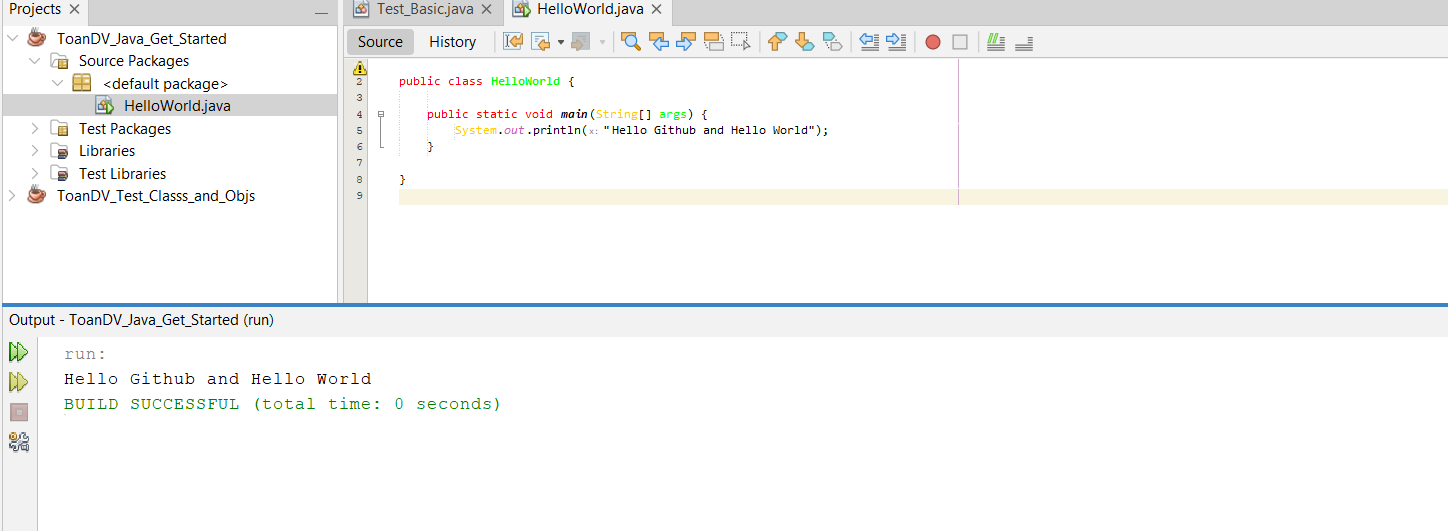
Và biên dịch trên cmd của mình javac ý lại là phiên bản mới hơn 

Thế nên sẽ gặp lỗi bên dưới chút mình sẽ đề cập.

Mình sẽ viết mã trên Apache Netbeans IDE 17 nhé.

Còn đây là biên dịch bằng Netbean

Với Class HelloWorld.



Mình đã tạo ra 1 file Notepad và đặt tên là



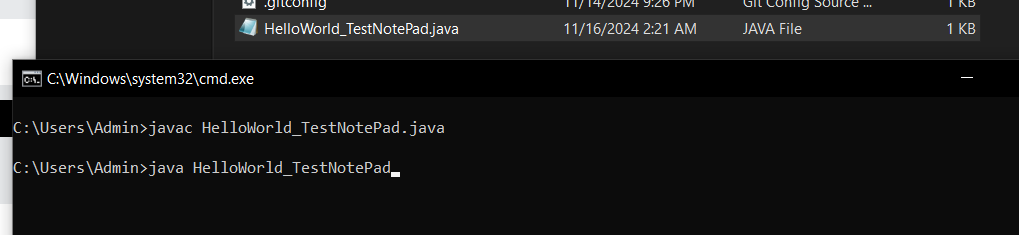
HelloWorld\_TestNotePad.java.

Tổ hợp phím Window + R, gõ cmd, javac HelloWorld\_TestNotePad ( lưu ý cả tên tệp và tên lớp phải giống nhau )

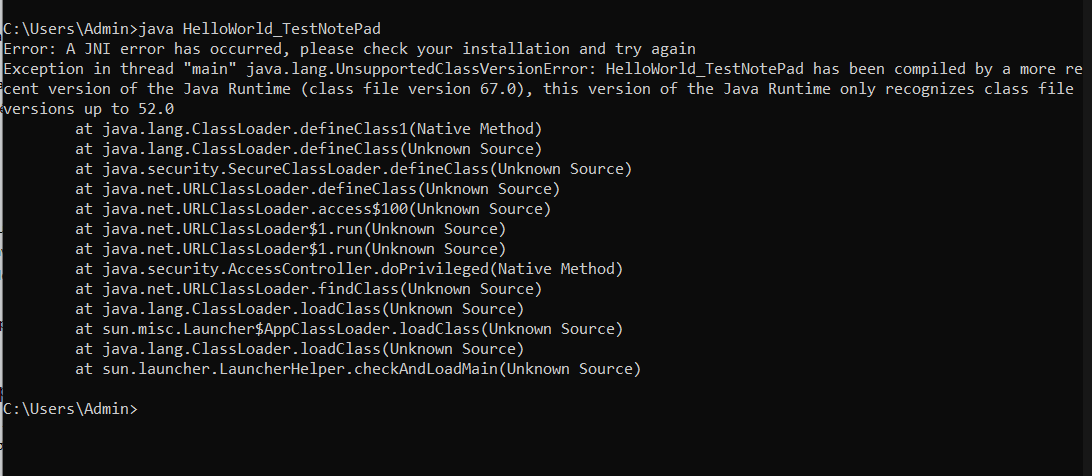


Điều này sẽ biên dịch mã của bạn. Nếu không có lỗi trong mã, dấu nhắc lệnh sẽ đưa bạn đến dòng tiếp theo.

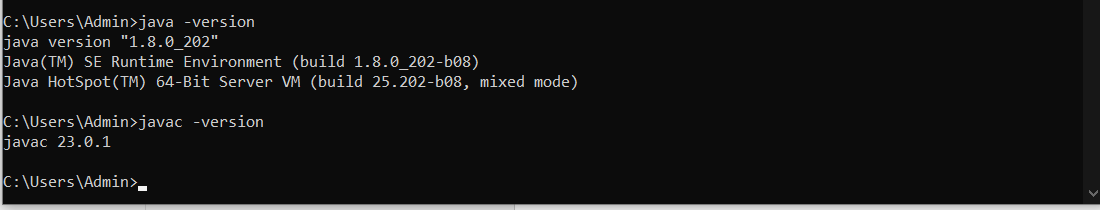
Kế đó mình gõ lệnh : java HelloWorld\_TestNotePad.java



Đây chính là cửa sổ cmd và tab của nơi chứa file HelloWorld\_TestNotePad.java

Và nó sẽ báo lỗi này 

Mình cần kiểm tra lại cả phiên bản java và javac



Lỗi đó đây.

### Sự khác biệt giữa java và javac

* **java**: Là công cụ dùng để **chạy chương trình Java**. Nó sử dụng Java Runtime Environment (JRE) để thực thi file bytecode (.class).
* **javac**: Là trình biên dịch dùng để **chuyển mã nguồn Java** (.java) thành file bytecode (.class).

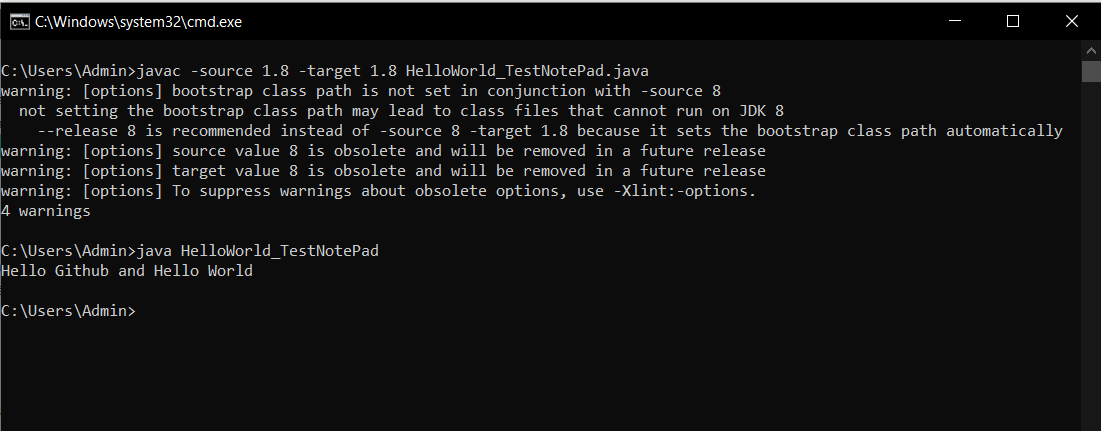
Nếu muốn chương trình chạy đúng với phiên bản hiển thị từ java -version, hãy:

* Biên dịch lại mã nguồn với phiên bản javac tương thích (

javac -source 1.8 -target 1.8 HelloWorld\_TestNotePad.java

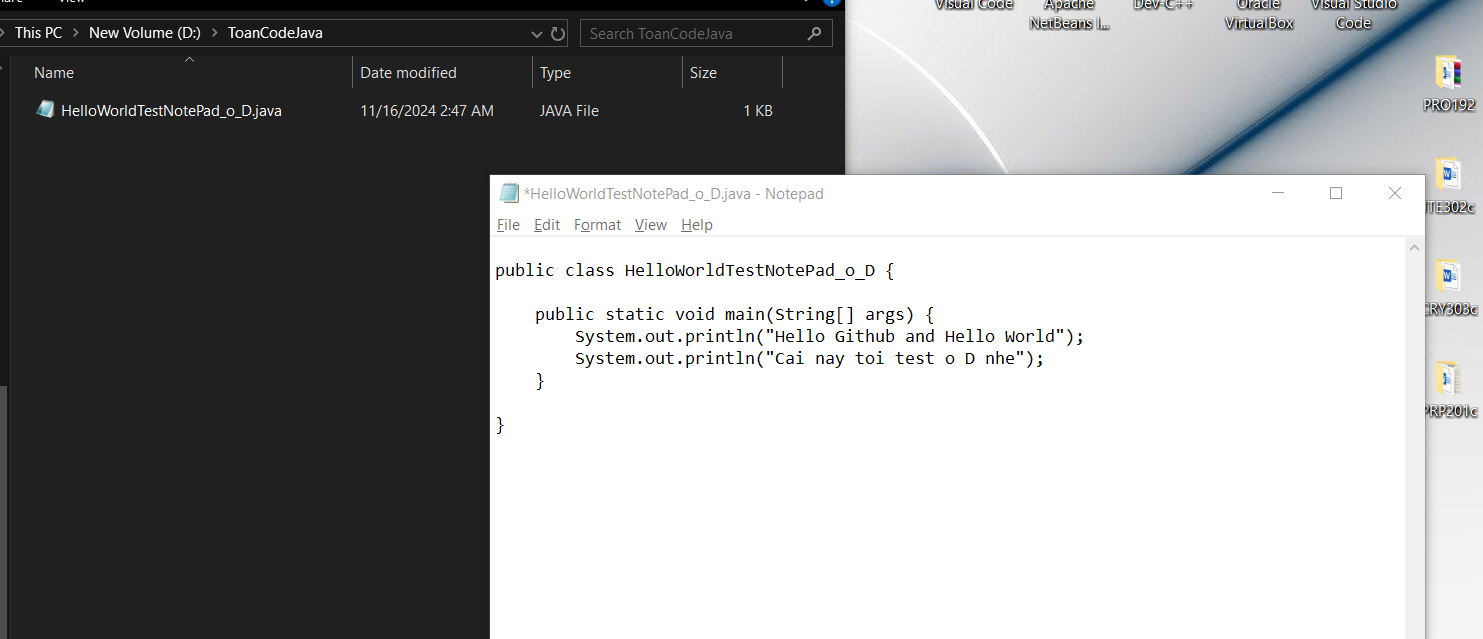
, nếu java là Java 8).

* Hoặc nâng cấp java lên phiên bản mới hơn để chạy mã đã biên dịch.

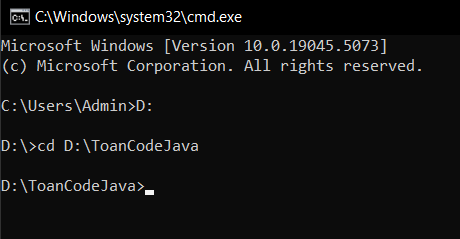
và đây là kết quả ta mong chờ hihi.

Bây giờ là ổ D: nè, đường dẫn D:\ToanCodeJava nhé, tên tệp là, tràn rammmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm, moáaaaaaaaaaaaaaaaa

OK ĐƯỢC RỒI, TIẾP TỤC NÀY ,

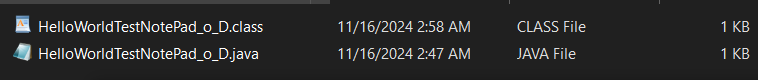


Trong cmd, mình gõ lệnh D: để nhảy vào ổ D, xong điều hướng đến ổ kìa bằng lệnh cd D:\ToanCodeJava



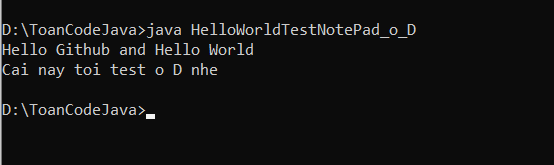
Tiếp theo mình sẽ javac HelloWorldTestNotePad\_o\_D.java và nó sẽ lại lỗi biên dịch nên mình chuyển trước luôn thành

javac -source 1.8 -target 1.8 HelloWorldTestNotePad\_o\_D.java

và nó đã tạo ra tệp nữa 

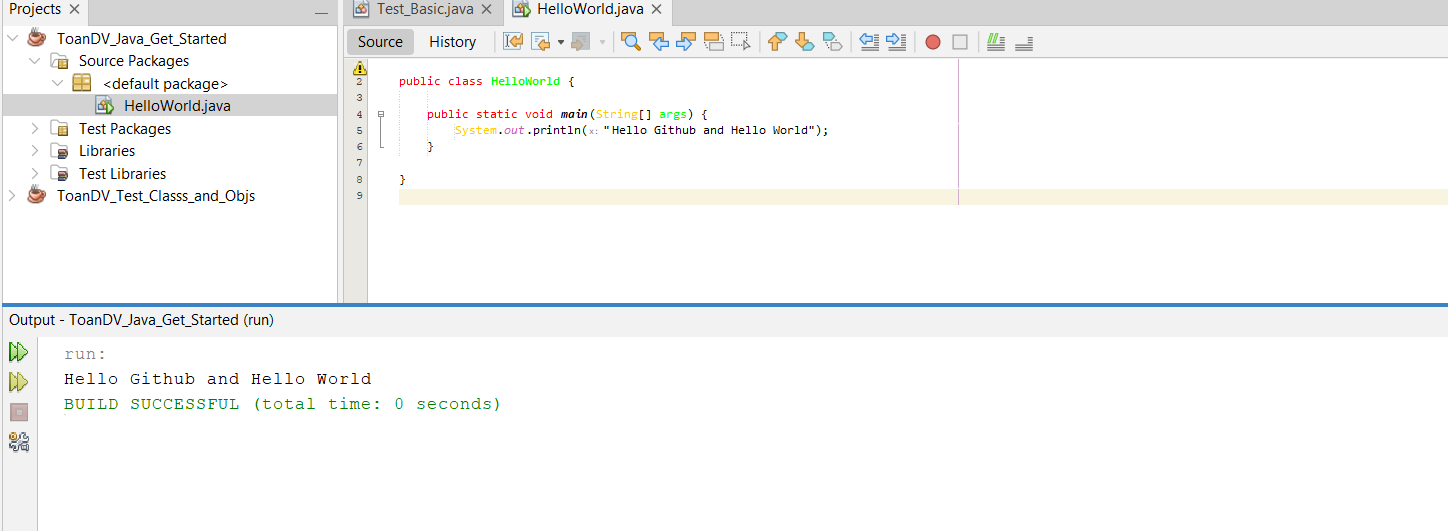
rồi giờ mình cho chạy nhé

java HelloWorldTestNotePad\_o\_D



## 3.Cú pháp Java ( Java Syntax )

Mỗi dòng mã chạy trong Java phải nằm trong trong class. Và tên lớp phải luôn bắt đầu bằng chữ cái đầu tiên viết hoa. Trong ví dụ của mình thì đặt tên lớp **là HelloWorld** .



### Nhưng tại sao nên dùng chữ cái hoa?

1. **Quy ước lập trình**: Theo quy chuẩn Java, tên lớp nên bắt đầu bằng chữ hoa để phân biệt dễ dàng với các biến hoặc phương thức (bắt đầu bằng chữ thường).
2. **Dễ đọc và bảo trì mã**: Nếu bạn làm việc nhóm hoặc chia sẻ mã, việc tuân thủ quy ước giúp người khác dễ hiểu hơn.
3. **Cộng đồng lập trình**: Đây là một chuẩn được hầu hết các lập trình viên Java trên thế giới sử dụng, nên tuân theo sẽ giúp bạn phù hợp với tài liệu, công cụ tự động kiểm tra mã, và đồng nghiệp.

### Phương pháp chính

Phương pháp này main()là bắt buộc và bạn sẽ thấy nó trong mọi chương trình Java:

public static void **main**(String[] args)

Bất kỳ mã nào bên trong main()phương thức sẽ được thực thi. Đừng lo lắng về các từ khóa trước và sau nó. Bạn sẽ biết chúng từng chút một khi đọc hướng dẫn này.

Hiện tại, bạn chỉ cần nhớ rằng mọi chương trình Java đều có class tên phải khớp với tên tệp và mọi chương trình phải chứa main()phương thức.

### System.out.println()

Trong main()phương pháp này, chúng ta có thể sử dụng println() phương pháp để in một dòng văn bản ra màn hình

public static void main(String[] args) {

System.out.println("Toan dang hoc Java ne");

}

## 4.Java Output / Print

### In văn bản(Print Text) :  sử dụng println()phương thức (method ) này để xuất giá trị hoặc in văn bản trong Java

**Dấu ngoặc kép ( Double Quotes)**

Văn bản phải được đặt trong dấu ngoặc kép ""

VD : System.out.println("Hello You");

**Phương thức Print() (** The Print() Method**)**

Ngoài ra còn có một print()phương pháp tương tự như println().

Sự khác biệt duy nhất là nó không chèn dòng mới vào cuối đầu ra

VD : system.out.print("Hello World! ");

### In Số ( Print Numbers )

sử dụng println()phương pháp này để in số

Tuy nhiên, không giống như văn bản, chúng ta không đặt số bên trong dấu ngoặc kép.

VD : System.out.println(3)

System.out.println(999)

Có thể thực hiện các phép tính toán học bên trong println()phương thức:

System.out.println(3 + 3);

System.out.println(6 + 99);

System.out.println(3 \* 3 - 44);

## 5.Chú thích (Java Comments)

Sử dụng chú thích để giải thích mã Java và làm cho mã dễ đọc hơn (make it more readable) . Nó cũng có thể được sử dụng để ngăn chặn việc thực thi khi kiểm tra mã thay thế (It can also be used to prevent execution when testing alternative code )

### Chú thích một dòng ( Single-line Comments )

Chú thích một dòng bắt đầu bằng hai dấu gạch chéo ( //).

Bất kỳ văn bản nào ở giữa //và cuối dòng đều bị Java bỏ qua (sẽ không được thực thi).

Ví dụ này sử dụng chú thích một dòng trước một dòng mã:

// This is a comment

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

Và chú thích một dòng cuối một dòng mã:

System.out.println("Hello World"); // This is a comment

### Chú thích nhiều dòng trong Java ( Java Multi-line Comments)

Chú thích nhiều dòng bắt đầu bằng /\*và kết thúc bằng \*/.

Bất kỳ văn bản nào nằm giữa /\*và \*/sẽ bị Java bỏ qua.

Ví dụ này sử dụng chú thích nhiều dòng (khối chú thích) để giải thích mã:

/\* The code below will print the words Hello World

to the screen, and it is amazing \*/

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

## 6.CÁC BIẾN TRONG JAVA ( Java Variables )

### Biến là nơi chứa đựng giá trị dữ liệu ( Variables are containers for storing data values )

Trong Java, có các loại biến khác nhau, ví dụ:

* **String** - lưu trữ văn bản, chẳng hạn như "Hello". Các giá trị String được bao quanh bởi dấu ngoặc kép.
* **int** - lưu trữ số nguyên (số không có phần thập phân), chẳng hạn như 123 hoặc -123.
* **float** - lưu trữ số thực (số thập phân), chẳng hạn như 19.99 hoặc -19.99.
* **char** - lưu trữ một ký tự đơn lẻ, chẳng hạn như 'a' hoặc 'B'. Các giá trị char được bao quanh bởi dấu ngoặc đơn.
* **boolean** - lưu trữ các giá trị với hai trạng thái: **true** (đúng) hoặc **false** (sai).

#### Khai báo (Tạo) Biến

Để tạo một biến, bạn cần chỉ định kiểu dữ liệu và gán giá trị cho nó.

Cú pháp:

type variableName = value;

**type**: Một trong các kiểu dữ liệu của Java (như int hoặc String).

**variableName**: Tên của biến (như x hoặc name).

**dấu bằng (=)**: Dùng để gán giá trị cho biến.

Để tạo một biến lưu trữ văn bản, hãy xem ví dụ sau:

#### **Ví dụ 1: Tạo biến lưu trữ văn bản**

Tạo một biến có tên **name** với kiểu **String** và gán giá trị "John" cho nó. Sau đó, sử dụng hàm println() để in giá trị của biến ra màn hình.

String name = "John";

System.out.println(name);

// Kết quả: John

#### **Ví dụ 2: Tạo biến lưu trữ số**

Tạo một biến có tên **myNum** với kiểu **int** và gán giá trị **15** cho nó.

int myNum = 15;

System.out.println(myNum);

// Kết quả: 15

#### **Ví dụ 3: Khai báo biến không gán giá trị ban đầu**

Bạn cũng có thể khai báo biến mà không gán giá trị ngay lập tức, và sau đó gán giá trị cho nó:

int myNum; // Khai báo biến myNum

myNum = 15; // Gán giá trị 15 cho myNum

System.out.println(myNum);

// Kết quả: 15

#### **Ví dụ 4 : Khai báo một giá trị và ghi đè lên để in ra kết quả khác**

**int Test4 = 99; // Khai báo biến Test4 với gán giá trị 99**

**System.out.println("Test4 luc chua bi ghi de ="+ Test4);**

**Test4 = 999; // Gán đè giá trị 999 cho Test4**

**System.out.println(Test4 + "da bi ghi de");**

Kết quả : Test4 luc chua bi ghi de =99

999da bi ghi de

#### Ví dụ 5 : Biến cuối cùng

#### Nếu bạn không muốn người khác (hoặc chính bạn) ghi đè lên các giá trị hiện có, hãy sử dụng final từ khóa (điều này sẽ khai báo biến là "final" hoặc "constant", nghĩa là không thể thay đổi và chỉ đọc) .

VD:

final int Test5 = 15;

//Test5 = 20; // will generate an error: cannot assign a value to a final variable

#### **Ghi chú quan trọng**

* Tên biến phải tuân theo quy tắc đặt tên (ví dụ: không được bắt đầu bằng số, không chứa ký tự đặc biệt ngoại trừ \_ hoặc $).
* Biến cần được khởi tạo (gán giá trị) trước khi sử dụng, nếu không sẽ gây lỗi khi biên dịch.

### In biến trong Java ( Display Variables )

Hiển thị biến

Phương pháp này println()thường được sử dụng để hiển thị các biến.

Để kết hợp cả văn bản và biến ( text and a variable ) , hãy sử dụng ký tự +

### Khai báo nhiều biến (Declare Many Variables)

Để khai báo nhiều biến cùng loại ( same type ) , bạn có thể sử dụng danh sách được phân tách bằng dấu phẩy

int x = 5, y = 6, z = 50;

System.out.println(x + y + z);

Một Giá Trị Cho Nhiều Biến

Bạn cũng có thể gán cùng một giá trị cho nhiều biến trong một dòng:

int x1, y1, z1;

x1 = y1 = z1 = 50;

System.out.println(x1 + y1 + z1);

### Định danh trong Java (Java Identifiers)

**Tất cả các Biến** trong Java phải được xác định bằng những tên duy nhất.Những tên duy nhất này được gọi là **định danh** (identifiers).Định danh có thể là các tên ngắn (như x và y) hoặc các tên mang tính mô tả hơn (như age, sum, totalVolume).

**Lưu ý:** Nên sử dụng các tên mang tính mô tả để tạo mã nguồn dễ hiểu và dễ bảo trì.

All Java variables must be identified with unique names.These unique names are called identifiers.Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code.

Các quy tắc chung để đặt tên biến là:

* Tên có thể chứa chữ cái, chữ số, dấu gạch dưới (\_), và ký hiệu đô la ($).
* Tên phải bắt đầu bằng một chữ cái.
* Tên nên bắt đầu bằng chữ thường và không được chứa khoảng trắng.
* Tên cũng có thể bắt đầu bằng ký hiệu $ hoặc \_.
* Tên phân biệt chữ hoa và chữ thường (ví dụ: myVar và myvar là hai biến khác nhau).
* Các từ dành riêng (như từ khóa trong Java, ví dụ: int hoặc boolean) không được sử dụng làm tên.

The general rules for naming variables are:

* Names can contain letters, digits, underscores, and dollar signs
* Names must begin with a letter
* Names should start with a lowercase letter, and cannot contain whitespace
* Names can also begin with $ and \_
* Names are case-sensitive ("myVar" and "myvar" are different variables)
* Reserved words (like Java keywords, such as int or boolean) cannot be used as names

### Biến Java - Ví dụ thực tế

Trong mấy file hết nhé.

## 7.Các kiểu dữ liệu trong Java

Kiểu dữ liệu Java

Như đã giải thích ở chương trước, một biến trong Java phải là một kiểu dữ liệu được chỉ định

Example

int myNum = 5; // Integer (whole number)

float myFloatNum = 5.99f; // Floating point number

char myLetter = 'D'; // Character

boolean myBool = true; // Boolean

String myText = "Hello"; // String

Kiểu dữ liệu được chia thành hai nhóm:

### Kiểu dữ liệu nguyên thủy - bao gồm byte, short, int, long, float, double, boolean và char

Các kiểu dữ liệu không nguyên thủy - chẳng hạn như String, Mảng và Lớp ( String, Arrays and Classes )

Kiểu dữ liệu nguyên thủy (Primitive Data Types).Có tám kiểu dữ liệu nguyên thủy trong Java:

**Kiểu dữ liệu Kích thước Mô tả**

**byte** 1 byte Lưu trữ số nguyên từ -128 đến 127

**short** 2 bytes Lưu trữ số nguyên từ -32,768 đến 32,767

**int** 4 bytes Lưu trữ số nguyên từ -2,147,483,648 đến 2,147,483,647

**long** 8 bytes Lưu trữ số nguyên từ -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807

**float** 4 bytes Lưu trữ số thập phân. Đủ để lưu 6 đến 7 chữ số thập phân

**double**  8 bytes Lưu trữ số thập phân. Đủ để lưu 15 đến 16 chữ số thập phân

**boolean** 1 bit Lưu trữ giá trị đúng (true) hoặc sai (false)

**char** 2 bytes Lưu trữ một ký tự hoặc giá trị ASCII

Các kiểu dữ liệu nguyên thủy này được sử dụng để định nghĩa các giá trị cơ bản trong Java

A primitive data type specifies the size and type of variable values, and it has no additional methods. There are eight primitive data types in Java:

**Data Type** Size Description

**byte** 1 byte Stores whole numbers from -128 to 127

**short** 2 bytes Stores whole numbers from -32,768 to 32,767

**int** 4 bytes Stores whole numbers from -2,147,483,648 to 2,147,483,647

**long** 8 bytes Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

**float** 4 bytes Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits

**double** 8 bytes Stores fractional numbers. Sufficient for storing 15 to 16 decimal digits

**boolean** 1 bit Stores true or false values

**char** 2 bytes Stores a single character/letter or ASCII values

Số ( Numbers )

Các loại số nguyên thủy (Primitive number types) được chia thành hai nhóm:

Kiểu số nguyên(**Integer types)** lưu trữ số nguyên, dương hoặc âm (như 123 hoặc -456), không có phần thập phân. Các kiểu hợp lệ là byte, short, int và long. Kiểu nào bạn nên sử dụng phụ thuộc vào giá trị số.

Kiểu dữ liệu được sử dụng để lưu trữ số thập phân hoặc số có phần lẻ (Floating Point Types) biểu diễn các số có phần thập phân, chứa một hoặc nhiều số thập phân. Có hai kiểu: float và double. Kiểu dữ liệu floatvà doublecó thể lưu trữ số phân số. Lưu ý rằng bạn nên kết thúc giá trị bằng "f" cho số float và "d" cho số double

### ****Sự khác biệt giữa**** float ****và**** double

1. **Kích thước và độ chính xác:**
   * float có kích thước nhỏ hơn và độ chính xác thấp hơn so với double.
   * double thường được sử dụng khi yêu cầu độ chính xác cao hơn, đặc biệt trong các tính toán khoa học.
2. **Hiệu suất:**
   * float tiêu tốn ít bộ nhớ hơn và thường nhanh hơn trong các ứng dụng yêu cầu ít độ chính xác.
   * double sử dụng nhiều bộ nhớ hơn nhưng hỗ trợ các tính toán phức tạp với độ chính xác cao.

### ****Lưu ý khi sử dụng****

 **Ký tự hậu tố f hoặc F:** Khi khai báo số kiểu float, cần thêm hậu tố f hoặc F để trình biên dịch nhận diện. Mặc định, số thập phân không có hậu tố sẽ được coi là kiểu double.  
Ví dụ:

float pi = 3.14f; // Đúng

double piDouble = 3.14; // Mặc định là double

 **Số thực nhỏ hoặc rất lớn:** Sử dụng ký pháp khoa học (scientific notation).  
Ví dụ:

double bigNumber = 1.23e10; // 1.23 x 10^10

float smallNumber = 4.56e-5f; // 4.56 x 10^-5

* Cả float và double lưu trữ toàn bộ số dưới dạng **floating-point representation** (dạng dấu chấm động).
* Dạng này biểu diễn số dưới công thức: So^ˊ=Cơso^ˊ×2So^ˊmu~Ví dụ:
* Số **1234.5678** được lưu trữ bằng cách chia thành:
  + **Cơ số:** 1.2345678
  + **Số mũ:** 10^3  
    Điều này giúp các kiểu float và double lưu trữ được cả số lớn và số nhỏ mà không phân biệt phần trước hay sau dấu phẩy.

### Kiểu dữ liệu Boolean của Java

Java có một boolean kiểu dữ liệu chỉ có thể lấy các giá trị true hoặc false.

### Ký **tự trong Java (Java Characters** )

Trong Java, **ký tự** được đại diện bởi kiểu dữ liệu nguyên thủy char. Ký tự này dùng để lưu trữ một ký tự đơn, như một chữ cái, một số, hoặc một ký hiệu.

**Các điểm chính về kiểu** char**:**

1. **Kích thước:**
   * char chiếm 2 byte (16-bit), vì Java sử dụng mã hóa **Unicode** để hỗ trợ các ký tự từ nhiều ngôn ngữ trên thế giới.
2. **Phạm vi giá trị:**
   * Ký tự char có thể lưu trữ giá trị từ \u0000 (0) đến \uffff (65,535), tương ứng với bảng mã Unicode.
3. **Ký hiệu và cách biểu diễn:**
   * Ký tự được đặt trong dấu nháy đơn ' '.
   * Có thể biểu diễn trực tiếp ký tự ('A', '1', '$') hoặc bằng mã Unicode ('\u0041' đại diện cho 'A').

Kiểu Strings

Kiểu Stringdữ liệu được sử dụng để lưu trữ một chuỗi ký tự (văn bản). Giá trị chuỗi phải được bao quanh bởi dấu ngoặc kép.

### Kiểu Dữ Liệu Không Nguyên Thủy (Non-Primitive Data Types)

Kiểu dữ liệu không nguyên thủy còn được gọi là kiểu tham chiếu ( reference types ) , vì chúng tham chiếu đến các đối tượng.

Sự khác biệt chính giữa kiểu dữ liệu nguyên thủy và kiểu dữ liệu không nguyên thủy là:

* Kiểu dữ liệu nguyên thủy trong Java được định nghĩa sẵn và tích hợp(predefined and built) trong ngôn ngữ, trong khi kiểu dữ liệu không nguyên thủy do lập trình viên tự tạo (trừ kiểu String).
* Kiểu dữ liệu không nguyên thủy có thể được sử dụng để gọi các phương thức thực hiện các thao tác nhất định(methods to perform certain operations), trong khi kiểu dữ liệu nguyên thủy thì không thể.
* Kiểu dữ liệu nguyên thủy luôn bắt đầu bằng chữ cái viết thường(*lowercase letter*) (ví dụ: int), trong khi kiểu dữ liệu không nguyên thủy thường bắt đầu bằng chữ cái viết hoa(*uppercase letter* ) (ví dụ: String).
* Kiểu dữ liệu nguyên thủy luôn chứa một giá trị, trong khi kiểu dữ liệu không nguyên thủy có thể có giá trị là null.

Các ví dụ về kiểu dữ liệu không nguyên thủy bao gồm **String**, **Mảng (Arrays)**, **Lớp (Classes)**, v.v. Bạn sẽ tìm hiểu thêm về những loại này trong các chương tiếp theo.

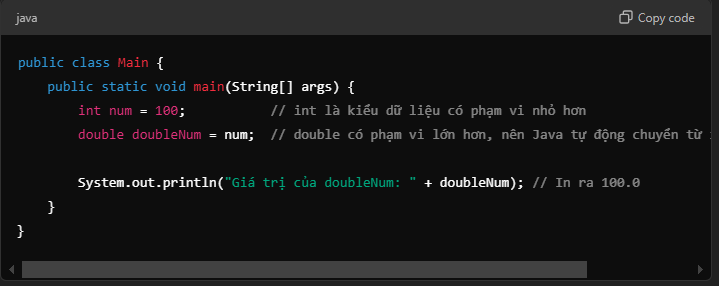
## 8.Chuyển đổi Kiểu Dữ Liệu(Java Type Casting)

Hay còn gọ là ép kiểu, là khi bạn gán giá trị của một kiểu dữ liệu nguyên thủy cho một kiểu khác.

Trong Java, có hai loại ép kiểu:

**Ép kiểu ngầm định** (Widening Casting ) trong Java xảy ra khi bạn chuyển một giá trị từ kiểu dữ liệu có phạm vi nhỏ hơn sang kiểu dữ liệu có phạm vi lớn hơn. Java tự động thực hiện việc chuyển đổi này mà không cần lập trình viên can thiệp.

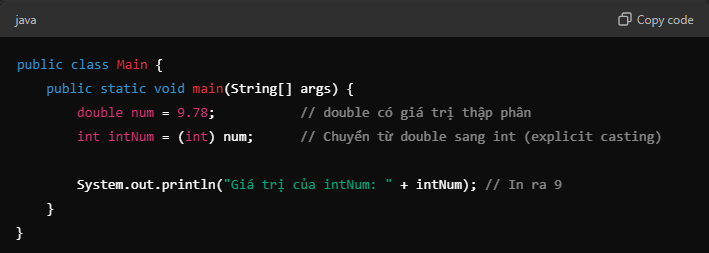
Ví dụ:



Trong ví dụ trên, int tự động chuyển sang double mà không cần lập trình viên can thiệp.

**Ép kiểu tường minh trong Java( Narrowing Casting)**

Chuyển đổi kiểu tường minh xảy ra khi bạn chuyển một giá trị từ kiểu dữ liệu có phạm vi lớn hơn sang kiểu dữ liệu có phạm vi nhỏ hơn. Khi thực hiện chuyển đổi kiểu tường minh, bạn cần sử dụng toán tử cast (type) để chỉ định kiểu dữ liệu đích.

Trong ví dụ này, giá trị của num là 9.78 (kiểu double), nhưng khi chuyển sang int, phần thập phân sẽ bị loại bỏ, và kết quả sẽ là 9. Lập trình viên phải dùng toán tử cast (int) để thực hiện chuyển đổi kiểu tường minh.

## 9.Toán Tử trong Java (Java Operators )

là các ký hiệu dùng để thực hiện các phép toán hoặc thao tác trên các biến và giá trị. Java hỗ trợ nhiều loại toán tử khác nhau, bao gồm toán tử số học, so sánh, logic, gán, và các toán tử khác. Dưới đây là các loại toán tử phổ biến trong Java:

Java chia các toán tử thành các nhóm sau:

Toán tử số học

Toán tử gán

Toán tử so sánh

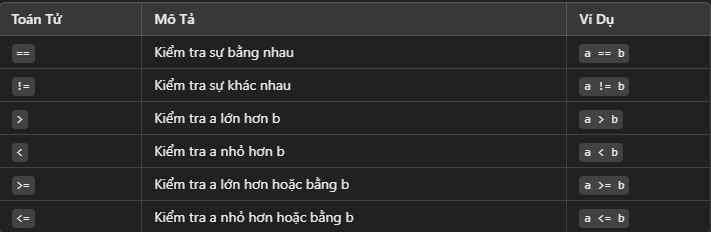
Toán tử logic

Toán tử bitwise

### 1. Toán Tử Số Học (Arithmetic Operators)

Toán tử số học dùng để thực hiện các phép toán cơ bản như cộng, trừ, nhân, chia, và chia lấy dư. 

### 2. Toán Tử So Sánh (Relational Operators)

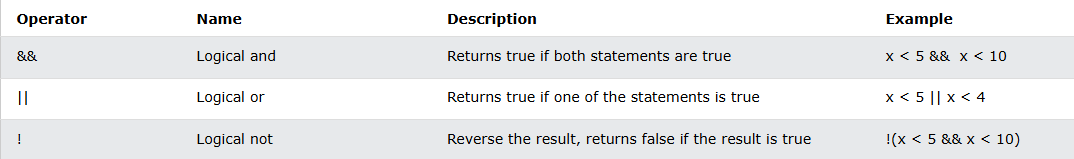
Toán tử so sánh được dùng để so sánh các giá trị và trả về giá trị boolean (true hoặc false). 

### 3. Toán Tử Logic (Logical Operators)

Toán tử logic dùng để thực hiện các phép toán logic trên các giá trị boolean.

Kiểm tra giá trị truehoặc false bằng toán tử logic.

Các toán tử logic được sử dụng để xác định logic giữa các biến hoặc giá trị



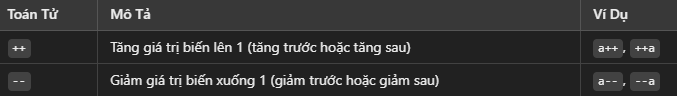
### Toán Tử Gán (Assignment Operators)

Toán tử gán dùng để gán giá trị cho biến.



### 5. Toán Tử Tăng Giảm (Increment/Decrement Operators)

Toán tử tăng giảm được sử dụng để tăng hoặc giảm giá trị của biến.



### 6.Toán Tử Điều Kiện (Ternary Operator)

Toán tử điều kiện là một phép toán rút gọn của câu lệnh if-else. Nó có cấu trúc như sau:

condition ? value\_if\_true : value\_if\_false

### 7. Toán Tử Bitwise (Bitwise Operators)

Toán tử bitwise thực hiện các phép toán ở mức bit trên các kiểu số nguyên.



### 8. Toán Tử instanceof

Toán tử này kiểm tra xem một đối tượng có phải là thể hiện của một lớp hoặc interface nào đó hay không.

VD:

String str = "Hello";

System.out.println(str instanceof String); // true

### Tóm Tắt

Toán tử số học: Sử dụng cho phép toán cơ bản như cộng, trừ, nhân, chia.

Toán tử so sánh: Dùng để so sánh các giá trị (==, !=, >, <, ...).

Toán tử logic: Dùng cho phép toán logic (&&, ||, !).

Toán tử gán: Dùng để gán giá trị cho biến (=, +=, -=, \*=, ...).

Toán tử tăng giảm: Tăng hoặc giảm giá trị biến (++ và --).

Toán tử điều kiện: Một phép toán rút gọn của câu lệnh if-else.

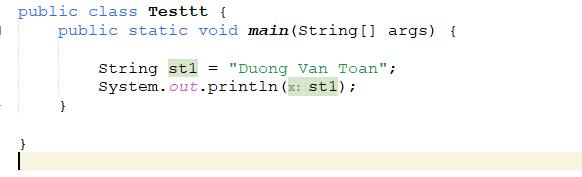
Toán tử bitwise: Thực hiện phép toán trên các bit của kiểu dữ liệu số nguyên.

Toán tử instanceof: Kiểm tra kiểu đối tượng.

Java cung cấp nhiều loại toán tử để giúp lập trình viên thực hiện các phép toán và thao tác trên dữ liệu một cách dễ dàng và hiệu quả.

## 10.Chuỗi trong Java ( Java Strings )

### Một String biến chứa một tập hợp các ký tự được bao quanh bởi dấu ngoặc kép.



### Tính chiều dài chuỗi (String Length )

String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

System.out.println("The length of the txt string is: " + txt.length());

### Phương pháp khác như viết hoa cả chuỗi hoặc viết thường cả chuỗi ( toUpperCase and toLowerCase )

String txt = "Hello World";

System.out.println(txt.toUpperCase()); // Outputs "HELLO WORLD"

System.out.println(txt.toLowerCase()); // Outputs "hello world"

### Tìm một ký tự trong một chuỗi

Phương pháp này indexOf() trả về vị trí của lần xuất hiện đầu tiên của văn bản được chỉ định trong chuỗi (bao gồm cả khoảng trắng).

// 01234567

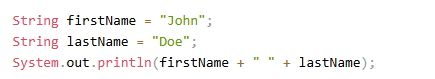
String txt = "Please locate where 'locate' occurs!";

System.out.println(txt.indexOf("locate")); // Outputs 7

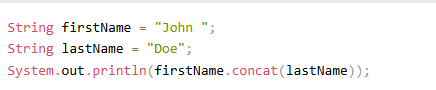
Bạn có thể tìm lần xuất hiện tiếp theo bằng cách cung cấp một chỉ số bắt đầu, ví dụ:

System.out.println(txt.indexOf("o", 9)); // Tìm "o" sau vị trí 8

### Nối chuỗi trong Java ( Java String Concatenation )



Nó cũng được viết theo cách



## 11.Math trong Java

Java cung cấp lớp Math trong gói java.lang, chứa các phương thức toán học để thực hiện các phép tính như lượng giác, số mũ, căn bậc hai, làm tròn, v.v. Lớp này rất tiện lợi vì các phương thức của nó đều là static (tĩnh), do đó bạn có thể gọi trực tiếp mà không cần tạo đối tượng.



## 12.Boolean trong Java

Trong Java, boolean là một kiểu dữ liệu nguyên thủy được sử dụng để biểu diễn hai trạng thái logic: true hoặc false. Kiểu boolean thường được dùng trong các biểu thức điều kiện, kiểm tra logic, và các câu lệnh điều kiện như if, while, hoặc for.

Các phép toán so sánh



Và các phép toán logic như

&&(AND) (5 > 3) && (10 > 5) true

|| (OR) hoặc

!(NOT) !(5 > 3) false

Lưu ý:

Thứ tự ưu tiên của các phép toán logic:

! (NOT) được ưu tiên trước.

&& (AND) được ưu tiên hơn || (OR).

Dùng dấu ngoặc () để rõ ràng và thay đổi thứ tự nếu cần.

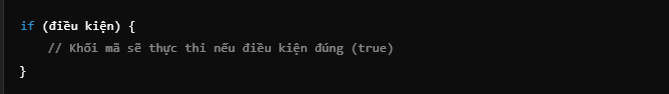
Boolean rất hữu ích trong lập trình điều kiện, giúp kiểm tra logic và điều khiển luồng thực thi chương trình.

## 13. Java If ... Else

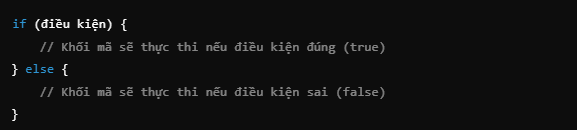
Câu lệnh if...else trong Java được sử dụng để thực hiện các thao tác có điều kiện. Nó cho phép chương trình kiểm tra một điều kiện và thực thi các đoạn mã khác nhau dựa trên việc điều kiện đó đúng (true) hay sai (false).

Cấu trúc của if trong Java:

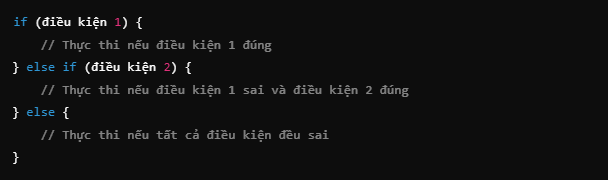
### A. Câu lệnh if:



### B. Câu lệnh if...else

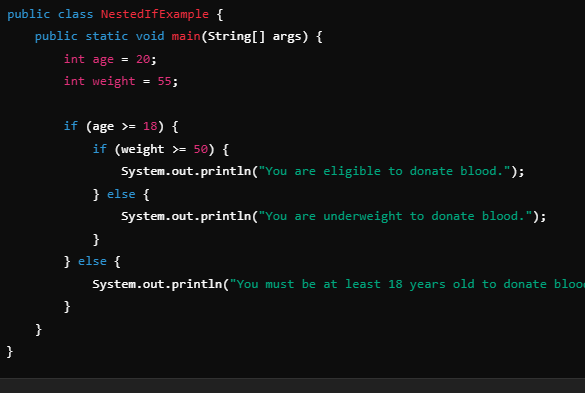


### C. if...else if...else



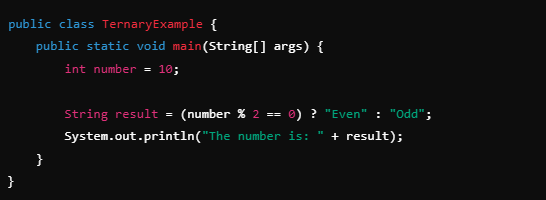
### D. Câu lệnh lồng nhau (Nested if):

Bạn có thể đặt một câu lệnh if...else bên trong một câu lệnh if...else khác



Ternary Operator (Toán tử 3 ngôi)

variable = (điều kiện) ? giá trị\_nếu\_đúng : giá trị\_nếu\_sai;

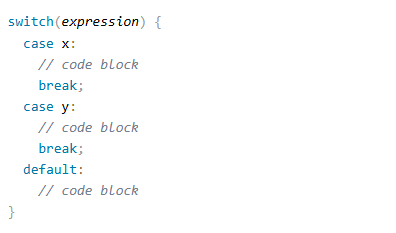


## 14. Switch trong Java

Thay vì viết nhiều if..else câu lệnh, bạn có thể sử dụng switchcâu lệnh.

Câu lệnh này switchchọn một trong nhiều khối mã để thực thi

### Cú pháp



**Từ khóa break**

Khi Java đạt đến break, nó sẽ thoát khỏi switch.

Điều này sẽ dừng việc thực thi thêm mã và thử nghiệm trường hợp (execution of more code and case testing) bên trong block.

Khi tìm thấy sự trùng khớp và công việc đã hoàn tất, đã đến lúc break. Không cần phải thử nghiệm thêm nữa(no need for more testing)

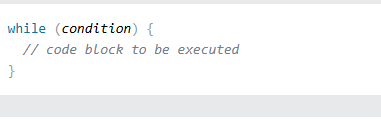
**Từ khóa default**

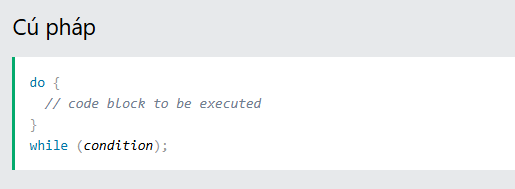
Từ default chỉ định một số mã để chạy nếu không có trường hợp nào khớp



15. While/Loop trong Java

**Vòng lặp while** này lặp qua một khối mã miễn là điều kiện được chỉ định là true



**Vòng lặp Do/While là một biến thể của vòng lặp while . Vòng lặp này sẽ thực thi khối mã một lần, trước khi kiểm tra xem điều kiện có đúng không, sau đó nó sẽ lặp lại vòng lặp miễn là điều kiện vẫn đúng.**

**Bài tập minh họa:**

**Bài 1: Đếm số lần nhập sai mật khẩu**

**Viết chương trình yêu cầu người dùng nhập mật khẩu và đếm số lần nhập sai trước khi nhập đúng.**

**Bài 2: In bảng cửu chương**

**Viết chương trình sử dụng do...while để in bảng cửu chương từ 1 đến 10.**