



# LẬP TRÌNH JAVA CƠ BẢN

TS. Trần Quý Nam - Giảng viên Khoa CNTT

**Thời gian: 4 tiết**

# Java là gì?

- Java là một ngôn ngữ lập trình HĐT được phát triển bởi Sun Microsystems, nay thuộc sở hữu của Oracle.
- Ban đầu được sử dụng để xây dựng ứng dụng điều khiển các bộ xử lý bên trong các thiết bị điện tử dân dụng như máy điện thoại cầm tay, lò vi sóng...
- Java = 1 ngôn ngữ lập trình = 1 công nghệ = 1 nền tảng phát triển

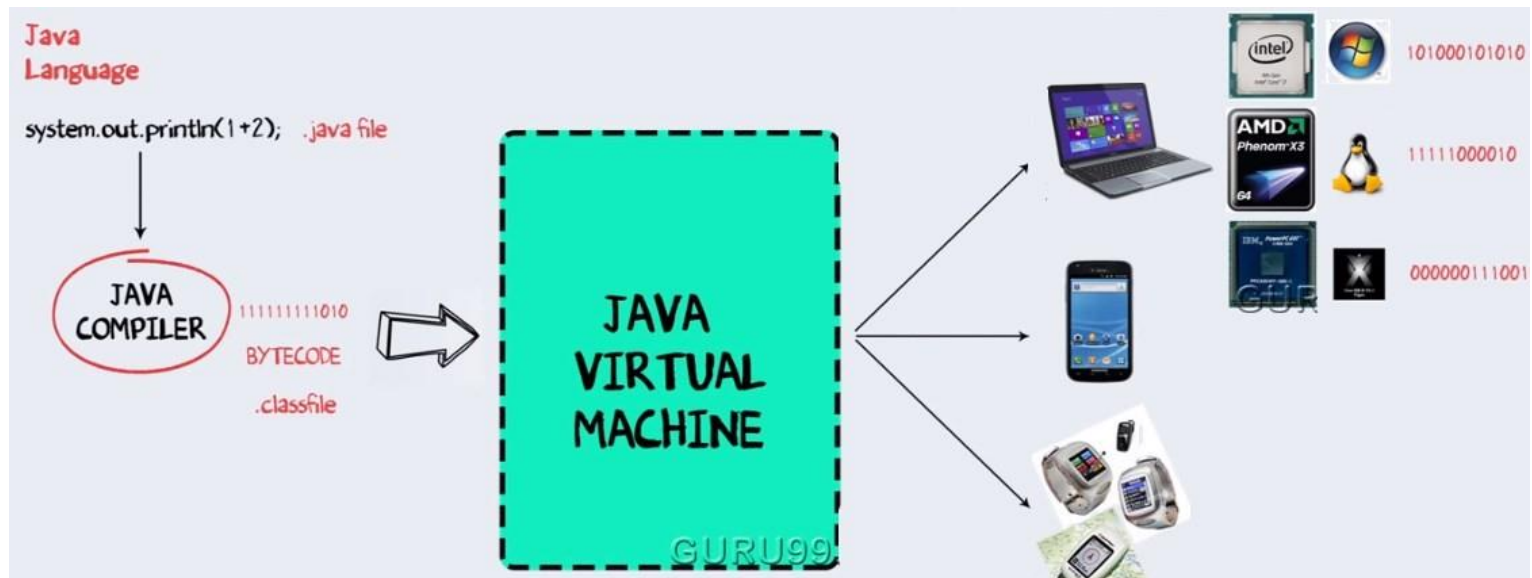


Green Team and James Gosling  
(the leader)

- Mỗi sản phẩm Java gồm:
  - Bộ công cụ phát triển: J2SDK, JDK hay SDK (development kit)
    - JDK = JRE + tools tiện ích (\*.exe) + tài liệu + thư viện
  - Môi trường chạy JRE (runtime environment): môi trường thực thi
    - JRE = JVM + Java runtime library (trên desktop)
    - Server JRE: JRE trên server
  - Máy ảo java JVM (virtual machine)
  - Các đặc tả chi tiết kỹ thuật, Ngôn ngữ lập trình, Các công nghệ đi kèm
- Java hỗ trợ các platform
  - Sun Solaris, Linux, Mac OS, FreeBSD & Microsoft Windows.

# Mô hình dịch của Java

- Mô hình dịch của Java
  - Tiêu chí "Viết (code) một lần, thực thi khắp nơi" ("Write Once, Run Anywhere" (WORA))
  - Mã nguồn được biên dịch thành bytecode rồi được thông dịch bởi JVM



# Môi trường, công cụ

- Môi trường phát triển và thực thi - JDK

(<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)

- **IDE (Integrated Development Enviroment)**

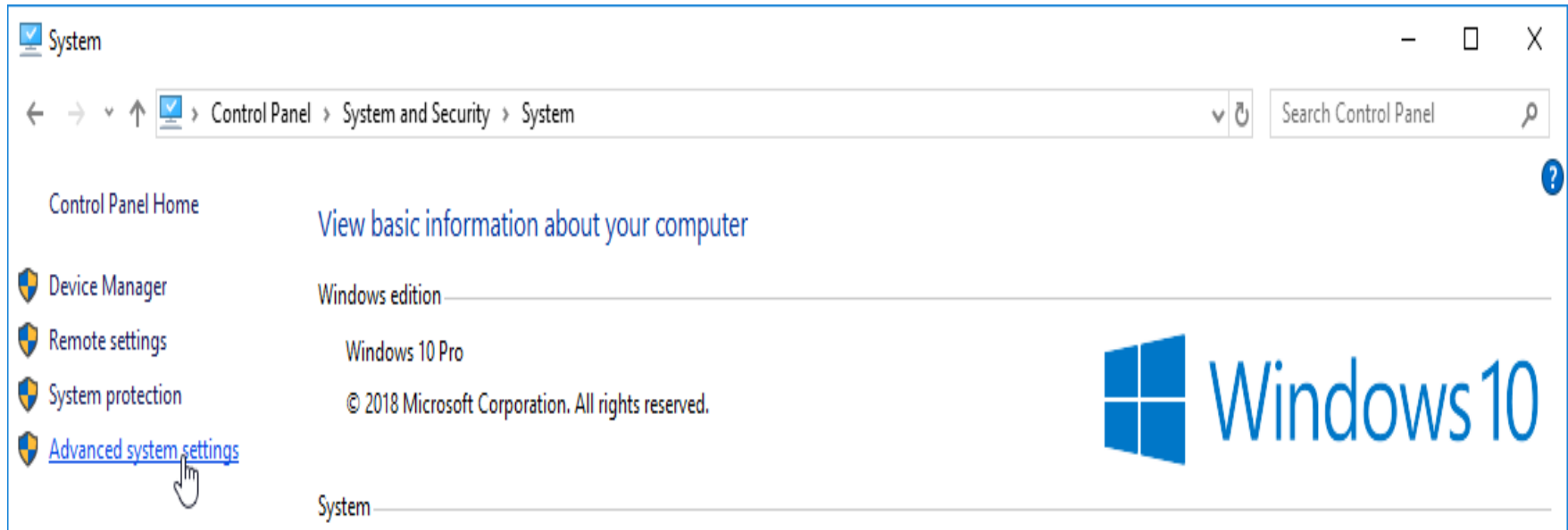
- ✓ NetBeans
- ✓ Eclipse
- ✓ Jcreator
- ✓ Jbuilder
- ✓ ...

# Cài đặt môi trường lập trình java

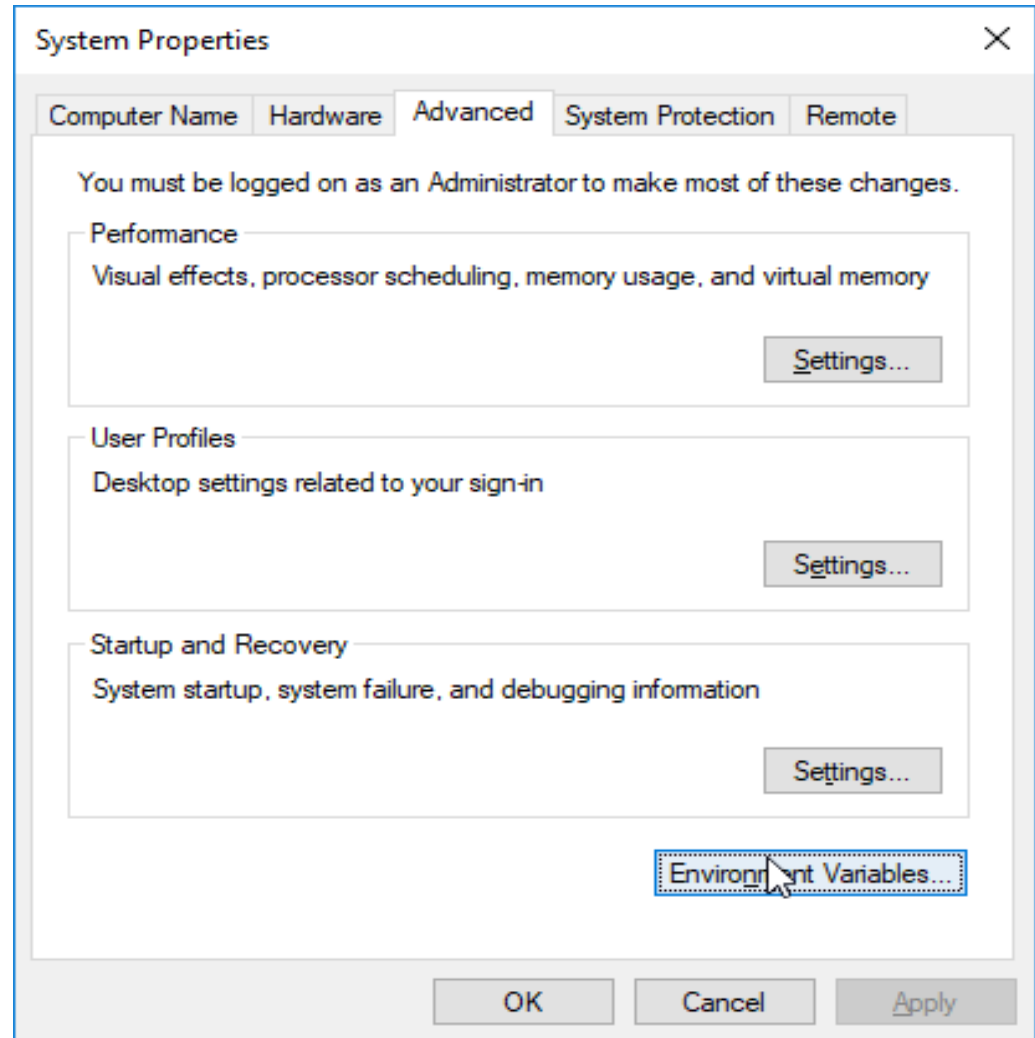
- Cài đặt JDK 8: <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>
- Cài đặt Eclipse: <https://www.eclipse.org/downloads/>
- Cài đặt Netbeans: <https://netbeans.apache.org/download/>
- TryIt của W3Schools:  
[https://www.w3schools.com/java/tryjava.asp?filename=demo\\_helloworld](https://www.w3schools.com/java/tryjava.asp?filename=demo_helloworld)



- **Bước 1:** Truy cập System Properties ( Control Panel > System and Security > System > Advanced System Settings)

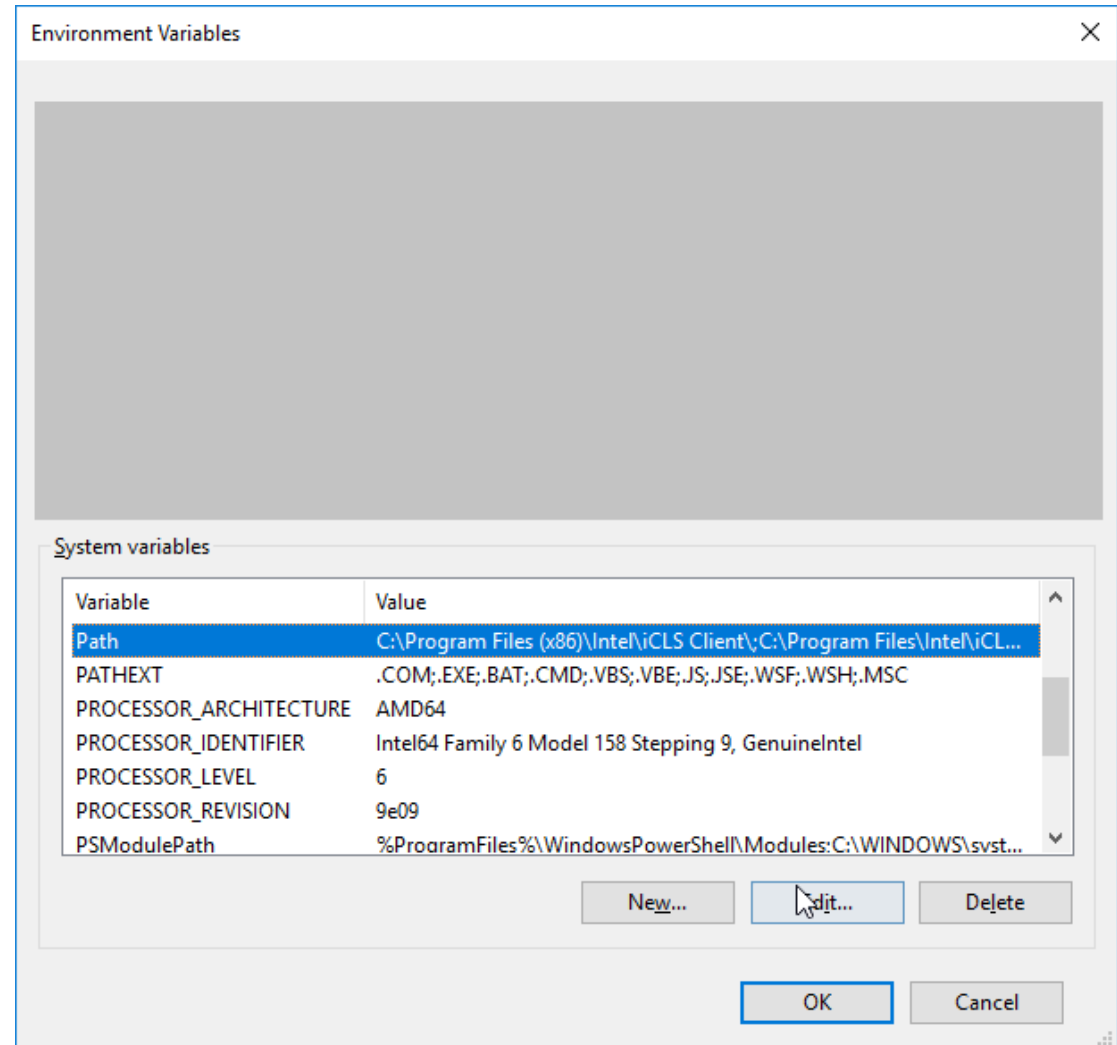


- **Bước 2:** Truy cập vào “Environment variables” ở dưới tab Advanced

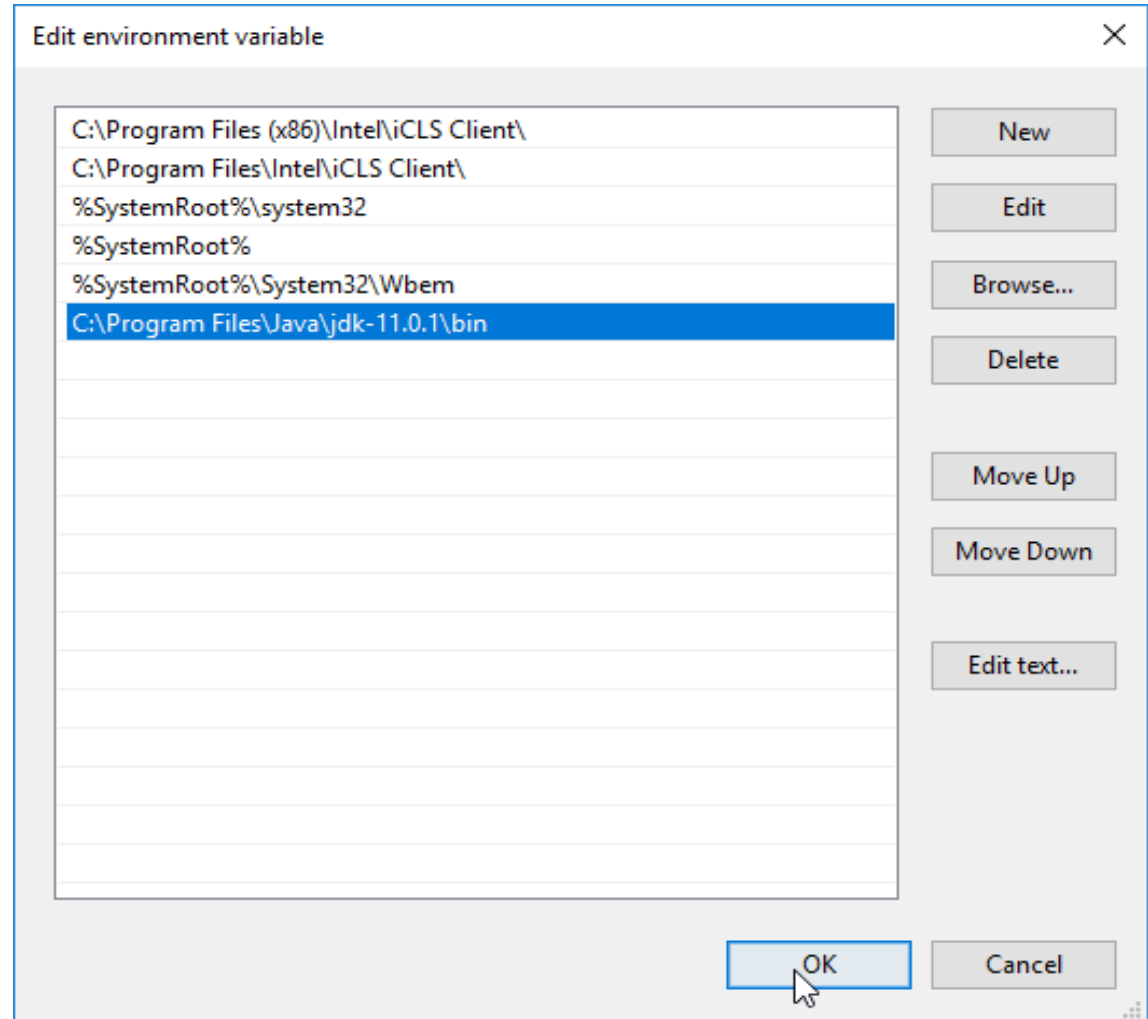




- **Bước 3:** Chọn biến Path trong danh sách các biến hệ thống (System variables) và ấn nút Edit



**Bước 4:** Ấn nút New sau đó copy đường dẫn Java đã cài đặt theo sau đó là \bin. Mặc định Java sẽ được cài đặt ở C:\Program Files\Java\<phiên bản Java>\bin. Bạn sẽ phải thêm một đường dẫn mới với giá trị ví dụ phiên bản Java 8.0.31: C:\Program Files\Java\jdk-8.0.31\bin. Ấn Ok để lưu cấu hình



- **Bước 5:** Kiểm tra
- Vào Command Prompt (cmd.exe) gõ lệnh để kiểm tra:

```
C:\Users\Your Name>java -version
```

- Nếu Java đã được cài thành công sẽ hiển thị ra tương tự như sau chỉ khác biệt phiên bản Java:

```
java version "11.0.1" 2018-10-16 LTS  
Java(TM) SE Runtime Environment 18.9 (build 11.0.1+13-LTS)  
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.1+13-LTS, mixed mode)
```

- Nhập mã Java vào panel bên trái
- Chạy mã lệnh Java bằng cách ấn nút Run
- Không hỗ trợ gợi ý code
- Không cần cài đặt JDK, IDE

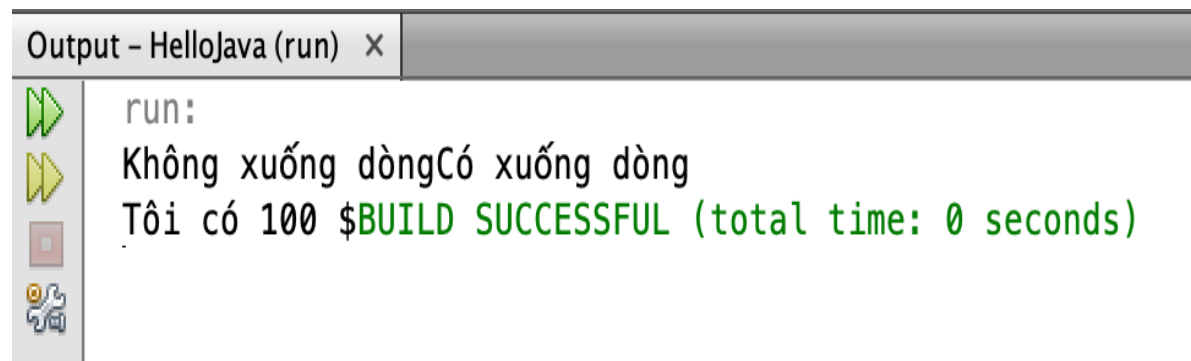


# In dữ liệu ra màn hình

- `System.out.print(<chuỗi cần in>)`: in một chuỗi ra màn hình trên một dòng
  - Ví dụ: `System.out.print("Xin chào");`
- `System.out.println(<chuỗi cần in>)`: in một chuỗi ra màn hình và xuống dòng
  - Ví dụ: `System.out.println("Xuống dòng");`
- `System.out.printf(<chuỗi định dạng>, <các tham số>)`: in một chuỗi ra màn hình theo định dạng tương tự `printf` của C
  - `%s` in chuỗi
  - `%d` in số nguyên
  - `%f` in số dấu phẩy động
  - Ví dụ: `System.out.printf("%s %d", "Chuỗi", 10);`

# Ví dụ in ra màn hình

```
public static void main(String[] args) {  
    System.out.print("Không xuống dòng");  
    System.out.println("Có xuống dòng");  
    System.out.printf("Tôi có %d $", 100);  
}
```

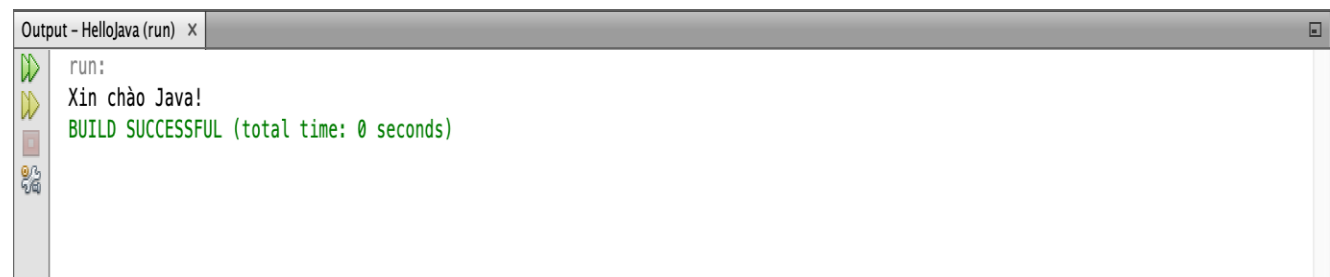


```
Output - HelloJava (run) x  
run:  
Không xuống dòng  
Có xuống dòng  
Tôi có 100 $  
BUILD SUCCESSFUL (total time: 0 seconds)
```

# Bài tập: Hiển thị “Xin chào Java”

```
package hellojava;  
  
public class HelloJava {  
    public static void main(String[] args) {  
        System.out.println("Xin chào Java!");  
    }  
}
```

- Thực hiện lưu tệp và chạy đoạn mã Java bằng cách vào menu Run > Run Project hoặc ấn F6
- Kết quả chạy sẽ như sau:



# Ví dụ

## Viết và thực thi chương trình Hello World

- Dùng Notepad soạn thảo đoạn lệnh bên dưới và lưu lại với tên

**HelloWorld.java**

Khai báo thư viện java.io

```
import java.io.*;
```

Định nghĩa lớp tên “**HelloWorld**”

```
class HelloWorld
```

Bắt đầu đoạn lệnh

```
{
```

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

```
{
```

```
    System.out.print(“Hello Class”);
```

```
}
```

```
}
```

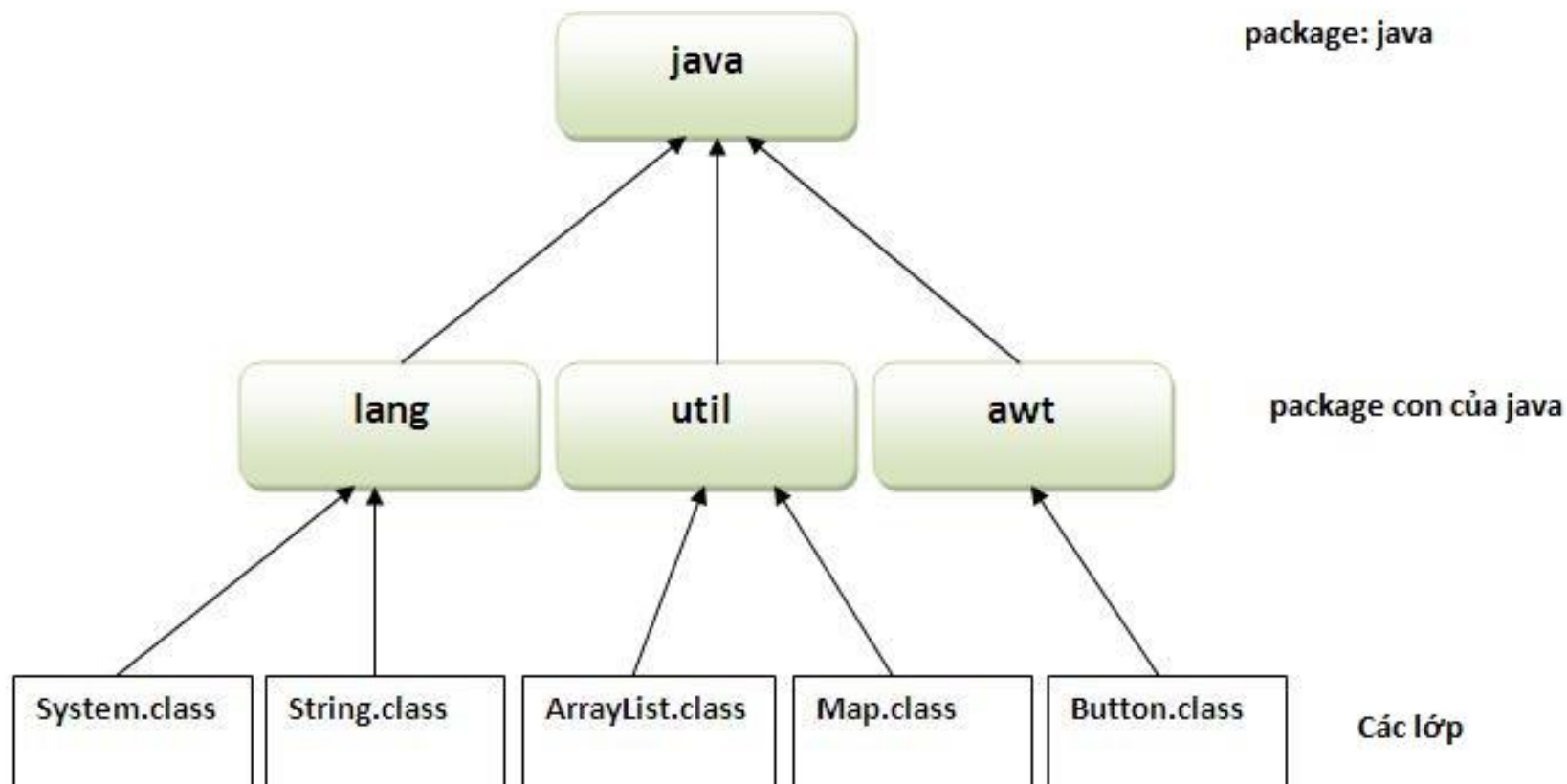
Kết thúc đoạn lệnh

Phương thức  
main

Xuất ra Console  
thông báo



- **Package** (gói) trong java là một nhóm các kiểu tương tự của các lớp, giao diện và các package con .
- Package trong java có thể được phân loại theo hai hình thức, package được dựng sẵn và package do người dùng định nghĩa.
- Có rất nhiều package được dựng sẵn như java, lang, AWT, javax, swing, net, io, util, sql, ..





# CĂN BẢN VỀ NGÔN NGỮ JAVA

- Biến là một **vùng nhớ** lưu các giá trị của chương trình
- Mỗi biến gắn với 1 kiểu dữ liệu và 1 định danh duy nhất là tên biến
- Tên biến phân biệt chữ hoa và chữ thường. Tên biến bắt đầu bằng 1 dấu `_`,  
\$, hay 1 ký tự, không được bắt đầu bằng 1 ký số.

## Khai báo

<kiểu dữ liệu> <tên biến>;

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

## Gán giá trị

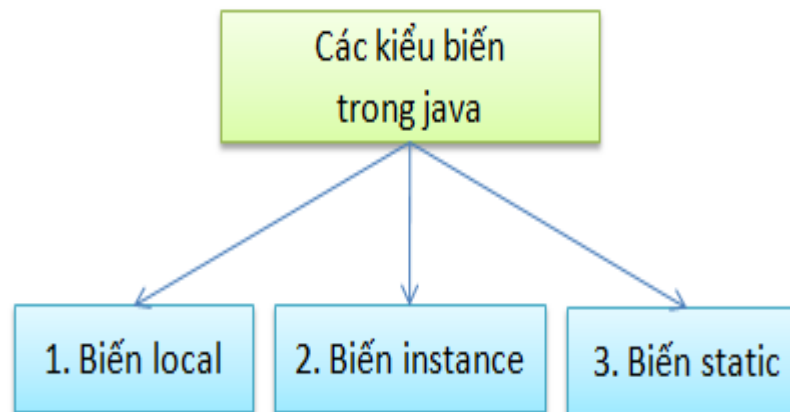
<tên biến> = <giá trị>;

- Cú pháp: `<kiểu> <tên biến> = <giá trị>;`
  - `<phạm vi>`: `public`, `protected`, `private`, `static`
  - `<kiểu>` là một trong các kiểu giá trị Java hoặc kiểu đối tượng tự tạo
- Khai báo một biến có tên là `<tên biến>` thuộc kiểu `<kiểu>` và gán giá trị của biến bằng `<giá trị>`
- Ví dụ khai báo ba biến chuỗi có kiểu là kiểu chuỗi (`String`), x có kiểu nguyên, f có kiểu số thực
  - `String` chuỗi = "Đây là chuỗi";
  - `int` x = 10;
  - `float` f = 8.5f;

- Biến trong Java có 2 loại: **member** variable và **local** variable.
- **member**: có thể được sử dụng mà không cần khởi tạo giá trị (được tự động gán giá trị mặc định).
- **local**: Java bắt buộc phải khởi tạo giá trị trước khi sử dụng. Nếu không sẽ tạo ra lỗi khi biên dịch.

# Các kiểu biến trong java

- Biến local được sử dụng trong các phương thức hoặc khối lệnh của Java không có **access modifier**
- Biến instance (thuộc tính) được khai báo trong lớp bên ngoài các phương thức và có thể khai báo **access modifier**
- Biến static được khai báo trong lớp bên ngoài phương thức, khối lệnh



```
public class Bien {  
    // Đây là biến static  
    public static float PI = 3.14f;  
    int n; // Đây là biến instance  
    public Bien () {  
        char c = 'c'; // Đây là biến local  
    }  
}
```



- Khai báo biến hằng, không thể thay đổi được giá trị bằng từ khoá *final*
- Ví dụ: khai báo biến x kiểu nguyên là final, sau đó thay đổi giá trị của x sẽ không thay đổi được IDE sẽ báo lỗi ngay ví dụ Netbean sẽ hiện đỏ ở dòng `x = 6`

```
final int x = 5;
```

```
x = 6; //Lỗi không thể thay đổi giá trị biến final
```

# Hằng

- Là một giá trị bất biến trong chương trình
- Tên đặt theo qui ước như tên biến
- Được khai báo dùng từ khóa **final**, và thường dùng tiếp vĩ ngữ đối với các hằng số (l, L, d, D, f, F)

- Ví dụ:

**final** int x = 10; // khai báo hằng số nguyên x = 10

**final** long y = 20L; // khai báo hằng số long y = 20

- Hằng ký tự: đặt giữa cặp nháy đơn **"**
- Hằng chuỗi: là một dãy ký tự đặt giữa cặp nháy đôi **""**

# Hàng ký tự đặc biệt

Ký tự	Ý nghĩa
\b	Xóa lùi (BackSpace)
\t	Tab
\n	Xuống hàng
\r	Dấu enter
\"	Nháy kép
\'	Nháy đơn
\\	\
\f	Đẩy trang
\uxxxx	Ký tự unicode

# Các kiểu dữ liệu của biến

- Kiểu dữ liệu trong Java chia làm 2 loại: kiểu dữ liệu nguyên thủy (primitive type) và kiểu dữ liệu không nguyên thủy (non-primitive type)
- Kiểu dữ liệu nguyên thủy bao gồm: byte, short, int, long, float, double, boolean, char
- Kiểu dữ liệu không nguyên thủy: ví dụ chuỗi String, mảng, lớp

# Kiểu dữ liệu nguyên thủy

Kiểu dữ liệu	Kích thước	Mô tả
byte	1 byte	Lưu các giá trị số từ -128 đến 127
short	2 bytes	Lưu các số từ -32768 đến 32767
int	4 bytes	Lưu các số từ -2,147,483,648 đến 2,147,483,647
long	8 bytes	Lưu các số giá trị từ -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807
float	4 bytes	Lưu các số dấu phẩy động có từ 6 đến 7 số sau dấu phẩy
double	8 bytes	Lưu các số dấu phẩy động có khoảng 15 số sau dấu phẩy
boolean	1 bit	Lưu giá trị true hoặc false
char	1 byte	Lưu giá trị ký tự ASCII

# Ví dụ

- Kí tự f sau giá trị kiểu số chỉ ra rằng đây là hằng số float, hằng kí tự trong dấu ‘’

```
byte b = 127;
```

```
short s = 890;
```

```
int i = 728723;
```

```
long l = 928392839;
```

```
float f = 74.3873483f;
```

```
double d = 8.12323232322323;
```

```
boolean flag = true;
```

```
char c = 'A';
```

# Đọc dữ liệu từ màn hình console

- Có nhiều cách để đọc dữ liệu từ màn hình: `BufferedReader`, `Scanner`, ...
- Sử dụng lớp `Scanner` để đọc dữ liệu
- `Scanner` cung cấp nhiều hàm để đọc dữ liệu khác nhau tùy theo kiểu dữ liệu bạn mong muốn: `String`, `Integer`, `Byte`, `Float`, ...
- Cách sử dụng:
  - Import thư viện: `import java.util.Scanner;`
  - Khởi tạo đối tượng `Scanner`: `Scanner scanner = new Scanner(System.in);`
  - Gọi các hàm để đọc dữ liệu: `next()`, `nextInt()`, `nextFloat()`, `nextByte()`, `nextBoolean()`, ...
    - Ví dụ: `int number = scanner.nextInt();`

```
public static void main(String[] args) {  
    System.out.print("Hãy nhập vào số nguyên: ");  
    Scanner scanner = new Scanner(System.in);  
    int number = scanner.nextInt();  
    System.out.print("Hãy nhập vào một chuỗi: ");  
    String s = scanner.next();  
    System.out.println("Căn bậc 2 của số vừa nhập  
là: " + Math.sqrt(number));  
}
```



# Ép kiểu trong java

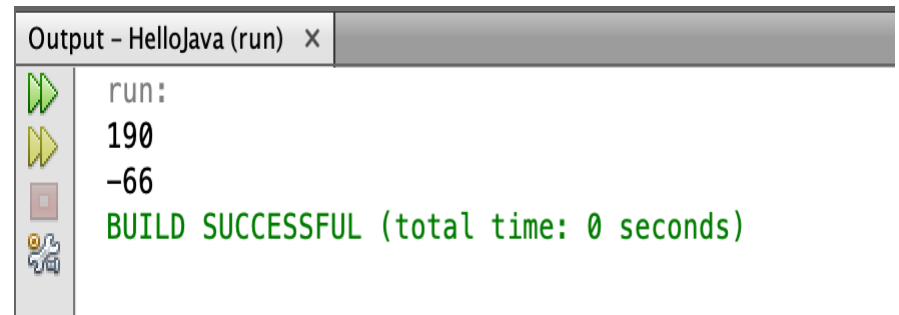
- Ép kiểu xảy ra khi bạn gán giá trị thuộc một kiểu nguyên thủy cho một kiểu giá trị kiểu nguyên thủy khác
- Có 2 loại ép kiểu:
  - Ép kiểu mở rộng: được thực hiện tự động khi chuyển đổi từ một kiểu có khoảng giá trị nhỏ hơn sang kiểu có khoảng giá trị lớn hơn như: byte -> short -> char -> int -> long -> float -> double
  - Ép kiểu hẹp: được thực hiện bởi lập trình viên chuyển đổi từ kiểu có khoảng giá trị lớn hơn về kiểu có khoảng giá trị bé hơn: double -> float -> long -> int -> char -> short -> byte

- Được thực hiện một cách tự động

```
public static void main(String[] args) {  
    int i = 100;  
    long l = i;  
    System.out.println(i);  
    System.out.println(l);  
}
```

- Thực hiện bằng cách sử dụng (<kiểu thu hẹp>) trước giá trị cần ép kiểu
- Lưu ý: ép kiểu hẹp có thể làm giá trị bị thay đổi nếu nó không nằm trong khoảng giá trị của <kiểu thu hẹp>
- Ví dụ:

```
public static void main(String[] args) {  
    short s = 190;  
    byte b = (byte)s;  
    System.out.println(s);  
    System.out.println(b);  
}
```



```
Output - HelloJava (run) x  
run:  
190  
-66  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Kiểu	Kích thước (bits)	Giá trị	Giá trị mặc định
<b>boolean</b>	[ <i>Note:</i> The representation of a boolean is specific to the Java Virtual Machine on each computer platform.]	true và false	<b>false</b>
<b>char</b>	16	'\u0000' to '\uFFFF' (0 to 65535)	<b>null</b>
<b>byte</b>	8	-128 to +127 ( $-2^7$ to $2^7 - 1$ )	<b>0</b>
<b>short</b>	16	-32,768 to +32,767 ( $-2^{15}$ to $2^{15} - 1$ )	<b>0</b>
<b>int</b>	32	-2,147,483,648 to +2,147,483,647 ( $-2^{31}$ to $2^{31} - 1$ )	<b>0</b>
<b>long</b>	64	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 ( $-2^{63}$ to $2^{63} - 1$ )	<b>0l</b>
<b>float</b>	32	1.40129846432481707e-45 to 3.4028234663852886E+38	<b>0.0f</b>
<b>double</b>	64	4.94065645841246544e-324 to 1.7976931348623157E+308	<b>0.0d</b>

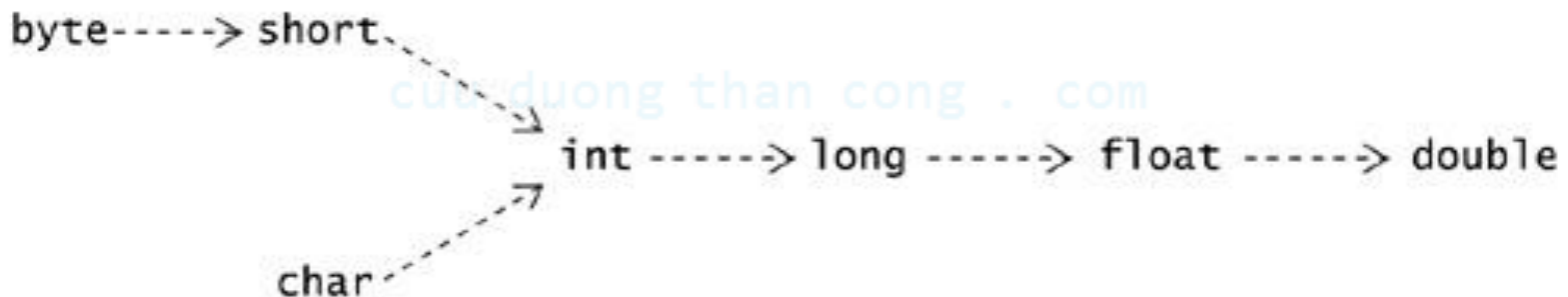
# Kiểu dữ liệu cơ sở (tt)

• **Chuyển đổi kiểu dữ liệu:** khi có sự không tương thích về kiểu dữ liệu (gán, tính toán biểu thức, truyền đối số gọi phương thức)

✓ Chuyển kiểu hẹp (lớn → nhỏ): ***cần ép kiểu***

***<tên biến 2> = (kiểu dữ liệu) <tên biến 1>;***

✓ Chuyển kiểu rộng (nhỏ → lớn): ***tự động chuyển***



# Kiểu dữ liệu cơ sở (tt)

- **Lưu ý**

1. Không thể chuyển đổi giữa kiểu boolean với int và ngược lại.

2. Nếu 1 toán hạng kiểu **double** thì

“Toán hạng kia chuyển thành **double**”

Nếu 1 toán hạng kiểu **float** thì

“Toán hạng kia chuyển thành **float**”

Nếu 1 toán hạng kiểu **long** thì

“Toán hạng kia chuyển thành **long**”

**Ngược lại** “Tất cả chuyển thành **int** để tính toán”

# Kiểu dữ liệu cơ sở (tt)

- Ví dụ minh họa

1. *byte*  $x = 5$ ;

2. *byte*  $y = 10$ ;

3. *byte*  $z = x + y$ ;

*// Dòng lệnh thứ 3 báo lỗi chuyển kiểu cần sửa lại*

*// byte z = (byte) (x + y);*

- Toán tử là thao tác được thực hiện lên các biến và giá trị
- Java chia toán tử thành các nhóm sau:
  - Toán tử số học
  - Toán tử gán
  - Toán tử so sánh
  - Toán tử logic
  - Toán tử bit



Toán tử	Tên	Mô tả	Ví dụ
+	Cộng	Cộng 2 giá trị	$x+y$
-	Trừ	Trừ một giá trị cho giá trị khác	$x-y$
*	Nhân	Nhân 2 giá trị	$x*y$
/	Chia	Chia một giá trị cho giá trị khác	$x/y$
%	Chia lấy dư	Chia và trả về phần dư	$x\%y$
++	Tăng giá trị	Tăng giá trị lên 1	$++x$
--	Giảm giá trị	Giảm giá trị đi 1	$--x$

- Được dùng để gán giá trị cho biến

Toán tử	Ví dụ	Giống với
=	$x = 5$	$x = 5$
+=	$x += 3$	$x = x + 3$
-=	$x -= 3$	$x = x - 3$
*=	$x *= 3$	$x = x * 3$
/=	$x /= 3$	$x = x / 3$
%=	$x \% = 3$	$x = x \% 3$
&=	$x \& = 3$	$x = x \& 3$
=	$x  = 3$	$x = x   3$
^=	$x \wedge = 3$	$x = x \wedge 3$
>>=	$x >> = 3$	$x = x >> 3$
<<=	$x << = 3$	$x = x << 3$

# Toán tử so sánh

Được  
sử dụng  
để so  
sánh  
hai số

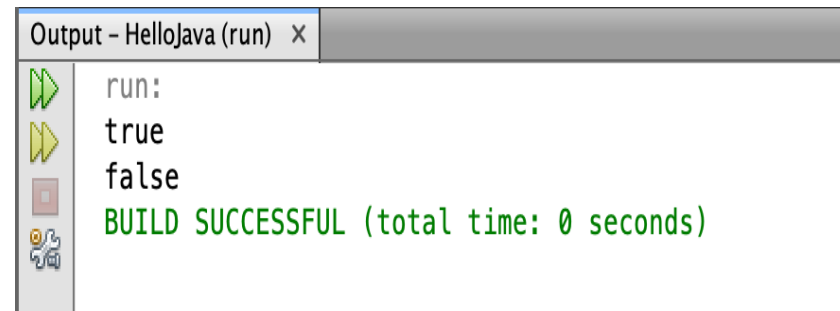
Toán tử	Tên	Ví dụ
$==$	So sánh bằng	$x == y$
$!=$	Không bằng	$x != y$
$>$	Lớn hơn	$x > y$
$<$	Nhỏ hơn	$x < y$
$>=$	Lớn hơn hoặc bằng	$x >= y$
$<=$	Nhỏ hơn hoặc bằng	$x <= y$

- Được sử dụng để xác định tính logic giữa các biến hoặc giá trị

Toán tử	Tên	Mô tả	Ví dụ
&&	Phép and	Trả về true nếu cả 2 giá trị là true	$x < 3 \ \&\& \ x < 9$
	Phép or	Trả về true nếu 1 giá trị là true	$x < 3 \    \ x < 9$
!	Phép not	Đảo ngược giá trị true, false	$!(x < 3 \ \&\& \ x < 9)$

- Kiểu boolean được khai báo bằng từ khoá boolean và nhận một trong hai giá trị true hoặc false
- Một số biểu thức Java trả về kết quả kiểu boolean: ví dụ phép toán so sánh

```
public static void main(String[] args) {  
    boolean flag = true;  
    System.out.println(flag);  
    System.out.println(4 > 5);  
}
```



```
Output - HelloJava (run) x  
run:  
true  
false  
BUILD SUCCESSFUL (total time: 0 seconds)
```

## •Toán tử số học

Toán tử	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia nguyên
%	Chia dư
++	Tăng 1
--	Giảm 1

# Toán tử, biểu thức (tt)

## • Phép toán trên bit

Toán tử	Ý nghĩa
&	AND
	OR
^	XOR
<<	Dịch trái
>>	Dịch phải
~	Bù bit

# Toán tử, biểu thức (tt)

- **Toán tử quan hệ & logic**

Toán tử	Ý nghĩa
<code>==</code>	So sánh bằng
<code>!=</code>	So sánh khác
<code>&gt;</code>	So sánh lớn hơn
<code>&lt;</code>	So sánh nhỏ hơn
<code>&gt;=</code>	So sánh lớn hơn hay bằng
<code>&lt;=</code>	So sánh nhỏ hơn hay bằng
<code>  </code>	OR (biểu thức logic)
<code>&amp;&amp;</code>	AND (biểu thức logic)
<code>!</code>	NOT (biểu thức logic)



# Toán tử, biểu thức (tt)

## • Toán tử gán

Toán tử	Ví dụ	Ý nghĩa
=	$a = b$	gán $a = b$
+=	$a += 5$	$a = a + 5$
-=	$b -= 10$	$b = b - 10$
*=	$c *= 3$	$c = c * 3$
/=	$d /= 2$	$d = d / 2$
%=	$e \% = 4$	$e = e \% 4$

Operators	Associativity
[] . () (method call)	Left to right
! ~ ++ -- + (unary) - (unary) () (cast) new	Right to left
* / %	Left to right
+ -	Left to right
<< >> >>>	Left to right
< <= > >= instanceof	Left to right
== !=	Left to right
&	Left to right
^	Left to right
	Left to right
&&	Left to right
	Left to right
?:	Right to left
= += -= *= /= %= &=  = ^= <<= >>= >>>=	Right to left

# Toán tử, biểu thức (tt)

## • Toán tử điều kiện

**Cú pháp:** <điều kiện> ? <biểu thức 1> : <biểu thức 2>

**Ví dụ:**

```
int x = 10; int y = 20;
```

```
int Z = (x < y) ? 30 : 40;
```

```
// Kết quả z = 30 do biểu thức (x < y) là
```

```
// đúng.
```



# CẤU TRÚC RỄ NHÁNH

- if: thực thi một khối lệnh nếu điều kiện sau if là đúng true
- else: chỉ ra đoạn lệnh sẽ được thực hiện nếu điều kiện trước đó là sai false
- else if : để kiểm tra điều kiện mới nếu điều kiện 1 là false

# If...else

## Cấu trúc *if ... else*

### Dạng 1:

```
if (<điều_kiện>) {  
    <khối_lệnh>;  
}
```

### Dạng 2:

```
if (<điều_kiện>) {  
    <khối_lệnh1>;  
}  
else {  
    <khối_lệnh2>;  
}
```

```
if (điều kiện 1) {  
    //Mã Java được thực hiện nếu điều kiện 1 đúng  
true  
}  
else if (điều kiện 2) {  
    //Mã lệnh thực hiện nếu điều kiện 2 đúng true và  
1 false  
}  
else {  
    //Mã lệnh thực hiện nếu cả 2 điều kiện là sai  
}
```

```
public static void main(String[] args) {  
    //Trò chơi dự đoán số  
    int x = 19;  
    if (x < 50) {  
        System.out.println("Giá trị nằm trong khoảng  
1 đến 50");  
    }  
    ...  
}
```



```
...  
else if (x < 75) {  
    System.out.println("Giá trị nằm trong khoảng  
50 đến 75");  
}  
else {  
    System.out.println("Giá trị nằm trong khoảng  
từ 75 đến 100");  
}  
}
```

# Câu lệnh switch case

- switch: kiểm tra trong nhiều đoạn code để xem đoạn code nào sẽ được thực hiện
- Biểu thức sau switch được tính một lần
- Giá trị của biểu thức sẽ được so sánh lần lượt với các giá trị sau case

- Cấu trúc *switch ... case*

```
switch (<biến>) {  
    case <giá trị_1>:  
        <khởi_lệnh_1>; break;  
    ....  
    case <giá trị_n>:  
        <khởi_lệnh_n>; break;  
    default:  
        <khởi_lệnh default>;  
}
```

# Ví dụ switch case

```
public static void main(String[] args) {  
    int x = 2;  
    switch (x) {  
        case 0:  
            System.out.println("Số  
không");  
            break;  
        case 1:  
            System.out.println("Số một");  
            break;  
        case 2:  
            System.out.println("Số hai");  
            break;  
        case 3:  
            System.out.println("Số ba");  
            break;  
        case 4:  
            System.out.println("Số bốn");  
            break;  
        case 5:  
            System.out.println("Số không");  
            break;  
        default:  
            System.out.println(x);  
            break;  
    }  
}
```

```
System.out.println("Số ba");  
break;  
case 4:  
    System.out.println("Số bốn");  
    break;  
case 5:  
    System.out.println("Số không");  
    break;  
default:  
    System.out.println(x);  
    break;
```

}

## Quy tắc viết code java

Tất cả các mã code Java đều phải đặt trong một lớp khai báo bằng từ khoá *class*.

Tên tệp chứa code Java phải **trùng tên lớp** được chỉ ra sau từ khoá class.

Ví dụ tên lớp là HelloWorld: class HelloWorld phải được lưu trong tệp HelloWorld.java

Hàm main bắt buộc phải có với các ứng dụng Java chạy trên desktop

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

# Java comments

Java có 2 cách để comment: comment trên một dòng và comment trên nhiều dòng

Comment một dòng sử dụng `//` các đoạn văn bản sau `//` cho đến cuối dòng sẽ không được thực thi bởi Java

`//Comment một dòng`

Comment nhiều dòng sử dụng `/* */` tất cả các văn bản nằm trong cặp dấu `/*` và `*/` đều sẽ bị bỏ qua bởi Java

`/*`

Đây là comment nhiều dòng

`*/`



# CẤU TRÚC VÒNG LẶP

- **Cấu trúc lặp**

- **Dạng 1:** while (<điều\_kiện\_lặp>) {...}

- **Dạng 2:** do {...  
                    } while (điều\_kiện);

- **Dạng 3:** for (khởi\_tạo\_biến\_đếm; đk\_lặp; tăng\_biến) {...}

- **Dạng 4 (Java 5)**

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

```
for (int i: arr) {...}
```



- **Cấu trúc lệnh nhảy jump:** dùng kết hợp nhãn (label) với từ khóa ***break*** và ***continue*** để thay thế cho lệnh ***goto*** (trong C).

***Ví dụ:***

```
label:
for (...) {
    for (...) {
        if (<biểu thức điều kiện>)
            break label;
        else
            continue label;
    }
}
```

Dùng để thực hiện một khối lệnh cho đến khi điều kiện được chỉ ra còn đúng.

Cú pháp:

```
while (điều kiện) {  
    //Khối lệnh sẽ thực hiện  
}
```

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



```
Output - HelloJava (run) x  
run:  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Là biến thể của vòng lặp while

**do while** sẽ thực hiện trước khối lệnh sau đó mới kiểm tra điều kiện nếu điều kiện đúng khối lệnh sẽ được tiếp tục thực hiện, nếu sai thì kết thúc

```
public static void main(String[] args) {  
    int i = 1;  
    do {  
        System.out.println(i);  
        i++;  
    }  
    while (i < 11);  
}
```



```
Output - HelloJava (run) ×  
run:  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Vòng lặp **for** lặp với số lần lặp đã xác định trước

Vòng lặp **for-each** lặp qua một mảng

Cú pháp:

Câu lệnh 1 được thực hiện 1 lần để khởi tạo biến đếm vòng lặp

Câu lệnh 2 định nghĩa điều kiện để thực hiện khối lệnh

Câu lệnh 3 được thực hiện sau khi khối lệnh trong for được thực hiện

```
for (câu lệnh 1; câu lệnh 2; câu lệnh 3) {  
    //Khối lệnh được thực hiện  
}
```

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

Dùng để lặp qua các phần tử của một mảng

Cú pháp:

Kiểu là kiểu giá trị các phần tử của mảng, ví dụ: int, long, string, ...

Tên biến là biến sẽ được nhận giá trị từng phần tử của mảng qua mỗi lần lặp

```
for (<kiểu> <tên biến> : <tên mảng>) {  
    //Khối lệnh được thực hiện  
}
```

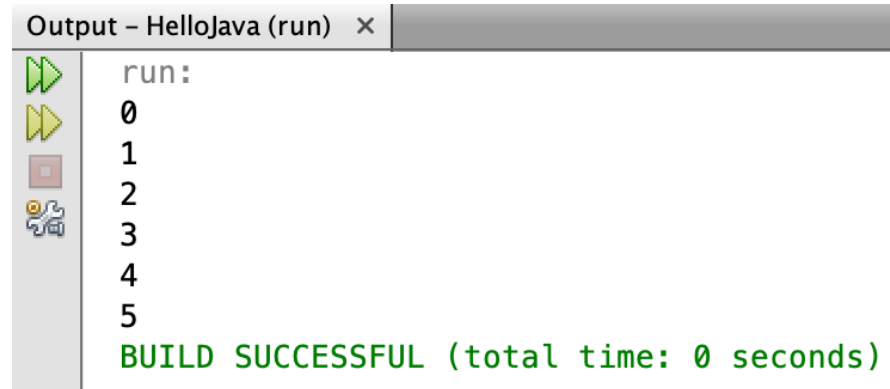


```
public static void main(String[] args) {  
    int[] arrs = { 1, 2, 3, 4, 5 };  
    for (int value : arrs) {  
        System.out.println(value);  
    }  
}
```

- **break** được sử dụng để nhảy ra khỏi vòng lặp:  
for, while, do while
- **continue** được dùng để nhảy ra khỏi một lần  
lặp nếu điều kiện nào đó là đúng và bắt đầu  
lần lặp tiếp theo

# Lệnh break và continue

```
public static void main(String[]  
args) {  
    for (int i = 0; i <= 10; i++) {  
        if (i > 5) break;  
        System.out.println(i);  
    }  
}
```



Output - HelloJava (run) x

```
run:  
0  
1  
2  
3  
4  
5  
BUILD SUCCESSFUL (total time: 0 seconds)
```

# Lệnh break và continue

```
public static void main(String[] args) {  
    for (int i = 0; i <= 10; i++) {  
        if (i > 2 && i < 5) continue;  
        System.out.println(i);  
    }  
}
```



```
Output - HelloJava (run) x  
run:  
0  
1  
2  
5  
6  
7  
8  
9  
10
```

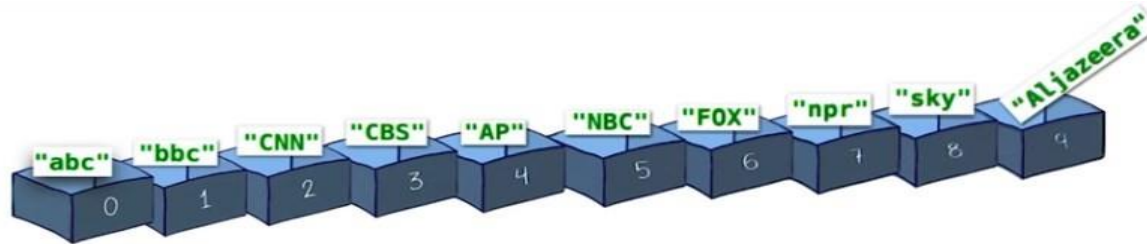


# DỮ LIỆU KIỂU MẢNG (ARRAY)

# Khái niệm mảng (array)

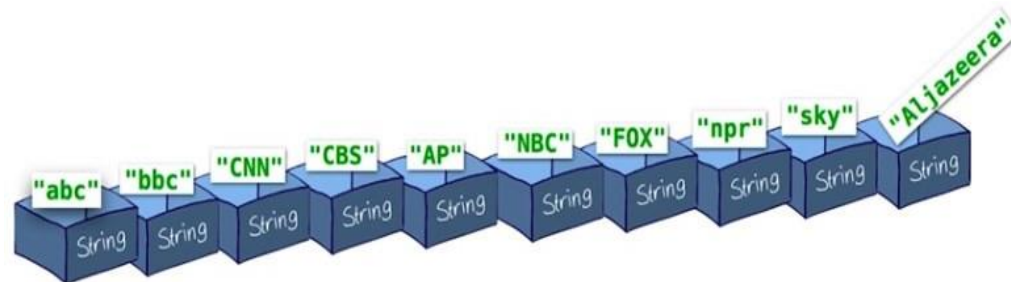
- Dùng để lưu một **tập hợp hữu hạn** các phần tử **cùng kiểu** (nguyên thủy hoặc đối tượng), **liền kề nhau** trong bộ nhớ.

- Mỗi mảng có 1 tên gọi
- Các phần tử được đánh số thứ tự, bắt đầu từ 0
- Truy cập từng phần tử trong mảng, dùng toán tử lấy chỉ số []
- Mỗi phần tử lúc này được xác định như 1 biến đơn



`newsOutlet`

`newsOutlet[0]` → "abc"  
`newsOutlet[1]` → "bbc"  
`newsOutlet[2]` → "CNN"



- Cú pháp: <kiểu giá trị>[] <tên mảng>
- Truy cập vào một phần tử của mảng thông qua chỉ số <tên mảng>[chỉ số]
- Thuộc tính **length** chỉ ra số phần tử của mảng
- Ví dụ:

```
int[] intarr; //Khai báo mảng kiểu số nguyên
```

```
int[] intarr2 = { 1, 2, 3, 4, 5 }; //Khai báo mảng số nguyên và  
khởi tạo có 5 phần tử
```

```
int[] intarr3 = new int[5]; //Khai báo mảng 5 số nguyên nhưng  
không khởi tạo
```



- Khai báo trước khi sử dụng, kích thước của một mảng sau khai báo sẽ không thể thay đổi
- Nếu không khởi tạo: tất cả các phần tử của mảng nhận giá trị mặc định tùy thuộc vào kiểu dữ liệu.
- Cú pháp:

```
kieu_dulieu[] ten_mang = new kieu_dulieu[KichThuoc_MANG];
```

```
kieu_dulieu ten_mang[] = new kieu_dulieu[KichThuoc_MANG];
```

```
kieu_dulieu[] ten_mang = {ds_gia_tri_cac_ptu};
```

// Khai báo kèm khởi tạo giá trị ban đầu

```
int [] numbers = {12, 1, 777, 3, 4, 0, 0, 121, 1, -4, 0, -100, 2};
```



12	1	777	3	4	0	0	121	1	-4	0	-100	2
0	1	2	3	4	5	6	7	8	9	10	11	12

- Dùng thuộc tính ***.length*** để lấy kích thước của một mảng
- Lưu ý không truy cập vào các chỉ số không thuộc mảng, ví dụ chỉ số âm, chỉ số  $\geq$  kích thước mảng.
- Duyệt tất cả các phần tử trong mảng: dùng ***vòng lặp***.

temperatures:

74	73	72	80
0	1	2	3

```
int size = temperatures.length; ← 4
```

```
System.out.println(temperatures[10]);
```

Error!

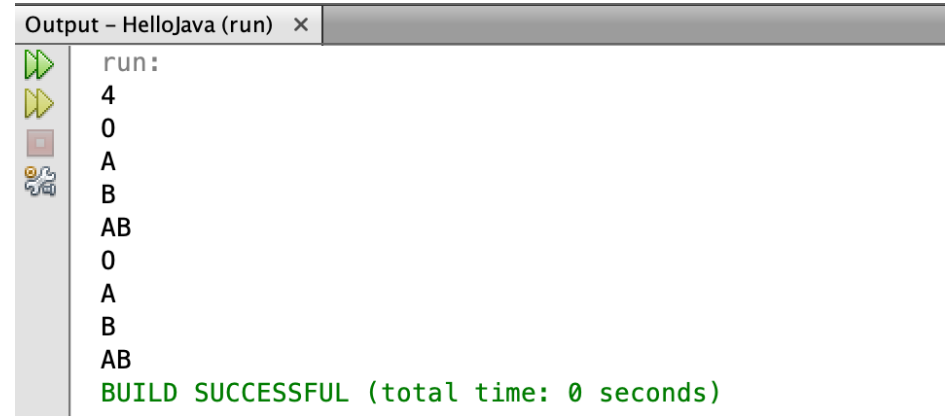
ArrayIndexOutOfBoundsException

```
for(int i=0; i<size; i++){  
    // truy cập temperatures[i];  
}
```

loop counter: (0, 1, 2 ...)

```
int MAX = 5;  
  
boolean bit[] = new boolean[MAX];  
  
float[] value = new float[2*3];  
  
int[] number = {10, 9, 8, 7, 6};  
  
System.out.println(bit[0]); // “false”  
  
System.out.println(value[3]); // “0.0”  
  
System.out.println(number[1]); // “9”
```

```
public static void main(String[] args) {  
    String[] nhommaus = { "O", "A", "B", "AB" };  
    System.out.println(nhommaus.length);  
    for (int i = 0; i < nhommaus.length; i++) {  
        System.out.println(nhommaus[i]);  
    }  
    for (String s : nhommaus) {  
        System.out.println(s);  
    }  
}
```



```
Output - HelloJava (run) x  
run:  
4  
0  
A  
B  
AB  
0  
A  
B  
AB  
BUILD SUCCESSFUL (total time: 0 seconds)
```

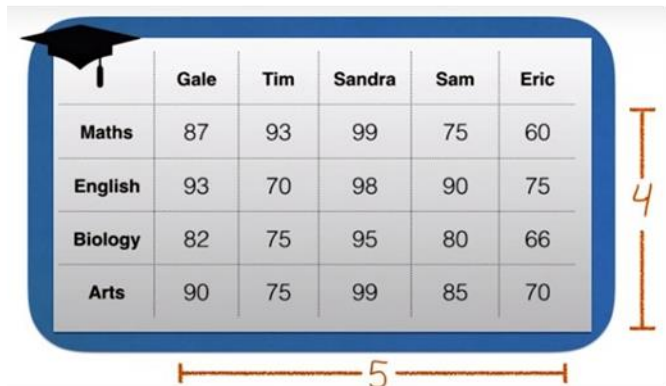


# MẢNG 2 CHIỀU (2D ARRAY)

# Mảng 2 chiều

- Trong Java, có thể khai báo và sử dụng một mảng N chiều.
- Khi mảng 2 chiều ( $N=2$ ), mảng giống một bảng với các dòng và cột.
- Mảng 2 chiều thao tác với ma trận

# Mảng 2 chiều



	Gale	Tim	Sandra	Sam	Eric
Maths	87	93	99	75	60
English	93	70	98	90	75
Biology	82	75	95	80	66
Arts	90	75	99	85	70

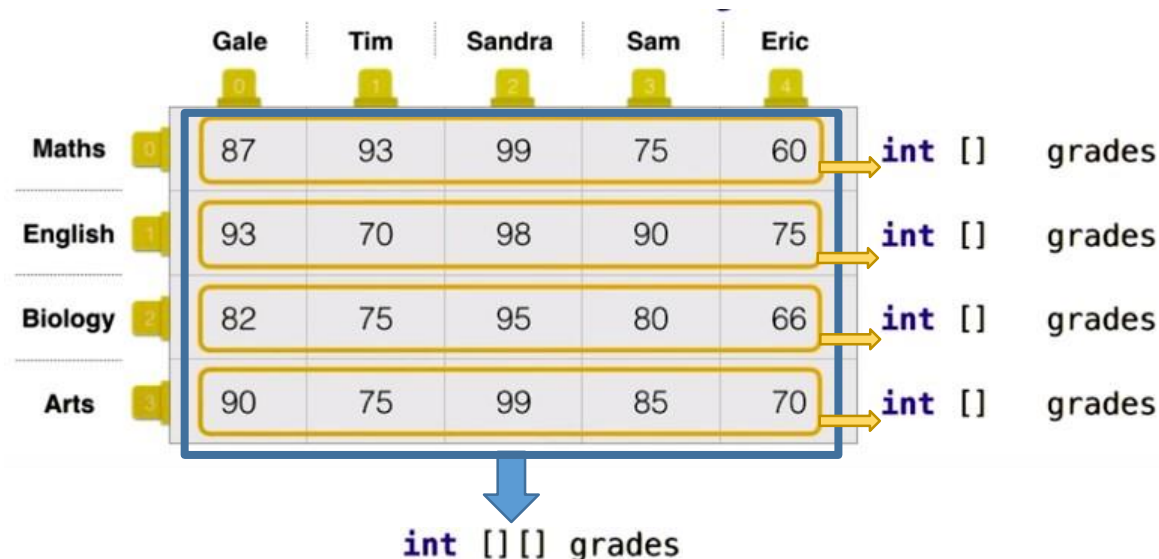
2D Array

		Gale	Tim	Sandra	Sam	Eric
		0	1	2	3	4
Maths	0	87	93	99	75	60
English	1	93	70	98	90	75
Biology	2	82	75	95	80	66
Arts	3	90	75	99	85	70



# Mảng 2 chiều

- Mảng 2 chiều: coi như 1 mảng 1 chiều của các phần tử A, mỗi phần tử A lại là 1 mảng các phần tử B.
- Cách khai báo mảng 2 chiều:  
`kieu_dulieu[][] ten_mang;`



# Truy xuất mảng 2 chiều

- Truy cập phần tử trong mảng:

`ten_mang[chi_so_hang][chi_so_cot]`

- Duyệt tất cả các phần tử trong mảng: Dùng 2 vòng lặp lồng nhau (tương tự lập trình C/C++)

# Truy xuất mảng 2 chiều

	Gale	Tim	Sandra	Sam	Eric
	0	1	2	3	4
Maths	87	93	99	75	60
English	93	70	98	90	75
Biology	82	75	95	80	66
Arts	90	75	99	85	70

grades[2][1] = 75

array #2  
(Biology)

item #1  
(Tim)

```
for(int i=0; i<4; i++){  
    for (int j=0; j<5; j++){  
        // truy cập grades[i][j];  
    }  
}
```

Mỗi phần tử của mảng lại là mảng khác.

*Ví dụ:*

```
int[][] haichieu = { { 1, 2 }, { 3, 4 }, { 5, 7 } };
```

```
System.out.println(haichieu[1][1]); //Giá trị 4
```

```
    for (int i = 0; i < haichieu.length; i++) {
```

```
        for (int j = 0; j < haichieu[i].length; j++) {
```

```
            System.out.println(haichieu[i][j]);
```

```
        }
```

```
    }
```

# XỬ LÝ CHUỖI (STRINGS)

- Được sử dụng để lưu trữ giá trị chuỗi (văn bản)
- Hằng số chuỗi được đặt trong dấu nháy kép “”
- Cộng chuỗi bằng toán tử +
- Các ký tự đặc biệt khi đưa vào chuỗi phải được escape sử dụng ký tự \

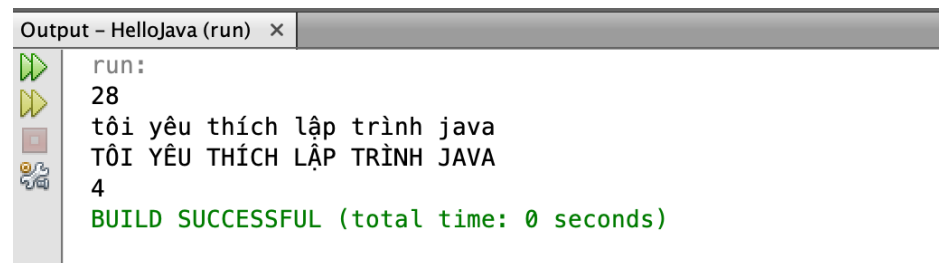
Escape ký tự	Kết quả	Mô tả
\'	'	Nháy đơn
\"	"	Nháy kép
\\	\	

# Các thao tác với chuỗi

- Độ dài chuỗi: `length()`
- Chuyển sang chữ hoa: `toUpperCase()`
- Chuyển sang chữ thường: `toLowerCase()`
- Tìm kiếm chuỗi: `indexOf(<chuỗi cần tìm>)` trả về chỉ số của lần xuất hiện đầu tiên của chuỗi cần tìm

- Khai báo một chuỗi s và thực hiện các thao tác lên chuỗi
- Kiểu chuỗi là kiểu lớp nên các thao tác thực hiện bằng cách sử dụng cú pháp **<đối tượng chuỗi>.<hàm chuỗi>**

```
public static void main(String[] args) {  
    String s = "Tôi yêu thích lập trình Java";  
    System.out.println(s.length());  
    System.out.println(s.toLowerCase());  
    System.out.println(s.toUpperCase());  
    System.out.println(s.indexOf("yêu"));  
}
```



Output - HelloJava (run) x

```
run:  
28  
tôi yêu thích lập trình java  
TÔI YÊU THÍCH LẬP TRÌNH JAVA  
4  
BUILD SUCCESSFUL (total time: 0 seconds)
```





# Xin cảm ơn!

**TRẦN QUÝ NAM**

*Tiến sỹ - Giảng viên Khoa CNTT  
namtq@dainam.edu.vn*