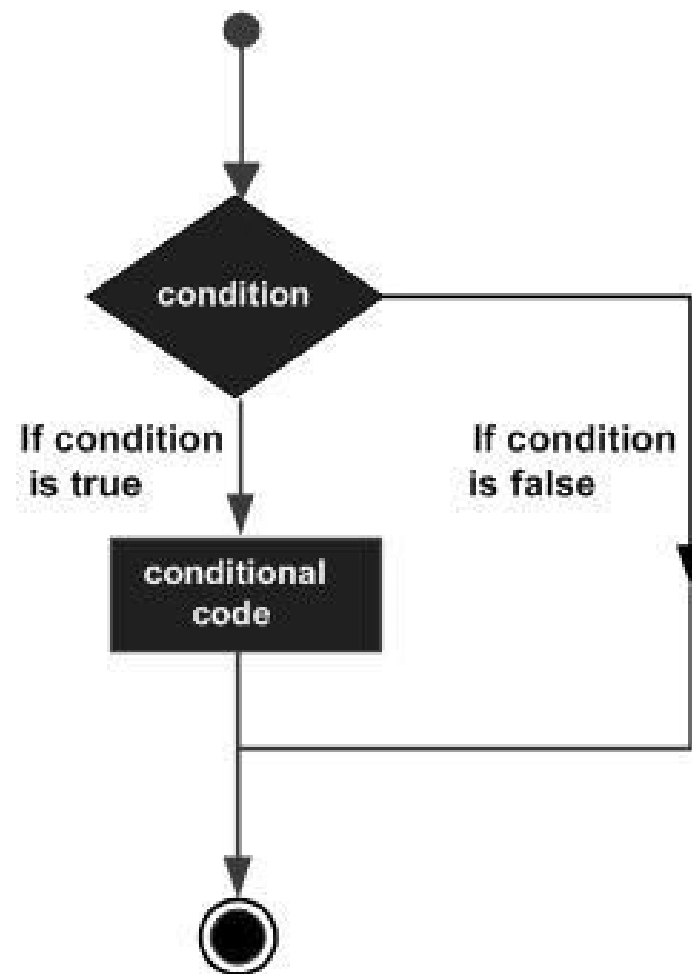
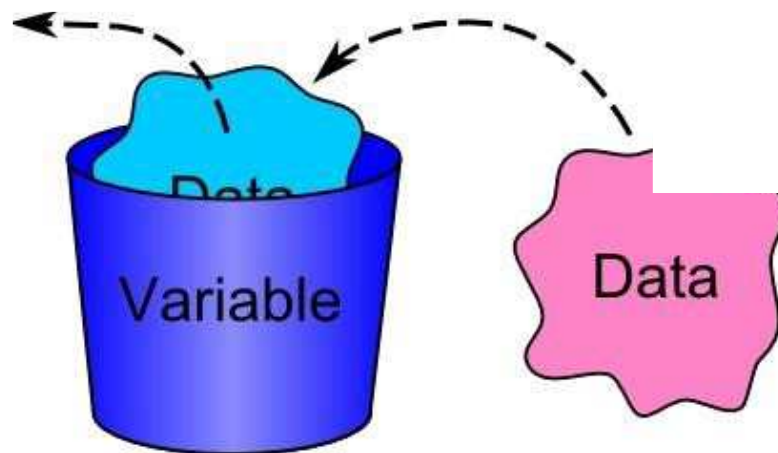


# **Bài 2**

## **Toán tử, Biểu thức điều kiện**

# Mục tiêu

- Khai báo biến trong Java
- Kiểu dữ liệu
- Nhập và xuất console
- Toán tử
- Mệnh đề điều kiện
- Cú pháp If-Else
- Cú pháp Switch-case

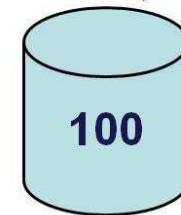


# Khai báo biến

**Biến** là gì?

- Là một **vùng lưu trữ** trên **bộ nhớ** máy tính chứa **dữ liệu** sử dụng trong chương trình Java.
- Những dữ liệu lưu trong biến tham gia quá trình **tính toán, truy cập, xuất...**
- Tùy vào vị trí đặt biến mà chúng có **phạm vi tồn tại** hay **truy cập** khác nhau.

count = 100;



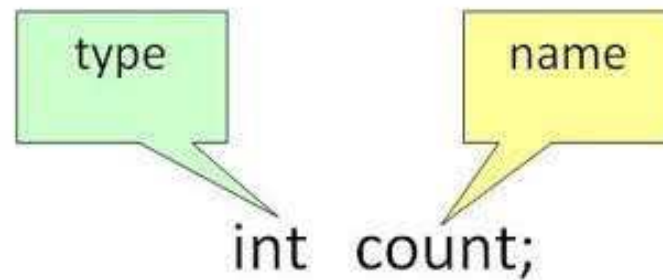
Container named  
"Count" holding  
a value 100

# Khai báo biến

---

Khai báo **Biến**?

Cú pháp khai báo biến như sau:



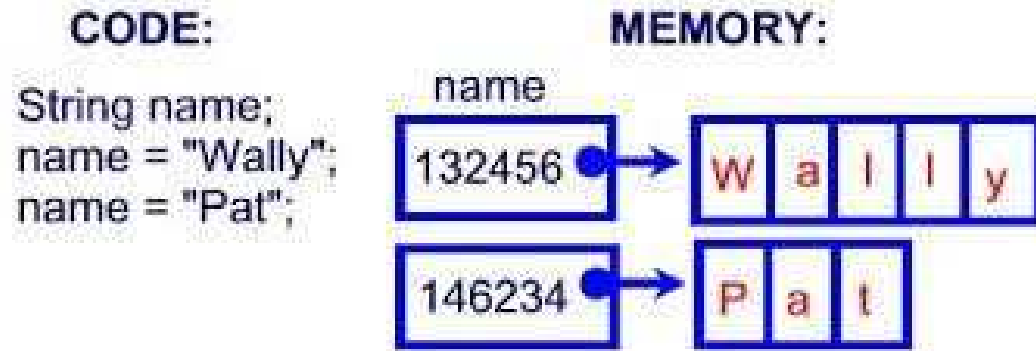
Trong đó:

- **int** (type): là kiểu dữ liệu
- **count** (name): là tên biến

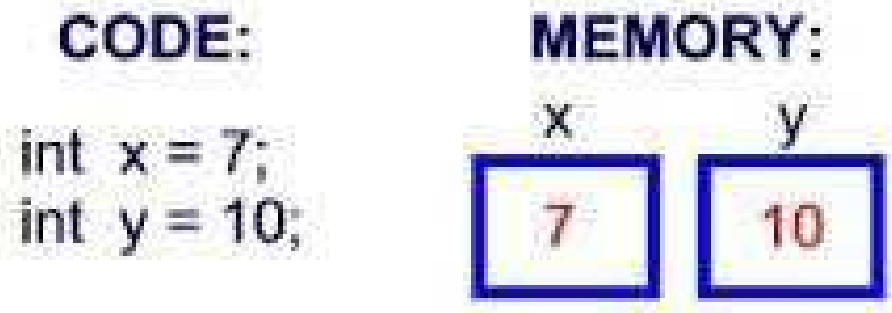
# Khai báo biến

Dữ liệu lưu trong **Biến**:

Dữ liệu kiểu chuỗi:

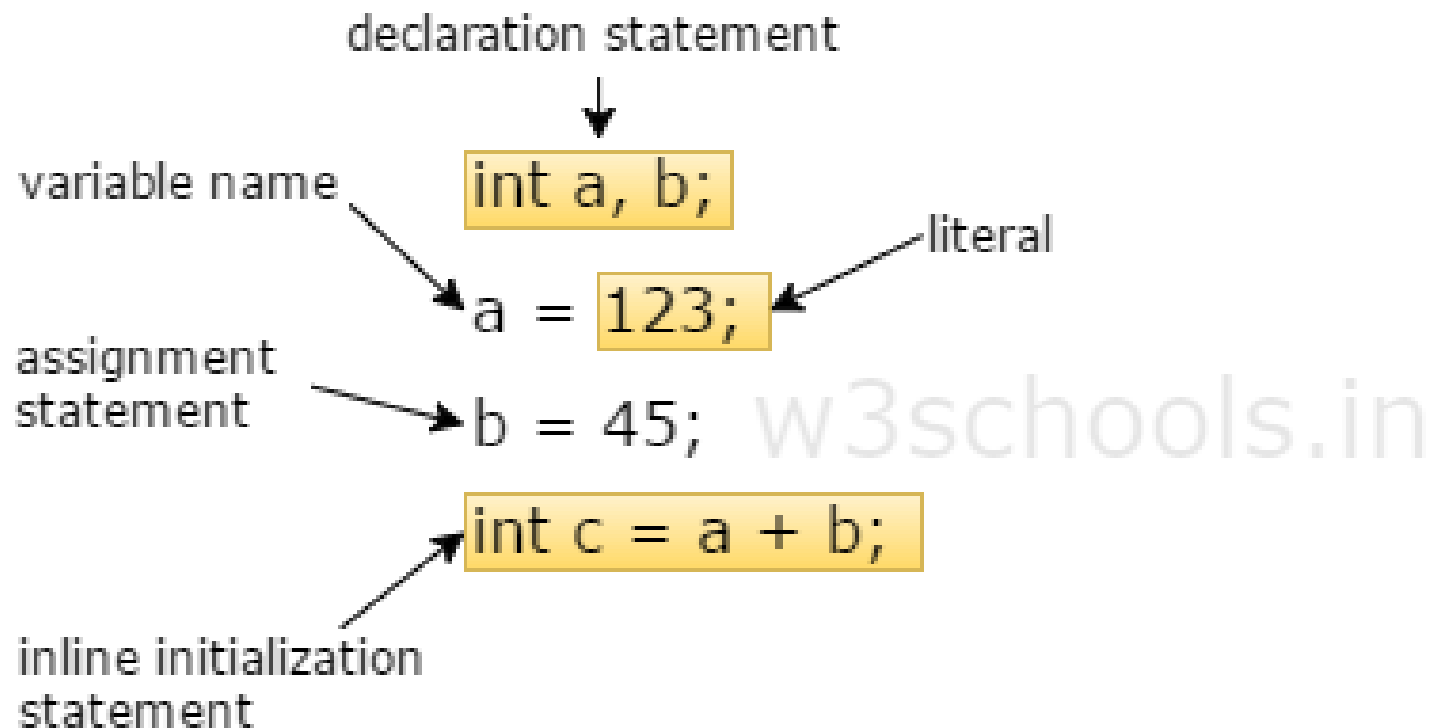


Dữ liệu kiểu số:



# Khai báo biến

Truy xuất, gán giá trị cho **Biến**:



Declaration and assignment statements

# Khai báo biến

---

**Quy tắc** đặt tên **Biến**:

- Có thể chứa các **ký tự** Unicode và **số**, bao gồm cả ký tự “\_” và dolla “\$”.
- Tên biến **không** chứa ký tự đặc biệt và khoảng trắng.
- Tên biến **không** được trùng với **từ khóa** Java.
- Tên biến **phân biệt** chữ **hoa – thường**.
- Tên biến nên viết theo quy tắc **Camel-case**.

# Khai báo biến

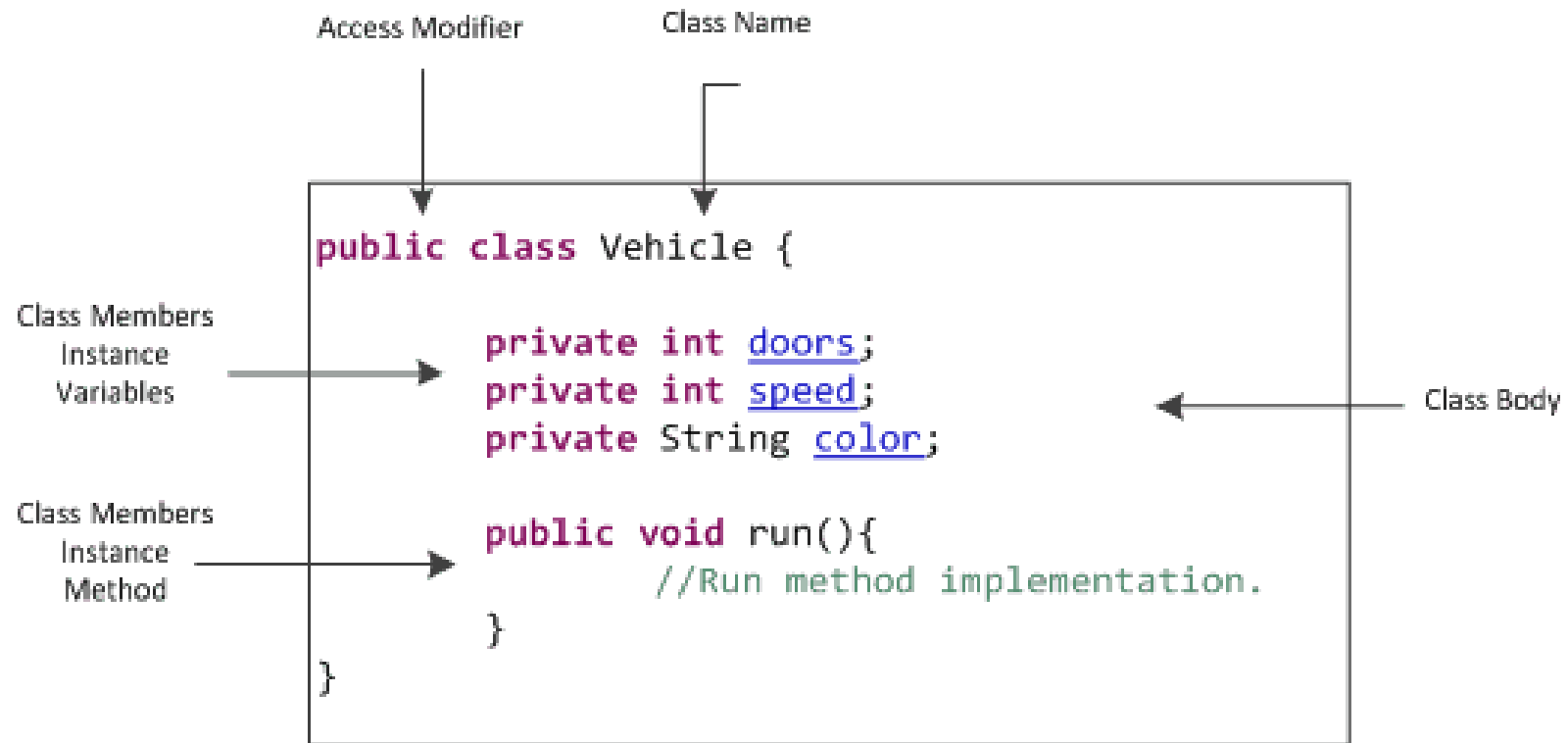
**Quy tắc** đặt tên **Biến**:

Tên biến	Đúng/Sai
rollNumber	Đúng
a2x5_w7t3	Đúng
\$yearly_salary	Đúng
_2010_tax	Đúng
\$\$_	Đúng
amount#Balance	Sai vì chứa ký tự đặc biệt #
double	Sai vì trùng với từ khóa Java
4short	Sai vì bắt đầu bằng chữ số



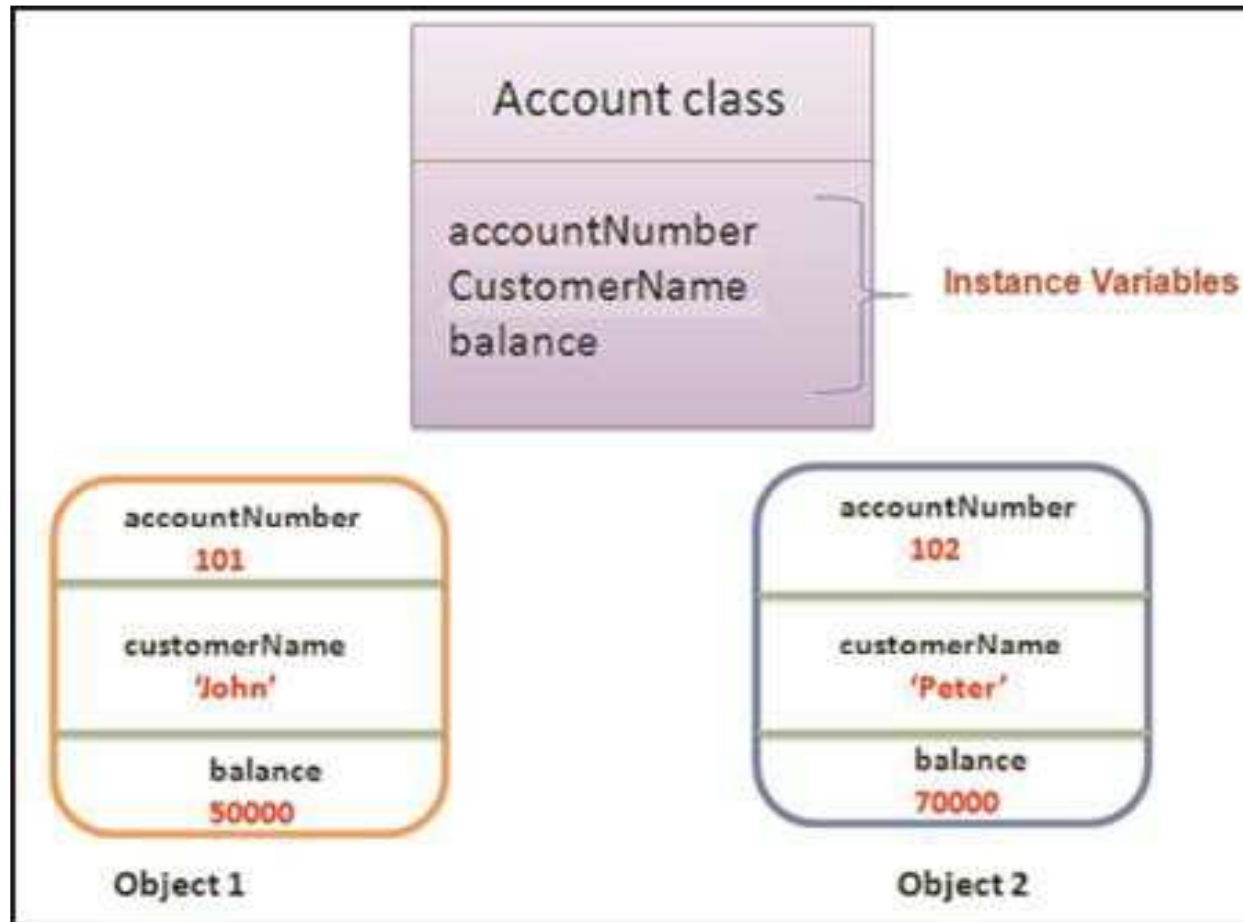
# Khai báo biến

Khai báo **Biến** là **thuộc tính** của lớp:



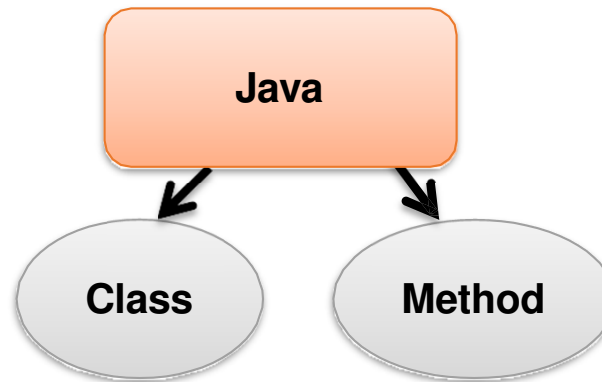
# Khai báo biến

Khởi tạo biến đối tượng từ lớp:



# Khai báo biến

Phạm vi và giới hạn tồn tại của biến:



- **Giới hạn lớp**: phạm vi giới hạn trong lớp, các phương thức có thể truy cập.
- **Giới hạn phương thức**: phạm vi chỉ nằm cục bộ trong phương thức, không thể truy cập từ phương thức khác.

# Khai báo biến

Phạm vi và giới hạn tồn tại của biến:

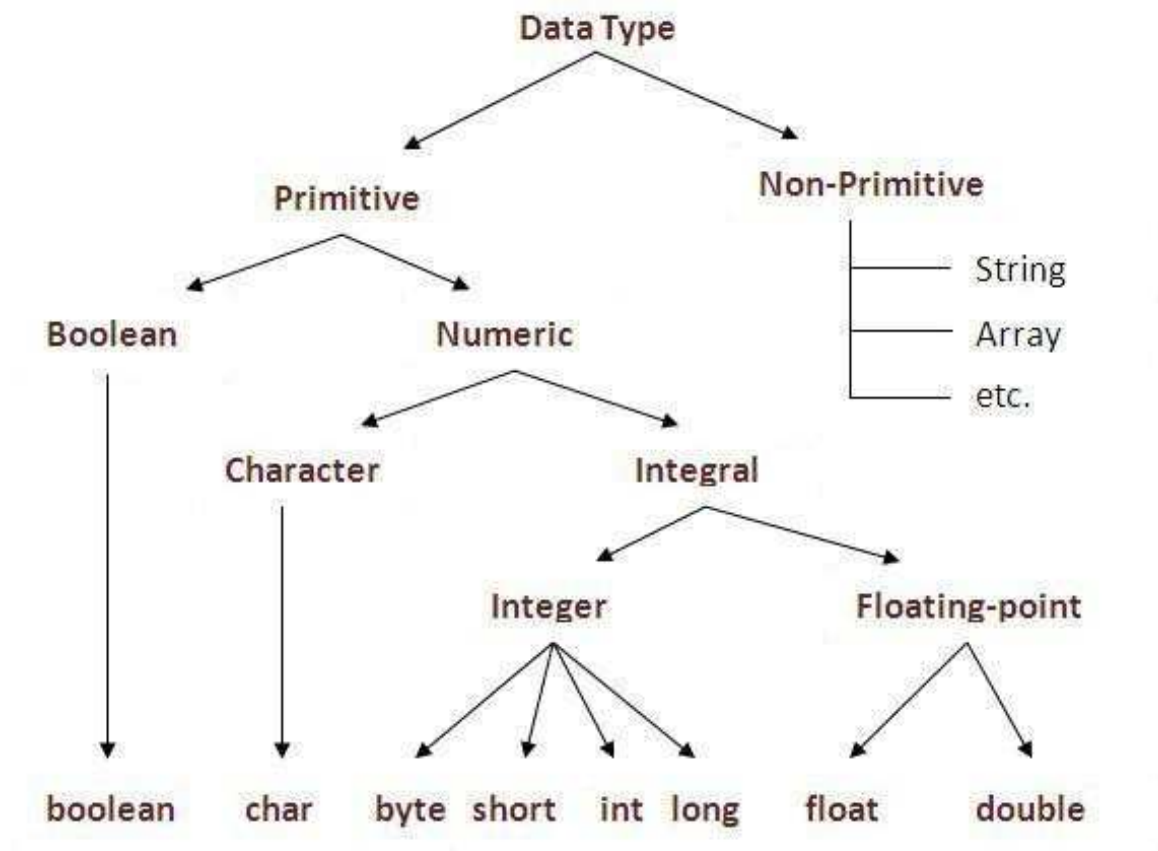
```
public class ScopeOfVariables {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
  
        // Known to code within main() method  
  
        int x;  
        x = 10;  
  
        { // Starts a block with new scope  
  
            int y = 20;  
  
            System.out.println("x and y: " + x + " " + y);  
  
            // Calculates value for variable x  
            x = y * 2;  
        } // End of the block  
  
        // y = 100; // Error! y not known here  
  
        // x is accesible  
        System.out.println("x is: " + x);  
    }  
}
```

Variable x is accessible within the main() block

Variable y is visible only within the block

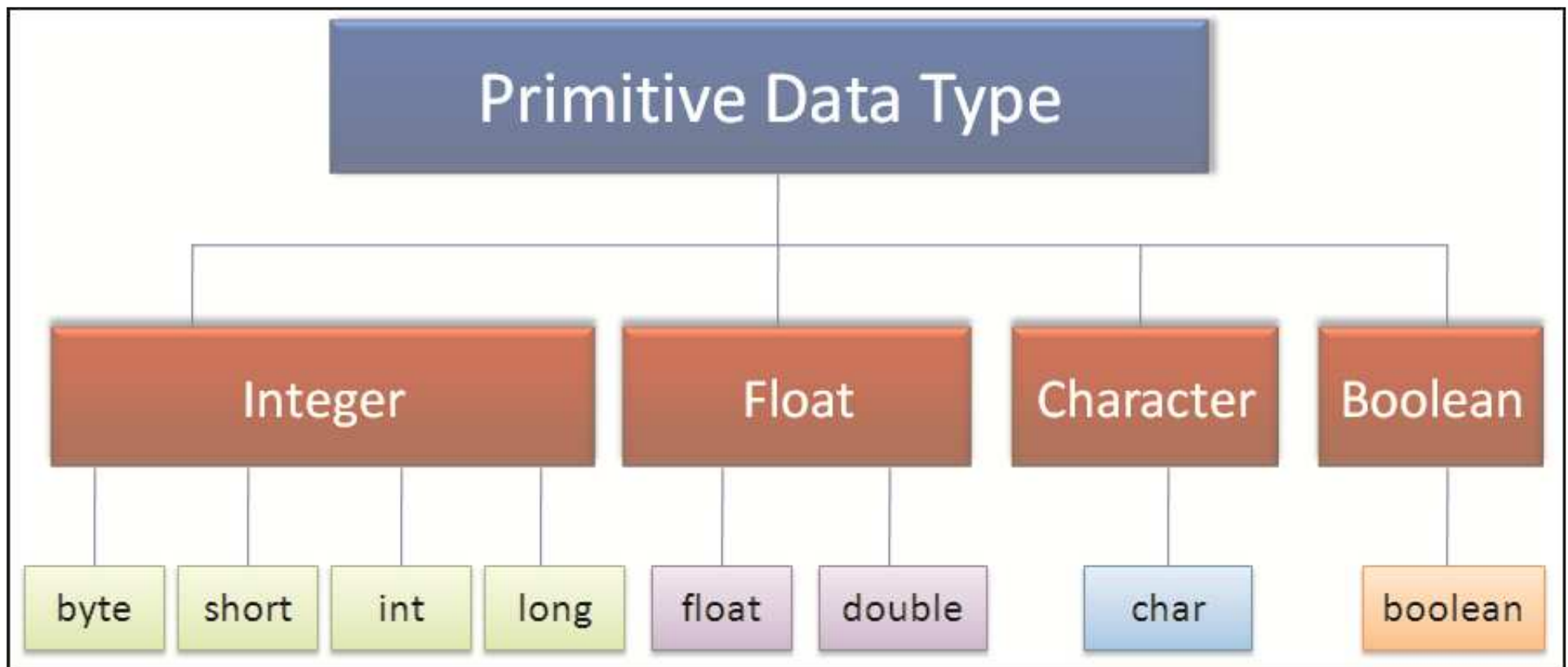
# Kiểu dữ liệu

**Java** có 2 loại kiểu dữ liệu: **Cơ bản** và **Tham chiếu**



# Kiểu dữ liệu

Kiểu dữ liệu **Cơ bản**:

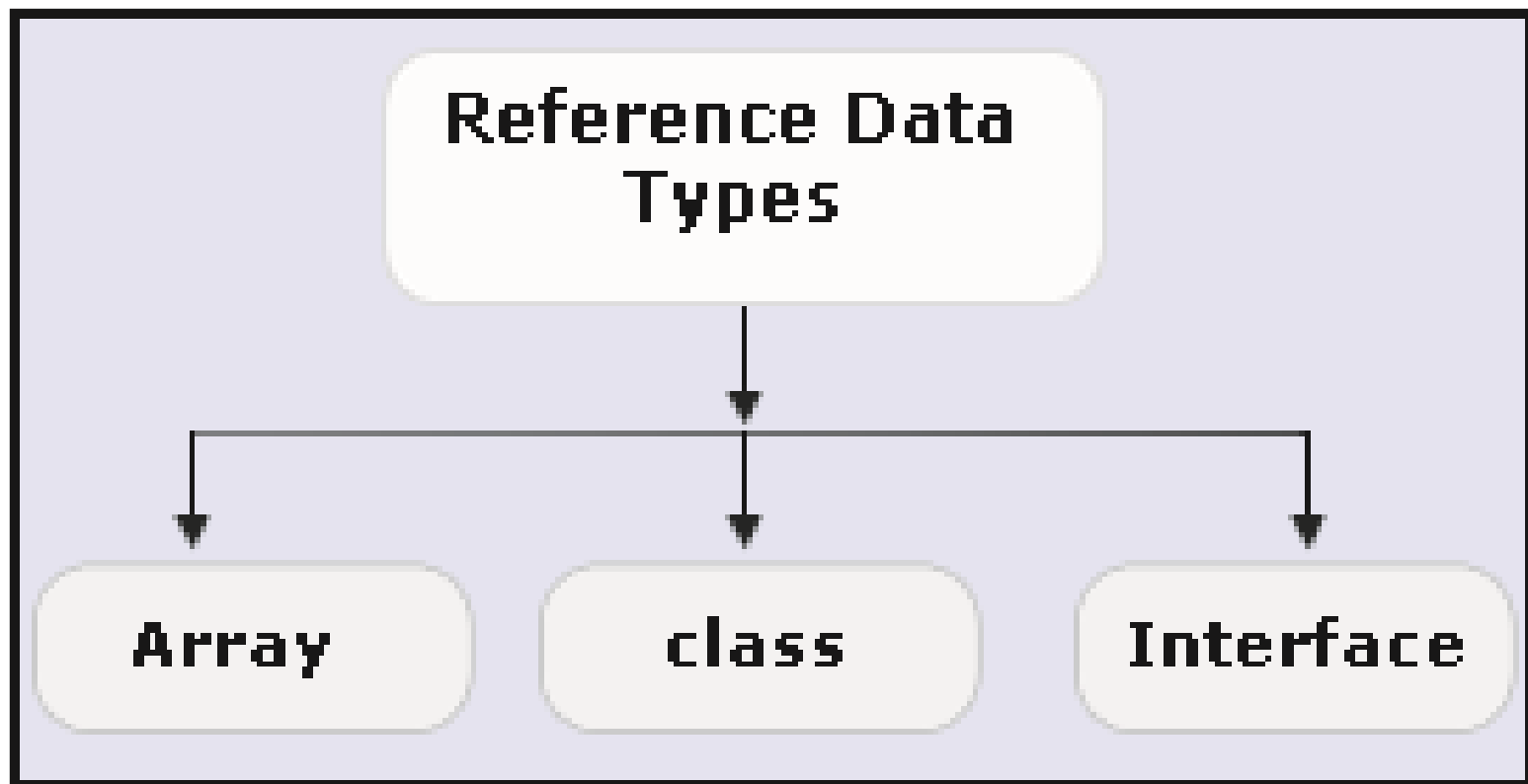


## PRIMITIVE DATA TYPE in JAVA

Type	Contains	Default	Size	Range
byte	Signed integer	0	8 bits	-128 to 127
short	Signed integer	0	16 bits	-32768 to 32767
int	Signed integer	0	32 bits	-2147483648 to 2147483647
float	IEEE 754 floating point	0.0f	32 bits	$\pm 1.4\text{E-}45$ to $\pm 3.4028235\text{E+}38$
long	Signed integer	0L	64 bits	-9223372036854775808 to 9223372036854775807
double	IEEE 754 floating point	0.0d	64 bits	$\pm 4.9\text{E-}324$ to $\pm 1.7976931348623157\text{E+}308$
boolean	true or false	FALSE	1 bit	NA
char	Unicode character	'\u0000'	16 bits	\u0000 to \uFFFF

# Kiểu dữ liệu

Kiểu dữ liệu **Tham chiếu**:





# Nhập xuất console

Để nhập dữ liệu từ màn hình console cần sử dụng lớp Scanner:

The diagram illustrates the steps to use the Scanner class in Java. It shows a code snippet with four numbered annotations:

- 1. import Scanner class**: Points to the `import java.util.Scanner;` line.
- 2. Construct Scanner object**: Points to the `Scanner in = new Scanner(System.in);` line.
- 3. Define variable to receive value**: Points to the `int x;` line.
- 4. read input**: Points to the `x = in.nextInt();` line.

```
import java.util.Scanner;

public class MyProgram
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        int x;
        ...
        x = in.nextInt();
        ...
    }
}
```

# Nhập xuất console

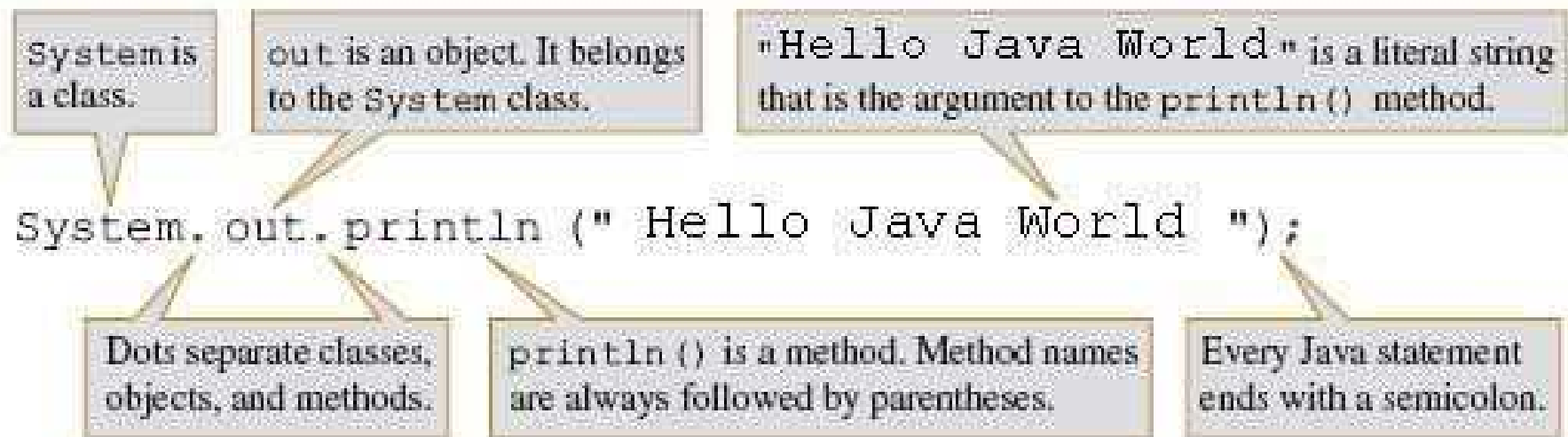
---

Để nhập dữ liệu từ màn hình console cần sử dụng lớp Scanner:

- # `nextInt()` --> for integer input
- # `nextFloat()` ---> for float input
- # `nextDouble()` --> for double input
- # `nextByte()` --> for byte input
- # `nextLine()` ----> for string input
- # `nextShort()` --> short integer input
- # `nextLong()` ---> for long integer input

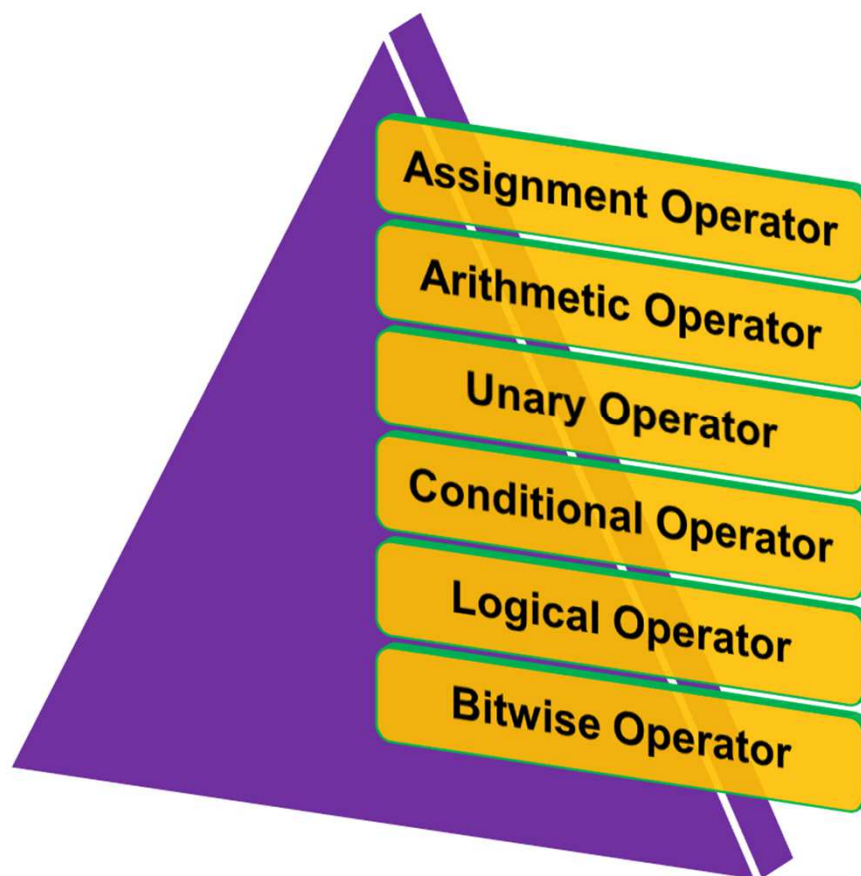
# Nhập xuất console

Để in dữ liệu ra màn hình console:



# Toán tử

**Toán tử** là: một tập hợp các ký hiệu sử dụng để chỉ thị các phép tính thực hiện trên dữ liệu.



# Toán tử

Toán tử **Toán học**:

Toán tử	Mô tả
+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia
%	Chia lấy dư (VD: $9/2$ kết quả trả về là 1)

# Toán tử

## Toán tử **Đơn**:

Toán tử	Mô tả
+	Số dương
-	Số âm
++	Tăng giá trị biến lên 1 đơn vị
--	Giảm giá trị biến đi 1 đơn vị
!	NOT - Nghịch đảo giá trị của biến luận lý (logic)

# Toán tử

## Toán tử **So sánh**:

Toán tử	Mô tả
<code>==</code>	So sánh bằng
<code>!=</code>	So sánh KHÔNG bằng
<code>&gt;</code>	Giá trị TRÁI lớn hơn PHẢI
<code>&lt;</code>	Giá trị TRÁI nhỏ hơn PHẢI
<code>&gt;=</code>	Giá trị TRÁI lớn hơn/bằng PHẢI
<code>&lt;=</code>	Giá trị TRÁI nhỏ hơn/bằng PHẢI

# Toán tử

## Toán tử **Logic** – luận lý:

Toán tử	Mô tả
&&	AND - Trả về TRUE nếu cả 2 biểu thức là TRUE
	OR - Trả về TRUE nếu 1 trong 2 biểu thức là TRUE



# Toán tử

## Toán tử **Bitwise**:

Toán tử	Mô tả
&	AND - So sánh hai bit, trả về 1 nếu cả hai bit là 1
	OR - So sánh hai bit, trả về 1 nếu một trong hai bit là 1
^	XOR – Trả về 1 nếu một trong hai bit là 1, trả về 0 nếu cả hai cùng là 0 hoặc cùng là 1
~	Nghịch đảo giá trị bit
>>	Dịch các bit sang PHẢI
<<	Dịch các bit sang TRÁI

# Toán tử

## Toán tử **Hỗn hợp**:

```
variable x = (expression) ? value if true : value if false
```

```
public class Test {  
  
    public static void main(String args[]) {  
        int a, b;  
        a = 10;  
        b = (a == 1) ? 20: 30;  
        System.out.println( "Value of b is : " + b );  
  
        b = (a == 10) ? 20: 30;  
        System.out.println( "Value of b is : " + b );  
    }  
}
```

# Toán tử

Độ ưu tiên Toán tử:

Toán tử	Mô tả
1.	Phép toán trong ngoặc ()
2.	Toán tử đơn: +, -, ++, --, ~, !
3.	Toán tử toán học và Bitwise như: *, /, %, +, -, >>, <<
4.	Toán tử so sánh: >, >=, <, <=, ==, !=
5.	Toán tử điều kiện và Bitwise như: &, ^,  , &&,
6.	Toán tử hỗn hợp và gán như: ?::, =, *=, /=, +=, -=

# Toán tử

1

- $(2*3+4/2) > 3 \ \&\& \ 3<5 \ || \ 10<9$
- Đầu tiên phép tính trong ngoặc được thực hiện trước.

2

- $((2*3)+(4/2)) > 3 \ \&\& \ 3<5 \ || \ 10<9$
- Phép chia và nhân được thực hiện trước khi phép cộng trong ngoặc thực hiện.

3

- $(6+2) > 3 \ \&\& \ 3<5 \ || \ 10<9$

4

- $(8>3) \ \&\& \ [3<5] \ || \ [10<9]$
- Tiếp theo các phép toán quan hệ được ưu tiên xử lý trước.

5

- Phép toán logic AND được thực hiện trước vì ưu tiên hơn và nằm về bên trái.
- $(\text{True} \ \&\& \ \text{True}) \ || \ \text{False}$

6

- Phép toán thực hiện cuối cùng là OR.
- $\text{True} \ || \ \text{False}$

7

- $\text{True}$

# Toán tử

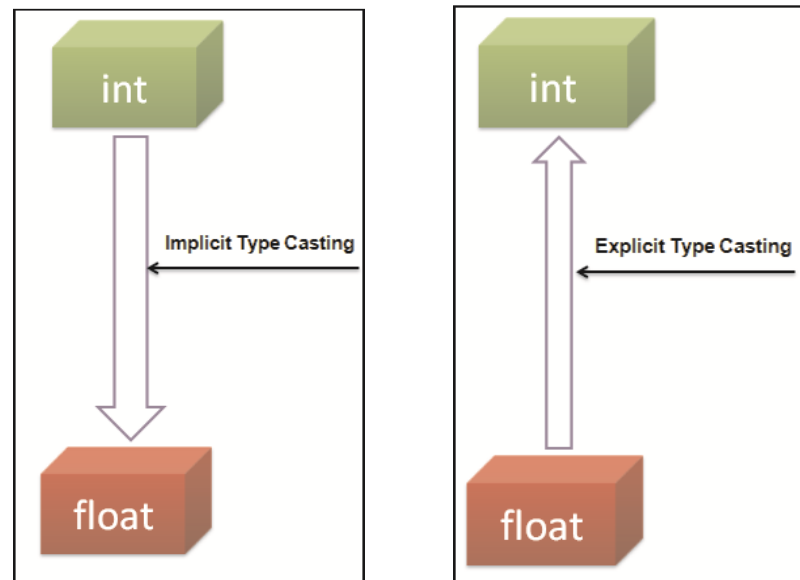
Độ ưu tiên theo hướng (trái-phải):

Associative	Operators
R to L	<code>++ -- + - ~ ! (&lt;data_type&gt;)</code>
L to R	<code>* / %</code>
L to R	<code>+ -</code>
L to R	<code>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</code>
L to R	<code>&lt; &gt; &lt;= &gt;= instanceof</code>
L to R	<code>== !=</code>
L to R	<code>&amp;</code>
L to R	<code>^</code>
L to R	<code> </code>
L to R	<code>&amp;&amp;</code>
L to R	<code>  </code>
R to L	<code>&lt;boolean_expr&gt; ? &lt;expr1&gt; :&lt;expr2&gt;</code>
R to L	<code>= *= /= %= += -= &lt;&lt;= &gt;&gt;= &gt;&gt;&gt;= &amp;= ^=  =</code>

# Toán tử

**Ép kiểu:** kiểu dữ liệu có thể chuyển đổi qua lại. Khi chuyển đổi Java yêu cầu phải ép kiểu.

- Ép kiểu không tường minh (**implicit**): khi ép kiểu dữ liệu có dung lượng thấp **lên** cao
- Ép kiểu tường minh (**explicit**): khi ép kiểu dữ liệu có dung lượng cao **xuống** thấp.



# Toán tử

---

Ví dụ ép kiểu:

## Không tường minh

```
...  
double dbl = 10;  
long lng = 100;  
int in = 10;  
dbl = in; // assigns the integer value to double variable  
lng = in; // assigns the integer value to long variable
```

## Tường minh

```
...  
float a = 21.3476f;  
int b = (int) a + 5;  
...
```

# Mệnh đề điều kiện

---

- Một chương trình Java là một tập hợp các câu lệnh được thực hiện tuần tự theo thứ tự mà nó xuất hiện.
- Sự thay đổi dòng chảy của lệnh được xử lý bằng các lệnh điều khiển khác nhau.
- Có 3 loại lệnh điều khiển được hỗ trợ là:

**1. Lệnh điều kiện**

**2. Lệnh lặp**

**3. Lệnh rẽ nhánh**



# Mệnh đề điều kiện

---

Mệnh đề điều kiện cho phép

- Thay đổi luồng thực hiện của chương trình.
- Đánh giá một điều kiện và căn cứ vào đó mà thực hiện chuỗi lệnh phù hợp.
- Có 2 mệnh đề điều kiện hỗ trợ trong Java:



**if Statement**

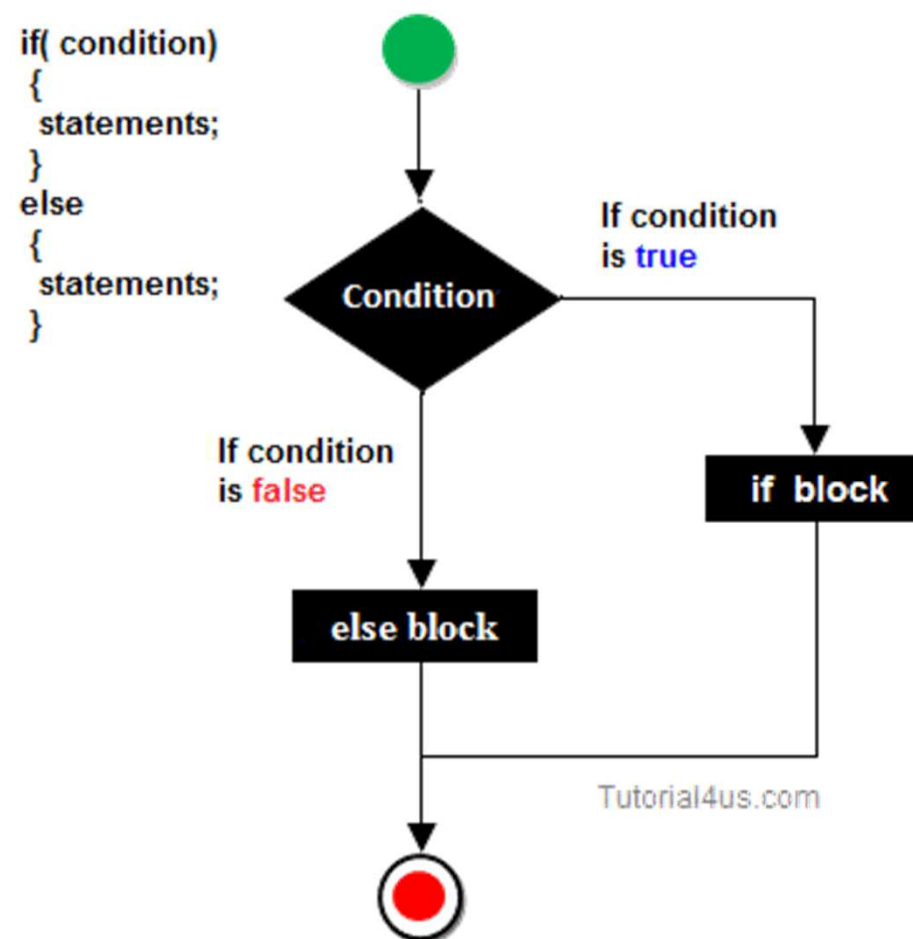


**Switch-case  
Statement**

# IF - ELSE

Mệnh đề **IF** có 3 tình huống:

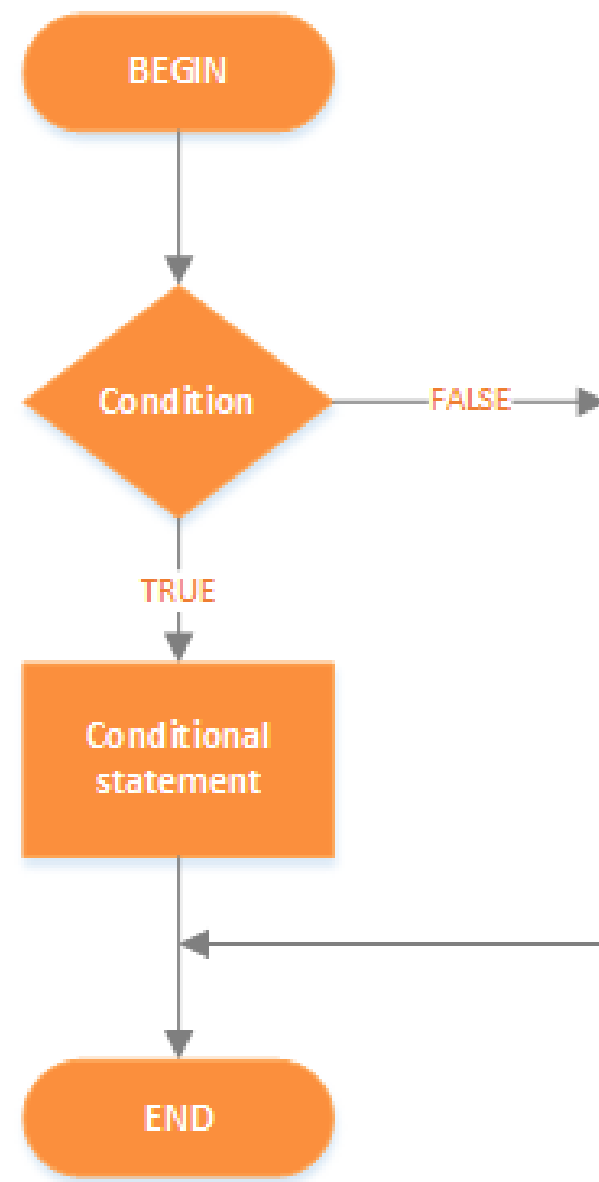
- 1) Xét điều kiện **đơn**.
- 2) Xét điều kiện **đôi**.
- 3) Xét **đa** điều kiện.



# IF - ELSE

Điều kiện ĐƠN:

```
if(a == 5){  
    // Khối lệnh thực hiện nếu a = 5  
}
```



# IF - ELSE

Điều kiện ĐƠN:

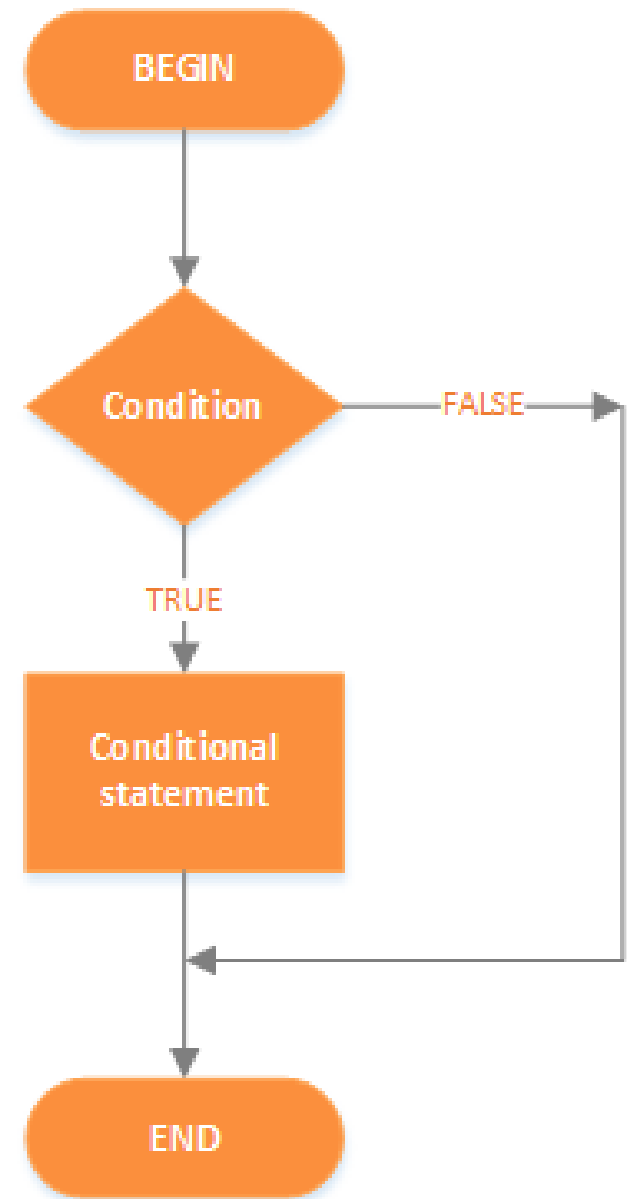
```
if(a == 5 && b == 10){
```

```
/*
```

```
Khởi lệnh thực hiện nếu a = 5  
và b = 10
```

```
*/
```

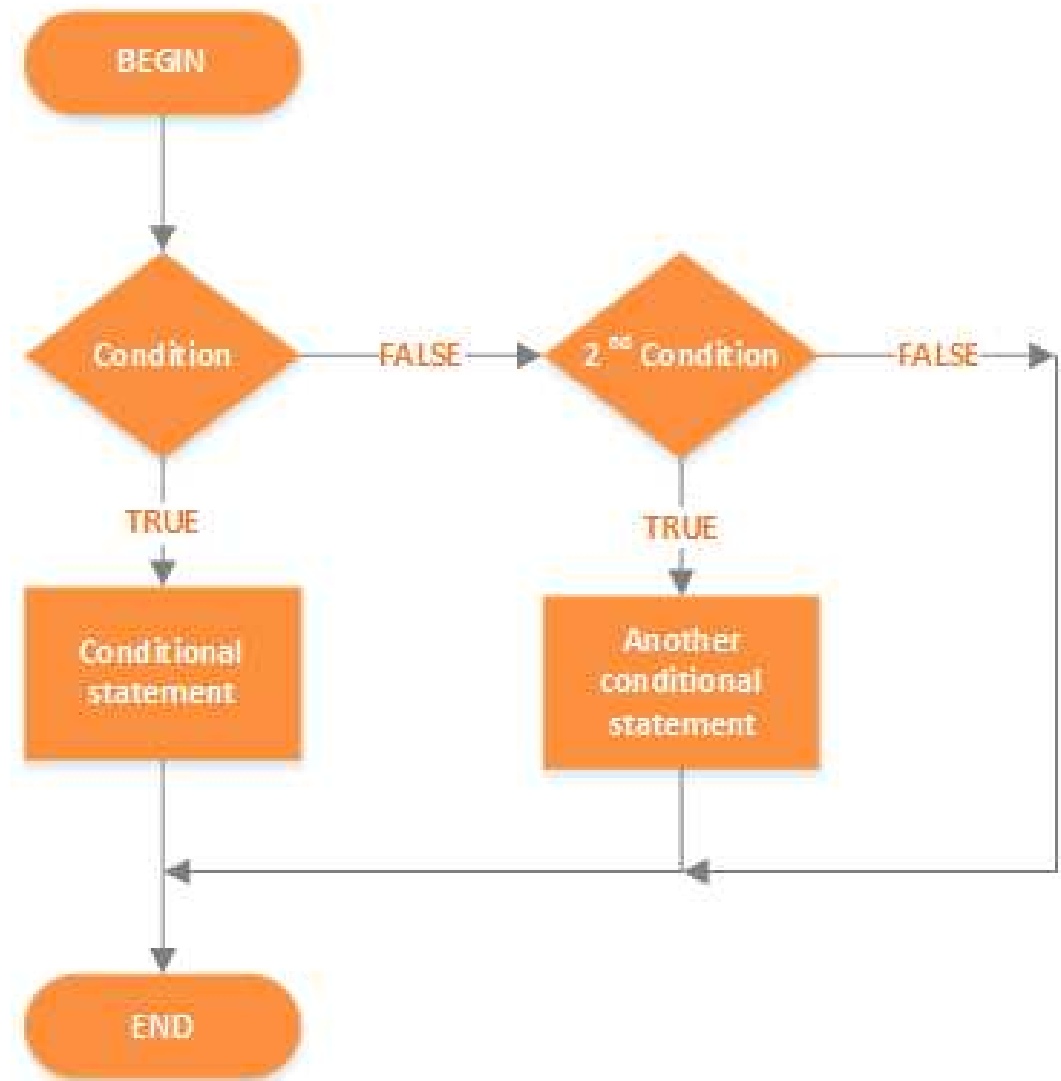
```
}
```



# IF - ELSE

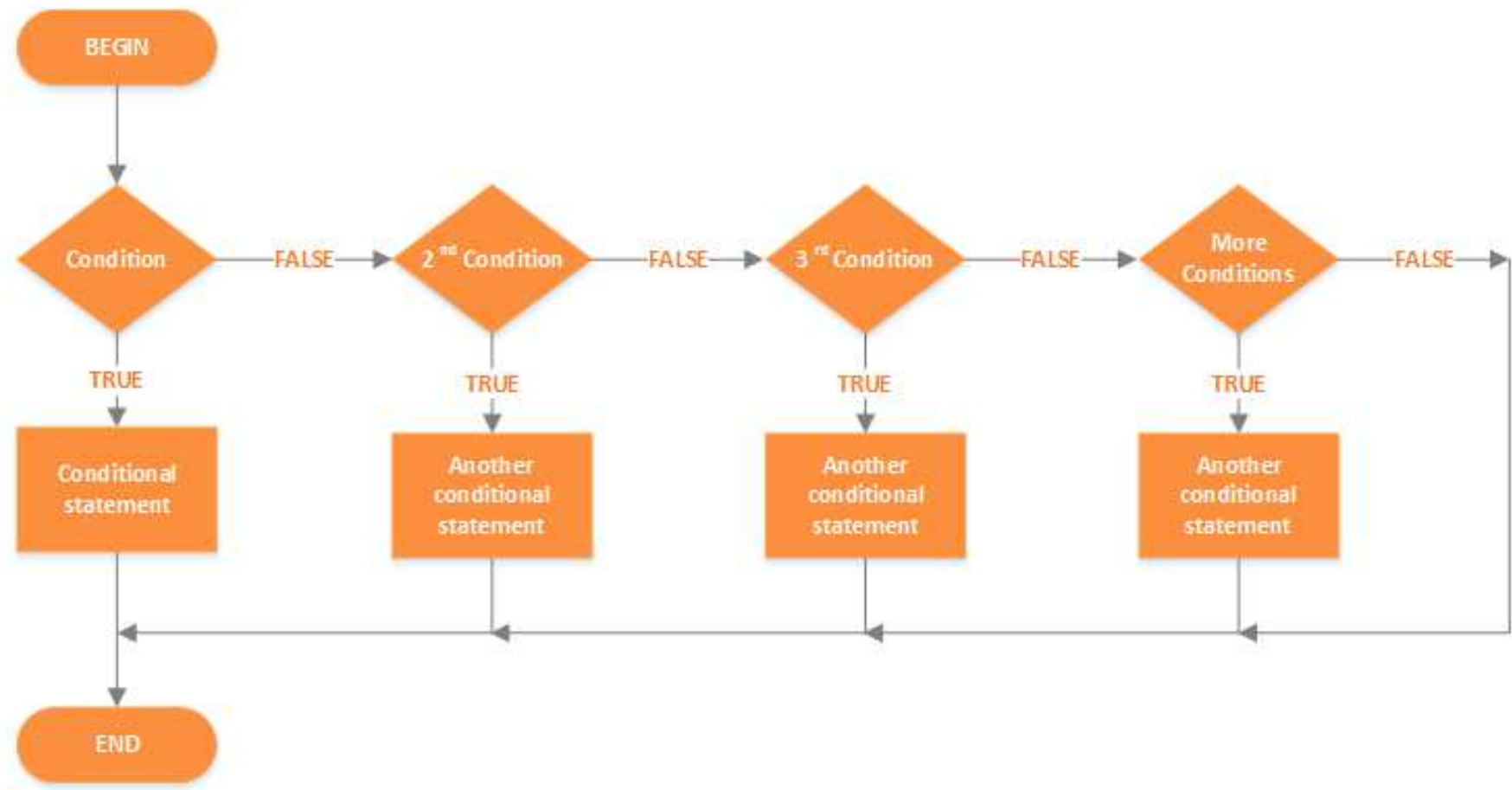
Điều kiện ĐÔI:

```
if (a == 5) {  
    // Khối lệnh thực hiện  
    // nếu a = 5  
}  
else {  
    // Khối lệnh thực hiện  
    // nếu biểu thức false  
}
```



# IF - ELSE

ĐA Điều kiện:



# IF - ELSE

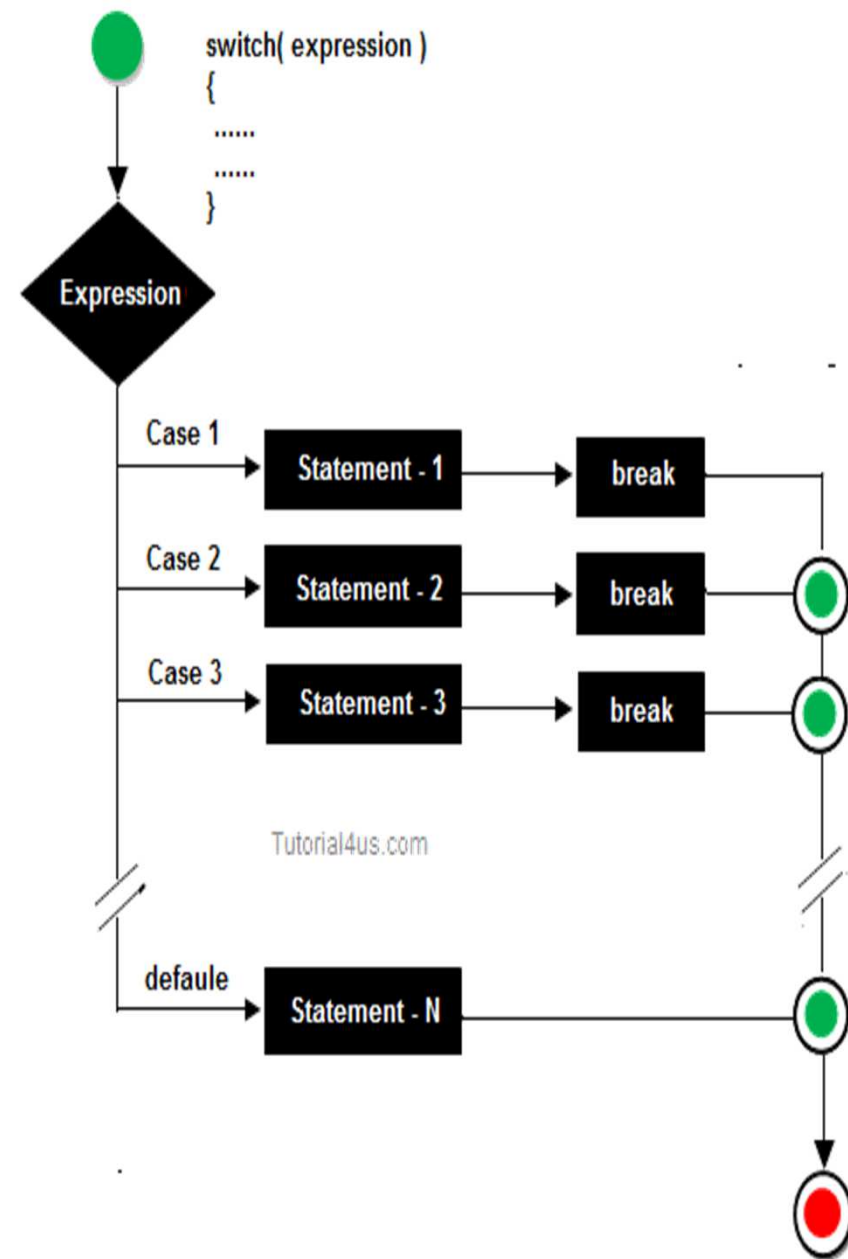
---

ĐA Điều kiện:

```
if (a == 5) {  
    // Khởi lệnh thực hiện nếu a = 5  
} else if (b == 2) {  
    // Khởi lệnh thực hiện nếu b = 2  
} else if (c == 9) {  
    // Khởi lệnh thực hiện nếu c = 9  
} else {  
    // Khởi lệnh thực hiện nếu không có biểu thức nào đúng  
}
```

# SWITCH-CASE

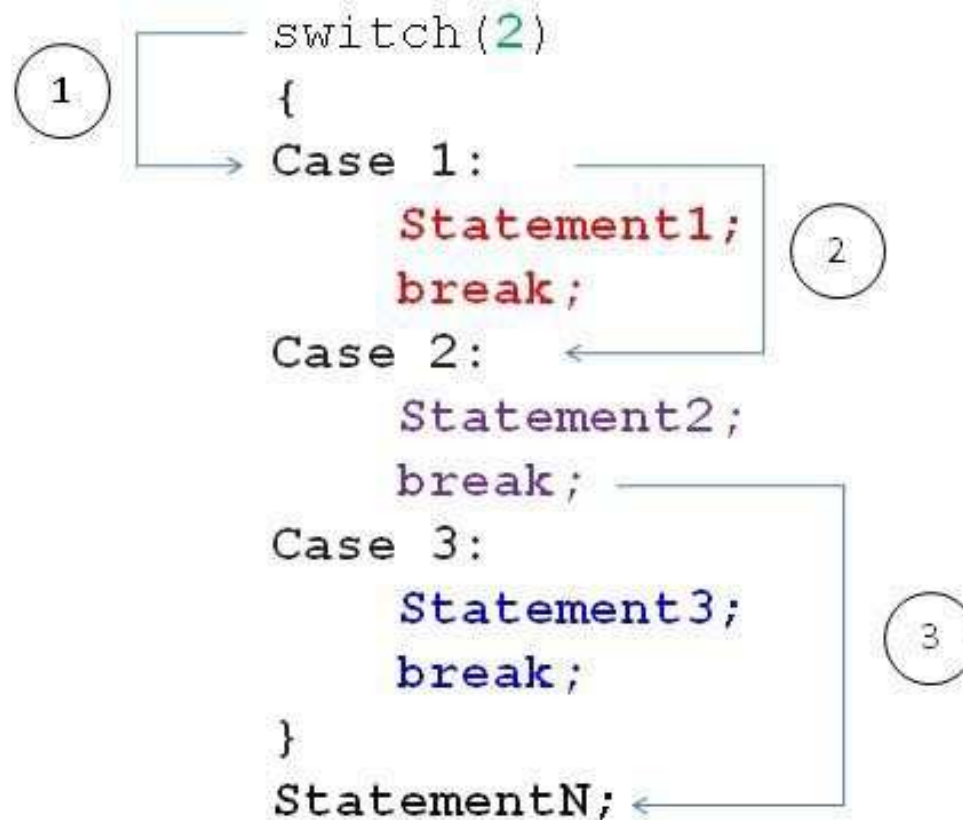
- Lệnh **SWITCH-CASE** được sử dụng để thực thi mã (code) từ nhiều điều kiện, trường hợp khác nhau.
- Lệnh này cơ bản giống với IF-ELSE bậc thang.
- Lệnh làm việc với các kiểu dữ liệu: byte, short, char, int và kiểu String ( $\geq$  JDK 7).





# SWITCH-CASE

- Cú pháp lệnh **SWITCH-CASE**:



# Tóm tắt bài học

---

- ✓ **Biến** là vùng **lưu trữ dữ liệu** trên máy tính được sử dụng trong quá trình tính toán.
- ✓ Phạm vi tồn tại và truy cập biến phụ thuộc vào vị trí khai báo trong **class** và **method**.
- ✓ Có 2 loại kiểu dữ liệu là: kiểu dữ liệu **cơ bản** và kiểu dữ liệu **tham chiếu**.
- ✓ Nhập xuất sử dụng **Scanner** và hàm **println()**.
- ✓ **Toán tử** là những **biểu tượng** giúp thao tác thực hiện một số chức năng trên dữ liệu.

# Tóm tắt bài học

---

- ✓ Chương trình Java là tập hợp các **câu lệnh** được **thực hiện tuần tự** theo thứ tự mà chúng xuất hiện.
- ✓ Java hỗ trợ 3 loại điều khiển luồng thực hiện câu lệnh gồm: **điều kiện**, **lặp** và **lệnh phân nhánh**.
- ✓ Lệnh **If-Else** định nghĩa khối lệnh nào được thực thi khi với mỗi trường hợp đúng-sai.
- ✓ Lệnh **Switch-case** là giải pháp cho tình huống có nhiều lựa chọn có kiểu giống nhau.