

# Java coding convention & các cấu trúc lệnh cơ bản

**Giảng viên :Cao Le Thanh**

## Mục tiêu bài học

- ❖ Hiểu rõ và sử dụng các kiểu dữ liệu nguyên thủy
- ❖ Hiểu rõ cách chuyển đổi chuỗi sang các kiểu nguyên thủy
- ❖ Hiểu cách sử dụng try ..catch để bắt lỗi
- ❖ Sử dụng thành thạo lệnh if, switch case
- ❖ Sử dụng thành thạo mảng và lệnh lặp
- ❖ Biết cách viết 1 chương trình đơn giản theo đúng quy ước



# Nội dung

- ❖ 1. Các kiểu dữ liệu nguyên thủy
- ❖ 2. Chuyển đổi kiểu dữ liệu
- ❖ 3. Các toán tử cơ bản
- ❖ 4. Lệnh rẽ nhánh ( IF , Switch)
- ❖ 5. Lệnh lặp & ngắt
- ❖ 6. Mảng
- ❖ 7. Tổ chức một chương trình java



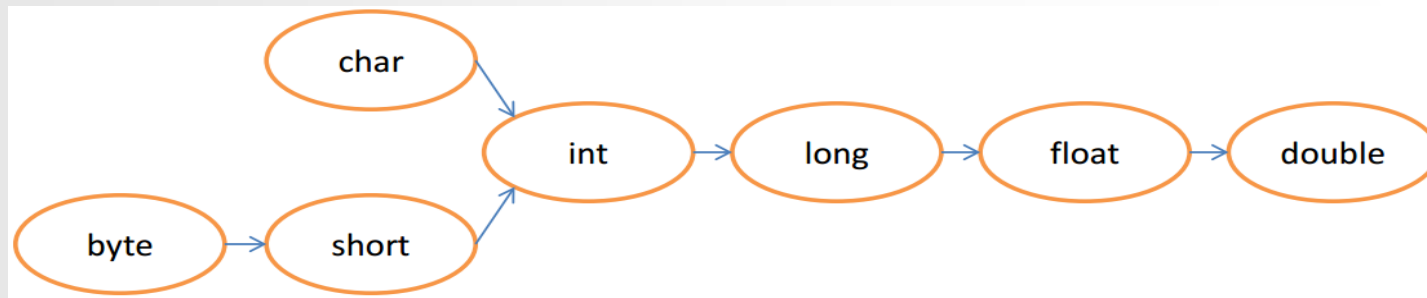
# 1. Các kiểu dữ liệu nguyên thủy

Kiểu	Mặc định	Bit	Khả năng lưu trữ	
			Giá trị nhỏ nhất	Giá trị lớn nhất
byte	0	8	$-2^7$	$+2^7-1$
short	0	16	$-2^{15}$	$+2^{15}-1$
int	0	32	$-2^{31}$	$+2^{31}-1$
long	0L	64	$-2^{63}$	$+2^{63}-1$
float	0.0F	32	$-3.40292347 \times 10^{38}$	$+3.40292347 \times 10^{38}$
double	0.0	64	$-1.79769313486231570 \times 10^{308}$	$+1.79769313486231570 \times 10^{308}$
char	'\u0000'	16	'\u0000'	'\uFFFF'
boolean	false	1	false	true

*Giá trị mặc định là giá trị sẽ được gán cho biến khi khai báo không khởi đầu giá trị cho biến*

## 1.2 Quy luật chuyển đổi kiểu

- ❖ Với các kiểu nguyên thủy thì ép kiểu tự động xảy ra theo chiều mũi tên



- ❖ Ví dụ :

```
int a = 5 ;
```

```
double b = 9.4 ;
```

```
b = a ; // ép tự động
```

```
a = (int) b ; // ép tường minh
```

## 2.3 Chuyển đổi kiểu chuỗi sang kiểu cơ bản

### ❖ Ví dụ 1 :

- String a = "3" ;
- String b= "4" ;
- String c = a+ b ;
- ➔ c = "34"

### ❖ Ví dụ 2 :

- Int a = Integer.parseInt("3");
- Int b = Integer.parseInt("4");
- Int c = a + b ;
- ➔ c = 7

```
byte Byte.parseInt(String)
```

```
short Short.parseInt(String)
```

```
int Integer.parseInt(String)
```

```
long Long.parseLong(String)
```

```
float Float.parseFloat(String)
```

```
double Double.parseDouble(String)
```

```
boolean Boolean.parseBoolean(String)
```

## 2.4 Kiểm soát lỗi chuyển kiểu

- ❖ Xét trường hợp

`int a = scanner.nextInt();`

hoặc

`int a = Integer.parseInt(s);`

- ❖ Điều gì sẽ xảy ra khi người dùng nhập không phải số hoặc chuỗi s không phải là chuỗi chứa số!

- ❖ Hãy sử dụng lệnh `try...catch` để kiểm soát các lỗi trên

```
static Scanner input= new Scanner(System.in);
public static void main(String[] args) {
    try{
        int a = input.nextInt();
        System.out.println("Bạn đã nhập đúng");
    }
    catch (Exception ex){
        System.out.println("Vui lòng nhập số !");
    }
}
```

## 2.5 Lớp bọc kiểu nguyên thủy (Wrapper)

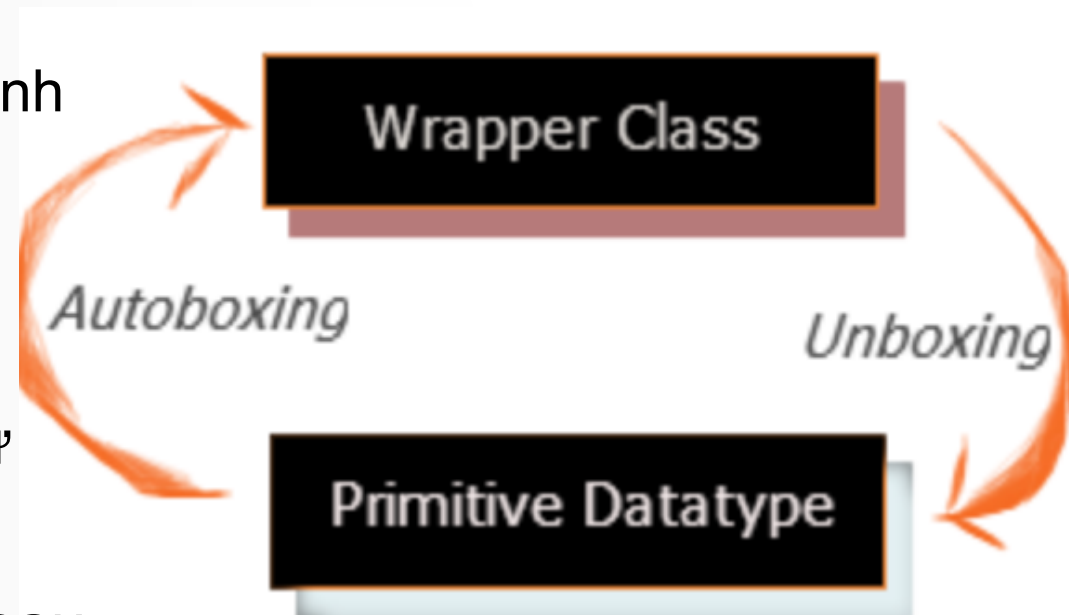
- ❖ Tương ứng với mỗi kiểu cơ bản Java định nghĩa một lớp bao để bao giá trị của kiểu tương ứng gọi là lớp bao kiểu.
- ❖ Rất nhiều hàm trong Java chỉ làm việc với đối tượng mà không làm việc với kiểu cơ bản

Nguyên Thủy	Lớp bao
byte	⇔ <b>Byte</b>
short	⇔ <b>Short</b>
int	⇔ <b>Integer</b>
long	⇔ <b>Long</b>
float	⇔ <b>Float</b>
double	⇔ <b>Double</b>
char	⇔ <b>Character</b>
boolean	⇔ <b>Boolean</b>




## 2.6 Bao( Boxing) / Mở bao(Unboxing)

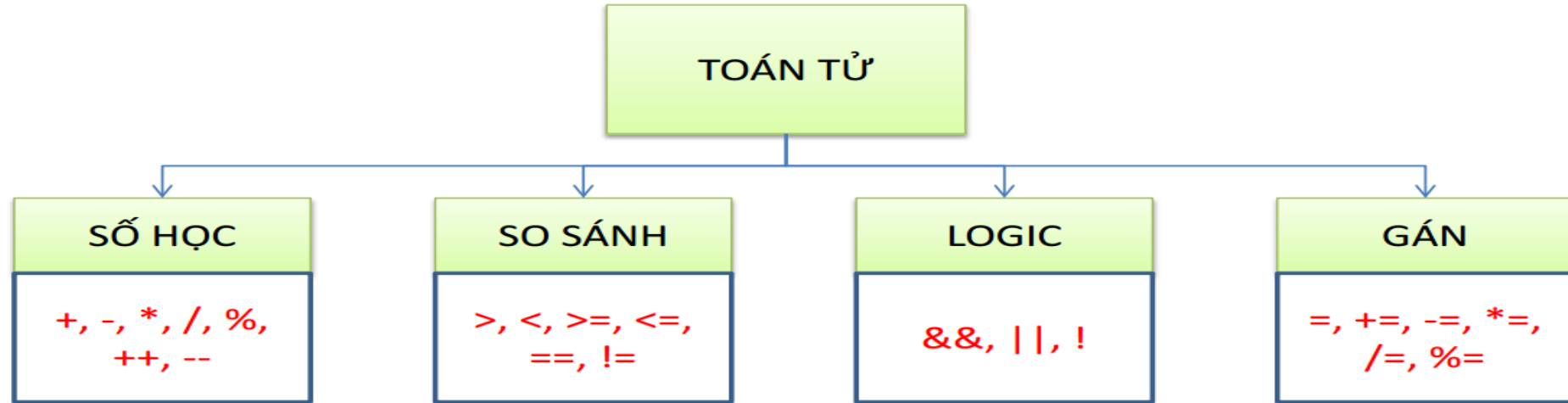
- ❖ Boxing là việc tạo đối tượng từ lớp bao để bọc giá trị nguyên thủy.
- ❖ Có 3 cách để bao giá trị nguyên thủy sau
  - `Integer a = Integer.valueOf(5)` // bao tường minh
  - `Integer a = new Integer(5)` // bao tường minh
  - `Integer a = 5` // bao ngầm định
- ❖ Unboxing là việc mở lấy giá trị nguyên thủy từ đối tượng của lớp bao
- ❖ Có 2 cách mở bao để lấy giá trị nguyên thủy sau
  - `int b = a.intValue()` // mở bao tường minh
  - `int b = a;` // mở bao ngầm định



## 2.7 Boxing/unboxing

Boxing	Unboxing	Ví dụ
Byte.valueOf(byte)	<<Byte>>.byteValue()	<pre>long a = 5L; Long b = Long.valueOf(a); long c = b.longValue();</pre> 
Short.valueOf(short)	<<Short>>.shortValue()	
Integer.valueOf(int)	<<Integer>>.intValue()	
Long.valueOf(long)	<<Long>>.longValue()	
Float.valueOf(float)	<<Float>>.floatValue()	
Double.valueOf(double)	<<Double>>.doubleValue()	
Boolean.valueOf(boolean)	<<Boolean>>.booleanValue()	

## 3.1 Toán tử và biểu thức



Biểu thức là sự kết hợp giữa toán tử và toán hạng. Kết quả của biểu thức là một giá trị.

**Giá trị của các biểu thức sau?**

```
int x = 11 % 4;
```

```
boolean a = 9 < 2 && true || 4 > 3;
```

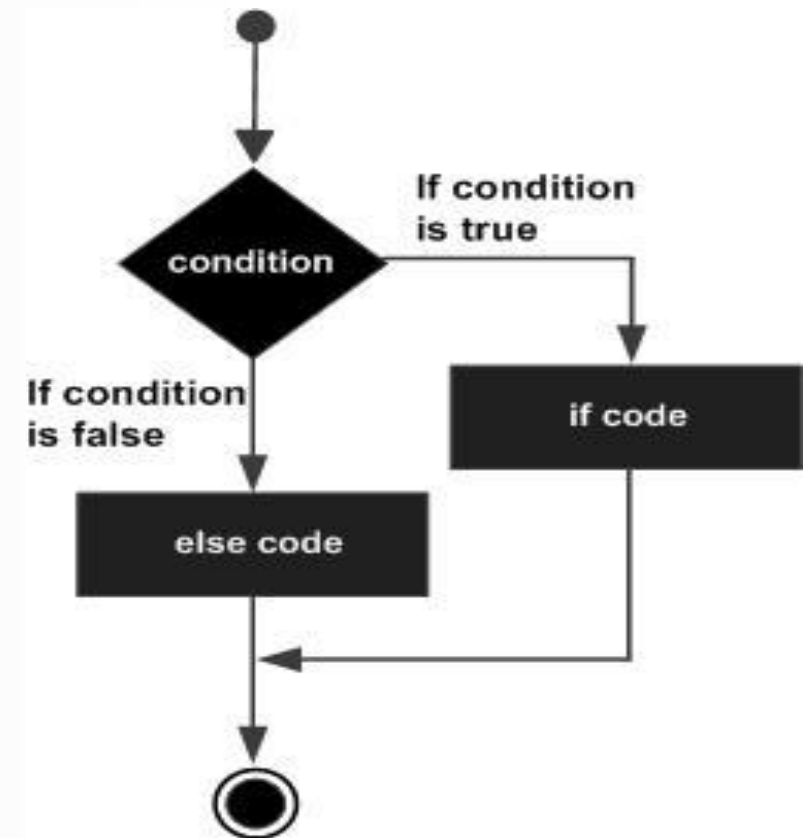
## 3.2 Danh sách các toán tử

Category	Operator	Name/Description	Example	Result
Arithmetic	+	Addition	3+2	5
	-	Subtraction	3-2	1
	*	Multiplication	3*2	6
	/	Division	10/5	2
	%	Modulus	10%5	0
	++	Increment and then return value	X=3; ++X	4
		Return value and then increment	X=3; X++	3
	--	Decrement and then return value	X=3; --X	2
		Return value and then decrement	X=3; X--	3
Logical	&&	Logical “and” evaluates to true when both operands are true	3>2 && 5>3	False
		Logical “or” evaluates to true when either operand is true	3>1    2>5	True
	!	Logical “not” evaluates to true if the operand is false	3!=2	True
Comparison	==	Equal	5==9	False
	!=	Not equal	6!=4	True
	<	Less than	3<2	False
	<=	Less than or equal	5<=2	False
	>	Greater than	4>3	True
	>=	Greater than or equal	4>=4	True
String	+	Concatenation(join two strings together)	“A”+“BC”	ABC



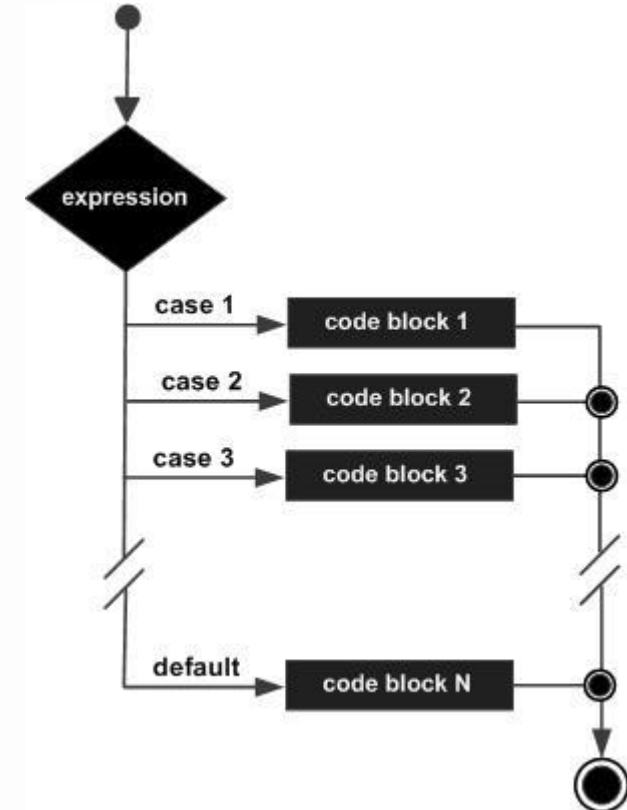
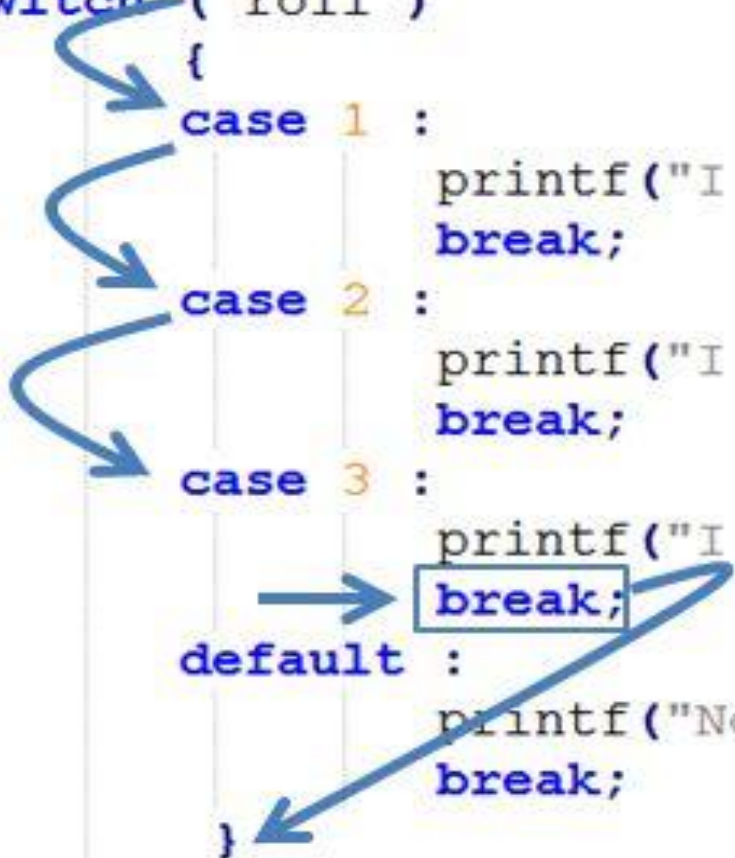
## 4.1 Lệnh rẽ nhánh (IF)

```
public static void main(String[] args) {  
  
    int user = 21;  
  
    if (user <= 18) {  
        System.out.println("User is 18 or younger");  
    }  
    else if (user > 18 && user < 40) {  
        System.out.println("User is between 19 and 39");  
    }  
  
    else {  
        System.out.println("User is older than 40");  
    }  
}
```



## 4.2 lệnh rẽ nhánh

```
int roll = 3 ;  
switch ( roll )  
{  
    case 1 :  
        printf("I am Pankaj");  
        break;  
    case 2 :  
        printf("I am Nikhil");  
        break;  
    case 3 :  
        printf("I am John");  
        break;  
    default :  
        printf("No student found");  
        break;  
}
```

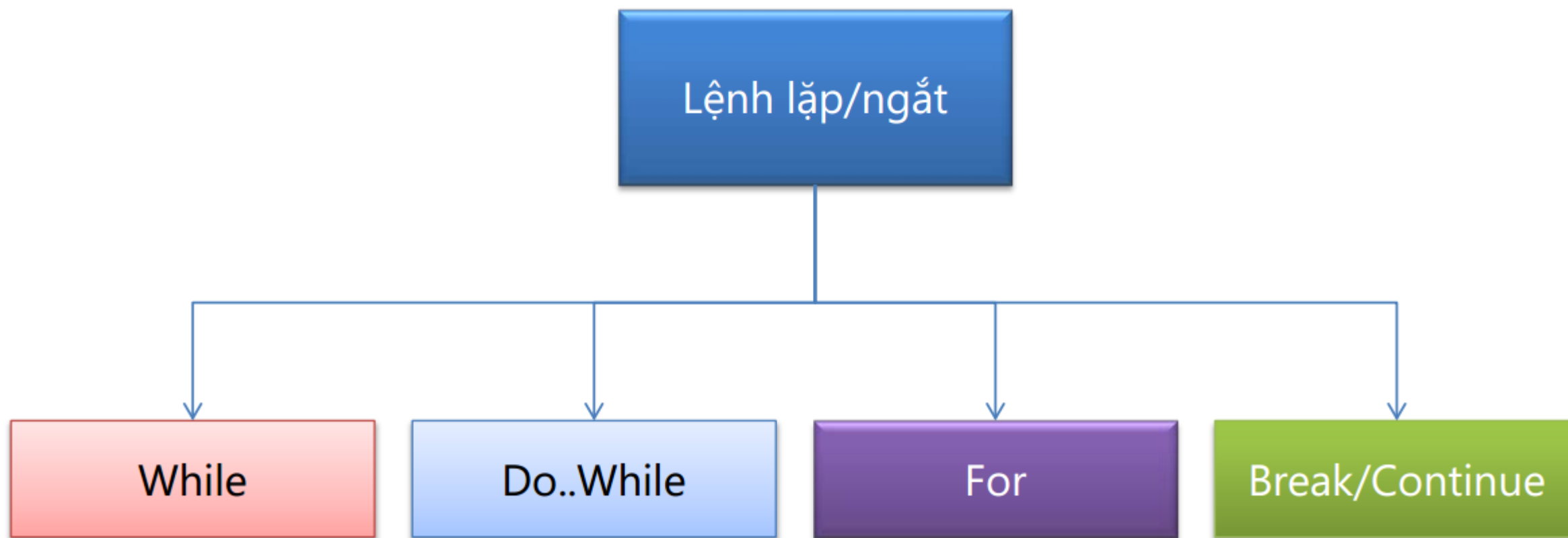


## 4.3 Thảo luận

- ❖ Thảo luận : Khi nào thì dùng IF, khi nào dùng switch case ?



## 5.1 Lệnh lặp





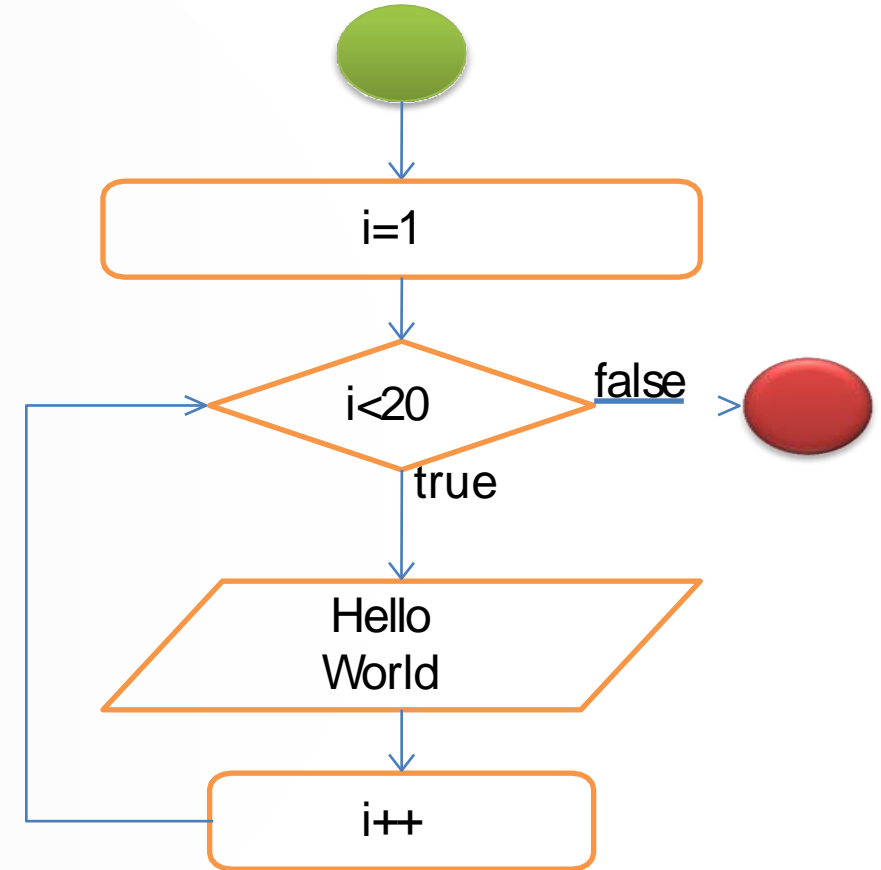
## 5.2 While

### ❑ Ví dụ

```
int i = 1;  
while (i < 20) { System.out.println("Hello World !");  
    i++;  
}
```

### ❑ Diễn giải:

- ❖ Đoạn mã trên xuất 19 dòng Hello World ra màn hình



## 5.3 Do

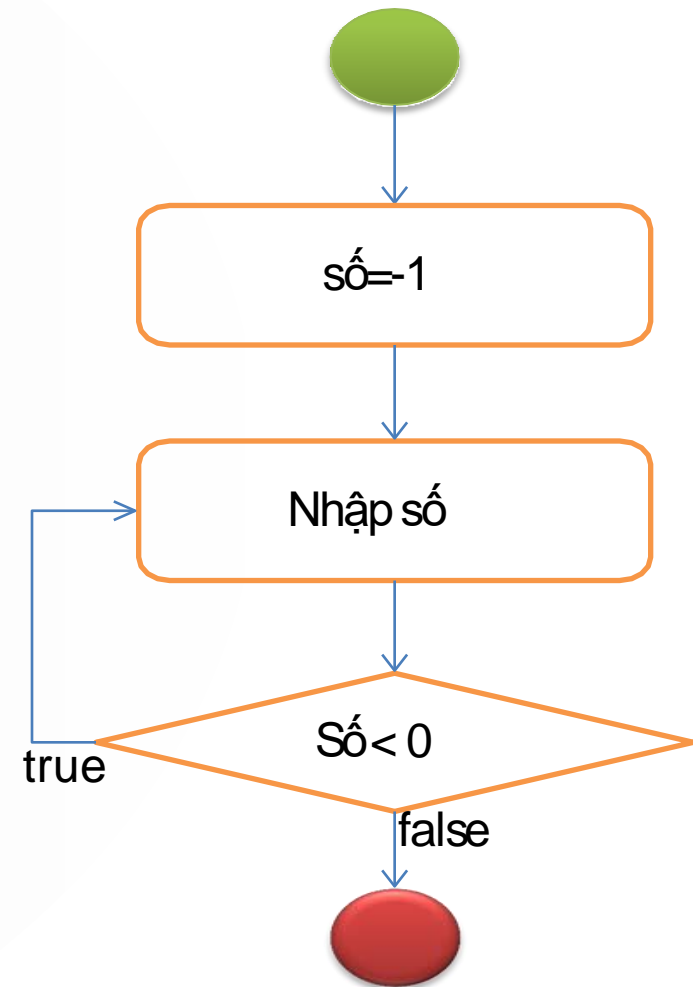
### ❑ Ví dụ

```
int so = -1; do {  
    so = scanner.nextDouble();  
}
```

```
while (so < 0);
```

### ❑ Diễn giải:

- ❖ Đoạn mã trên chỉ cho phép nhập số nguyên dương từ bàn phím.

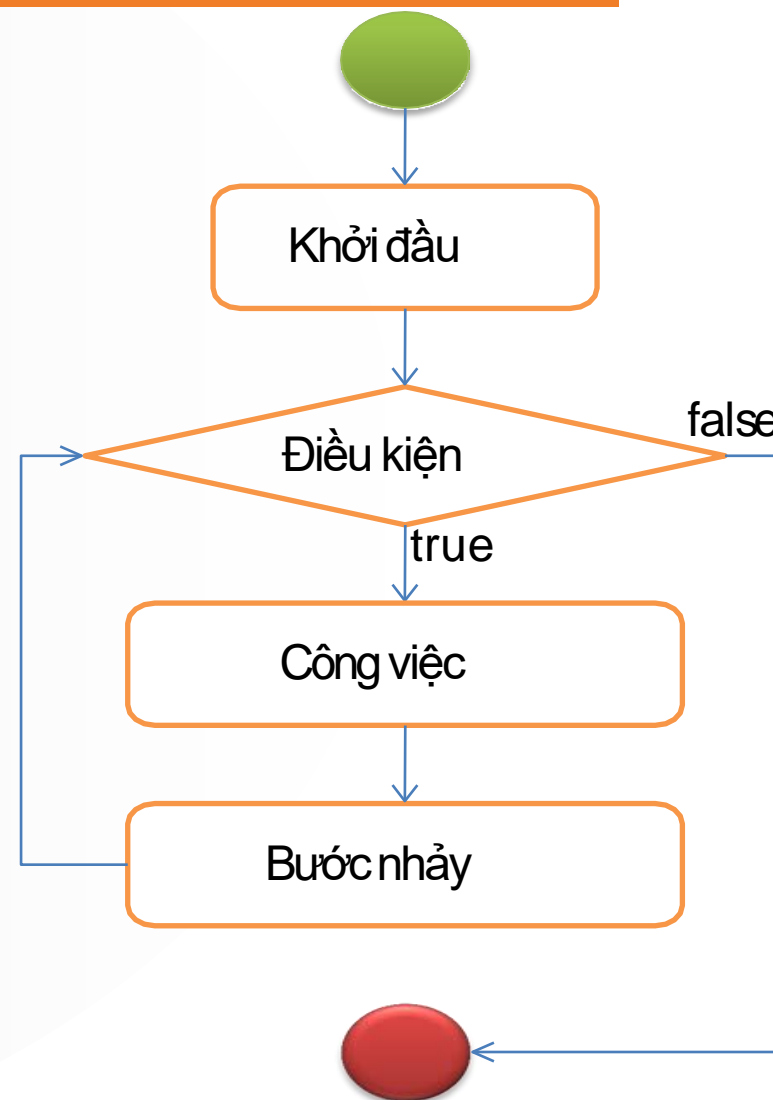


## 5.4 For

### ❑ Cú pháp

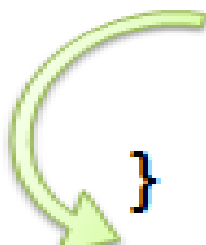
```
for (khởi đầu ; điều kiện; bước nhảy){  
    // công việc  
}
```

### ❑ Diễn giải




## 5.5 Ngắt

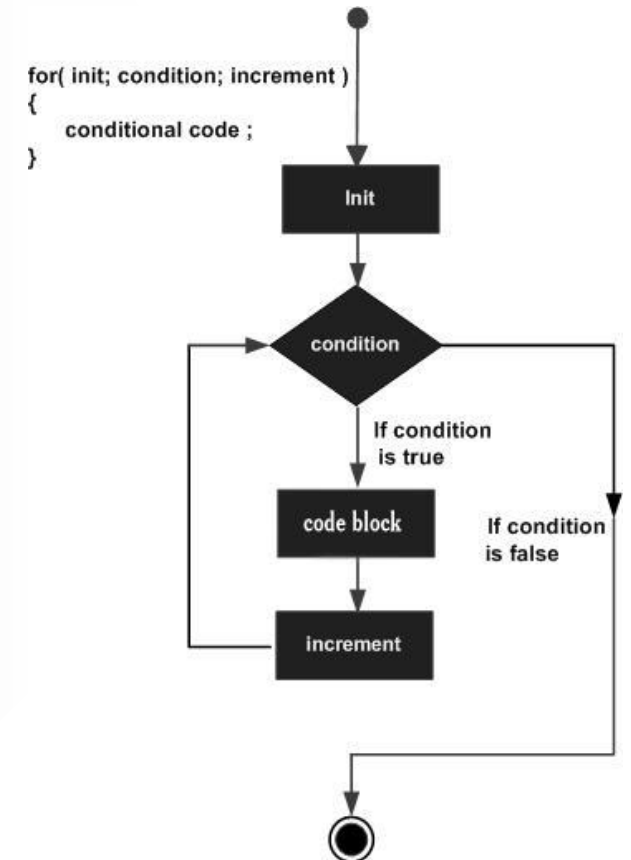
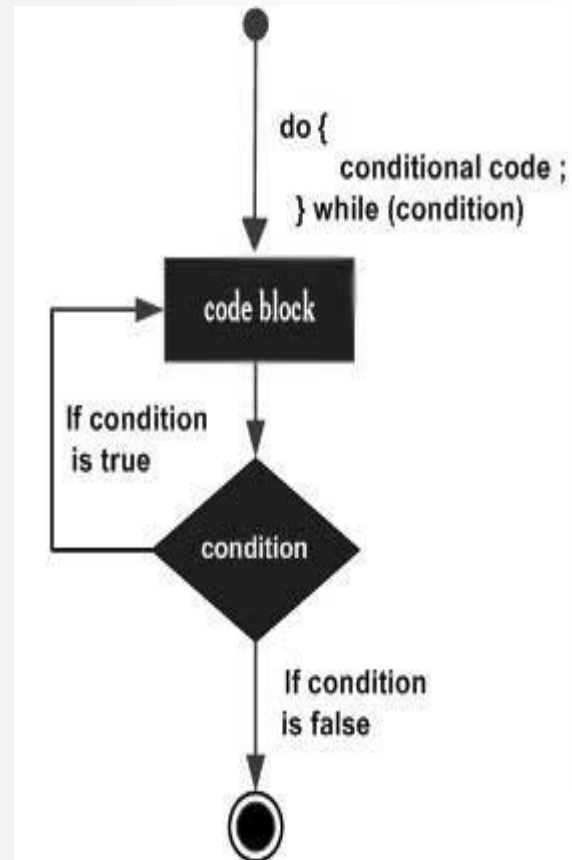
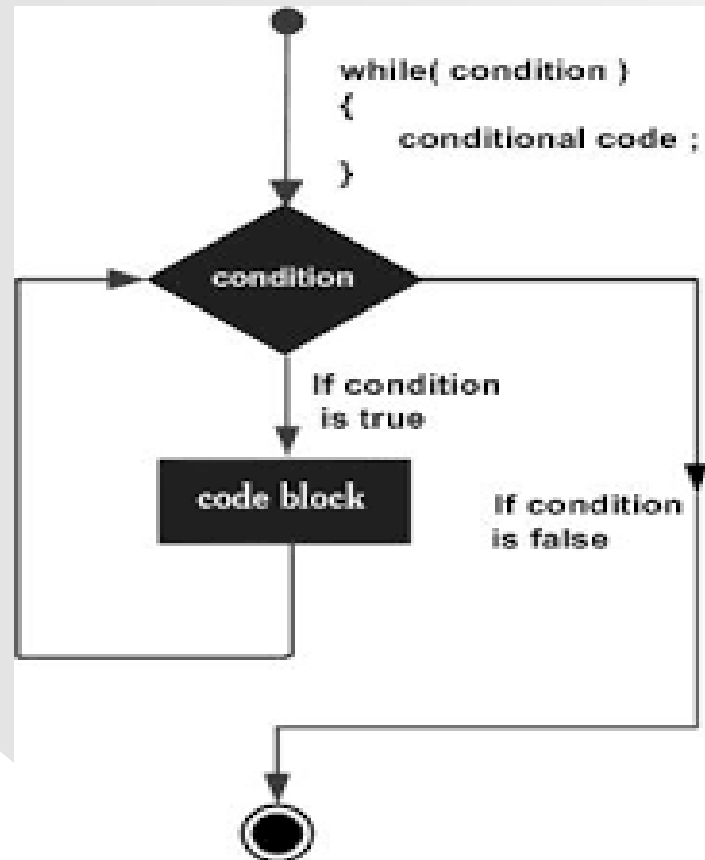
```
<Lệnh lặp>  
{  
    ...  
    break;  
    ...  
}
```



```
<Lệnh lặp>  
{  
    ...  
    continue;  
    ...  
}
```



## 5.6 Demo & discuss



## 6.1 Mảng

- ❑ Mảng là cấu trúc lưu trữ nhiều phần tử có cùng kiểu dữ liệu

0	1	2	3	4	5	6	7	8
5	7	9	1	45	1	9	9	2

Indices Elements ←

- ❑ Để truy xuất các phần tử cần biết chỉ số (index). Chỉ số được đánh từ 0.
- ❑ Các thao tác mảng
  - ❖ Khai báo
  - ❖ Truy xuất (đọc/ghi) phần tử
  - ❖ Lấy số phần tử
  - ❖ Duyệt mảng
  - ❖ Sắp xếp các phần tử mảng

## 6.3 Khai báo mảng

### ❑ Khai báo không khởi tạo

- ❖ `int[] a;` // mảng số nguyên chưa biết số phần tử
- ❖ `int b[];` // mảng số nguyên chưa biết số phần tử
- ❖ `String[] c = new String[5];` // mảng chứa 5 chuỗi

### ❑ Khai báo có khởi tạo

- ❖ `double[] d1 = new double[]{2, 3, 4, 5, 6};` // mảng số thực, 5 phần tử, đã được khởi tạo
- ❖ `double[] d2 = {2, 3, 4, 5, 6};` // mảng số thực, 5 phần tử, đã được khởi tạo

## 6.3 Truy xuất mảng

- ❑ Sử dụng chỉ số (**index**) để phân biệt các phần tử. Chỉ số mảng tính từ 0.
  - ❖ `int a[] = {4, 3, 5, 7};`
  - ❖ `a[2] = a[1] * 4; // 45*4=180`
  - ❖ Sau phép gán này mảng là {4, 3, 12, 7};
- ❑ Sử dụng thuộc tính **length** để lấy số phần tử của mảng
  - ❖ `a.length` có giá trị là 9



## 6.5 Duyệt mảng bằng for

```
int[] a = {4, 3, 5, 9};  
for(int i=0; i<a.length; i++){  
    System.out.println(a[i]);  
}
```

← **for(;;)**

**for-each** →

```
int[] a = {4, 3, 5, 9};  
for (int x: a){  
    System.out.println(x);  
}
```

## 6.6 Ví dụ

❑ Ví dụ sau tính tổng các số chẵn của mảng.

- ❖ Lấy từng phần tử từ mảng với for-each
- ❖ Nếu là số chẵn thì cộng vào tổng

```
int[] a = {9, 3, 8, 7, 3, 9, 4, 2};  
  
double tong = 0;  
for(int x : a){  
    if(x % 2 == 0){  
        tong += x;  
    }  
}  
  
System.out.print("Tổng: " + tong);
```



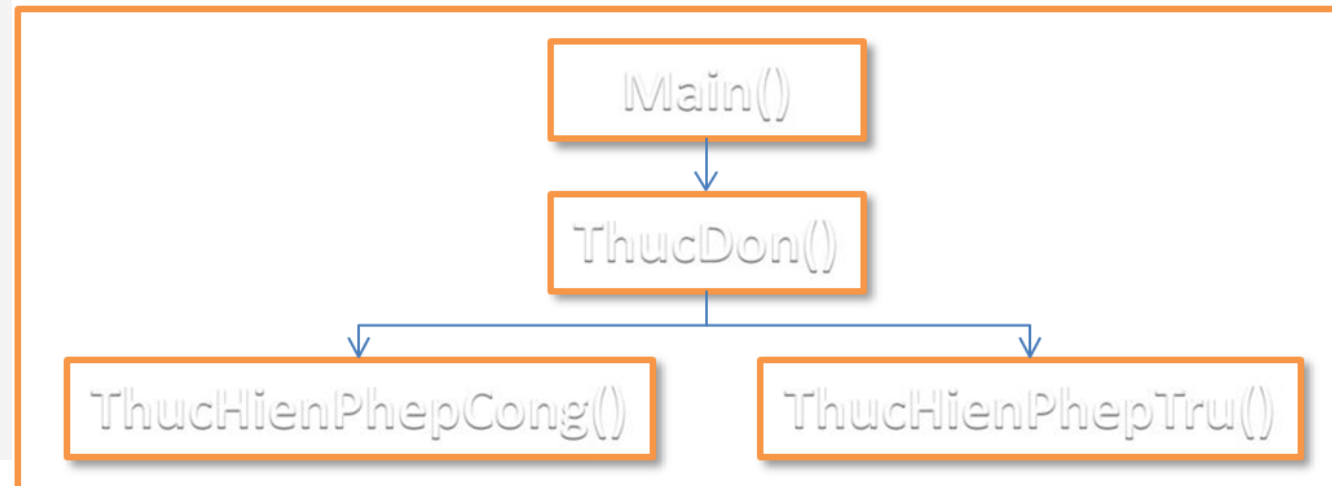
## 6.7 Thao tác nâng cao

```
Int[] a = {1, 9, 2, 8, 3, 7, 4, 6, 5};
```

Phương thức	Mô tả/ví dụ
<code>&lt;T&gt; List&lt;T&gt; <b>asList</b>(T... a)</code>	Chuyển một mảng sang List với kiểu tương ứng. Ví dụ: <b>List&lt;Integer&gt; b = Arrays.asList(a);</b>
<code>int <b>binarySearch</b>(Object[] a, Object key)</code>	Tìm vị trí xuất hiện đầu tiên của một phần tử trong mảng. Ví dụ: <b>int i = Arrays.binarySearch(a, 8);</b>
<code>void <b>sort</b>(Object[] a)</code>	Sắp xếp các phần tử theo thứ tự tăng dần. Ví dụ: <b>Arrays.sort(a);</b>
<code>String <b>toString</b>(Object[] a)</code>	Chuyển mảng thành chuỗi được bọc giữ cặp dấu ngoặc kép, các phần tử mảng cách nhau dấu phẩy. Ví dụ: <b>String s = Arrays.toString(a);</b>
<code>void <b>fill</b>(Object[] a, Object val)</code>	Gán 1 giá trị cho tất cả các phần tử mảng. Ví dụ: <b>Arrays.fill(a, 9);</b>



## 7.1 Tổ chức chương trình Java & quy ước



```
import java.util.Scanner;

public class ChuongTrinh {
    public static void main(String[] args) {
        thucDon();
    }

    public static void thucDon() {
    }

    public static void thucHienPhepCong() {
    }

    public static void thucHienPhepTru() {
    }
}
```

Hiển thị thực đơn chính  
của chương trình

## 7.2 Thiết kế thực đơn

```
System.out.println(">> MÁY TÍNH CÁ NHÂN <<");  
System.out.println("+-----+");  
System.out.println("| 1. Cộng          |");  
System.out.println("| 2. Trừ           |");  
System.out.println("| 3. Kết thúc      |");  
System.out.println("+-----+");  
System.out.println(" >> Chọn chức năng? ");
```

```
Scanner scanner = new Scanner(System.in);  
int answer = scanner.nextInt();  
if(answer == 1){  
    thucHienPhepCong();  
}  
else if(answer == 2){  
    thucHienPhepTru();  
}  
else if(answer == 3){  
    System.exit(0);  
}
```

Gọi phương thức thực hiện  
phép cộng

Gọi phương thức thực hiện  
phép trừ

Thoát ứng dụng

## 7.4 Quy ước viết code

<u>Kiểu</u>	<u>Quy ước</u>	<u>Ví dụ</u>
<b>Class ( <u>Lớp</u> )</b>	<u>Viết hoa chữ đầu theo Pascal case , không viết tắt , rõ nghĩa , dùng danh từ</u>	Class <b>H</b> oc <b>S</b> inh Class <b>S</b> tudent
<b>Methods ( <u>Phương thức</u> )</b>	<u>Viết thường theo came case, dùng động từ mô tả</u>	run() ; <u>tínhTienLuong()</u>
<b>Variables ( <u>Biến</u> )</b>	<u>Viết đủ nghĩa , dùng danh từ hoặc tính từ</u>	<u>int doDai ;</u> <u>int length;</u>
<b>Constants ( <u>Hằng</u> )</b>	<u>Dùng danh từ, viết hoa có gạch nối</u>	<u>Int DO_DAI_TOI_DA = 10 ;</u> <u>Int MAX_LENGTH =10 ;</u>



# Tổng kết

- ❖ 1. Các kiểu dữ liệu nguyên thủy
- ❖ 2. Chuyển đổi kiểu dữ liệu
- ❖ 3. Các toán tử cơ bản
- ❖ 4. Lệnh rẽ nhánh ( IF , Switch)
- ❖ 5. Lệnh lặp & ngắt
- ❖ 6. Mảng
- ❖ 7. Tổ chức một chương trình java

