



# NỘI DUNG

## LẬP TRÌNH HDT

*Chương 1: Tổng quan về lập trình HDT*

*Chương 2: Giới thiệu về lập trình Java*

*Chương 3: Lập trình Java cơ sở*

*Chương 4: Lập trình hướng đối tượng với Java*

*Chương 5: Xử lý ngoại lệ*



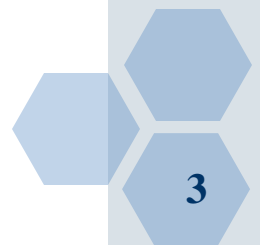
# CHƯƠNG 3. LẬP TRÌNH JAVA CƠ SỞ





# LẬP TRÌNH JAVA CƠ SỞ

- ❖ **Bộ ký tự, từ khoá, tên**
- ❖ Các kiểu và cấu trúc dữ liệu cơ bản
- ❖ Hằng, biến toán tử, biểu thức
- ❖ Nhập xuất cơ sở
- ❖ Các cấu trúc điều khiển
- ❖ Mảng
- ❖ Hàm – Truyền tham số và các lời gọi hàm

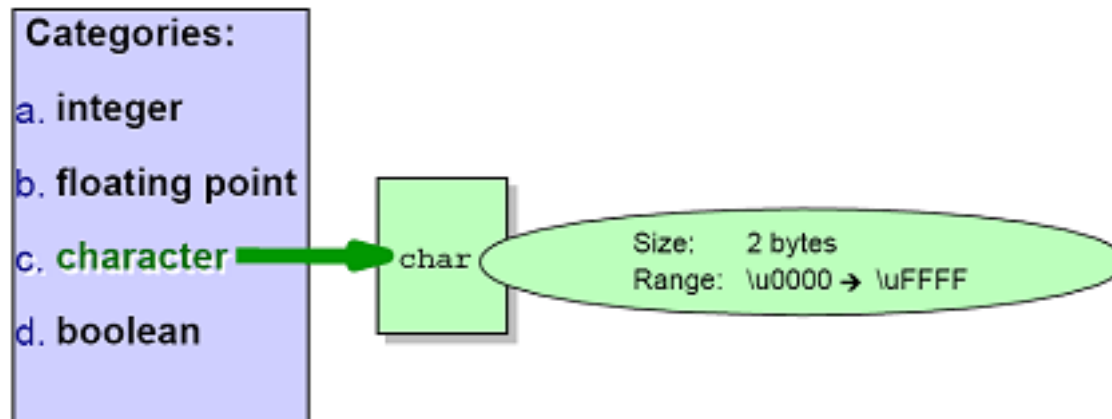




# Bộ ký tự, từ khoá, tên

## a) Bộ ký tự:

- Dùng các ký tự unicode để đặt tên cho các định danh.
- Giá trị mặc định là giá trị zero (`\u0000`)





# Từ khoá của Java

Từ khóa	Mô tả
<i>abstract</i>	Sử dụng để khai báo lớp, phương thức trừu tượng
<i>boolean</i>	Kiểu dữ liệu logic
<i>break</i>	Được sử dụng để kết thúc vòng lặp hoặc cấu trúc switch
<i>byte</i>	kiểu dữ liệu số nguyên
<i>case</i>	được sử dụng trong lệnh switch
<i>cast</i>	Chưa được sử dụng (để dành cho tương lai)
<i>catch</i>	được sử dụng trong xử lý ngoại lệ
<i>char</i>	kiểu dữ liệu ký tự
<i>class</i>	Dùng để khai báo lớp
<i>const</i>	Chưa được dùng
<i>continue</i>	được dùng trong vòng lặp để bắt đầu một vòng lặp mới
<i>default</i>	được sử dụng trong lệnh switch
<i>do</i>	được dùng trong vòng lặp điều kiện sau
<i>double</i>	kiểu dữ liệu số thực
<i>else</i>	khả năng lựa chọn thứ hai trong câu lệnh if
<i>extends</i>	chỉ rằng một lớp được kế thừa từ một lớp khác
<i>false</i>	Gia trị logic
<i>final</i>	Dùng để khai báo hằng số, phương thức không thể ghi đè, hoặc lớp không thể kế thừa
<i>finally</i>	phần cuối của khối xử lý ngoại lệ



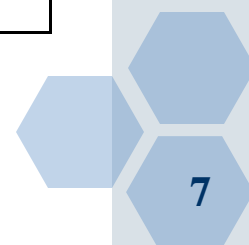
# Từ khoá của Java

<i>float</i>	kiểu số thực
<i>for</i>	Câu lệnh lặp
<i>goto</i>	Chưa được dùng
<i>if</i>	Câu lệnh lựa chọn
<i>implements</i>	chỉ rằng một lớp triển khai từ một giao diện
<i>import</i>	Khai báo sử dụng thư viện
<i>instanceof</i>	kiểm tra một đối tượng có phải là một thể hiện của lớp hay không
<i>interface</i>	sử dụng để khai báo giao diện
<i>long</i>	kiểu số nguyên
<i>native</i>	Khai báo phương thức được viết bằng ngôn ngữ biên dịch C++
<i>new</i>	tạo một đối tượng mới



# Từ khoá của Java

<i>null</i>	một đối tượng không tồn tại
<i>package</i>	Dùng để khai báo một gói
<i>private</i>	đặc tả truy xuất
<i>protected</i>	đặc tả truy xuất
<i>public</i>	đặc tả truy xuất
<i>return</i>	Quay từ phương thức về chỗ gọi nó
<i>short</i>	kiểu số nguyên
<i>static</i>	Dùng để khai báo biến, thuộc tính tĩnh
<i>super</i>	Truy xuất đến lớp cha
<i>switch</i>	lệnh lựa chọn
<i>synchronized</i>	một phương thức độc quyền truy xuất trên một đối tượng
<i>this</i>	Chỉ chính lớp đó
<i>throw</i>	Ném ra ngoại lệ
<i>throws</i>	Khai báo phương thức ném ra ngoại lệ
<i>true</i>	Giá trị logic
<i>try</i>	sử dụng để bắt ngoại lệ
<i>void</i>	Dùng để khai báo một phương thức không trả về giá trị
<i>while</i>	Dùng trong cấu trúc lặp

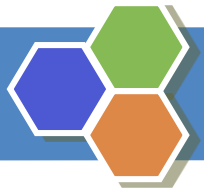




# Định danh (tên)

- ❖ Tên dùng để xác định duy nhất một đại lượng trong chương trình.
- ❖ *Quy tắc đặt tên:*
  - Không trùng với từ khoá
  - Không bắt đầu bằng một số, tên phải bắt đầu bằng ký tự hoặc \$, \_
  - Không chứa dấu cách, các ký tự toán học như +, -, \*, /
  - Không trùng với một định danh khác trong cùng một phạm vi.






# Định danh (tên)

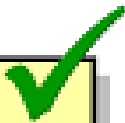
## ❖ *Chú ý:*

- Tên nên đặt sao cho có thể mô tả được đối tượng trong thực tế.
- Java phân biệt chữ hoa, chữ thường
- Trong java có thể đặt tên với độ dài tùy ý.

An-Identifier  
2nd\_Identifier  
goto  
10\$



An\_Identifier  
a\_2nd\_Identifier  
Go2  
\$10

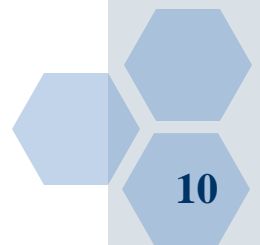




# Định danh (tên)

## ❖ Quy ước với định danh:

- Bắt đầu bằng chữ cái.
- Gói (package): tất cả sử dụng chữ thường.
  - theexample
- Lớp (Class): viết hoa chữ cái đầu tiên trong các từ ghép lại VD: TheExample
- Phương thức/ Thuộc tính (method/field): Bắt đầu bằng chữ thường, viết hoa chữ cái đầu tiên trong các từ còn lại. VD: theExample
- Hằng (constants): Tất cả viết hoa
  - THE\_EXAMPLE





# Chú thích (Comment)

## ❖ Chú thích trên một dòng:

- Tất cả các ký tự sau // cho đến cuối dòng là chú thích.

## ❖ Chú thích trên nhiều dòng

- Phần nằm giữa /\* và \*/ là chú thích

## ❖ Chú thích trong tư liệu:

- Tạo ra tư liệu dạng HTML cho chương trình
- Bắt đầu bằng /\*\* và kết thúc bằng \*/
- Ví dụ:

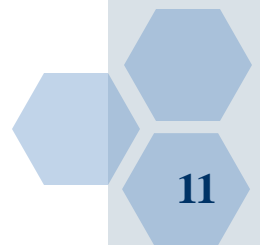
```
/**
```

```
 * Lớp này cài đặt giao diện Demo
```

```
 * Tác giả: Trần An
```

```
 * Version 1.0
```

```
 */
```





# LẬP TRÌNH JAVA CƠ SỞ

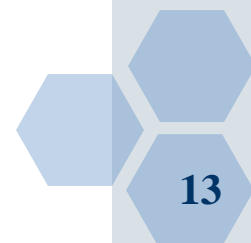
- ❖ Bộ ký tự, từ khoá, tên
- ❖ **Các kiểu và cấu trúc dữ liệu cơ bản**
- ❖ Hằng, biến, toán tử, biểu thức
- ❖ Nhập xuất cơ sở
- ❖ Các cấu trúc điều khiển
- ❖ Mảng
- ❖ Hàm – Truyền tham số và các lời gọi hàm



# Các kiểu dữ liệu cơ bản

## ❖ Trong Java kiểu dữ liệu được chia thành 2 loại:

- Kiểu dữ liệu nguyên thủy (primitive)
  - Số nguyên (integer)
  - Số thực (float)
  - Ký tự (char)
  - Giá trị logic (boolean)
- Kiểu dữ liệu tham chiếu (reference)
  - Mảng (array)
  - Lớp (Class)
  - Giao diện (Interface)



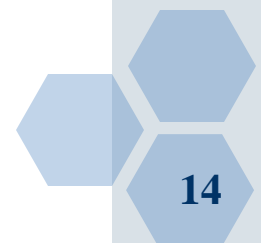


# Kiểu dữ liệu nguyên thủy

- ❖ **Mọi biến đều phải khai báo một kiểu dữ liệu**
  - Các kiểu dữ liệu cơ bản chứa một giá trị đơn
  - Kích thước và định dạng phải phù hợp với kiểu của nó.
- ❖ **Java chia thành các kiểu dữ liệu nguyên thủy:**

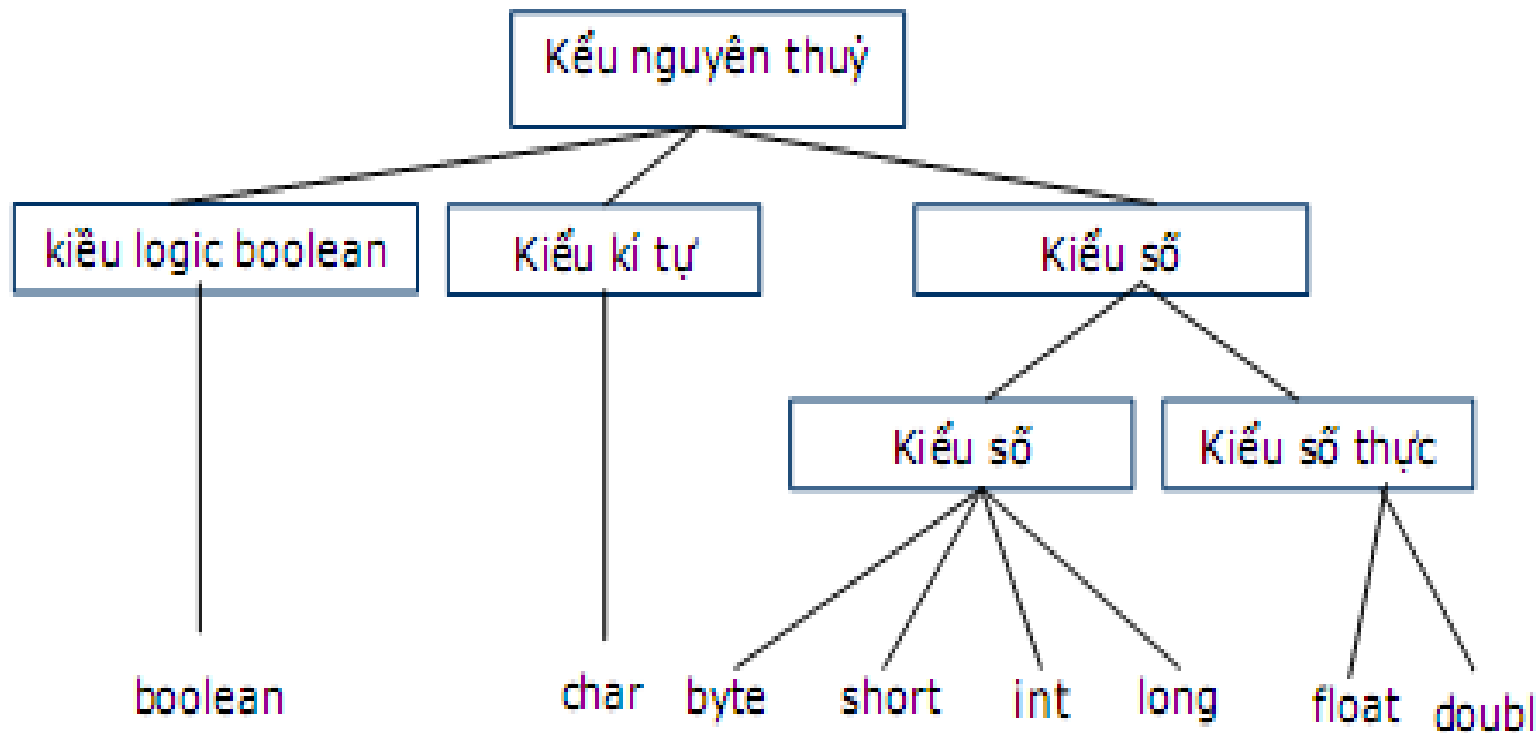
## Categories:

- a. integer
- b. floating point
- c. character
- d. boolean





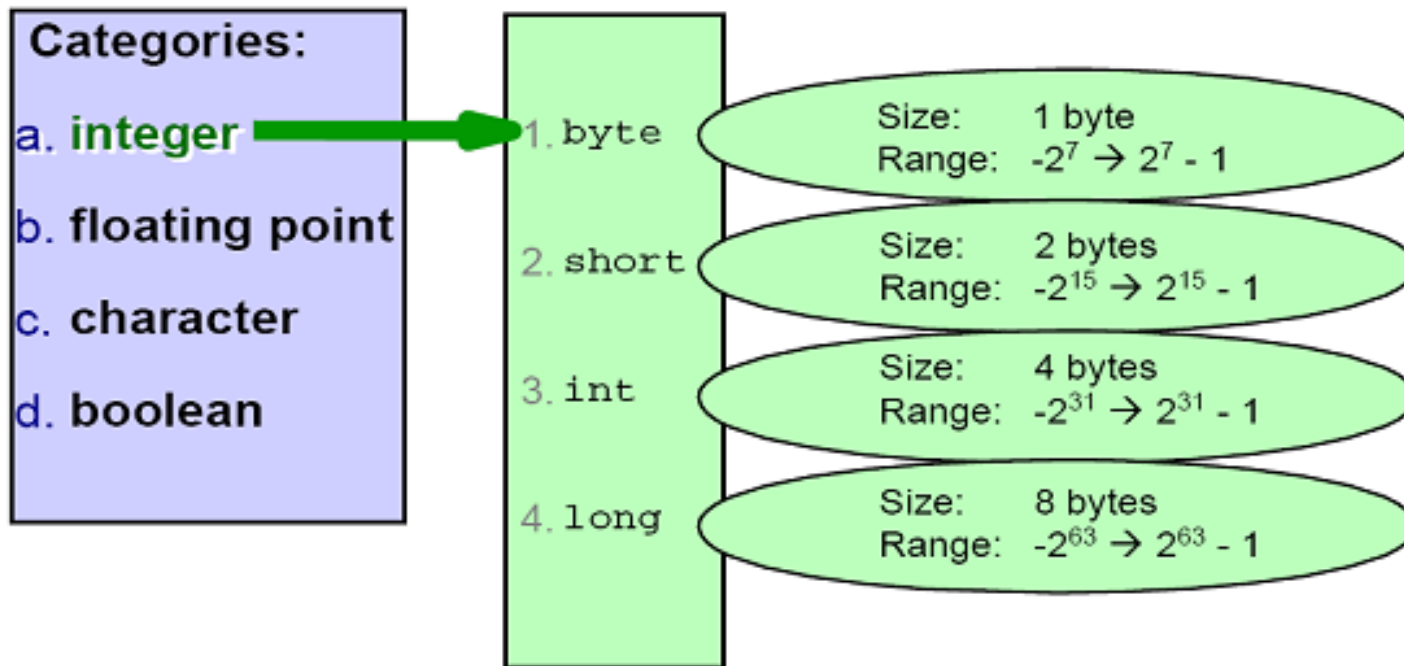
# Kiểu dữ liệu nguyên thủy





# Số nguyên

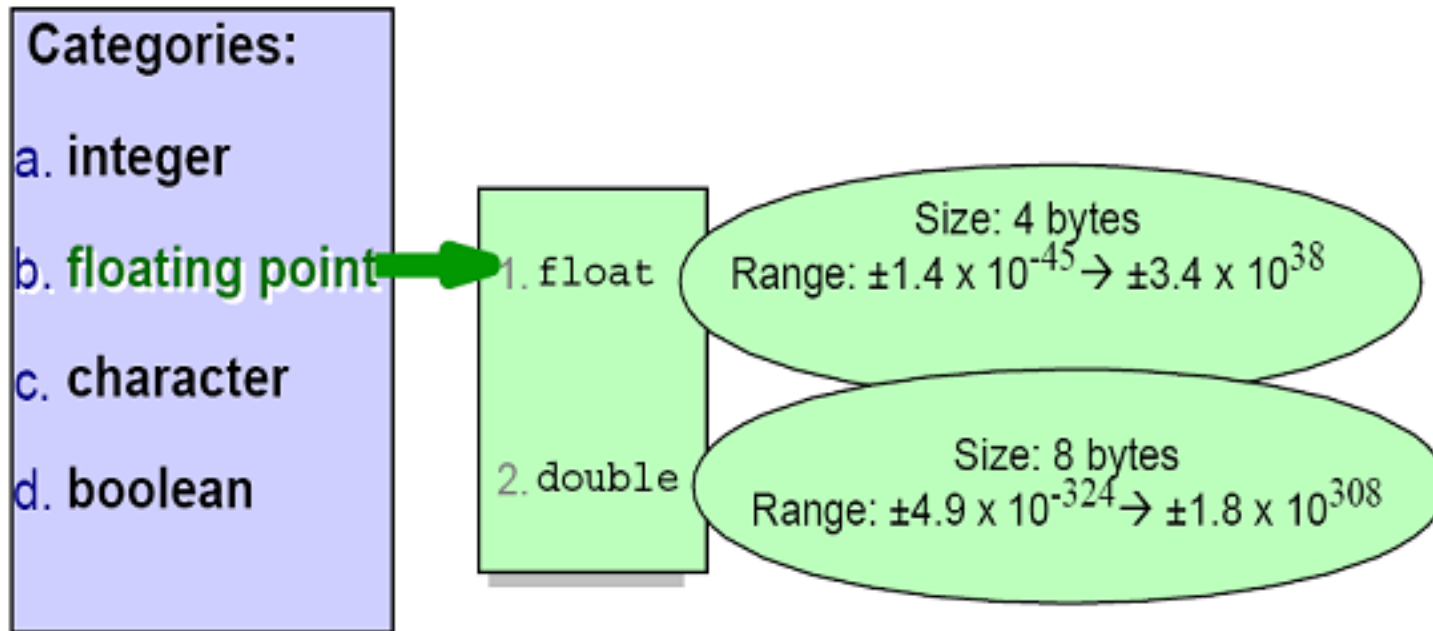
- ❖ Số nguyên có dấu
- ❖ Khởi tạo với giá trị 0







## ❖ Khởi tạo với giá trị 0.0





# Số thực

❖ **float** kết thúc bằng ký tự **f** (hoặc **F**)

VD: 7.1f

❖ **double** kết thúc bằng ký tự **d** (hoặc **D**)

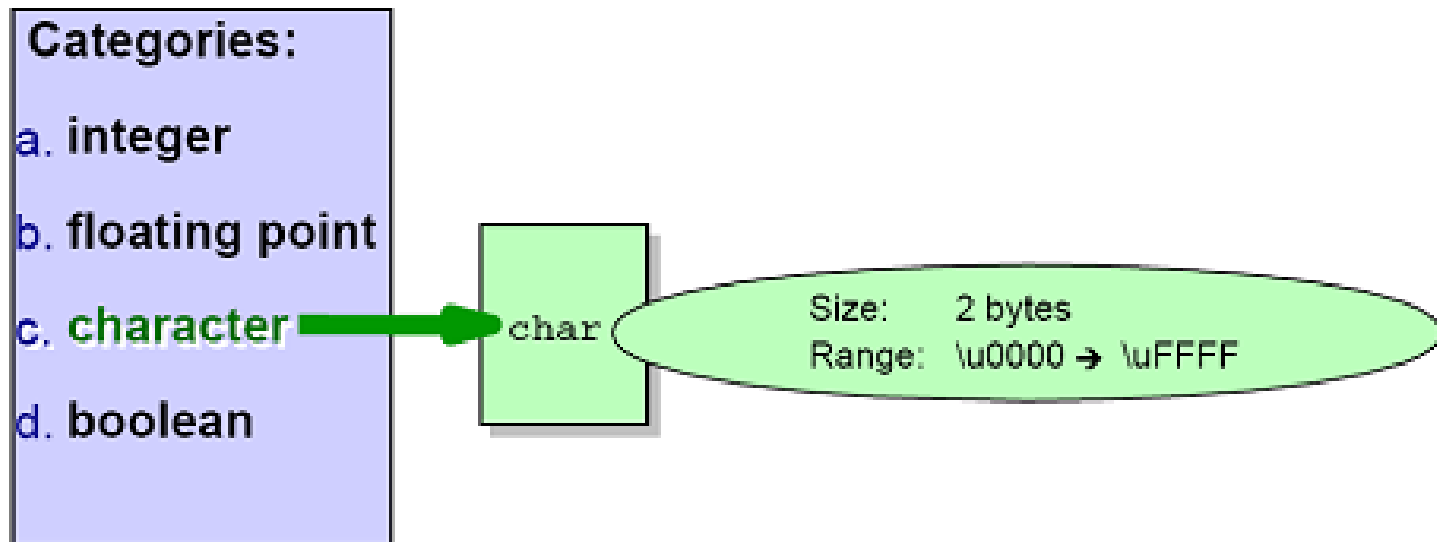
VD: 7.1d

❖ Một giá trị thực mà không có ký tự kết thúc đi kèm sẽ có kiểu là **double**

VD: 7.1 giống như 7.1d



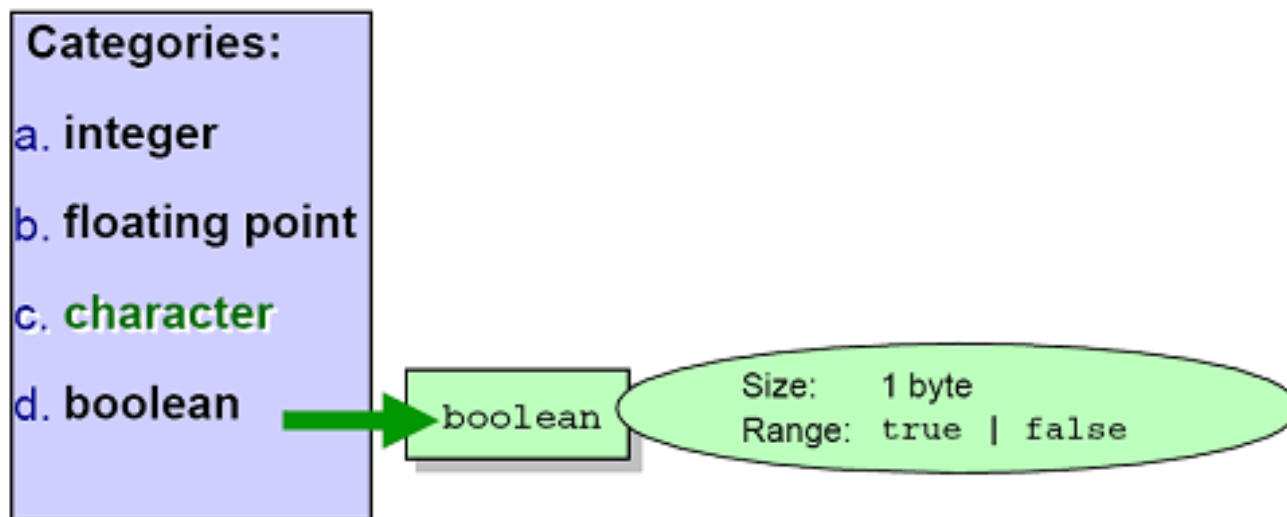
- ❖ Ký tự **Unicode** không dấu, được đặt giữa hai dấu nháy đơn.
- ❖ Giá trị mặc định là giá trị zero (`\u0000`)





# Giá trị logic

- ❖ Giá trị **boolean** được xác định rõ ràng trong **Java**.
  - Một giá trị **int** không thể sử dụng thay cho giá trị **boolean**.
  - Có thể lưu trữ giá trị hoặc **true** hoặc **false**.
- ❖ Biến **boolean** được khởi tạo là **false**





# Boolean, ký tự và xâu ký tự

## ❖ Boolean:

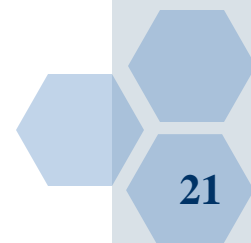
- True
- False

## ❖ Ký tự:

- Được đặt giữa 2 dấu nháy đơn.
- VD: 'a', 'A' hoặc '\uffff'

## ❖ Xâu ký tự:

- Được đặt giữa hai dấu nháy kép
- VD: "Hello word", "Xin chào", ...





# Các kiểu dữ liệu cơ bản

Từ khoá	Mô tả	Kích cỡ	Tối thiểu	Tối đa	Lớp bao
(kiểu số nguyên)					
byte	số nguyên một byte	8 bit	-128	127	Byte
short	số nguyên ngắn	16 bit	$-2^{15}$	$2^{15}-1$	Short
int	số nguyên	32 bit	$-2^{31}$	$2^{31}-1$	Integer
long	số nguyên dài	64 bit	$-2^{63}$	$-2^{63}-1$	Long
(kiểu số thực)					
float	kiểu thực với độ chính xác đơn	32 bit	IEEE754	IEEE754 4	Float
double	Double-precision floating point	64 bit	IEEE754	IEEE754 4	Double
(kiểu khác)					
char	kiểu kí tự	16 bit	Unicode 0	Unicode $2^{16}-1$	Character
boolean	kiểu logic	true hoặc false	-	-	Boolean
void	-	-	-	-	Void

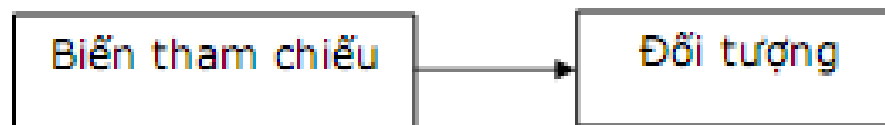


# Các kiểu dữ liệu cơ bản

## ❖ Kiểu tham chiếu: có 3 kiểu

Kiểu dữ liệu	Mô tả
Mảng (Array)	Tập hợp các dữ liệu cùng kiểu.
Lớp (Class)	Là sự cài đặt mô tả về một đối tượng trong bài toán.
Giao diện (Interface)	Là một lớp thuần trừu tượng được tạo ra cho phép cài đặt đa thừa kế trong Java.

## ❖ Biến kiểu tham chiếu: chứa địa chỉ của đối tượng mà nó trỏ đến.





# Chuyển đổi kiểu dữ liệu

- ❖ **Java** là ngôn ngữ định kiểu chặt chẽ.
  - Gán sai kiểu giá trị cho 1 biến có thể dẫn đến các lỗi biên dịch hoặc các ngoại lệ của **JVM**.
- ❖ **JVM** có thể ngầm định chuyển từ một kiểu dữ liệu hẹp sang một kiểu rộng hơn.
- ❖ Để chuyển sang một kiểu dữ liệu hẹp hơn, cần phải định kiểu rõ ràng.

```
int a, b;  
short c;  
a = b + c;
```

```
int d;  
short e;  
e = (short)d;
```

```
double f;  
long g;  
f = g;  
g = f; //error
```





# LẬP TRÌNH JAVA CƠ SỞ

- ❖ Bộ ký tự, từ khoá, tên
- ❖ Các kiểu và cấu trúc dữ liệu cơ bản
- ❖ **Hằng, biến, toán tử, biểu thức**
- ❖ Nhập xuất cơ sở
- ❖ Các cấu trúc điều khiển
- ❖ Mảng
- ❖ Hàm – Truyền tham số và các lời gọi hàm

- ❖ **Hằng** là một giá trị **bất biến** trong chương trình.
- ❖ Tên **hằng** được đặt theo qui ước: viết hoa tất cả các chữ.
- ❖ **Hằng số nguyên**: trường hợp giá trị hằng ở dạng “long” ta thêm vào cuối chuỗi số chữ “l” hoặc “L”. VD: 1L, 5L.
- ❖ **Hằng số thực**: trường hợp giá trị hằng có kiểu float ta thêm tiếp vị ngữ “f” hoặc “F”, còn kiểu double thì thêm tiếp vị ngữ “d” hoặc “D”.
- ❖ **Hằng boolean**: có 2 hằng là **true** và **false**.
- ❖ **Hằng ký tự**: là 1 ký tự đơn nằm giữa 2 dấu nháy đơn.
  - VD: ‘a’: hằng ký tự a

- ❖ **Hằng chuỗi:** là tập hợp các ký tự được đặt giữa hai dấu nháy kép “”.
- ❖ Một **hằng chuỗi** không có ký tự nào là một hằng chuỗi rỗng.
- ❖ **Cú pháp khai báo hằng:**  
 $\text{final} + \text{kiểu dữ liệu} + \text{tên hằng} = \text{giá trị cần gán}.$   
**VD:** `final int NAM_SINH = 1994`

## ❖ Một số hằng ký tự đặc biệt:

Ký tự	Ý nghĩa
\b	Xoá lùi (BackSpace)
\t	Tab
\n	Xuống hàng
\r	Dấu enter
\”	Nháy kép
\’	Nháy đơn
\\	Số ngược
\f	Đầy trang
\uxxxx	Ký tự unicode

- ❖ **Biến** là vùng nhớ dùng để lưu trữ các giá trị của chương trình.
- ❖ Mỗi **biến** gắn liền với 1 kiểu dữ liệu và một định danh duy nhất gọi là tên biến.
- ❖ Tên **biến** phải bắt đầu bằng một chữ cái hoặc dấu gạch dưới hoặc dấu \$.
- ❖ Tên biến không có khoảng trắng ở giữa tên.
- ❖ Tên biến không trùng với từ khoá trong java
- ❖ Biến có thể khai báo ở bất kỳ nơi đâu trong chương trình.

## ❖ Có 4 loại biến:

- Biến thành phần: là các thành phần của lớp và được khởi tạo giá trị mỗi khi một đối tượng của lớp được tạo ra
- Các biến tham chiếu đối tượng: được sử dụng để xử lý các đối tượng
  - Cú pháp: `Hocsinh hs = new Hocsinh`
- Các biến tĩnh (static): đại diện cho cả lớp
- Các biến cục bộ: được khai báo trong các phương thức và trong các khối

## ❖ Khai báo biến

- **Cú pháp:**

*[Kiểu dữ liệu] [tên biến]*

- VD: `int giaTri; giaTri=5;` hoặc `int giaTri=5.`

## ❖ Khởi tạo giá trị cho các biến

- Các biến tĩnh trong lớp luôn được khởi tạo với các giá trị mặc định nếu ko được gán gt tường minh
- Các biến thành phần cũng được khởi tạo mặc định mỗi khi đối tượng của lớp có thành phần đó được khởi tạo nếu ko được gán gt tường minh
- Biến tham chiếu được gán mặc định là null nếu ko tạo lập tường minh theo toán tử new hoặc constructor



- ❖ **Bài 1:** Trong các tên biến sau, tên biến nào khai báo sai: a, b, \_c, 3a, %s, \*d, \_e, class, \_else, super, \$super, Public, Return, If, \_case, New , \$New.
- ❖ **Bài 2:** Khai báo 2 biến nguyên kiểu int, gán giá trị bất kỳ cho 2 biến, tính tổng 2 số, gán tổng vào biến t, in giá trị biến t ra ngoài màn hình.
- ❖ **Bài 3:** Khai báo hằng  $PI=3.14$  kiểu số thực, với biến r là bán kính đường tròn, kiểu số thực. Hãy viết chương trình tính diện tích và chu vi hình tròn, in kết quả ra màn hình.

❖ ***Kết quả bài 1:*** Tên biến khai báo sai là: 3a, %s, \*d, class, super,

❖ ***Bài 2:***

```
public class Bai2
{
    public static void main (String [] args)
    {
        int a=50, b=75, t=a+b;
        System.out.println("Tong 2 so la:" +t);
    }
}
```

## ❖ Bài 3:

```
public class Bai3
{
    public static void main (String [] args)
    {
        final float PI=3.14f;
        float r=4.5f;
        System.out.println("Chu vi hình tron:"+(2*r*PI));
        System.out.println("Dien tích hình tron:"+(r*r*PI));
    }
}
```



# Ví dụ

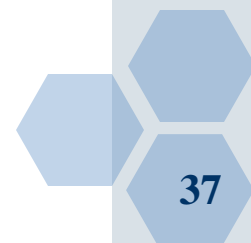
```
public class tong
{
    public static void main(String[] args)
    {
        final double PI=3.14;
        double r=5, chuvi,dt;
        chuvi=2*PI*r;
        dt=PI*r*r;
        System.out.println("Chu vi là: "+chuvi);
        System.out.println("Dien tich là: "+dt);

    }
}
```



# Toán tử

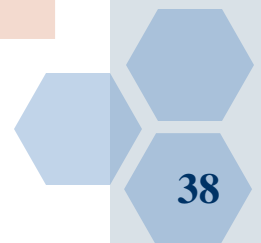
- ❖ Kết hợp các giá trị đơn hoặc các biểu thức con thành những biểu thức mới, phức tạp hơn và có thể trả về giá trị.
- ❖ **Java** cung cấp nhiều dạng toán tử sau:
  - Toán tử **số học**
  - Toán tử **bit**, toán tử **quan hệ**
  - Toán tử **logic**
  - Toán tử **điều kiện**
  - Toán tử **gán**
  - Toán tử **ép kiểu**





# Toán tử số học

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





# Toán tử

**Bài 1:** Cho 2 số nguyên a và b khởi tạo với giá trị bất kỳ

```
public class SoHoc
```

```
{
```

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

```
    {
```

```
        int a, b, du, nguyen;
```

```
        a = 10;
```

```
        b = 3;
```

```
        du = a%b;
```

```
        nguyen = a/b;
```

```
        System.out.println("Phần dư (a:b) là: " + du);
```

```
        System.out.println("Phần nguyên (a:b) là: " + nguyen);
```

```
        a++;
```

```
        System.out.println("Giá trị a đã tăng lên 1, giá trị mới là: " + a);
```

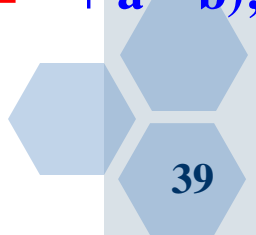
```
        b--;
```

```
        System.out.println("Giá trị của b đã giảm đi 1, giá trị mới là: " + b);
```

```
        System.out.println("Với 2 giá trị a, b mới trên, Tích (a x b) = " + a * b);
```

```
    }
```

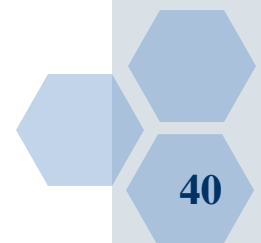
```
}
```





# Toán tử quan hệ, logic

Toán tử	Ý nghĩa
==	So sánh bằng
!=	So sánh khác
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hoặc bằng
<=	So sánh nhỏ hơn hoặc bằng
	OR (Biểu thức logic)
&&	AND (Biểu thức logic)
!	NOT (Biểu thức logic)







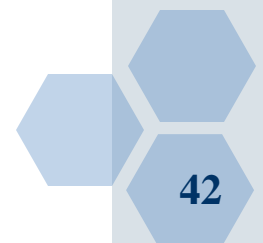
# Ví dụ

```
public class QuanHeLogic
{
    public static void main(String[] args)
    {
        Boolean soSanh;
        int a, b;
        a = 5;
        b = 10;
        soSanh = (a == b);
        System.out.println("Kết quả so sánh " + a + "=" + b + " không? " + soSanh);
        soSanh = (a < b);
        System.out.println("Kết quả so sánh " + a + "<" + b + " không? " + soSanh);
        soSanh = (a != b);
        System.out.println("Kết quả so sánh " + a + "#" + b + " không? " + soSanh);
        soSanh = (a >= b);
        System.out.println("Kết quả so sánh " + a + ">=" + b + " không? " + soSanh);
        soSanh = (a < b) || (a == b);
        System.out.println("Kết quả so sánh " + a + "<=" + b + " không? " + soSanh);
        soSanh = !true;
        System.out.println("Biến soSanh được gán bằng giá trị phủ định của true, giá trị đó là: "+soSanh);
    }
}
```



# Toán tử trên bit

Toán tử	Ý nghĩa
&	AND
	OR
^	XOR
<<	Dịch trái
>>	Dịch phải
>>>	Dịch phải và điền 0 vào bit trống
~	Phủ định bit. Trả về giá trị phủ định của một bit





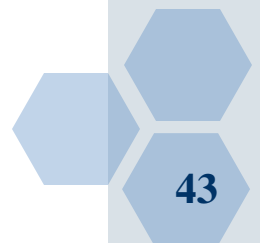
# Toán tử điều kiện

## ❖ Cú pháp:

**<Biểu thức 1>? <Biểu thức 2>: <Biểu thức 3>;**

**Trong đó:**

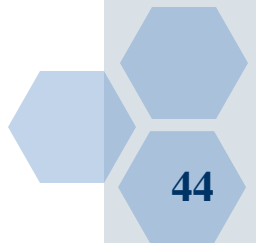
- **Biểu thức 1:** Biểu thức logic. Trả về giá trị **True** hoặc **False**.
- **Biểu thức 2:** Là giá trị trả về nếu <Biểu thức 1> xác định là **True**.
- **Biểu thức 3:** Là giá trị trả về nếu <Biểu thức 1> xác định là **False**.





# Toán tử ép kiểu

- ❖ **Ép kiểu rộng:** từ kiểu nhỏ sang kiểu lớn (không mất thông tin)
- ❖ **Ép kiểu hẹp:** Từ kiểu lớn sang kiểu nhỏ (có khả năng mất mát thông tin)
- ❖ **Cú pháp:**  
 $\langle \text{Tên biến} \rangle = (\text{Kiểu dữ liệu}) \langle \text{tên biến} \rangle;$   
`int a; float b;`  
`b=a;`  
`a=(int)b;`





# Toán tử ép kiểu

```
public class EpKieu
{
    public static void main(String[] args)
    {
        float soThuc;
        int soNguyen=5;
        soThuc = 10.6f;
        soThuc=soNguyen;
        soNguyen = (int)soThuc;
        System.out.println("Số thực vào là: " + soThuc);
        System.out.println("Số nguyên ép kiểu từ số thực là: " + soNguyen);
    }
}
```



## Toán tử gán (=)

❖ Dùng để gán một giá trị vào một biến và có thể gán nhiều giá trị cho nhiều biến cùng một lúc.

• VD:

```
int var=20;
```

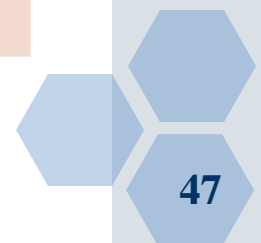
```
int p, q, r, s;
```

```
p=q=r=s=var;
```



# Thứ tự ưu tiên của các toán tử

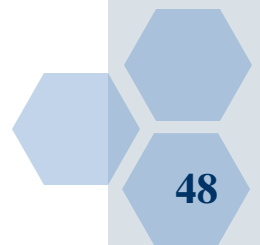
Thứ tự	Toán tử
1	Các toán tử đơn như ++,--
2	Các toán tử số học và các toán tử dịch bit như *, /, +, -, <<, >>
3	Các toán tử quan hệ như >, <, >=, <=, ==, !=
4	Các toán tử logic và các toán tử trên bit như &&,   , &,  , ^
5	Các toán tử gán như =, *=, ?=, +=, -=





# Một số hàm toán học

- ❖ Gợi ý cách xem nhiều hàm toán học có sẵn trong Java: gõ “Math.”(có dấu chấm phía cuối), rồi ấn Ctrl+”dấu cách”.
- ❖ **VD:** Math.min(a,b): tìm giá trị nhỏ nhất của 2 số a và b.  
Math.sqrt(b): Tính căn bậc 2 của số b.
- BT:** Tìm giá trị tuyệt đối, căn bậc 2 và max, min, mũ 3 của hai số thực a và b.

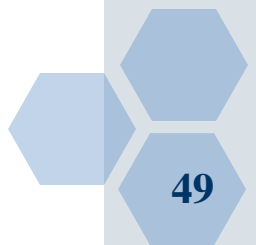






# LẬP TRÌNH JAVA CƠ SỞ

- ❖ Bộ ký tự, từ khoá, tên
- ❖ Các kiểu và cấu trúc dữ liệu cơ bản
- ❖ Hằng, biến, toán tử, biểu thức
- ❖ **Nhập xuất cơ sở**
- ❖ Các cấu trúc điều khiển
- ❖ Mảng
- ❖ Hàm – Truyền tham số và các lời gọi hàm





# Nhập liệu từ màn hình Console

## ❖ Thư viện:

`java.util.Scanner;`

## Sử dụng:

```
Scanner scan = new Scanner (System.in);
```

```
String str = scan.nextLine();
```



# Nhập liệu từ màn hình Console

Phương thức	Kiểu dữ liệu trả về
<code>nextInt()</code>	Trả về kiểu <code>Int</code>
<code>nextByte()</code>	Trả về kiểu <code>Byte</code>
<code>nextShort()</code>	Trả về kiểu <code>Short</code>
<code>nextLong()</code>	Trả về kiểu <code>Long</code>
<code>nextFloat()</code>	Trả về kiểu <code>float</code>
<code>double nextDouble()</code>	Trả về kiểu <code>double</code>
<code>nextBoolean()</code>	Trả về kiểu <code>boolean</code>
<code>nextLine()</code>	Trả về kiểu <code>String</code> (nhập chuỗi)
<code>next()</code>	



# Xuất ra màn hình console

## ❖ Cú pháp

```
System.out.print(...);
```

```
System.out.println(...);
```



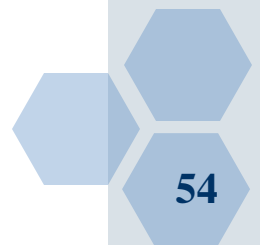
# Bài tập

- ❖ **Bài 1:** Các biến  $a, b, c$  là các số nguyên. Tìm phần nguyên khi chia các số này cho 2, tìm phần dư khi chia các số này cho 3, in kết quả ra màn hình. Tăng giá trị  $a, b, c$  mỗi biến lên 1, in giá trị 3 số ra màn hình.
- ❖ **Bài 2:** Cho  $a, b, c$  là độ dài 3 cạnh tam giác,  $a, b, c$  được nhập từ bàn phím. Viết chương trình tính diện tích  $s$  của tam giác, in kết quả ra màn hình.
- ❖  $P = (a+b+c)/2;$
- ❖  $S = \text{Sqrt}(p*(p-a)*(p-b)*(p-c))$



# Bài tập

- ❖ **Bài 4:** Viết chương trình nhập vào họ, tên sau đó in ra và màn hình họ và tên.





# LẬP TRÌNH JAVA CƠ SỞ

- ❖ Bộ ký tự, từ khoá, tên
- ❖ Các kiểu và cấu trúc dữ liệu cơ bản
- ❖ Hằng, biến, toán tử, biểu thức
- ❖ Nhập xuất cơ sở
- ❖ **Các cấu trúc điều khiển**
- ❖ Mảng
- ❖ Hàm – Truyền tham số và các lời gọi hàm



# Cấu trúc điều kiện If...else

## ❖ Dạng 1:

```
if (<điều kiện>)  
{  
    <khối lệnh>;  
}
```

**Chú ý:** <điều kiện> ở đây là dạng logic, nó chỉ có thể là **True** hoặc **False**.

VD: Nếu số a chia hết cho 2 thì in ra thông báo đây là số chẵn. `if(a%2==0) { system.out.println("a là số chẵn"); }`







# Cấu trúc điều kiện If...else

```
package cautruc;  
Public class CauTruc  
{  
    public static void main(String[] args)  
    {  
        int a=6;  
        if(a%2==0)  
        {  
            System.out.println ( a+“ là số chẵn”);  
        }  
    }  
}
```



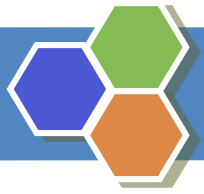
# Cấu trúc điều kiện If...else

## ❖ Dạng 2:

```
if (<điều kiện>
{
    <khối _lệnh1>;
} else
{
    <Khối _lệnh2>;
}
```

❖ Biểu thức điều kiện nhận giá trị **boolean**

❖ Mệnh đề **else** là tùy chọn



# Cấu trúc điều kiện If...else

## ❖ Dạng 3:

```
if (<điều kiện>
{
    <khối_lệnh1>;
} else if
{
    <Khối_lệnh2>;
} else
{
    <khối_lệnh3>
}
```



# Cấu trúc điều kiện If...else

```
int i=9;
if ( i > 0 ) {
    System.out.println("i lon hon 0");
}
```

```
int i= - 9;
if ( i>0 ) {
    System.out.println("i lon hon 0");
} else {
    System.out.println(" i be hon 0");
}
```

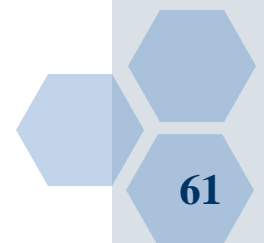
```
int i= -9;
if (i > 0 ) {
    System.out.println("i lon hon 0");
} else if ( i == 0 ){
    System.out.println("i bang 0");
} else {
    System.out.println("i be hon 0");
}
```



# Cấu trúc điều kiện If...else

## VD: Kiểm tra số chẵn-lẻ

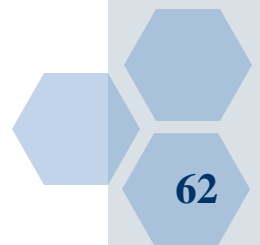
```
class CheckNumber
{
    public static void main(String args[])
    {
        int num =10;
        if (num %2 == 0)
            System.out.println (num+ "la so chan");
        else
            System.out.println (num + "la so le");
    }
}
```





# Bài tập

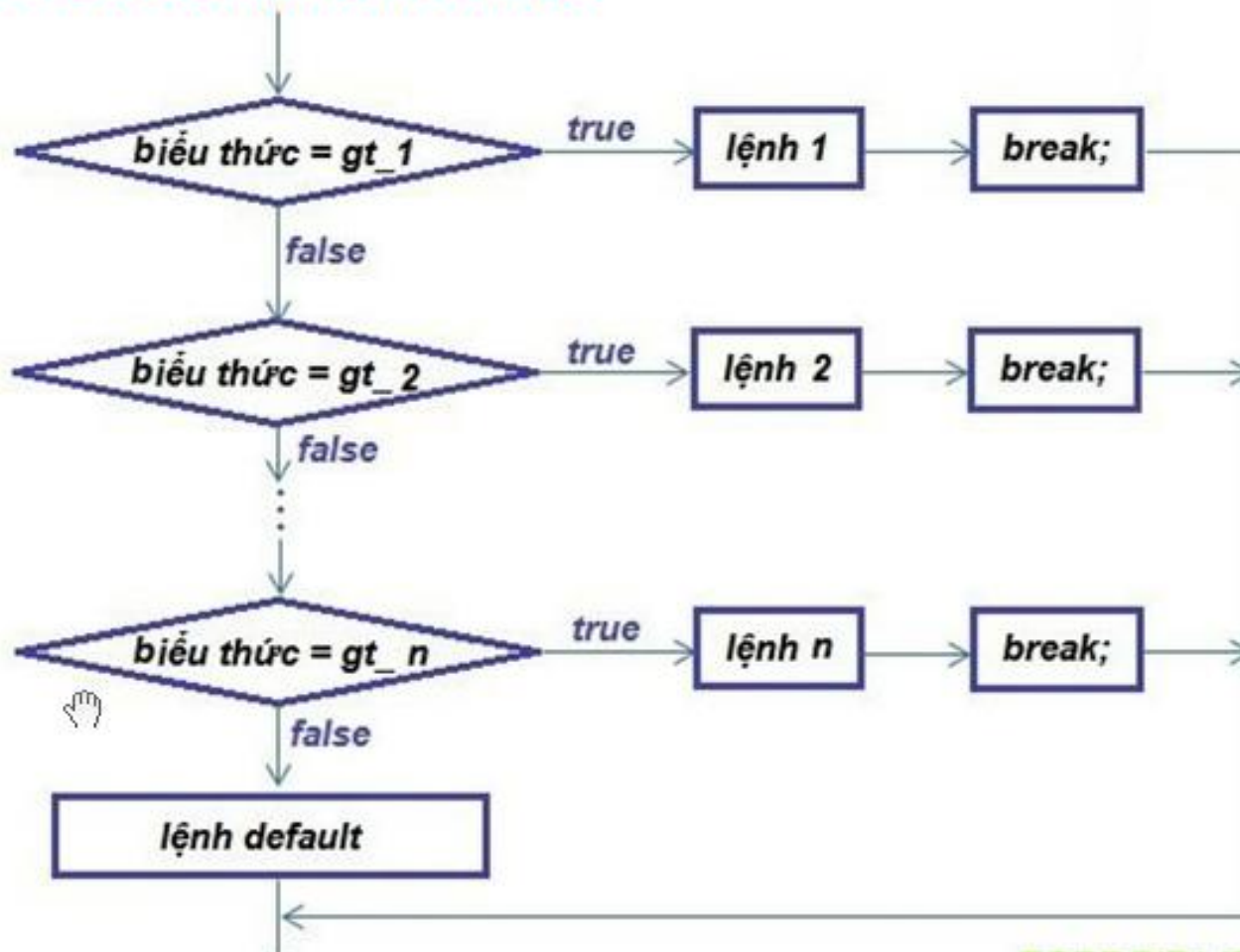
- ❖ **Bài 1:** Nhập từ bàn phím giá trị biến  $a$  trong chương trình,  $a$  là số nguyên. Xét xem  $a$  có chia hết cho 5 không? Nếu có thông báo ra màn hình là chia hết cho 5, nếu không hãy tìm thương và số dư và in ra màn hình.
- ❖ **Bài 2:** Giải phương trình bậc 1 :  $ax+b=0$  với  $a, b$  là các số thực.
- ❖ **Bài 3:** Giải phương trình bậc 2:  $ax^2 + bx + c = 0$  với  $a, b, c$  là các số thực.





# Cấu trúc Switch...Case

## Cấu trúc Switch...Case:





# Cấu trúc Switch...Case

- ❖ Kiểm tra một biến đơn với nhiều giá trị khác nhau và thực hiện trường hợp tương ứng.
- ❖ **break:** thoát khỏi lệnh **Switch...Case**.
- ❖ **default:** kiểm soát các giá trị nằm ngoài các giá trị case.





# Cấu trúc Switch...Case

## ❖ Cú pháp

```
switch (<biến>) {  
    case <giá trị_1> :  
        <khởi_lệnh_1>;  
        break;  
    case <giá trị_2>:  
        <khởi_lệnh_2>;  
        break;  
    ...  
    case <giá trị_n>:  
        <khởi_lệnh_n>;  
        break;  
    default:  
        <khởi_lệnh_default>;  
}
```

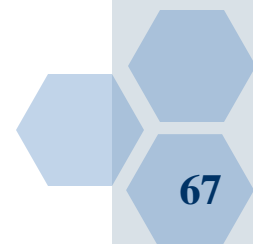


## Cấu trúc Switch...Case

- ❖ VD: Viết chương trình nhập biến nguyên  $a$  là 1 giá trị bất kỳ. Nếu  $a=1$  thì in ra màn hình là “Chủ nhật”;  $a=2$  in ra là “Thứ Hai”, ...,  $a=7$  in ra là “Thứ Bảy”. Nếu  $a$  không trong  $[1,..,7]$  thì in ra thông báo bạn đã gán sai giá trị.



- ❖ **Dùng switch ...case để tạo menu chương trình quản lý thông tin sinh viên:**
- ❖ **1. Nhập thông tin sinh viên**
- ❖ **2. Hiển thị thông tin sinh viên**
- ❖ **3. Thêm thông tin sinh viên**
- ❖ **4. Sửa thông tin sinh viên**
- ❖ **5. Xóa thông tin sinh viên**
- ❖ **6. Tìm kiếm sinh viên theo mã**
- ❖ **7. Thoát khỏi chương trình**





# Ví dụ

```
public class SwitchDemo {  
    public static void main(String[] args) {  
        int a = 3;  
  
        switch (a) {  
            case 1:  
                System.out.println("Chủ nhật");  
                break;  
            case 2:  
                System.out.println("Thứ Hai");  
                break;  
            case 3:  
                System.out.println("Thứ Ba");  
                break;  
            case 4:  
                System.out.println("Thứ Tư");  
                break;  
            case 5:  
                System.out.println("Thứ Năm");  
                break;  
            case 6:  
                System.out.println("Thứ Sáu");  
                break;  
            case 7:  
                System.out.println("Thứ Bảy");  
                break;  
            default:  
                System.out.println("Bạn đã gán sai giá trị, chỉ đ  
uộc gán số nguyên từ 1 tới 7");  
                break;  
        }  
    }  
}
```



# Vòng lặp while và do while

## ❖ Dạng 1:

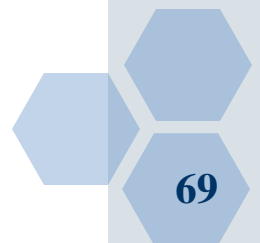
**while (điều kiện lặp)**

**{**

**<khối lệnh>;**

**}**

❖ Lưu ý: Dạng này xét điều kiện trước, đúng rồi mới thực hiện khối lệnh.





# Vòng lặp while và do while

## ❖ Dạng 2:

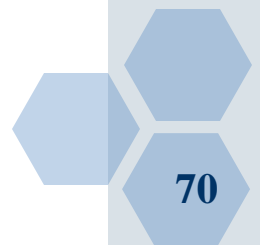
do

{

<khối lệnh>;

}while (điều kiện )

- ❖ Lưu ý: Dạng này thực hiện khối lệnh trước rồi xét điều kiện, nếu sai thì không thực hiện nữa.
- ❖ Như vậy, nếu điều kiện sai ngay từ đầu thì khối lệnh luôn được thực hiện ít nhất 1 lần.





# Vòng lặp while và do while

```
int x = 2;  
while (x < 2) {  
    x++;  
    System.out.println(x);  
}
```

```
int x = 2;  
do {  
    x++;  
    System.out.println(x);  
} while (x < 2);
```



# Vòng lặp do ... while

- ❖ Ví dụ 1: Viết chương trình nhập vào một số lớn hơn 10
- ❖ Ví dụ 2: Viết chương trình tính giai thừa của số 5







# Vòng lặp while và do while

❖ **VD:** Viết chương trình tính giai thừa của số 5.

```
class WhileDemo{
    public static void main(String args[]){
        int a = 5, fact = 1;
        while (a >= 1){
            fact *=a;
            a--;
        }
        System.out.println("The Factorial of 5
                           is "+fact);
    }
}
```



# Ví dụ

```
public static void main(String[] args)
{
    int k=0,n,dem,j=0;
    while(j<15){
        k++;
        dem=0;
        for(n=2;n<=k;n++)
        {
            if(k%n==0)
            {
                dem++;
            }
        }
        if(dem==1)
        {
            System.out.println(k+"" );
            j++;// Số lần in được cộng thêm 1
        }
    }
}
```



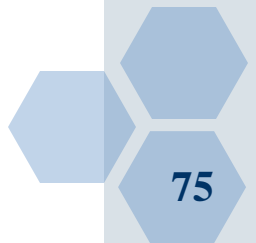
## Vòng lặp for(...)

### ❖ Cú pháp:

```
for(khởi_tạo_biến_đếm; đk_lặp; tăng_biến)
{
    <khởi_lệnh>;
}
```

VD:

```
for (int index = 0; index < 10; index++) {
    System.out.println(index);
}
```





# Vòng lặp for (...)

- ❖ **VD1**: Tính tổng của n số tự nhiên đầu tiên
- ❖ **VD2**: Viết chương trình in ra tổng của 5 số chẵn đầu tiên.





# Vòng lặp for

❖ **VD:** Viết chương trình in ra tổng của 5 số chẵn đầu tiên.

```
class ForDemo
{
    public static void main(String args[])
    {
        int i=1, sum=0;
        for (i=1;i<=10;i+=2)
            sum+=i;
        System.out.println ("Sum of first five
                             old numbers is " + sum);
    }
}
```





# Một số lệnh khác

## ❖ Lệnh **break**

- Lệnh **break** thường được sử dụng kết hợp lệnh lặp
- Lệnh **break** dùng để thoát khỏi vòng lặp
- Nếu có nhiều lệnh lặp lồng nhau thì lệnh **break** chỉ thoát vòng lặp trực tiếp chứa nó
- Lệnh **break** cũng dùng để thoát khỏi lệnh **switch ... case**
- Trong trường hợp khối được gán nhãn, **break** sẽ cho qua phần còn lại của khối và tiếp tục thực hiện lệnh đứng sau khối đó





# Một số lệnh khác

## ❖ Lệnh break

- Ví dụ: Kết quả chương trình sau?

```
package caulenhdk;
import java.util.Scanner;{
    int i;
    for (i=1; i <=5; i++)
    {
        System.out.print("Bat dau vong" +i);
        System.out.println("Chao ban");
        if (i==3) break;
        System.out.println("Ket thuc vong" +i);
    }
    System.out.print("Het vong lap");
}
```



# Một số lệnh khác

## ❖ Lệnh **continue**

- Lệnh **continue** dùng trong các chu trình lặp **for**, **while**, **do-while** để dừng sự thực hiện của lần lặp hiện thời và bắt đầu lặp lại lần tiếp theo nếu điều kiện lặp còn thỏa mãn





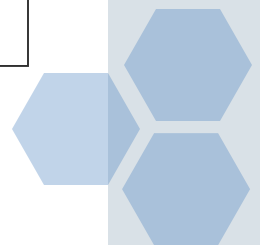


# Một số lệnh khác

## ❖ Lệnh lặp continue

- Ví dụ: Chương trình in ra màn hình các số 1,2,3 và 5 cùng với căn bậc 2 của chúng, với  $i = 4$  thì ko thực hiện
- Ví dụ

```
class TiepTuc
{
    public static void main(String[] args) {
        int i;
        for (i=1; i <= 5; i++)
        {
            if (i==4) continue;
            System.out.println(i+"\\t"+Math.sqrt(i));
        }
    }
}
```





# Một số lệnh khác

## ❖ Lệnh continue

- Ví dụ: Kết quả của đoạn chương trình sau?

```
for (i = 1; i <= 4; i++)  
    for (j = 1; j <= 10; j++)  
    {  
        cout<<j;  
        if (j != 10) continue;  
        cout<<endl;  
    }
```





# LẬP TRÌNH JAVA CƠ SỞ

- ❖ Bộ ký tự, từ khoá, tên
- ❖ Các kiểu và cấu trúc dữ liệu cơ bản
- ❖ Hằng, biến, toán tử, biểu thức
- ❖ Nhập xuất cơ sở
- ❖ Các cấu trúc điều khiển
- ❖ **Mảng**
- ❖ Hàm – Truyền tham số và các lời gọi hàm



# Mảng

- ❖ Mảng là tập hợp nhiều phần tử có cùng tên, cùng kiểu dữ liệu và mỗi phần tử trong mảng được truy xuất thông qua chỉ số của nó trong mảng.
- ❖ Cách khai báo:
  - `<Kiểu dữ liệu> <tên mảng>[];`
  - Hoặc `<Kiểu dữ liệu>[]<tên mảng>;`
  - VD:
    - `int arrInt1[];`
    - `int [] arrInt2;`
    - `int [] arrInt3, arrInt4, arrInt5;`

❖ Lưu ý:

- Trong Java việc khai báo mảng không thể sử dụng được ngay như trong C/C++ mà cần phải cấp phát vùng nhớ để tạo mảng.
- Kích thước của mảng chưa xác định khi khai báo.
- Việc khai báo biến mảng thì chưa hoàn toàn tạo ra cấu trúc đó mà mới khai báo cấu trúc để tham chiếu
- .

## ❖ Tạo lập đối tượng mảng

- Để cấp phát bộ nhớ cho mảng ta dùng từ khoá “new” và phải xác định số phần tử của mảng đó.
- Cú pháp  
$$\langle \text{Tên mảng} \rangle = \text{new} \langle \text{Kiểu các phần tử} \rangle [\langle \text{Số phần tử} \rangle];$$
- Ví dụ:
  - `int mangInt [];`
  - `mangInt = new int[100];`
  - `hocSinh = new hocSinh[50];`
- Có thể kết hợp cả khai báo với tạo lập mảng như sau:
  - $\langle \text{Kiểu các phần tử 1} \rangle \langle \text{Tên mảng} \rangle [] = \text{new} \langle \text{Kiểu các phần tử 2} \rangle [\langle \text{Số phần tử} \rangle];$

## ❖ Lưu ý:

- <Kiểu các phần tử 1> và <Kiểu phần tử 2> là hai kiểu tương thích với nhau.
- Đối với kiểu lớp thì <Kiểu các phần tử 2> phải là lớp con của <Kiểu các phần tử 1>
- Khi một mảng được tạo lập thì tất cả các phần tử của nó được khởi tạo giá trị mặc định
- Ví dụ:
  - `float a[] = new float[20];`

## ❖ Khởi tạo các mảng

### ■ Cú pháp

- $\langle \text{Kiểu các phần tử} \rangle [\langle \text{Tên mảng} \rangle] = \{ \langle \text{Kiểu các phần tử 2} \rangle \};$

### ■ Ví dụ

- `int[] a = { 1, 3, 5, 7, 9};`



## ❖ Khởi tạo các mảng

### ■ Cú pháp

- `<Kiểu các phần tử>[]<Tên mảng> = {<Kiểu các phần tử 2>;`

### ■ Ví dụ

- `int[] a = {1, 3, 5, 7, 9}`
- `char arrChar[] = {'a', 'b', 'c'};`
- `String arrString[] = {"Nguyen Van A", "Vu Van B", "Vu Van C"};`

## ❖ Lưu ý: Để lấy kích thước mảng bằng cách truy cập thuộc tính : `Arrayname.length()`



# Khởi tạo mảng

❖ Có thể khởi tạo giá trị ban đầu cho các phần tử của mảng khi nó được khai báo:

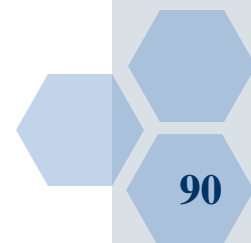
❖ VD:

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

```
char arrChar[] = {'a', 'b', 'c'};
```

```
String arrString[] = {"Nguyen Van A", "Vu Van B",  
"Vu Van C"};
```

❖ Lưu ý: Để lấy kích thước mảng bằng cách truy cập thuộc tính : `Arrayname.length()`





# Khởi tạo mảng

```
1 //Khởi tạo mảng một chiều kiểu long
2 long[] arr1 = {1, 3, 5, 7, 9}; //a.length = 5
3 //Khởi tạo mảng một chiều kiểu float
4 float[] arr2 = {1.3, 3.2, 5.5}; //a.length = 3
5 //Khởi tạo mảng một chiều kiểu double
6 double[] arr2 = {2.3, 7.2, 9.5}; //a.length= 3
7 //Khởi tạo mảng một chiều kiểu string
8 String[] ngay = {
9     "chủ nhật", "thứ hai", "thứ ba",
10    "thứ tư", "thứ năm", "thứ sáu", "thứ bảy"
11 }; //a.length= 7
```



# Truy cập mảng

- ❖ Chỉ số mảng trong Java bắt đầu từ 0 nên phần tử đầu tiên có chỉ số là 0 và phần tử  $n$  có chỉ số  $n-1$ .
- ❖ Các phần tử của mảng được truy xuất thông qua chỉ số của nó đặt giữa cặp dấu ngoặc vuông [].
- VD:

```
int arrInt[] = {1, 2, 3};  
int x = arrInt[0]; // x sẽ có giá trị là 1.  
int y = arrInt[1]; // y sẽ có giá trị là 2.  
int z = arrInt[2]; // z sẽ có giá trị là 3.
```

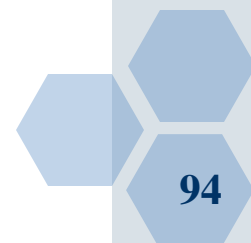


# Nhập mảng và xuất mảng một chiều

```
1 //Nhập mảng
2 System.out.print("Số phần tử của mảng là ");
3 int n = Integer.parseInt(scan.nextLine());
4 int [] a = new int [n]; //a.Length = n
5 for (int i = 0; i < a.length; i++) {
6     System.out.print("a["+i+"]=");
7     a[i] = Integer.parseInt(scan.nextLine());
8 }
9 //Xuất mảng dùng for
10 System.out.println("Xuất mảng dùng for");
11 System.out.println("Số phần tử của mảng " + a.length);
12 for (int i = 0; i < a.length; i++)
13 {
14     System.out.println(a[i]);
15 }
```



- ❖ **VD: Nhập số lượng phần tử trong mảng.**
- ❖ **Tạo mảng và nhập giá trị các phần tử trong mảng đó.**
- ❖ **In các giá trị phần tử mảng ra ngoài màn hình**
- ❖ **Tính tổng các giá trị phần tử mảng và in ra ngoài màn hình.**





## Ví dụ

```
import java.util.Scanner;
public class nhapxuatmang
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Số phần tử của mảng là:");
        int n=sc.nextInt();
        int [] a=new int[n];
        for(int i=0; i<a.length; i++)
        {
            System.out.println("a["+i+"]=");
            a[i]=sc.nextInt();
        }
    }
}
```



## Ví dụ

```
// Xuất mảng
System.out.println("Xuất mảng có số phần tử
là:"+a.length);
for(int i=0; i<a.length; i++)
{
    System.out.println(a[i]);
}
// Tính tổng các phần tử trong mảng
int s=0;
for(int i=0; i<a.length;i++)
{
    s=s+a[i];
}
System.out.println("Tổng s:"+s);
```





# Tính tổng các phần tử trong mảng 1 chiều

```
// Tính tổng dùng for
int s=0;
for(int i=0; i<a.length; i++)
{
    s=s+a[i];
}
System.out.println("s="+s);
```



# Sắp xếp mảng tăng dần

```
1  int i, j;
2  int min, temp;
3  for (i = 0; i < a.length - 1; i++)
4  {
5      min = i;
6      for (j = i + 1; j < a.length; j++)
7      {
8          if (a[j] < a[min])
9          {
10             min = j;
11          }
12      }
13      temp = a[i];
14      a[i] = a[min];
15      a[min] = temp;
16 }
```



# Mảng

```
public class ViDuArray {  
    public static void main(String[] args) {  
        int arrInt[];  
        arrInt = new int[4];  
        arrInt[0] = 9;  
        arrInt[1] = 17;  
        arrInt[2] = 13;  
        arrInt[3] = 14;  
  
        String arrString[] = {"Vu Van A", "Nguyen Van B", "Nguyen Van C"};  
  
        System.out.println("Mảng số nguyên: ");  
        for (int i = 0; i < 4; i++) {  
            System.out.print(arrInt[i] + " ");  
        }  
        System.out.println("\nMảng các chuỗi: ");  
  
        for (int i = 0; i < 3; i++) {  
            System.out.println(arrString[i] + " ");  
        }  
        System.out.println("");  
    }  
}
```

- ❖ **VD:** Cho 1 dãy số tự nhiên, viết chương trình sắp xếp dãy này theo thứ tự giảm dần.

```
public class mang01 {  
    public static void main(String[] args) {  
        int [] a = {3,1,7,0,10};  
        int N=5,k,j,temp;  
        for (k=0;k<N-1;k++)  
        {  
            for (j=k+1;j<N;j++)  
            {  
                if (a[k]<a[j])  
                {  
                    temp=a[j];  
                    a[j]=a[k];  
                    a[k]=temp;  
                }  
            }  
        }  
        for (k=0;k<N;k++)  
            System.out.print(a[k]+" ");  
    }  
}
```



# Bài tập

## ❖ **Viết chương trình cho phép:**

- Nhập vào mảng 1 chiều.
- Tính tổng các phần tử trong mảng
- Tìm phần tử lớn nhất trong mảng
- Sắp xếp mảng tăng dần
- Xuất mảng ra ngoài màn hình.

**Bài 1:** Cho 1 dãy số tự nhiên, in ra màn hình tất cả các số nguyên tố của dãy này.

**Bài 2:** Cho 1 dãy số nguyên bất kỳ và 1 số  $c$  cho trước. Hãy đếm có bao nhiêu số trong dãy  $>c$ ,  $=c$  và  $<c$ .

**Bài 3:** Cho trước 1 xâu ký tự bất kỳ, hãy đếm xem trong xâu có bao nhiêu lần xuất hiện “ab”



# Các thao tác liên quan đến mảng

## 1. Nhập mảng:

- Khai báo, tạo và khởi tạo giá trị cho mảng trong 1 lệnh

### Ví dụ:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```



Lưu ý: chỉ trong 1 lệnh, nhiều hơn 1 lệnh là **SAI**:

```
double[] myList;  
myList = {1.9, 2.9, 3.4, 3.5}; // SAI
```

- Sử dụng vòng lặp:  
for (int i = 0; i < myList.length; i++)  
    myList[i] = i;



# Các thao tác liên quan đến mảng

## 2. Xuất mảng:

```
public static void xuatMang(int[] arr,int n)
{
    for(int i=0;i<n;i++){
        System.out.print("\t"+arr[i]);
    }
}
```





# Các thao tác liên quan đến mảng

## 3. Tìm kiếm:

```
public static int timX(int[] arr,int n,int x)
{
    for(int i=0;i<n;i++)
        if(arr[i]==x)
            return i;
    return -1;
}
```



# Các thao tác liên quan đến mảng

## 4. Sắp xếp:

```
public static void sapxepMang(int[] arr,int n){  
    int tam;  
    for(int i=0;i<n-1;i++){  
        for(int j=i+1;j<n;j++){  
            if(arr[i]>arr[j]){  
                tam=arr[i];  
                arr[i]=arr[j];  
                arr[j]=tam;  
            }  
        }  
    }  
}
```



# Các thao tác liên quan đến mảng

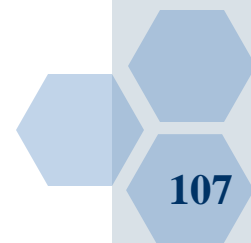
## 5. Xóa

```
public static int xoaX(int[] arr,int n,int x){  
    int vitri=timX(arr,n,x);  
    if (vitri!=-1){  
        System.arraycopy(arr, vitri+1, arr, vitri, arr.length-vitri-1);  
        n=n-1;  
    }  
    else  
        System.out.print("Khong tim thay "+x+" trong mang");  
    return n;  
}
```



Lưu ý: lệnh dùng để sao chép nội dung 2 mảng

```
System.arraycopy(src , int srcPos , dest , int destPos , length);
```





# Các thao tác liên quan đến mảng

## 6. Sửa

```
public static void suaX(int[] arr,int n,int x){  
    int vitri=timX(arr,n,x);  
    if (vitri!=-1){  
        System.out.print("Nhap gia tri moi:");  
        arr[vitri]=readInt();  
    }  
    else  
        System.out.print("Khong tim thay "+x+" trong mang");  
}
```



# Mảng 2 chiều

Chỉ số cột

↓

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

Chỉ số dòng →



# Mảng 2 chiều

## ❖ Khai báo:

```
KieuDuLieu [][] tenBien;  
Hoặc KieuDuLieu tenBien [][];
```

```
1 //Khai báo mảng hai chiều kiểu int  
2 int[][] arr1;  
3 //Khai báo mảng hai chiều kiểu long  
4 long[][] arr2;  
5 //Khai báo mảng hai chiều kiểu float  
6 float[][] arr3;  
7 //Khai báo mảng hai chiều kiểu double  
8 double[][] arr4;  
9 //Khai báo mảng hai chiều kiểu boolean  
10 boolean[][] arr5;  
11 //Khai báo mảng hai chiều kiểu String  
12 String[][] arr6;
```



# Mảng 2 chiều

## ❖ Cấp phát vùng nhớ

Cách 1	<code>KieuDuLieu[][] tenMang = new KieuDuLieu [n][m] ;</code>
Cách 2	<code>KieuDuLieu[][] tenMang; tenMang = new KieuDuLieu [n][m] ;</code>
n	Số dòng : <code>tenMang.length</code>
m	Số cột : <code>tenMang[i].length</code>

```
1 //Khai báo và cấp phát mảng hai chiều kiểu int
2 int[][] arr1 = new int[3][5];
3 int soDong = a.length; //soDong = 3
4 int soCot = a[i].length; //soCot = 5
5 //Khai báo và cấp phát mảng kiểu long
6 long[][] arr2 = new long[5][6];
7 int soDong = a.length; //soDong = 5
8 int soCot = a[i].length;; //soCot = 6
9 //Khai báo và cấp phát mảng hai chiều kiểu float
10 float[][] arr3 = new float[7][9];
11 int soCot = a.length; //soDong = 7
12 int soCot = a[i].length; //soCot = 9
```



# Mảng 2 chiều

## ❖ Khởi tạo:

```
1 //Cách 1
2 int[][] a =
3     {
4         { 1, 2 },
5         { 3, 4 },
6         { 5, 6 },
7         { 7, 8 }
8     };
9 int soDong = a.length; //soDong = 4
10 int soCot = a[0].length; //soCot = 2
11 //Cách 2
12 int [][] a = new int[4][2];
13 int soDong = a.length; //soDong = 4
14 int soCot = a[0].length; //soCot = 2
15 int k = 1;
16 for (int i = 0; i < soDong; i++)
17 {
18     for (int j = 0; j < soCot; j++)
19     {
20         a[i][j]=k++;
21     }
22 }
```





# Mảng 2 chiều

## ❖ Nhập mảng 2 chiều:

```
1 //Nhập mảng
2 int[][] a;
3 System.out.println("Nhập mảng");
4 System.out.print("Mời nhập vào số dòng:");
5 int n = Integer.parseInt(scan.nextLine());
6 System.out.print("Mời nhập vào số cột:");
7 int m = Integer.parseInt(scan.nextLine());
8 a=new int[n][m];
9 for (int i = 0; i < a.length; i++){
10     for (int j = 0; j < a[i].length; j++){
11         System.out.print("a["+i+"]["+j+"]=");
12         a[i][j] = Integer.parseInt(scan.nextLine());
13     }
14 }
```



# Mảng 2 chiều

## ❖ Xuất mảng 2 chiều

```
1 //Xuất mảng
2 System.out.println("Xuất mảng");
3 System.out.println("Số dòng : " + a.length);
4 System.out.println("Số cột : {0}", a[0].length);
5 for (int i = 0; i < a.length; i++){
6     for (int j = 0; j < a[i].length; j++) {
7         System.out.print(a[i][j]+"\\t");
8     }
9     System.out.println();
10 }
```



# Mảng 2 chiều

## ❖ Tính tổng các phần tử trong mảng 2 chiều:

```
1  int s = 0;
2  for (int i = 0; i < a.length; i++){
3      for (int j = 0; j < a[i].length; j++){
4          s = s + a[i][j];
5      }
6  }
7  System.out.println("s=" + s);
```



# Hàm

- ❖ Hàm là một nhóm các lệnh thực hiện cùng một nhiệm vụ hay đóng gói một chức năng nào đó để ta có thể tái sử dụng nhiều lần trong chương trình.
- ❖ Vai trò của hàm

```
1 package ham;  
2 public class Ham {  
3     public static void main(String[] args) {  
4         int a,b;  
5         a = 2;  
6         b = 3;  
7         int tong = a + b;  
8     }  
9  
10 }
```

- ❖ Hàm đóng gói các chức năng dưới dạng tổng quát, có thể chạy khắp các chương





# Hàm

## ❖ Cú pháp

- `public static <kiểu trả về><tên hàm>(danh sách các tham số)`

## ❖ Ví dụ

```
1 package hamcokieutravecothamsotruyenvao;
2
3 public class HamCoKieuTraVeCoThamSoTruyenVao {
4
5     public static void main(String[] args) {
6         System.out.println(TinhTong(2, 3));
7     }
8     public static int TinhTong(int a, int b){
9         int tong = a + b;
10        return tong;
11    }
12 }
```





# Hàm

## ❖ Các loại hàm

- Hàm không có kiểu trả về và không truyền tham số vào
- Hàm có kiểu trả về và không có tham số truyền vào
- Hàm không có kiểu trả về nhưng có tham số truyền vào
- Hàm có kiểu trả về và có tham số truyền vào





# Hàm

## ❖ Hàm không có kiểu trả về và không truyền tham số vào

- Thường được sử dụng trong quá trình chúng ta muốn in ra một giá trị nào đó.
- Hàm này không nên dùng trong trường hợp muốn tính toán

```
1 public static void showName(){  
2     String name;  
3     System.out.println("Moi ban nhap ten");  
4     Scanner sc = new Scanner(System.in);  
5     name = sc.nextLine();  
6     System.out.println("Ten cua ban la : " + name);  
7 }
```





# Hàm

## ❖ Hàm có kiểu trả về và không có tham số truyền vào

- Sau khi chạy sẽ trả về một giá trị nào đó

```
1 package hamcokieutravevakhongcothamsotruyenvao;  
2  
3 import java.util.Scanner;  
4  
5 public class HamCoKieuTraVeVaKhongCoThamSoTruyenVao {  
6  
7     public static void main(String[] args) {  
8         System.out.println(TinhTong());  
9     }  
10    public static int TinhTong(){  
11        int a, b;  
12        Scanner sc = new Scanner(System.in);  
13        System.out.println("Moi ban nhap gia tri cho a : ");  
14        a = sc.nextInt();  
15        System.out.println("Moi ban nhap gia tri cho b : ");  
16        b = sc.nextInt();  
17        int tong = a + b;  
18        return tong;  
19    }  
20 }
```







# Hàm

## ❖ Hàm không có kiểu trả về nhưng có tham số truyền vào

```
1 package hamkhongcokieutravecothamsotruyenvao;  
2  
3 public class HamKhongCoKieuTraVeCoThamSoTruyenVao {  
4  
5     public static void main(String[] args) {  
6         TinhTong(2,3);  
7     }  
8     public static void TinhTong(int a, int b){  
9         int tong = a + b;  
10        System.out.println("Tong cua a va b la : " + tong);  
11    }  
12  
13 }
```





# Hàm

## ❖ Hàm có kiểu trả về và có tham số truyền vào

```
1 package hamcokieutravecothamsotruyenvao;
2
3 public class HamCoKieuTraVeCoThamSoTruyenVao {
4
5     public static void main(String[] args) {
6         System.out.println(TinhTong(2, 3));
7     }
8     public static int TinhTong(int a, int b){
9         int tong = a + b;
10        return tong;
11    }
12 }
```





# Truyền tham số cho hàm

## ❖ **Viết chương trình cho phép:**

- Nhập vào mảng 1 chiều.
- Tính tổng các phần tử trong mảng
- Tìm phần tử lớn nhất trong mảng
- Sắp xếp mảng tăng dần
- Xuất mảng ra ngoài màn hình.