



**Java™**

# **LẬP TRÌNH JAVA 1**

## **BÀI 6: HÀM**

### **PHẦN 1**

- ⊙ Giải thích được cấu trúc hàm
- ⊙ Sử dụng một số hàm dựng sẵn trong Java
- ⊙ Viết và sử dụng hàm
- ⊙ Nạp chồng hàm
- ⊙ Phân biệt tham biến, tham trị
- ⊙ Sử dụng tham số biến đổi
- ⊙ Tạo các thư viện tiện ích



- ❑ Hàm (hay còn gọi là phương thức tĩnh, phương thức mức lớp) là một module mã thực hiện một công việc cụ thể nào đó.
- ❑ Ví dụ sau đây là các hàm đã được sử dụng
  - ❖ **System.arraycopy**(src, srcPos, dest, destPos, length);
    - Sao chép mảng
  - ❖ **Arrays.sort**(array)
    - Sắp xếp mảng
  - ❖ **Arrays.binarySearch**(array, element)
    - Tìm kiếm phần tử trong mảng
  - ❖ **Math.sqrt**(value)
    - Tính căn bậc hai

- ❑ Để tìm số lớn nhất trong 2 số, chúng ta có thể viết đoạn mã bên
- ❑ Nếu chúng ta tìm số lớn nhất giữa hai số khác là c và d thì chúng ta viết đoạn mã gần giống
- ❑ Như vậy mã viết y hệt nhau, chỉ khác là 2 số nào mà thôi
- ❑ Để thuận tiện cho việc tìm số lớn nhất của 2 số, java cung cấp hàm `Math.max(x, y)`

```
if(a > b){  
    ketqua = a;  
}  
else{  
    ketqua = b;  
}
```

```
if(c > d){  
    ketqua = c;  
}  
else{  
    ketqua = d;  
}
```

## ❑ Tìm số lớn nhất của 2 số nguyên

```
int a = 5, b = 7, c = 9, d = 6, ketqua1, ketqua2;  
if(a > b){  
    ketqua1 = a;  
}  
else{  
    ketqua1 = b;  
}  
if(c > d){  
    ketqua2 = c;  
}  
else{  
    ketqua2 = d;  
}
```



```
int a = 5, b = 7, c = 9, d = 6, ketqua1, ketqua2;  
ketqua1 = Math.max(a, b);  
ketqua2 = Math.max(c, d);
```

- ❑ Chúng ta sử dụng đoạn mã sau để sắp xếp mảng `a[]`

```
for(int i=0; i<a.length-1; i++){  
    for(int j=i+1; j<a.length; j++){  
        if(a[i] > a[j]){  
            int temp = a[i];  
            a[i] = a[j];  
            a[j] = temp;  
        }  
    }  
}
```

- ❑ Nếu muốn sắp xếp mảng `b[]` chúng ta lại viết lại đoạn mã trên với `b[]`.
- ❑ Hàm `Arrays.sort(array)` là module mã sắp xếp mảng. Chúng ta có thể dùng khi cần sắp xếp mảng.

- ❑ Hàm có thể hiểu là một chương trình con thực hiện một công việc cụ thể nào đó.
  - ❖ Hàm có thể có hoặc không có tham số. Tham số là dữ liệu đầu vào cần thiết để hàm thực hiện.
  - ❖ Hàm có thể có hoặc không có kết quả trả về

## ❑ Ví dụ

- ❖ `int c = Math.max(a, b)`
  - Có 2 tham số và có kết quả trả về
- ❖ `Arrays.sort(array)`
  - Có một tham số và không có kết quả trả về
- ❖ `double r = Math.random()`
  - Không có tham số và có kết quả trả về

## ☐ Tái sử dụng

- ❖ Khi cần dùng, chỉ việc gọi hàm

## ☐ Mã rõ ràng, ngắn gọn

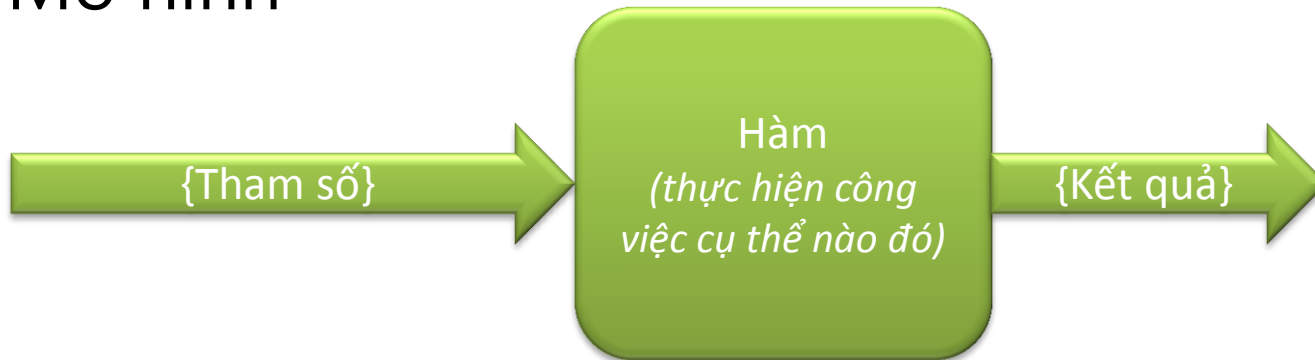
- ❖ Viết mã tách rời sau đó gọi hàm

## ☐ Dễ quản lý, bảo trì

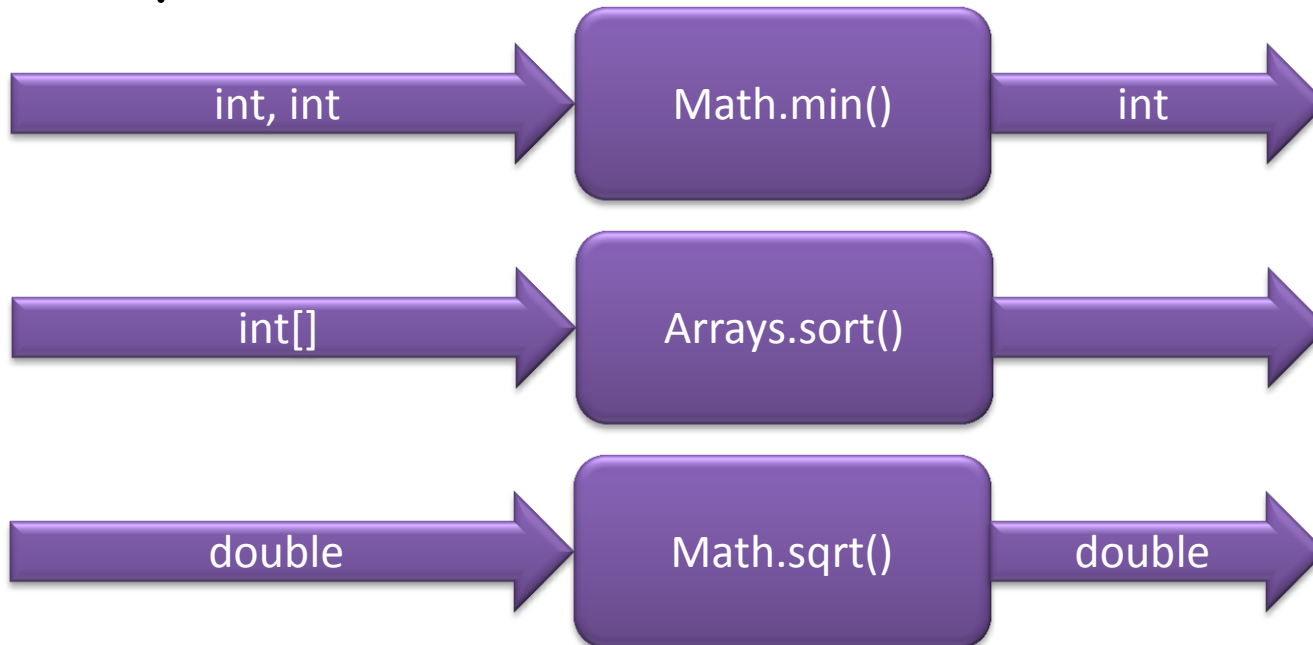
- ❖ Nâng cấp mã 1 chỗ



## □ Mô hình



## □ Ví dụ



`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: <code>List&lt;Integer&gt; b = Arrays.asList(a);</code>
<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: <code>int i = Arrays.binarySearch(a, 8);</code>
<code>void <b>sort</b>(Object[] a)</code>	Sắp xếp các phần tử theo thứ tự tăng dần. Ví dụ: <code>Arrays.sort(a);</code>
<code>String <b>toString</b>(Object[] a)</code>	Chuyển mảng thành chuỗi được bọc giữ cặp dấu [] và các phần tử mảng cách nhau dấu phẩy. Ví dụ: <code>String s = Arrays.toString(a);</code>
<code>void <b>fill</b>(Object[] a, Object val)</code>	Gán 1 giá trị cho tất cả các phần tử mảng. Ví dụ: <code>Arrays.fill(a, 9);</code>

- ❑ Java cung cấp các hàm tiện ích giúp chúng ta thực hiện các phép tính khó một cách dễ dàng như:
  - ❖ Làm tròn số
  - ❖ Tính căn bậc 2
  - ❖ Tính lũy thừa
  - ❖ ...
- ❑ Ví dụ sau đây tính căn bậc 2 của 7
  - ❖ `double a = Math.sqrt(7)`
- ❑ Ngoài `Math.sqrt()` còn rất nhiều hàm khác được trình bày ở slide sau.

Hàm	Diễn giải	Ví dụ
Math.min(a, b)	Lấy số nhỏ nhất của 2 số a và b	<code>x=Math.min(5, 3.5) =&gt; x=3.5</code>
Math.max(a, b)	Lấy số lớn nhất của 2 số a và b	<code>x=Math.max(5, 3.5) =&gt; x=5</code>
Math.pow(a, n)	Tính $a^n$ (a lũy thừa n)	<code>x=Math.pow(5, 3) =&gt; x=75</code>
Math.sqrt(a)	Tính $\sqrt{a}$ (căn bậc 2 của a)	<code>x=Math.sqrt(16) =&gt; x=4</code>
Math.abs(a)	Lấy giá trị tuyệt đối của a	<code>x=Math.abs(-5) =&gt; x=5</code>
Math.ceil(a)	Lấy số nguyên trên của a	<code>x=Math.ceil(3.5) =&gt; x=4</code>
Math.floor(a)	Lấy số nguyên dưới của a	<code>x=Math.floor(3.5) =&gt; x=3</code>
Math.round(a)	Làm tròn số của a	<code>x=Math.round(3.5) =&gt; x=4</code>
Math.random()	Sinh số ngẫu nhiên từ 0 đến 1	<code>x=Math.random() =&gt; x=0..1</code>

❑ Sinh số ngẫu nhiên từ 15 đến 30

```
double so = 15 + 15 * Math.random();  
int rand = (int) Math.round(so);  
System.out.printf("Số ngẫu nhiên: %.0f", so);
```



# DEMO

1. Sinh số ngẫu nhiên từ 5 đến 12  
Xuất số đó và căn bậc 2 của nó ra màn hình
2. Nhập 2 số thực a và b từ bàn phím  
Tính và xuất a lũy b, giá trị nhỏ nhất của 2 số



## ❑ Cú pháp

```
static <datatype> <name> ([<parameters>]) {  
    <method body>  
    [return value]  
}
```

## ❑ Trong đó:

- ❖ <datatype>: kiểu dữ liệu trả về
- ❖ <name>: tên hàm
- ❖ [<parameters>]: danh sách tham số (có thể có hoặc không)
- ❖ <method body>: thân hàm – mã thực hiện
- ❖ [return value]: trả kết quả về (có thể có hoặc không, nếu <datatype> là void thì không cần return, ngược lại bắt buộc phải có)

- ❑ Ví dụ sau sẽ đọc họ tên và số điện thoại từ bàn phím.

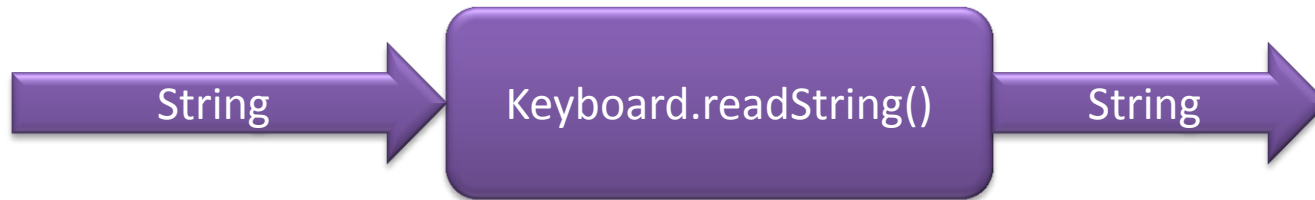
```
System.out.print("Họ và tên: ");  
String fullname = scanner.nextLine();  
  
System.out.print("Số điện thoại: ");  
String mobile = scanner.nextLine();
```

- ❑ Ta nhận thấy rằng mỗi lần đọc chuỗi ta xuất một dòng nhắc nhở người nhập trước khi đọc chuỗi từ bàn phím. Việc này được lặp lại nhiều lần.
- ❑ Mong muốn xây dựng hàm đọc chuỗi `readString()` để đơn giản hóa công việc viết mã

**`String fullname = Keyboard.readString("Họ và tên")`**




## Mô hình hàm



## Xây dựng hàm

```
public class Keyboard {  
    static Scanner scanner = new Scanner(System.in);  
    public static String readString(String message) {  
        System.out.print(message);  
        String input = scanner.nextLine();  
        return input;  
    }  
}
```

## Sử dụng hàm



```
String fullname = Keyboard.readString("Họ và tên: ");  
String mobile = Keyboard.readString("Số điện thoại: ");
```



# DEMO

Xây dựng thư viện đọc dữ liệu từ bàn phím

- + readString(String): String
- + readInt(String): int
- + readDouble(String): double
- + readBoolean(String): boolean





**Java™**

# **LẬP TRÌNH JAVA 1**

## **BÀI 6: HÀM**

### **PHẦN 2**

- ❑ Tráo 2 phần tử trong mảng
  - ❖ `XArrays.swap(int[] array, int i, int j)`
- ❑ Tìm số nhỏ nhất
  - ❖ `XArrays.min(int[] array): int`
- ❑ Tìm số lớn nhất
  - ❖ `XArrays.max(int[] array):int`
- ❑ Xóa phần tử khỏi mảng
  - ❖ `XArrays.remove(int[] array, int i): int[]`
- ❑ Bổ sung phần tử vào mảng
  - ❖ `XArrays.add(int[] array, int x):int[]`

```
public static void swap(int[] a, int i, int j){  
    int temp = a[i];  
    a[i] = a[j];  
    a[j] = temp;  
}
```

```
public class XArrays {  
    public static void swap(int[] a, int i, int j){  
    public static int min(int[] a){  
    public static int[] add(int[] a, int x){  
    public static int[] remove(int[] a, int i){  
}
```

```
public static int min(int[] a){  
    int min = a[0];  
    for(int i = 0; i < a.length; i++){  
        if(a[i] < min){  
            min = a[i];  
        }  
    }  
    return min;  
}
```

```
public static int[] add(int[] a, int x){  
    int b[] = new int[a.length + 1];  
    System.arraycopy(a, 0, b, 0, a.length);  
    b[a.length] = x;  
    return b;  
}
```

```
public static int[] remove(int[] a, int i){  
    int b[] = new int[a.length - 1];  
    System.arraycopy(a, 0, b, 0, i);  
    System.arraycopy(a, i+1, b, i, a.length-i-1);  
    return b;  
}
```

```
public static void main(String[] args) {  
    int[] a = {1,2,3,4,5,6};  
    XArrays.swap(a, 1, 4);  
    a = XArrays.add(a, 100);  
    a = XArrays.remove(a, 3);  
    System.out.println(Arrays.toString(a));  
}
```

1	2	3	4	5	6
---	---	---	---	---	---

1	5	3	4	2	6
---	---	---	---	---	---

1	5	3	4	2	6	100
---	---	---	---	---	---	-----

1	5	3	2	6	100
---	---	---	---	---	-----

- ❑ Trong một lớp có thể có nhiều hàm cùng tên nhưng khác nhau về tham số (kiểu, số lượng và thứ tự)

```
public class MyClass{  
    static void method()  
    static void method(int x)  
    static void method(float x)  
    static void method(int x, double y)  
}
```

- ❖ Trong lớp MyClass có 4 phương thức cùng tên là method nhưng khác nhau về tham số.
- ❑ Với lời gọi hàm là **MyClass.method(5)** sẽ chạy hàm có 1 tham số nguyên

❑ Tráo 2 phần tử của cả mảng số thực, số nguyên và chuỗi

```
public static void swap(int[] a, int i, int j){  
    int temp = a[i];  
    a[i] = a[j];  
    a[j] = temp;  
}
```

```
public static void swap(double[] a, int i, int j){  
    double temp = a[i];  
    a[i] = a[j];  
    a[j] = temp;  
}
```

```
public static void swap(String[] a, int i, int j){  
    String temp = a[i];  
    a[i] = a[j];  
    a[j] = temp;  
}
```



## ❑ Hãy viết bổ sung các hàm sau vào thư viện XArrays

### ❖ Tìm số nhỏ nhất

➤ XArrays.min(double[] array):double

### ❖ Tìm số lớn nhất

➤ XArrays.max(double[] array):double

### ❖ Xóa phần tử tại vị trí i khỏi mảng

➤ XArrays.remove(double[] array, int i):double[]

➤ XArrays.remove(String[] array, int i):String[]

### ❖ Bổ sung phần tử vào mảng

➤ XArrays.add(double[] array):double[]

➤ XArrays.add(String[] array):String[]

- ❑ Khi truyền tham số vào hàm, nếu hàm có làm thay đổi giá trị của tham số thì giá trị của tham số sau khi gọi hàm có bị thay đổi hay không?
  - ❖ Không thay đổi gọi là tham trị
  - ❖ Có thay đổi gọi là tham biến
- ❑ Java qui định các tham số
  - ❖ Có kiểu nguyên thủy là tham trị
  - ❖ Có các kiểu sau đây là tham biến
    - Mảng
    - Lớp
    - Interface

```
public static void main(String[] args) {  
    int a[] = {10}, b = 10;  
    m1(a);  
    m2(b);  
    System.out.printf("a[0]=%d, b=%d", a[0], b);  
}  
static void m1(int x[]) {  
    x[0] = 5;  
}  
static void m2(int x) {  
    x = 5;  
}
```

—————→ a[0]=5, b=10

- ❑ Tham số biến đổi là tham số khi truyền vào hàm với số lượng tùy ý (phải cùng kiểu).
- ❑ Tham số biến đổi phải được định nghĩa sau cùng

```
static void m(int...x){...}
```

Truyền tham số

m(2,6,8)

m(2)

int[] x = {2,6,8}  
m(x)

- ❑ Bản chất của tham số biến đổi là mảng nhưng khi truyền tham số bạn có thể truyền vào nguyên mảng hoặc liệt kê các phần tử

```
public static void main(String[] args) {  
    int[] a = {1,2,3,4,5};  
    int t1 = sum(a);  
    int t2 = sum(1,2,3,4,5);  
    int t3 = sum(1,2);  
    System.out.printf("t1=%d, t2=%d, t3=%d", t1, t2, t3);  
}  
  
static int sum(int...args) {  
    int tong = 0;  
    for(int x:args){  
        tong+=x;  
    }  
    return tong;  
}
```

→ t1=15, t2=15, t3=3



# DEMO

1. Hiện thực hóa phương thức `sum()`
2. Thêm phương thức ghép `n` chuỗi thành 1 chuỗi



- ☑ Giải thích được cấu trúc hàm
- ☑ Sử dụng một số hàm dựng sẵn trong Java
- ☑ Viết và sử dụng hàm
- ☑ Nạp chồng hàm
- ☑ Phân biệt tham biến, tham trị
- ☑ Sử dụng tham số biến đổi
- ☑ Tạo các thư viện tiện ích

