



**Java™**

# **LẬP TRÌNH JAVA 1**

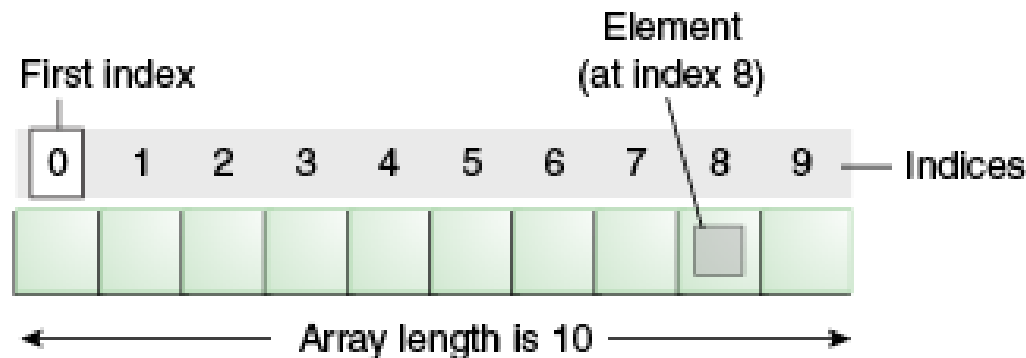
## **BÀI 5: MẢNG**

### **PHẦN 1**

- ⊙ Hiểu được cấu trúc của mảng
- ⊙ Phân biệt được mảng 1 chiều và mảng nhiều chiều
- ⊙ Thực hiện được các thao tác mảng
  - ⊙ Khai báo
  - ⊙ Truy xuất phần tử
  - ⊙ Lấy số phần tử
  - ⊙ Duyệt mảng
  - ⊙ Sắp xếp các phần tử mảng
- ⊙ Sử dụng được lớp tiện ích Arrays



- ❑ Mảng là tập hợp các phần tử cùng kiểu.
- ❑ Mảng có số lượng phần tử cố định và được cấp phát vùng nhớ liên tục.
- ❑ Truy xuất các phần tử mảng bằng chỉ số, bắt đầu là 0
- ❑ Ví dụ:
  - ❖ `int[] a = {5,8,22,1,7,6,11,25,33,9}`



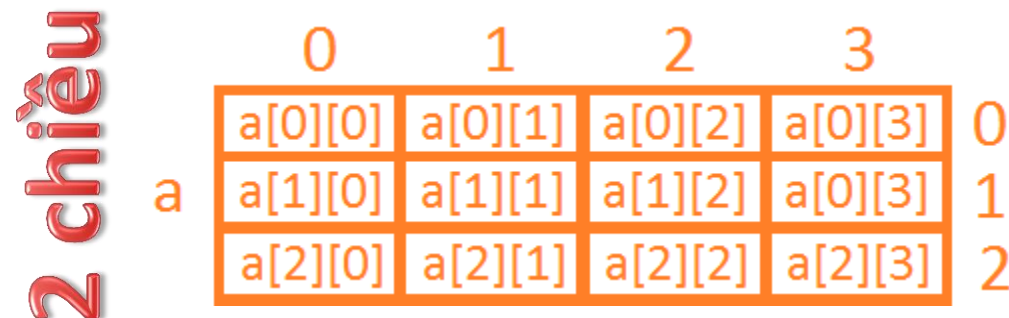
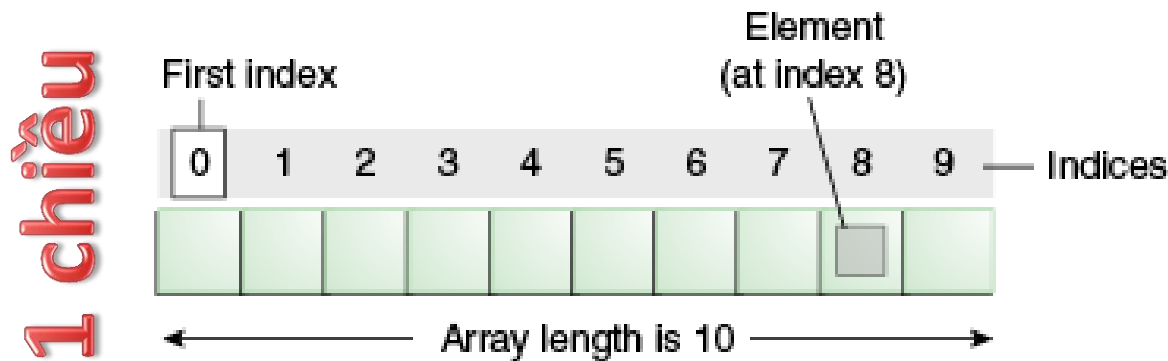
## ❑ Lợi ích của mảng

- ❖ Sử dụng mảng để nắm giữ nhiều giá trị thay vì phải khai báo nhiều biến.
- ❖ Truy xuất nhanh
- ❖ Dễ dàng đọc dữ liệu từ các phần tử và sắp xếp

## ❑ Bất lợi của mảng

- ❖ Số phần tử cố định nên không thể bổ sung hoặc bớt
- ❖ Cấp phát liên tục nên cần phải có vùng bộ nhớ liên tục đủ lớn để cấp phát.

- ❑ Mảng một chiều
- ❑ Mảng nhiều chiều (mảng của các mảng)



- ❑ Khai báo mảng
  - ❖ `datatype[] arr;`
  - ❖ `datatype arr[];`
- ❑ Khởi tạo kích thước mảng
  - ❖ `arr = new datatype[size];`
- ❑ Khai báo và khởi tạo kích thước cho mảng
  - ❖ `datatype[] arr = new datatype[size];`
- ❑ Khai báo và khởi tạo các phần tử
  - ❖ `datatype[] arr = {elem1, elem2,...};`
- ❑ Chú ý:
  - ❖ `datatype[] a, b; // khai báo 2 mảng`
  - ❖ `datatype a[], b; // khai báo một mảng và 1 biến`

```
int a[] = {4, 3, 5, 7}; // khai báo và khởi tạo  
a[2] = a[1] * 4; // 3*4=12  
System.out.println(a.length); // xuất số phần tử của mảng
```

- ❑ Sử dụng chỉ số (**index**) để phân biệt các phần tử.  
Chỉ số mảng tính từ 0.
  - ❖ Sau phép gán a[2] mảng là {4, 3, **12**, 7};
- ❑ Sử dụng thuộc tính **length** để lấy số phần tử của mảng

- ❑ Có thể sử dụng bất kỳ vòng lặp nào để duyệt mảng. Tuy nhiên 2 vòng lặp thường được sử dụng để duyệt mảng là for và for-each.

```
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);  
}
```



```
int a[] = new int[3]; // khai báo và khởi tạo kích thước mảng  
a[0] = 10; // gán giá trị cho phần tử đầu tiên  
a[1] = 20;  
a[2] = 70;  
  
for(int i = 0; i < a.length; i++){ // duyệt mảng  
    System.out.println(a[i]); // xuất phần tử thứ i ra màn hình  
}
```

```
int a[] = {10, 20, 70}; // khai báo và khởi tạo các phần tử  
  
for(int i = 0; i < a.length; i++){ // duyệt mảng  
    System.out.println(a[i]); // xuất phần tử thứ i ra màn hình  
}
```



# DEMO

Nhập mảng số nguyên từ bàn phím  
và xuất mảng vừa nhập ra màn hình



# VÍ DỤ TÌM SỐ NHỎ NHẤT

```
int arr[] = { 33, 3, 4, 5 };  
int min = arr[0];  
for(int i = 1; i < arr.length; i++){  
    if (min > arr[i]){  
        min = arr[i];  
    }  
}  
System.out.println(min);
```

☐ arr.length?

☐ arr[i]?



# DEMO

Nhập mảng số nguyên

+ Tính và xuất trung bình cộng

+ Xuất lập phương các phần tử



- ❑ Hãy viết chương trình thực hiện các công việc sau đây
  - ❖ Tìm số lớn nhất
  - ❖ Tính trung bình cộng các số chẵn trong mảng
- ❑ Vẽ lưu đồ thuật toán cho mỗi yêu cầu

*Nên chia nhóm để thực hiện*

- ❑ Chúng ta có thể sắp xếp tăng dần hoặc giảm dần các phần tử trong mảng

```
int a[] = {8,2,6,2,9,1,5};  
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 thay đổi toán tử so sánh thành  $<$  thì thuật toán trở thành sắp xếp tăng dần.



# DEMO

Nhập 2 mảng họ tên và điểm.  
Xuất 2 mảng giảm theo điểm





**Java™**

# **LẬP TRÌNH JAVA 1**

## **BÀI 5: MẢNG**

### **PHẦN 2**



❑ **System.arraycopy**(src, srcPos, dest, destPos, length) được sử dụng để copy mảng

❖ Trong đó:

- src là mảng nguồn
- dest là mảng đích
- srcPos là vị trí bắt đầu copy
- destPost là vị trí bắt đầu của mảng đích
- length là số phần tử cần copy

❑ Ví dụ tạo mảng b từ mảng a bỏ số 4

```
int a[] = {1, 2, 3, 4, 5, 6};
```

```
int b[] = new int[5];
```

```
System.arraycopy(a, 0, b, 0, 3);
```

```
System.arraycopy(a, 4, b, 3, 2);
```




b={1,2,3,5,6}

```
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 [] và 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>

**Chú ý: `binarySearch()` chỉ làm việc với mảng đã được sắp xếp tăng dần**

```
int[] a = {9, 3, 8, 7, 3, 9, 4, 2};  
  
System.out.println("Mảng gốc: " + Arrays.toString(a));  
  
Arrays.sort(a);  
System.out.println("Sau sort: " + Arrays.toString(a));  
  
int i = Arrays.binarySearch(a, 8);  
System.out.println("Vị trí của 8 là " + i);  
  
Arrays.fill(a, 0);  
System.out.println("Sau fill: " + Arrays.toString(a));
```



Mảng gốc: [9, 3, 8, 7, 3, 9, 4, 2]  
Sau sort: [2, 3, 3, 4, 7, 8, 9, 9]  
Vị trí của 8 là 5  
Sau fill: [0, 0, 0, 0, 0, 0, 0, 0]



# DEMO

Nhập mảng 5 SV và xuất tăng  
dần theo alphabet



	0	1	2	3	
a	a[0][0]	a[0][1]	a[0][2]	a[0][3]	0
	a[1][0]	a[1][1]	a[1][2]	a[1][3]	1
	a[2][0]	a[2][1]	a[2][2]	a[2][3]	2

**Mảng 2 chiều**

Index	0	1	2	3	4
0	65,340	12,483	138,189	902,960	633,877
1	5,246	424,642	650,380	821,254	866,122
2	89,678	236,781	601,691	329,274	913,534
3	103,902	4,567	733,611	263,010	85,550
4	2,778	658,305	128,788	978,155	620,702
5	45,024	55,058	705,586	89,672	384,605
6	780	47,538	523,784	556,801	617,107
7	32,667	350,890	834,753	638,108	85,188
8	56,083	145,582	775,040	548,322	756,587
9	41,123	543,542	537,738	513,048	418,482

**Mảng 3 chiều**

- ❑ Mảng nhiều chiều hay còn gọi là mảng của các mảng
- ❑ Mỗi phần tử của mảng là một mảng khác
- ❑ Khai báo 2 chiều
  - ❖ `datatype[][] arr;`
  - ❖ `datatype arr[][];`
  - ❖ `datatype [] arr[];`
- ❑ Khai báo có khởi tạo
  - ❖ `datatype[][] arr=new datatype[size1][size2];`
  - ❖ `datatype[][] arr = {{elem1, elem2}, {elem1, elem2}}`

```
// Khai báo và khởi tạo
int arr[][] = { { 1, 2, 3 }, { 2, 4, 5 }, { 4, 4, 5 } };

// Xuất mảng 2 chiều
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        System.out.print(arr[i][j] + " ");
    }
    System.out.println();
}
```



# DEMO

Nhập mảng số nguyên 2 chiều 2 hàng,  
3 cột từ bàn phím và xuất ra màn hình  
mảng đã nhập





// Tạo 2 mảng

```
int a[][] = { { 1, 3, 4 }, { 3, 4, 5 } };
```

```
int b[][] = { { 1, 3, 4 }, { 3, 4, 5 } };
```

// Tạo mảng lưu kết quả

```
int c[][] = new int[2][3];
```

// Cộng các phần tử và xuất ra màn hình

```
for(int i = 0; i < 2; i++) {  
    for(int j = 0; j < 3; j++) {  
        c[i][j] = a[i][j] + b[i][j];  
        System.out.print(c[i][j] + " ");  
    }  
    System.out.println();  
}
```

1	3	4
3	4	5

 + 

1	3	4
3	4	5

 = 

2	6	8
6	8	10

- ❑ Thảo luận và viết mã thực hiện 2 yêu cầu sau
  - ❖ Nhân 2 ma trận
  - ❖ Xoay ma trận vuông 90 độ (hàng  $i$  thành cột  $i$ )

- ☑ Hiểu được cấu trúc của mảng
- ☑ Phân biệt được mảng 1 chiều và mảng nhiều chiều
- ☑ Thực hiện được các thao tác mảng
  - ☑ Khai báo
  - ☑ Truy xuất phần tử
  - ☑ Lấy số phần tử
  - ☑ Duyệt mảng
  - ☑ Sắp xếp các phần tử mảng
- ☑ Sử dụng được lớp tiện ích Arrays

