

# JAVA CORE



# Contents



OVERVIEWS ARRAYS



PASSING ARRAY FOR METHOD



SORT BY IN ARRAY



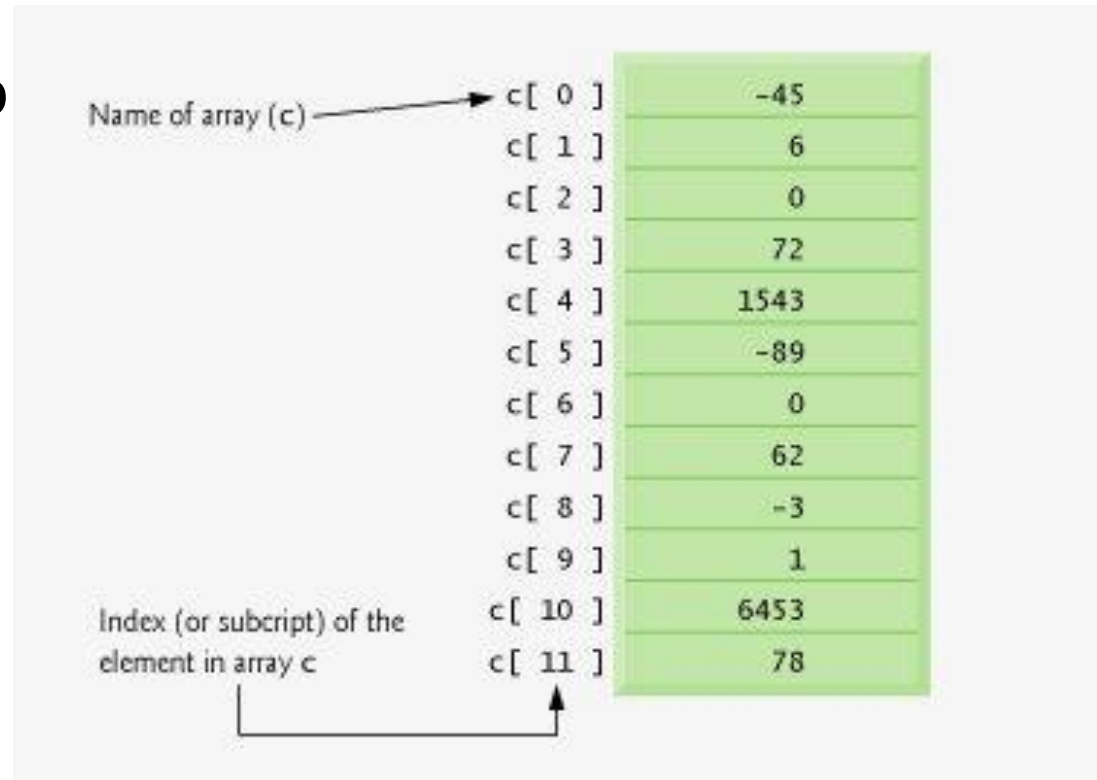
EXCERCISES

# OVERVIEWS ARRAY

# Mảng trong Java

## Intro

- ❖ Chứa các giá trị có cùng kiểu dữ liệu.
- ❖ Đối tượng mảng lưu trữ theo kiểu tham chiếu.
- ❖ Chỉ số mảng phải là số nguyên dương.



# Mảng trong Java

## Khai báo biến Mảng

❖ Cú pháp:

*dataType[] arrayRefVar;*

hoặc

*dataType arrayRefVar[];*

// Kiểu viết này được phép, ít sử dụng

❖ Ví dụ:

*Double[] myList;*

hoặc

*Double myList[];*

## Khởi tạo Mảng

- ❖ Khi khai báo biến Mảng sẽ chưa được cấp phát bộ nhớ. Chỉ được cấp phát khi đã khởi tạo Mảng.
- ❖ Nếu 1 biến không có sự tham chiếu thì giá trị mặc định của biến đó là **null**.
- ❖ Cú pháp:

*arrayRefVar = new dataType[arraySize];*

*Hoặc*

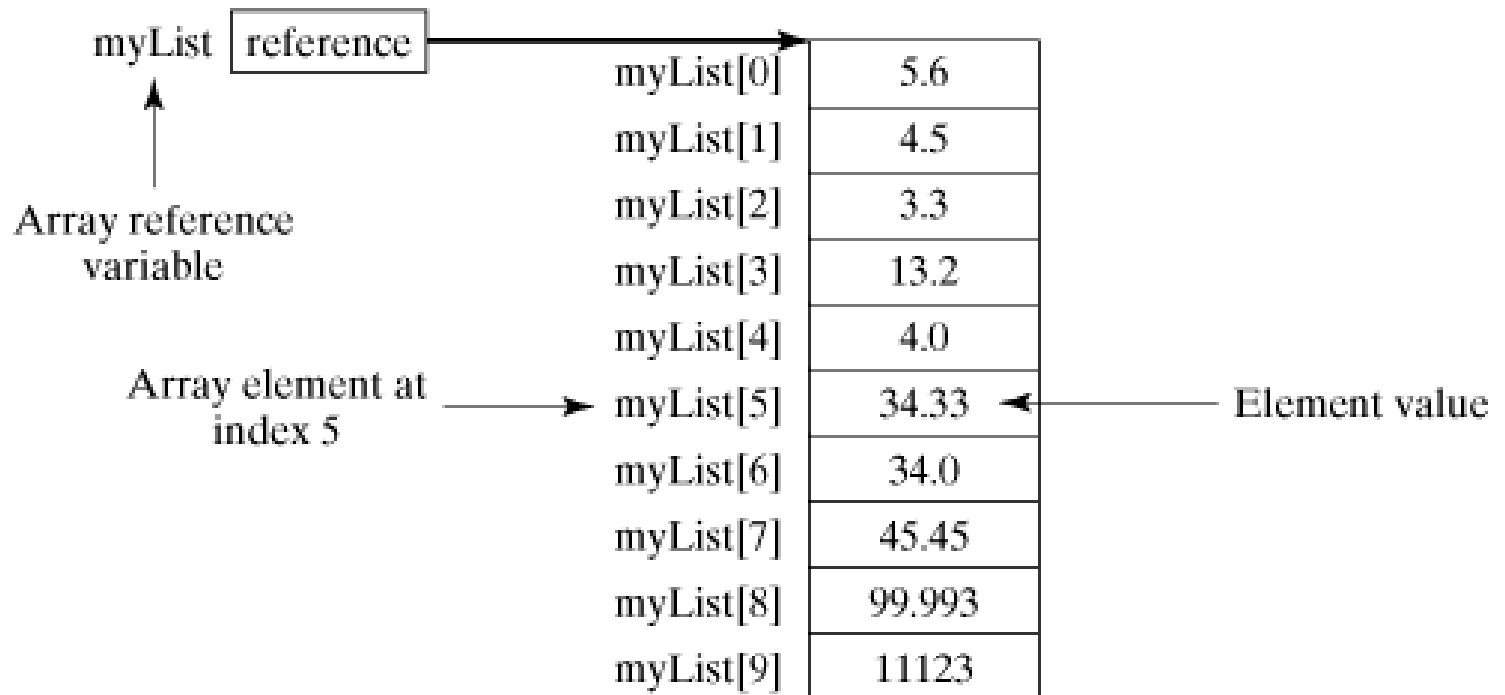
*dataType arrayRefVar[] = new dataType[arraySize];*

# Mảng trong Java

## Ví dụ

```
double[] myList = new double[10];
```

```
double[] myList = new double[10];
```



## Kích thước Mảng và giá trị mặc định

- ❖ Kích thước Mảng không thể thay đổi sau khi đã khởi tạo mảng.
- ❖ Nhận về thông tin kích thước mảng `arrayRefVar.length`
- ❖ Cho ví dụ, `myList.length` là 10.
- ❖ Gán giá trị mặc định 0 cho kiểu số, `'\u0000'` cho kiểu ký tự, và `false` cho kiểu luận lý Boolean.



## Khởi tạo Mảng

❖ Cú pháp:

```
dataType[] arrayRefVar = {value0, value1, ..., valuek};
```

❖ Cho ví dụ:

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

❖ Tương đương với phát biểu bên dưới:

```
double[] myList = new double[4];
```

```
myList[0] = 1.9;
```

```
myList[1] = 2.9;
```

```
myList[2] = 3.4; myList[3] = 3.5;
```

## Xử lý với Mảng

### ❖ Ví dụ

*// Duyệt và khởi tạo mảng **myList** giá trị mặc định từ 0.0 và 99.0*

```
for (int i = 0; i < myList.length; i++) {  
    myList[i] = Math.random() * 100;  
}
```

*// Duyệt mảng sử dụng **foreach**. Cú pháp:*

```
for (<kiểu dữ liệu tương ứng> <tên biến>: <Biến mảng>) {  
    // Xử lý với các phần tử  
}  
  
for (double element: myList) {  
    System.out.println(element);  
}
```

## Ví dụ

### ❖ Ví dụ 01:

*(Khởi tạo mảng với giá trị nhập vào) Duyệt và khởi tạo mảng **myList** với kích thước mảng là **10** (kiểu số nguyên). Giá trị do người dùng nhập vào từ bàn phím. Hiển thị các giá trị trong mảng và tính tổng các giá trị đó.*

### ❖ Ví dụ 02:

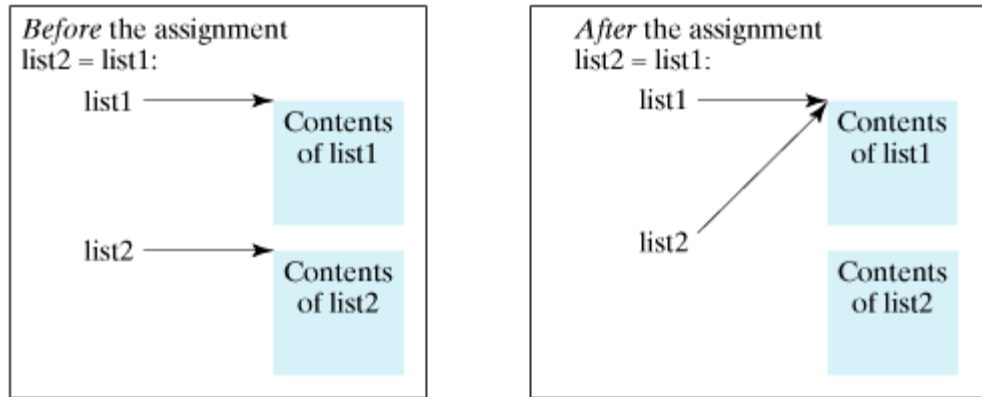
*(Tìm giá trị phần tử lớn nhất trong mảng) Duyệt và khởi tạo mảng **myList** với kích thước mảng là **10** (kiểu số nguyên). Giá trị do người dùng nhập vào từ bàn phím. Tìm giá trị phần tử lớn nhất trong mảng **myList**.*

### ❖ Ví dụ 03:

*(Tìm vị trí phần tử trong mảng) Duyệt và khởi tạo mảng **myList** với kích thước mảng là **10** (kiểu số nguyên). Giá trị do người dùng nhập vào từ bàn phím. Tìm vị trí phần tử chứa giá trị nhỏ nhất và lớn nhất đó.*

## Sao chép Mảng

❖ Nếu bạn cần sao chép mảng hoặc 1 phần của mảng : `list2 = list1;`



❖ Không sao chép nội dung của mảng tham chiếu bởi `list1` đến `list2`.

❖ Dữ liệu của `list2` không còn tham chiếu. Nó bị coi là rác. Sẽ tự động bị xóa bởi trình xử lý `Java Virtual Machine`.

## Sao chép Mảng

### ❖ *Giải pháp:*

- ✓ Sử dụng duyệt mảng nguồn và gán từng giá trị cho mảng đích:

```
int[] sourceArray = {2, 3, 1, 5, 10};  
int[] targetArray = new int[sourceArray.length];  
for (int i = 0; i < sourceArray.length; i++) {  
    targetArray[i] = sourceArray[i]; }  
}
```

- ✓ Sử dụng phương thức tĩnh (static) `arraycopy` trong lớp `System`.

\* Cú pháp:

```
arraycopy(sourceArray, srcPos, targetArray, tarPos, length);
```

```
System.arraycopy(sourceArray, 0, targetArray, 0, sourceArray.length);
```

## Sao chép Mảng

### ❖ Ví dụ:

(Khởi tạo mảng với giá trị nhập vào) Duyệt và khởi tạo mảng *sourceList* với kích thước mảng là **10** (kiểu số nguyên). Giá trị do người dùng nhập vào từ bàn phím. Thực thi sao chép mảng nguồn *sourceList* đến mảng đích *targetList*. Duyệt mảng đích *targetList* và in ra danh sách phần tử.

- Gợi ý:
  - Sử dụng 2 cách để sao chép mảng.

# PASSING ARRAY FOR METHOD

## Truyền tham số Mảng cho Methods

❖ Có thể truyền tham số mảng cho phương thức. Cho ví dụ, hiển thị các phần tử trong mảng số nguyên:

```
public static void printArray(int[] array) {  
    for (int i = 0; i < array.length; i++) {  
        System.out.print(array[i] + " ");  
    }  
}
```

❖ Có thể gọi và truyền cho mảng. Cho ví dụ:

```
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

❖ Sự khác nhau giữa truyền giá trị của biến kiểu cơ sở và truyền biến mảng:

✓ Tham số là kiểu dữ liệu cơ sở **primitive type**, truyền giá trị cho tham số.

✓ Tham số là kiểu dữ liệu mảng **an array type**, tham số truyền vào chứa tham chiếu tới mảng. Đây là kiểu truyền tham chiếu cho phương thức.



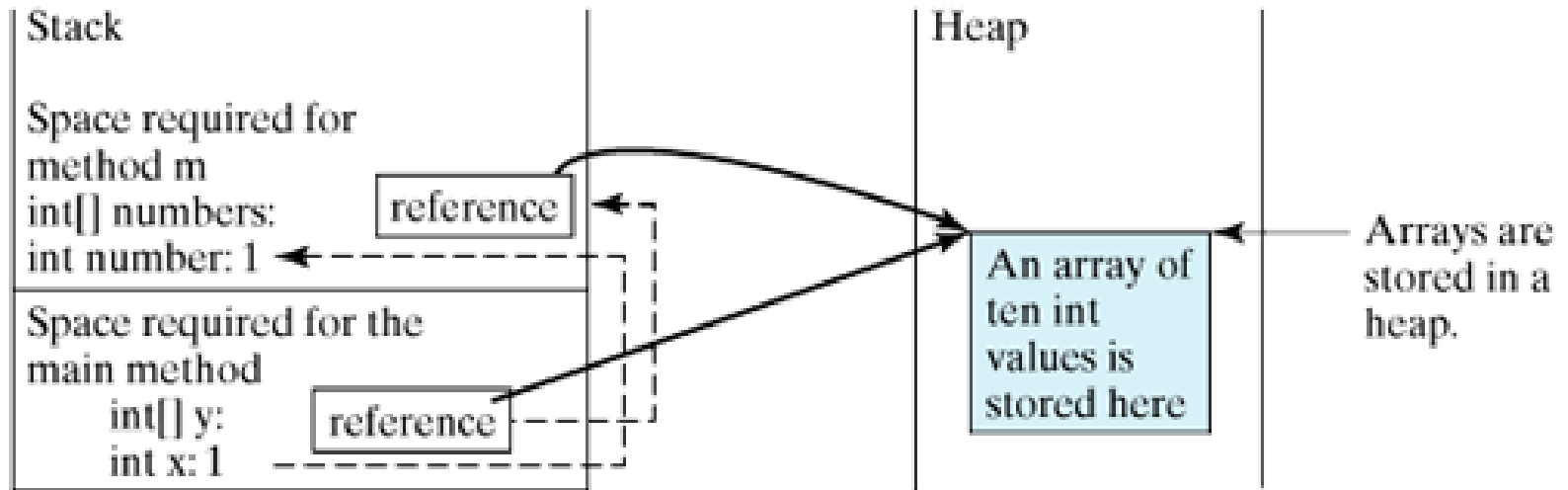
## Truyền mảng tới Methods

❖ Cho ví dụ:

```
public class Test {  
    public static void main(String[] args) {  
        int x = 1; // x giá trị kiểu số nguyên  
        int[] y = new int[10]; // y đại diện cho mảng chứa giá trị int  
  
        m(x, y); // Gọi phương thức m với tham số truyền vào x và y  
        System.out.println("x is " + x);  
        System.out.println("y[0] is " + y[0]);  
    }  
  
    public static void m(int number, int[] numbers) {  
        number = 1001; // Gán 1 giá trị mới cho số  
        numbers[0] = 5555; // Gán 1 giá trị mới cho mảng số  
    }  
}
```

# Mảng trong Java

## Mô tả ví dụ



- **Chú ý:**

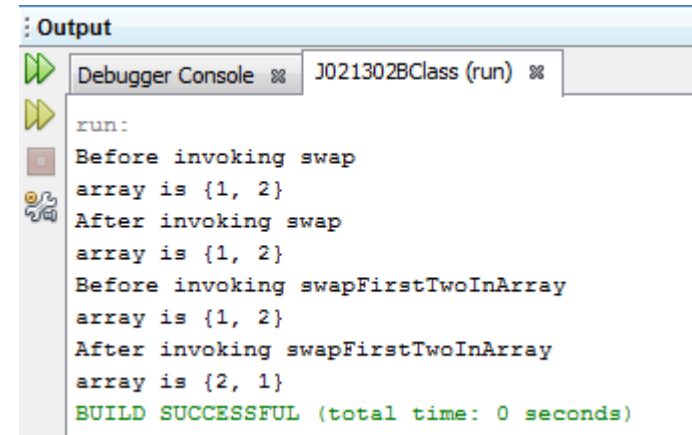
- JVM lưu trữ đối tượng mảng trong vùng nhớ gọi là *heap*

# Mảng trong Java

## Ví dụ

```
/** Hoán đổi 2 biến */  
public static void swap(int n1, int n2) {  
    int temp = n1;  
    n1 = n2;  
    n2 = temp;  
}
```

```
/** Hoán đổi 2 vị trí đầu tiên trong mảng */  
public static void swapFirstTwoInArray(int[] array) {  
    int temp = array[0];  
    array[0] = array[1];  
    array[1] = temp;  
}
```

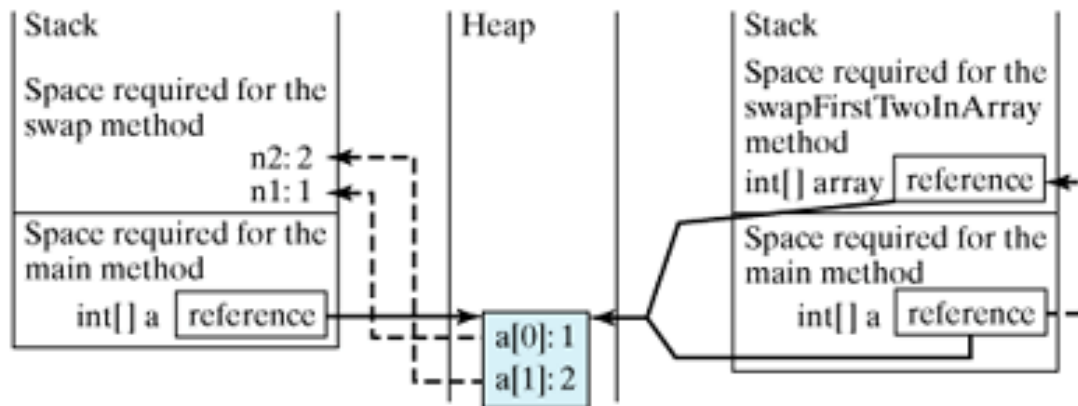


```
run:  
Before invoking swap  
array is {1, 2}  
After invoking swap  
array is {1, 2}  
Before invoking swapFirstTwoInArray  
array is {1, 2}  
After invoking swapFirstTwoInArray  
array is {2, 1}  
BUILD SUCCESSFUL (total time: 0 seconds)
```

# Mảng trong Java

## Mô tả ví dụ

**When passing an array to a method, the reference of the array is passed to the method.**



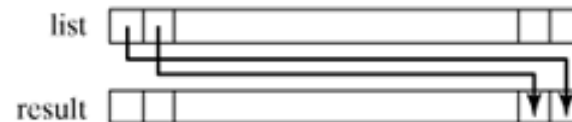
Invoke `swap(int n1, int n2)`. The arrays are stored in a heap. The primitive type values in `a[0]` and `a[1]` are passed to the swap method.

Invoke `swapFirstTwoInArray(int[] array)`. The reference value in `a` is passed to the `swapFirstTwoInArray` method.

## Trả về kiểu mảng từ 1 phương thức

❖ Có thể truyền tham số mảng cho phương thức. Một phương thức trả về kiểu mảng.  
Cho ví dụ:

```
1 public static int[] reverse(int[] list) {  
2     int[] result = new int[list.length];  
3  
4     for (int i = 0, j = result.length - 1;  
5         i < list.length; i++, j--) {  
6         result[j] = list[i];  
7     }  
8  
9     return result;  
10 }
```



❖ Chạy thử:

```
int[] list1 = {1, 2, 3, 4, 5, 6};  
int[] list2 = reverse(list1);
```

# SORT BY IN ARRAY

## Sắp xếp Mảng

### ❖ Bubble Sort :

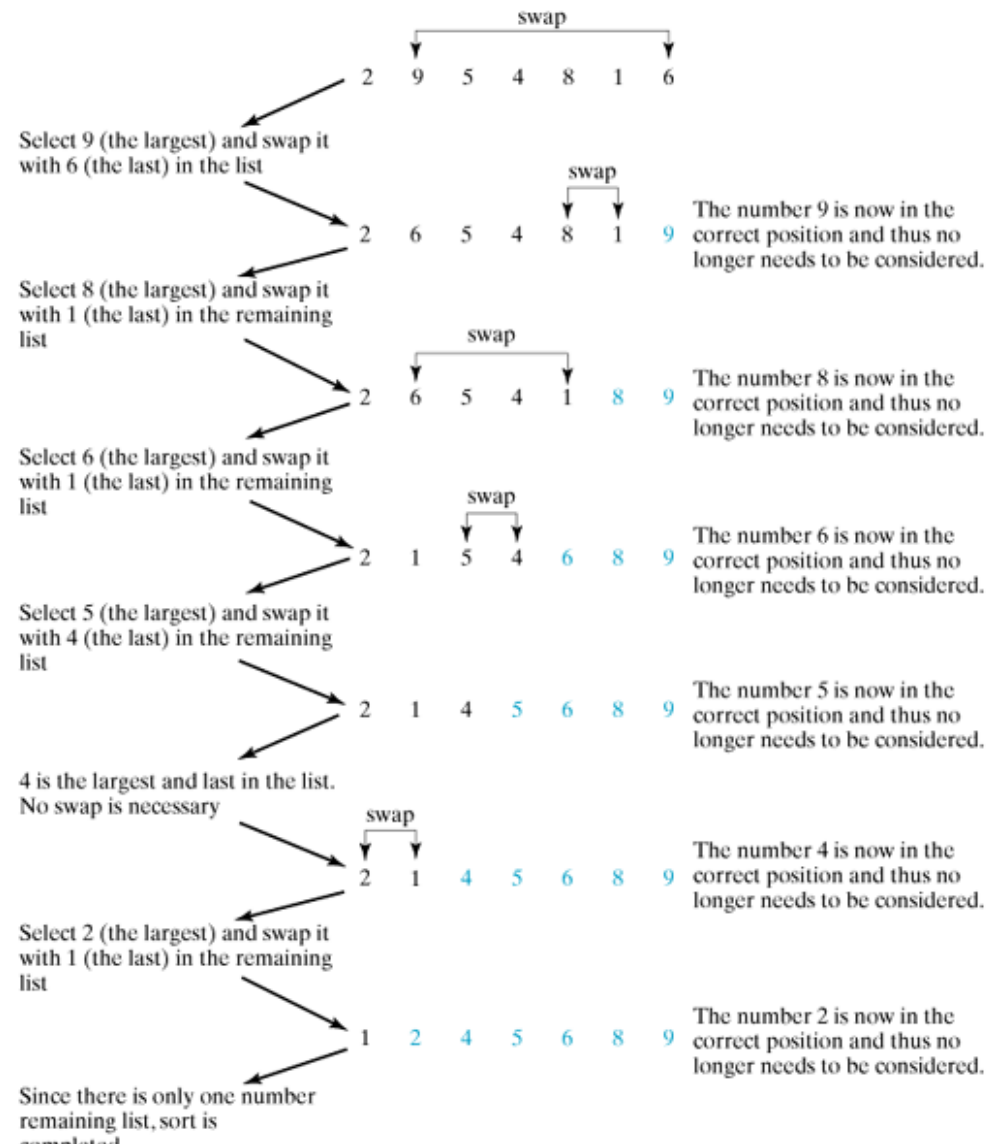
- Bạn muốn sắp xếp 1 danh sách theo thứ tự Tăng hoặc giảm dần. Cho 1 danh sách giá trị cần sắp xếp {2, 9, 5, 4, 8, 1, 6} sử dụng sắp xếp nổi bọt (Bubble Sort).

```
for (int i = 0; i < arr.Length - 1; i++)  
{  
    for (int j = i + 1; j <= arr.Length - 1; j++)  
    {  
        if (int.Parse(arr[i]) > int.Parse(arr[j]))  
        {  
            strTemp = arr[i];  
            arr[i] = arr[j];  
            arr[j] = strTemp;  
        }  
    }  
}
```

## Sắp xếp Mảng

### ❖ Selection Sort :

- Bạn muốn sắp xếp 1 danh sách theo thứ tự Tăng hoặc giảm dần. Cho 1 danh sách giá trị cần sắp xếp {2, 9, 5, 4, 8, 1, 6} sử dụng selection sort.





## Sắp xếp Mảng

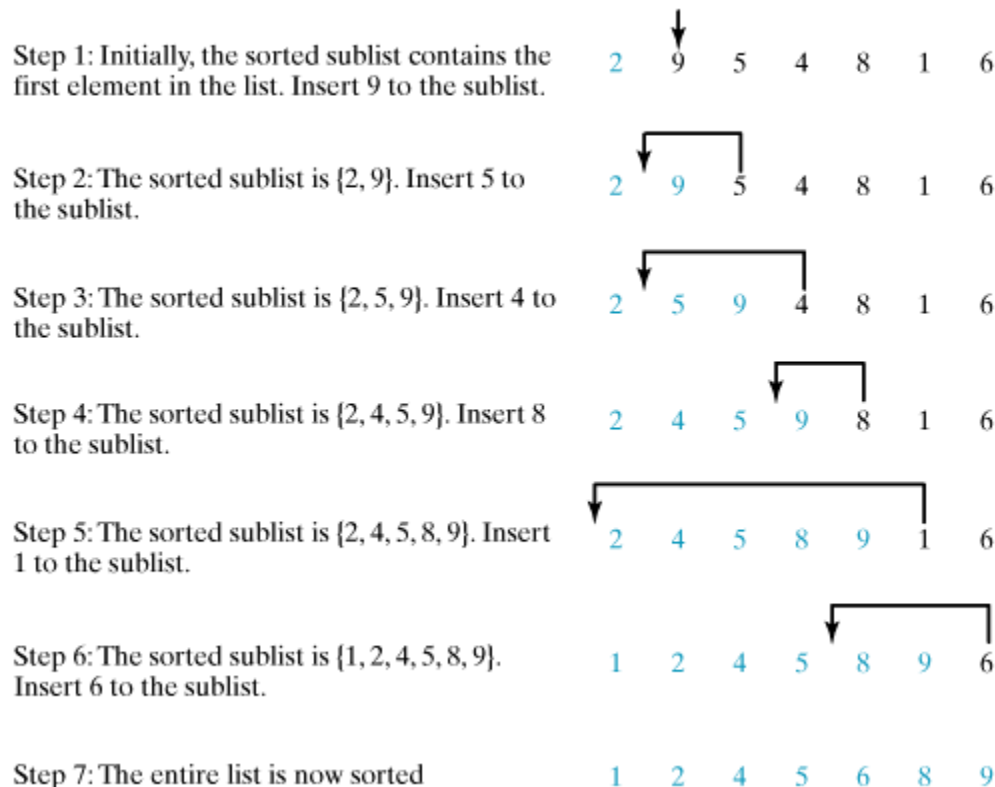
### ❖ Ví dụ Selection Sort:

```
public static void selectionSort(double[] list) {  
    for (int i = list.length - 1; i >= 1; i--){  
        // Find the maximum in the list[0..i]  
        double currentMax = list[0];  
        int currentMaxIndex = 0;  
        {for (int j = 1; j <= i; j++) {  
            if (currentMax < list[j]) {  
                currentMax = list[j];  
                currentMaxIndex = j;  
            }  
        }  
        // Swap list[i] with list[currentMaxIndex] if necessary;  
        if (currentMaxIndex != i) {  
            list[currentMaxIndex] = list[i];  
            list[i] = currentMax;  
        }  
    }  
}
```

## Sắp xếp Mảng

### ❖ Cho ví dụ Insertion Sort :

**Insertion sort repeatedly inserts a new element into a sorted sublist.**



## Sắp xếp Mảng

### ❖ Ví dụ Insertion Sort:

```
public static void insertionSort(double[] list) {  
    for (int i = 1; i < list.length; i++) {  
        /** insert list[i] into a sorted sublist list[0..i-1] so t  
            list[0..i] is sorted. */  
        double currentElement = list[i];  
        int k;  
        for (k = i - 1; k >= 0 && list[k] > currentElement; k--) {  
            list[k + 1] = list[k];  
        }  
  
        // Insert the current element into list[k+1]  
        list[k + 1] = currentElement;  
    }  
}
```

## Lớp mảng Array

❖ The *java.util.Arrays* lớp chứa tập các phương thức tĩnh (static) cho việc *sắp xếp* và *tìm kiếm mảng*, *so sánh mảng*, và *điền phần tử mảng*. Phương thức nạp chồng (overload) cho tất cả kiểu dữ liệu cơ sở.

❖ **Gọi *sort(number)*:**

```
double[] numbers = {6.0, 4.4, 1.9, 2.9, 3.4, 3.5};  
java.util.Arrays.sort(numbers); // Sort the whole array
```

```
char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};  
java.util.Arrays.sort(chars, 1, 3); // Sort part of the array
```

❖ **Gọi *equals* :**

```
int[] list1 = {2, 4, 7, 10}; int[] list2 = {2, 4, 7, 10};  
int[] list3 = {4, 2, 7, 10};
```

```
System.out.println(java.util.Arrays.equals(list1, list2)); // true  
System.out.println(java.util.Arrays.equals(list2, list3)); // false
```

## Lớp mảng Array

❖ *Gọi fill:*

```
int[] list1 = {2, 4, 7, 10};  
int[] list2 = {2, 4, 7, 10};  
java.util.Arrays.fill(list1, 5); // fill 5 to the whole array  
java.util.Arrays.fill(list2, 1, 3, 8); // fill 8 to a partial array
```

# Mảng trong Java

## Mảng 2 chiều

❖ Cú pháp:

*dataType[][] arrayRefVar;*

hoặc

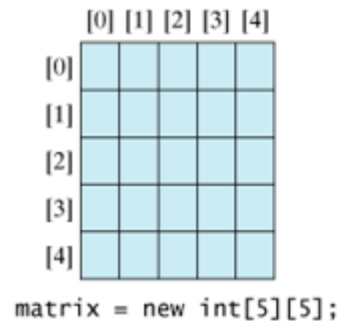
*dataType arrayRefVar[][];*

❖ Cho ví dụ:

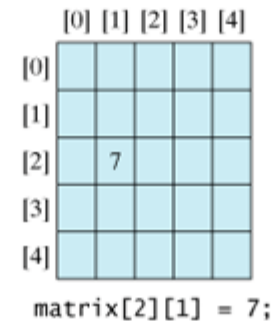
*int[][] matrix;*

hoặc

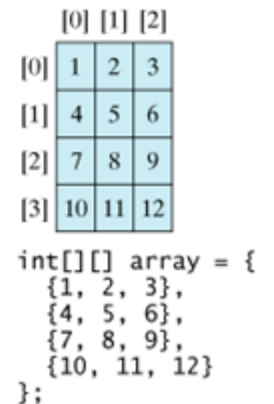
*int matrix[][];*



(a)



(b)



(c)

*matrix = new int[5][5];*

# Mảng trong Java

## Mảng 2 chiều

❖ Cho ví dụ, tạo đối tượng mảng với giá trị được khởi tạo

```
int[][] array = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9},  
    {10, 11, 12}  
};
```

(a)

Equivalent

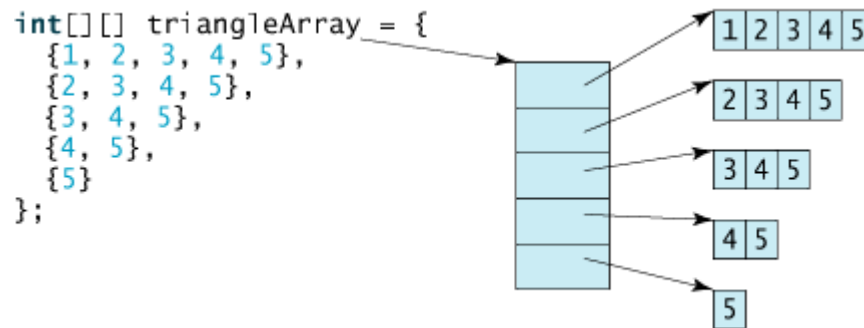
```
int[][] array = new int[4][3];  
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;  
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;  
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;  
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

(b)

# Mảng trong Java

## Mảng Ragged

❖ Cho ví dụ, đây là phần khởi tạo mảng Ragged:



❖ Không yêu cầu khai báo kích thước mảng

```
int[][] triangleArray = new int[5][];  
triangleArray[0] = new int[5];  
triangleArray[1] = new int[4];  
triangleArray[2] = new int[3];  
triangleArray[3] = new int[2];  
triangleArray[4] = new int[1];
```



# Mảng trong Java

## Mảng nhiều chiều

❖ Cú pháp:

*dataType[][][] objets = new dataType[arrise size][arrise size][arrise size]*

❖ Cho ví dụ:

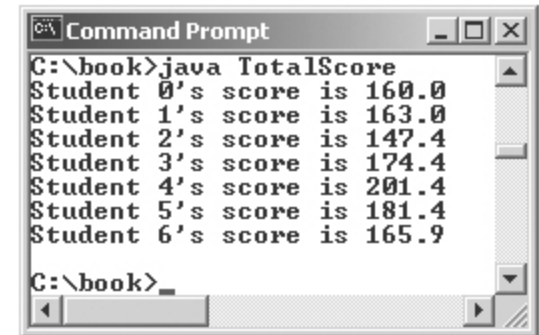
*double[][][] scores = new double[10][5][2];*

```
1 public class TotalScore {
2     /** Main method */
3     public static void main(String args[]) {
4         double[][][] scores = {
5             {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
6             {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
7             {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
8             {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
9             {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
10            {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}},
11            {{1.5, 29.5}, {6.4, 22.5}, {14, 30.5}, {10, 30.5}, {16, 6.0}}};
12
13            // Calculate and display total score for each student
14            for (int i = 0; i < scores.length; i++) {
15                double totalScore = 0;
16                for (int j = 0; j < scores[i].length; j++)
17                    for (int k = 0; k < scores[i][j].length; k++)
18                        totalScore += scores[i][j][k];
19
20                System.out.println("Student " + i + "'s score is " +
21                    totalScore);
22            }
23        }
24    }
```

Diagram illustrating the 3D array structure:

- `scores[0][3][0]` points to the first element of the third sub-array in the first main array.
- `scores[0][3][1]` points to the second element of the third sub-array in the first main array.
- `scores[6]` points to the entire sixth sub-array.
- `[4][1]` points to the second element of the fourth sub-array.

### Output



```
C:\book>java TotalScore
Student 0's score is 160.0
Student 1's score is 163.0
Student 2's score is 147.4
Student 3's score is 174.4
Student 4's score is 201.4
Student 5's score is 181.4
Student 6's score is 165.9
C:\book>
```

## Câu hỏi & Bài tập

# Objects Oriented Programming

