

# Chương 5: Mảng



# Đặt vấn đề

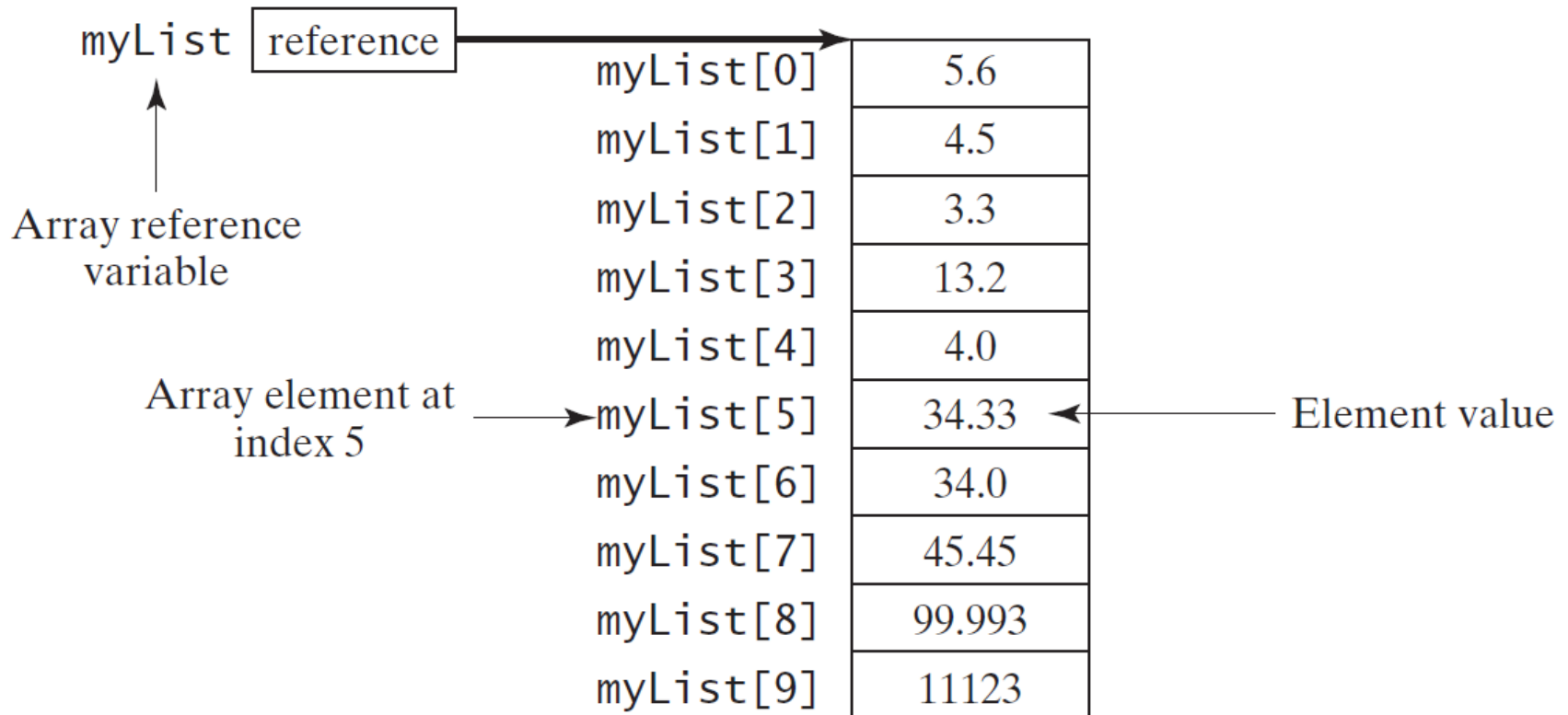
Nhập 100 số nguyên, tính giá trị trung bình của chúng và cho biết có bao nhiêu số lớn hơn giá trị trung bình?



# Giới thiệu

Mảng là một cấu trúc dữ liệu biểu diễn một tập hợp các phần tử cùng kiểu dữ liệu.

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



# Khai báo biến mảng

☞ `datatype[] arrayRefVar;`

Ví dụ:

```
double[] myList;
```

☞ `datatype arrayRefVar[];` // cách này có thể  
dung, nhưng ít người thích

Ví dụ:

```
double myList[];
```



# Tạo mảng

```
arrayRefVar = new datatype[arraySize];
```

Ví dụ:

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

`myList[0]` tham chiếu đến phần tử đầu tiên trong mảng.

`myList[9]` tham chiếu đến phần tử cuối cùng trong mảng.

# Kết hợp khai báo và tạo

☞ `datatype[] arrayRefVar = new  
datatype[arraySize];`

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

☞ `datatype arrayRefVar[] = new  
datatype[arraySize];`

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



# Kích thước của mảng

Khi một mảng được tạo ra thì kích thước của nó là cố định (không thể thay đổi). Cú pháp lấy kích thước của mảng

`arrayRefVar.length`

Ví dụ:

`myList.length` ➔ 10



# Giá trị mặc định

Khi một mảng được tạo ra, các phần tử của nó được gán các giá trị mặc định:

- Kiểu dữ liệu số (int, long, float, double): 0
- Kiểu char: '\u0000'
- Kiểu boolean: false





# Biến chỉ mục

Các phần tử của mảng được truy cập thông qua chỉ mục (index).

Chỉ mục của mảng bắt đầu từ 0 đến *arrayRefVar.length-1*.

Mỗi phần tử trong mảng được đại diện bằng cú pháp sau, còn được gọi là một *biến chỉ mục*:

```
arrayRefVar[index];
```



# Sử dụng biến chỉ mục

Sau khi một mảng được tạo ra, một biến chỉ mục có thể được sử dụng như một biến thông thường.

Ví dụ:

```
myList[2] = myList[0] + myList[1];
```



# Khởi tạo mảng

☞ Khai báo, tạo và khởi tạo trong một bước:

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

Cách ngắn gọn trên tương đương với các câu lệnh sau:

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

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

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

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

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



# LƯU Ý

Phải khai báo, tạo và khởi tạo mảng trong một dòng lệnh.

Việc tách các thao tác này ra sẽ phát sinh lỗi.

Ví dụ sau bị lỗi:

```
double[] myList;
```

```
myList = { 1.9, 2.9, 3.4, 3.5};
```



# Chạy chương trình

Khai báo biến mảng **values**, tạo mảng, và gán tham chiếu của nó đến biến **values**

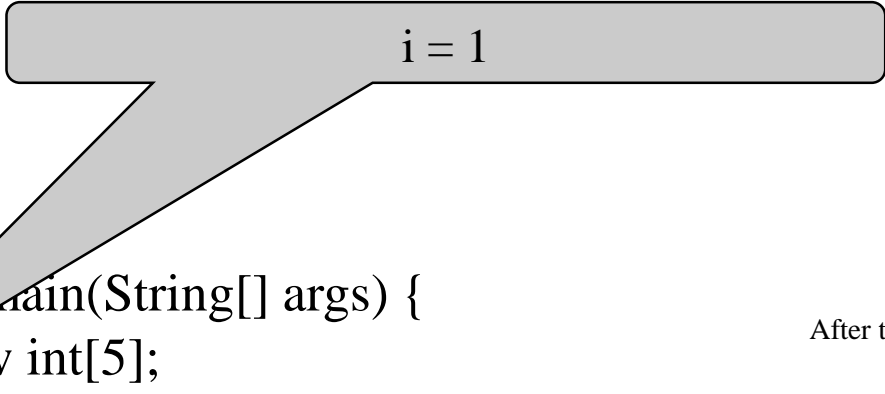
```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0



# Chạy chương trình



$i = 1$

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0



# Chạy chương trình

$i (=1) < 5 \rightarrow \text{true}$

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the array is created

0	0
1	0
2	0
3	0
4	0



# Chạy chương trình

value[1] = 1

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the first iteration

0	0
1	1
2	0
3	0
4	0





# Trace Program with Arrays

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

i = 2

After the first iteration

0	0
1	1
2	0
3	0
4	0



# Chạy chương trình

```
public class Test {  
    public static void main(String[]  
        args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] +  
            values[4];  
    }  
}
```

$i (= 2) < 5 \rightarrow \text{true}$

After the first iteration

0	0
1	1
2	0
3	0
4	0



# Trace Program with Arrays

values[2] = 3 (2 + 1)

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0



# Chạy chương trình

i = 3.

```
public class Test {
    public static void main(String[] args) {
        int[] values = new int[5];
        for (int i = 1; i < 5; i++) {
            values[i] = i + values[i-1];
        }
        values[0] = values[1] + values[4];
    }
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0



# Chạy chương trình

$i (=3) < 5 \rightarrow \text{true}$

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the second iteration

0	0
1	1
2	3
3	0
4	0



# Chạy chương trình

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

values[3] = 6 (3 + 3)

After the third iteration

0	0
1	1
2	3
3	6
4	0

# Chạy chương trình

i = 4

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0



# Chạy chương trình

$i (=4) < 5 \rightarrow \text{true}$

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the third iteration

0	0
1	1
2	3
3	6
4	0





# Chạy chương trình

values[4] = 10 (4 + 6)

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the fourth iteration

0	0
1	1
2	3
3	6
4	10



# Chạy chương trình

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

i = 5

After the fourth iteration

0	0
1	1
2	3
3	6
4	10



# Chạy chương trình

$i (=5) < 5 \rightarrow \text{false}$ . Kết thúc vòng lặp

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < 5; i++) {  
            values[i] = i + values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```

After the fourth iteration

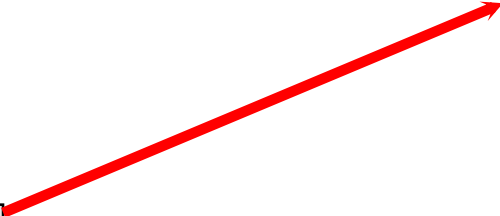
0	0
1	1
2	3
3	6
4	10



# Chạy chương trình

values[0] = 11 (1 + 10)

```
public class Test {  
    public static void main(String[] args) {  
        int[] values = new int[5];  
        for (int i = 1; i < values.length; i++) {  
            values[i] = i * values[i-1];  
        }  
        values[0] = values[1] + values[4];  
    }  
}
```



0	11
1	1
2	3
3	6
4	10



# Processing Arrays

1. Khởi tạo mảng với các giá trị được nhập vào
2. Khởi tạo mảng với các giá trị ngẫu nhiên
3. In mảng
4. Tính tổng các phần tử
5. Tìm phần tử lớn nhất
6. *Trộn mảng ngẫu nhiên*
7. *Dịch chuyển các phần tử mảng*



# Khởi tạo mảng với các giá trị được nhập vào

```
java.util.Scanner input = new java.util.Scanner(System.in);  
System.out.print("Nhập " + myList.length + " giá trị: ");  
for (int i = 0; i < myList.length; i++) {  
    myList[i] = input.nextDouble();  
}
```



# Khởi tạo mảng với các giá trị ngẫu nhiên

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



# In mảng

```
for (int i = 0; i < myList.length; i++) {  
    System.out.print(myList[i] + " ");  
}
```





# Tính tổng các phần tử

```
double total = 0;  
for (int i = 0; i < myList.length; i++) {  
    total += myList[i];  
}
```



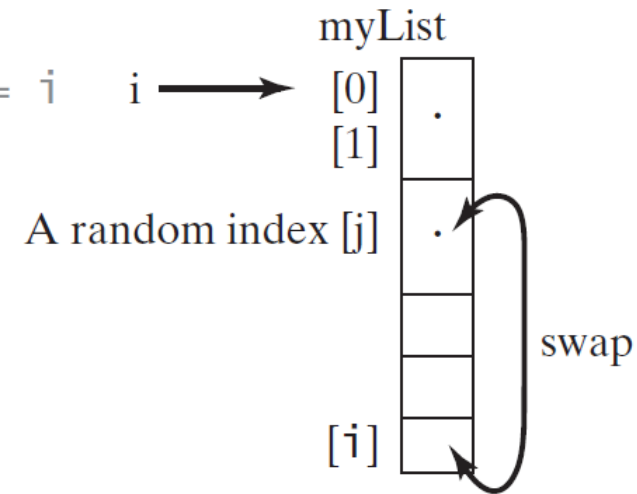
# Tìm phần tử lớn nhất

```
double max = myList[0];  
for (int i = 1; i < myList.length; i++) {  
    if (myList[i] > max) max = myList[i];  
}
```



# Trộn mảng ngẫu nhiên

```
for (int i = myList.length - 1; i > 0; i--) {  
    // Generate an index j randomly with 0 <= j <= i  
    int j = (int)(Math.random()  
        * (i + 1));  
  
    // Swap myList[i] with myList[j]  
    double temp = myList[i];  
    myList[i] = myList[j];  
    myList[j] = temp;  
}
```



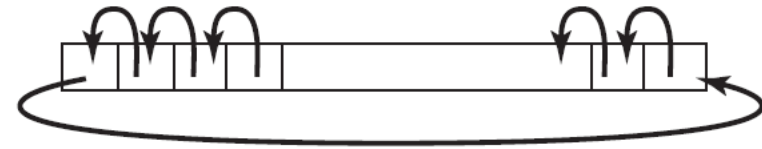
# *Dịch chuyển các phần tử mảng*

```
double temp = myList[0]; // Retain the first element
```

```
// Shift elements left  
for (int i = 1; i < myList.length; i++) {  
    myList[i - 1] = myList[i];  
}
```

```
// Move the first element to fill in the last position  
myList[myList.length - 1] = temp;
```

myList



# Vòng lặp for-each

JDK 1.5 giới thiệu một vòng lặp for mới có thể duyệt qua tất cả các phần tử của mảng một cách tuần tự mà không cần sử dụng *biến chỉ mục*.

Ví dụ: in các phần tử của mảng myList:

```
for (double value: myList)
    System.out.println(value) ;
```

Cú pháp tổng quát

```
for (elementType value: arrayRefVar) {
    // Process the value
}
```

Vẫn phải sử dụng một biến chỉ mục nếu muốn duyệt mảng theo một thứ tự khác hoặc thay đổi các phần tử trong mảng.

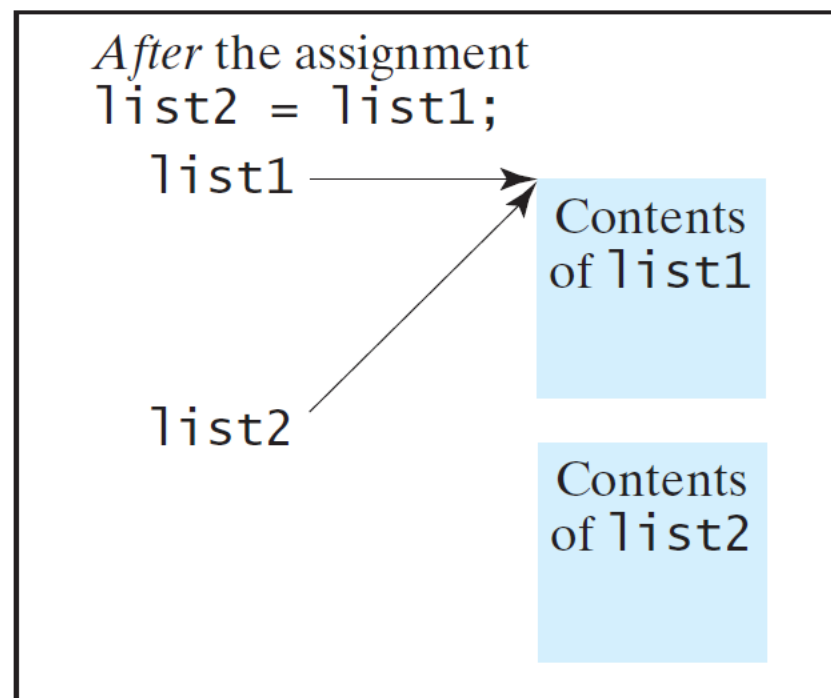
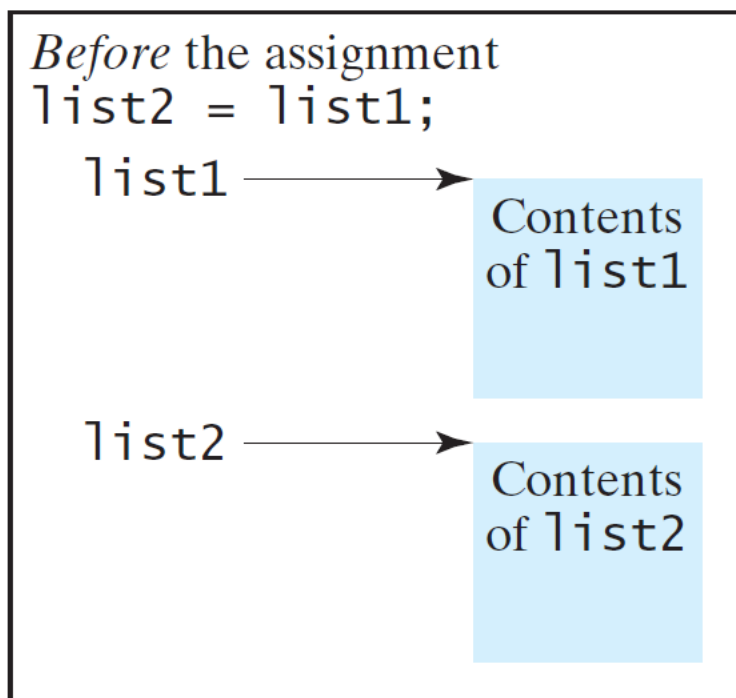


# Sao chép mảng

Thông thường, trong một chương trình, chúng ta cần sao chép một mảng hoặc một phần của mảng.

Chúng ta thường **cố gắng sử dụng phép gán (=)**

*list2 = list1;*



# Sao chép mảng

Sử dụng vòng lặp:

```
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];
```



# Phương thức `arraycopy`

```
arraycopy(sourceArray, src_pos,  
          targetArray, tar_pos, length);
```

Ví dụ:

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





# Truyền mảng cho phương thức

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

Gọi phương thức:

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

Gọi phương thức:

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

Mảng nằm danh



# Mảng nặc danh

## (Anonymous Array)

Câu lệnh:

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

tạo một mảng theo cú pháp sau:

```
new dataType[]{literal0, literal1, ..., literalk};
```

Không có một biến tham chiếu tường minh đến mảng → gọi là *mảng nặc danh*.



# Truyền giá trị (tham trị)

Java sử dụng *truyền giá trị* để truyền các đối số cho một phương thức.


Có những điểm khác nhau quan trọng trong việc truyền một giá trị của các biến kiểu cơ sở so với truyền một mảng.

☞ Với một tham số có kiểu cơ sở → giá trị thật được truyền. Việc thay đổi giá trị của tham số cục bộ bên trong phương thức không ảnh hưởng đến giá trị của biến bên ngoài phương thức.

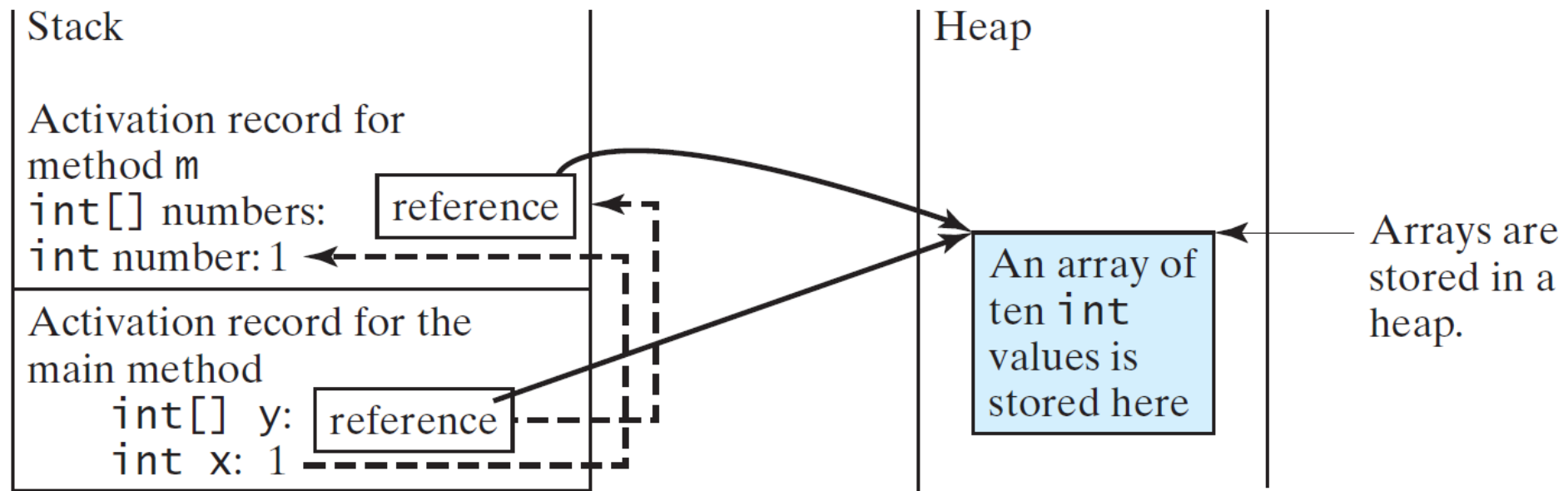
☞ Với một tham số kiểu mảng → giá trị của tham số chứa một tham chiếu đến một mảng, tham chiếu này được truyền vào phương thức. Bất kỳ thay đổi nào đến mảng xảy ra bên trong phương thức thì sẽ ảnh hưởng đến mảng gốc được truyền dưới dạng đối số.

# Ví dụ

```
public class Test {  
    public static void main(String[] args) {  
        int x = 1; // x represents an int value  
        int[] y = new int[10]; // y represents an array of int values  
  
        m(x, y); // Invoke m with arguments x and 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; // Assign a new value to number  
        numbers[0] = 5555; // Assign a new value to numbers[0]  
    }  
}
```

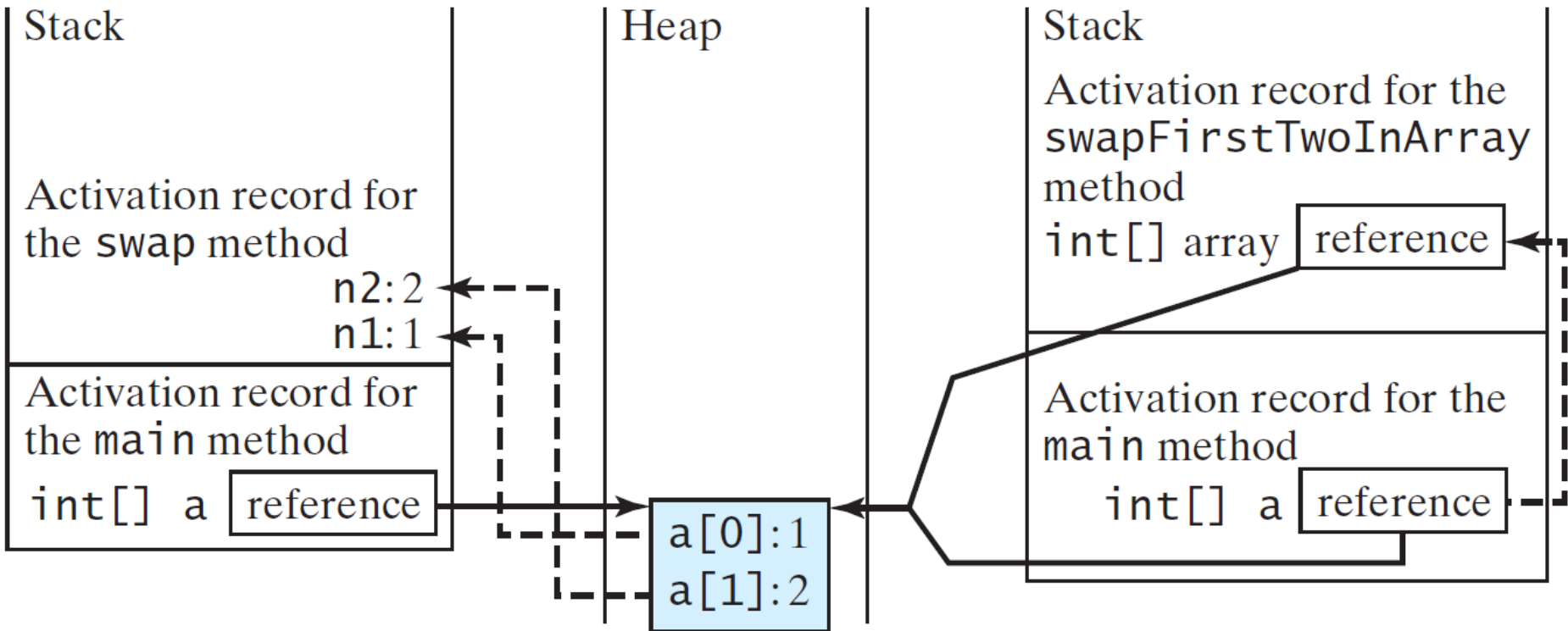


# Ngăn xếp thực thi



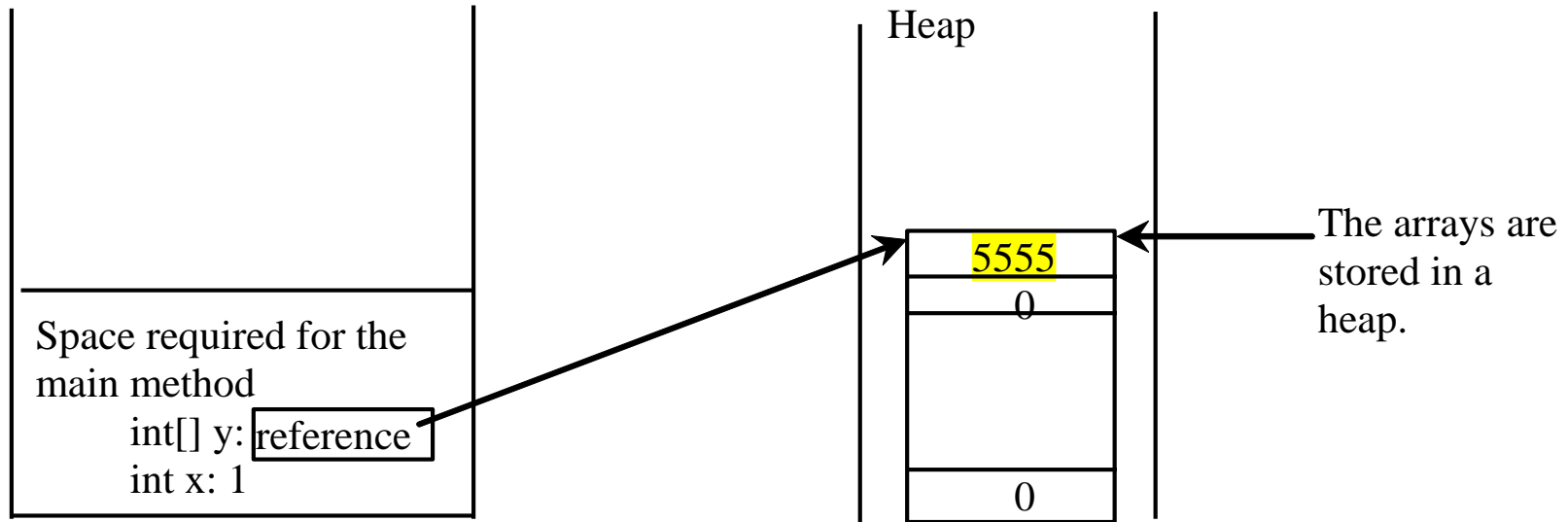
Khi gọi thực thi  $m(x, y)$ , các giá trị của  $x$  và  $y$  được truyền đến **number** và **numbers**. Vì  $y$  chứa giá trị tham chiếu đến mảng nên **numbers** cũng chứa giá trị tham chiếu đến cùng mảng.

# Ngăn xếp thực thi



Khi gọi thực thi **m(x, y)**, các giá trị của **x** và **y** được truyền đến **number** và **numbers**. Vì **y** chứa giá trị tham chiếu đến mảng nên **numbers** cũng chứa giá trị tham chiếu đến cùng mảng.

# Heap



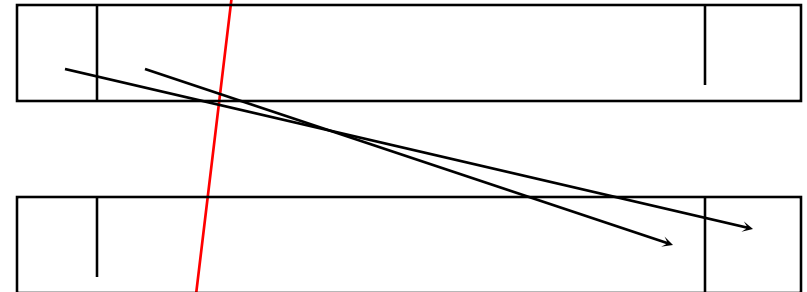
JVM lưu trữ mảng ở vùng nhớ gọi là *heap* (thường sử dụng cho cấp phát bộ nhớ động – các khối vùng nhớ được cấp phát và giải phóng trong một arbitrary order).

# Trả về một mảng từ phương thức

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

list

result



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





# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

Khai báo và tạo mảng **result**

list

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

result

0	0	0	0	0	0
---	---	---	---	---	---



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
         i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 0 và j = 5

list

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

result

0	0	0	0	0	0
---	---	---	---	---	---



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

$i (= 0) < 6 \rightarrow \text{true}$

list

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

result

0	0	0	0	0	0
---	---	---	---	---	---



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
         i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 0 và j = 5  
Gán list[0] cho result[5]

list

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

result

0	0	0	0	0	1
---	---	---	---	---	---



# Ví dụ

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

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
        i < list.length; i++, j--) {
        result[j] = list[i];
    }

    return result;
}
```

i = 1 và j = 4

list

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

result

0	0	0	0	0	1
---	---	---	---	---	---



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

$i (=1) < 6 \rightarrow \text{true}$

list

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

result

0	0	0	0	0	1
---	---	---	---	---	---

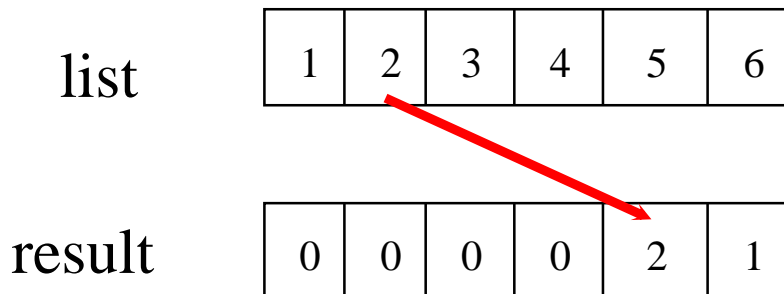


# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 1 và j = 4  
Gán list[1] cho result[4]



# Ví dụ

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

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
        i < list.length; i++, j--) {
        result[j] = list[i];
    }

    return result;
}
```

i = 2 và j = 3

list

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

result

0	0	0	0	2	1
---	---	---	---	---	---





# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i (=2) < 6 → true

list

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

result

0	0	0	0	2	1
---	---	---	---	---	---

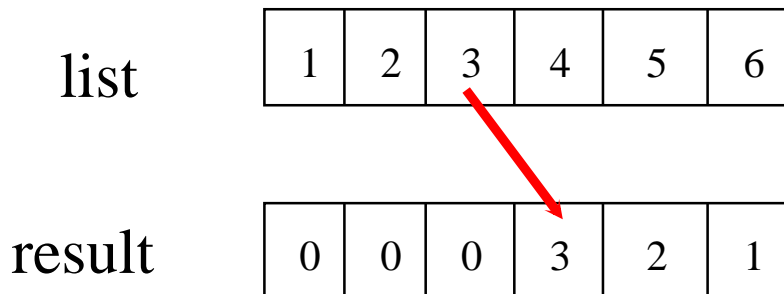


# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 2 và j = 3  
Gán list[i] cho result[j]



# Ví dụ

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

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
        i < list.length; i++, j--) {
        result[j] = list[i];
    }

    return result;
}
```

i = 3 và j = 2

list

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

result

0	0	0	3	2	1
---	---	---	---	---	---



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

$i (=3) < 6 \rightarrow \text{true}$

list

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

result

0	0	0	3	2	1
---	---	---	---	---	---

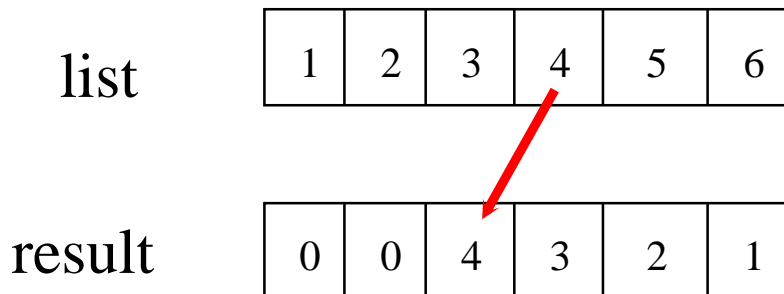


# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
         i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 3 và j = 2  
Gán list[i] cho result[j]



# Ví dụ

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

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
        i < list.length; i++, j--) {
        result[j] = list[i];
    }

    return result;
}
```

i = 4 và j = 1

list

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

result

0	0	4	3	2	1
---	---	---	---	---	---



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

$i (=4) < 6 \rightarrow \text{true}$

list

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

result

0	0	4	3	2	1
---	---	---	---	---	---



# Ví dụ

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

```
int[] list2 = reverse(list1);
```

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];
```

```
    for (int i = 0, j = result.length - 1;
```

```
        i < list.length; i++, j--) {
```

```
        result[j] = list[i];
```

```
    }
```

```
    return result;
```

```
}
```

i = 4 và j = 1  
Gán list[i] cho result[j]

list

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

result

0	5	4	3	2	1
---	---	---	---	---	---





# Ví dụ

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

```
public static int[] reverse(int[] list) {
    int[] result = new int[list.length];

    for (int i = 0, j = result.length - 1;
        i < list.length; i++, j--) {
        result[j] = list[i];
    }

    return result;
}
```

i = 5 và j = 0

list

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

result

0	5	4	3	2	1
---	---	---	---	---	---



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

$i (=5) < 6 \rightarrow \text{true}$

list

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

result

0	5	4	3	2	1
---	---	---	---	---	---

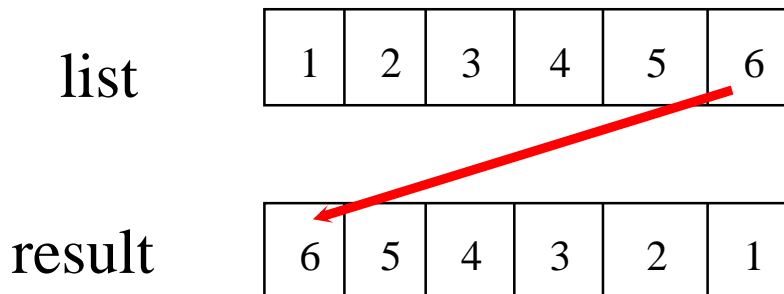


# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
         i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 5 và j = 0  
Gán list[i] cho result[j]



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

i = 6 và j = -1

list

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

result

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



# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

$i (=6) < 6 \rightarrow \text{false}$ . Kết thúc vòng lặp.

list

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

result

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

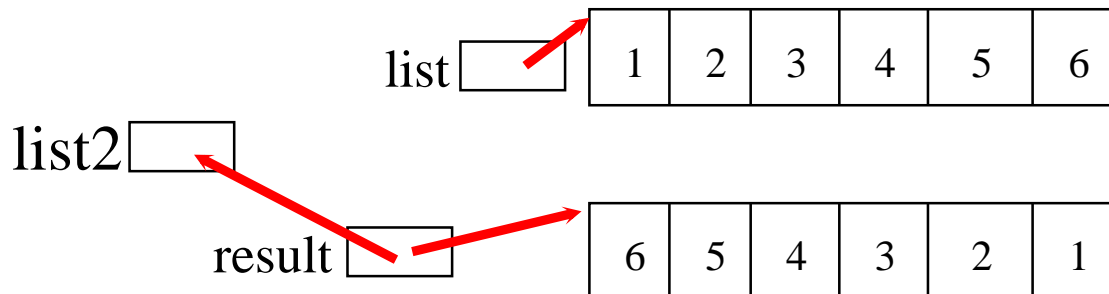


# Ví dụ

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

```
public static int[] reverse(int[] list) {  
    int[] result = new int[list.length];  
  
    for (int i = 0, j = result.length - 1;  
        i < list.length; i++, j--) {  
        result[j] = list[i];  
    }  
  
    return result;  
}
```

Trả về result



# Tìm kiếm trong mảng

Tìm kiếm là quá trình tìm một phần tử cụ thể trong một mảng.

Có nhiều thuật toán tìm kiếm. Ở đây, chúng ta sẽ tìm hiểu 2 thuật toán *tìm kiếm tuyến tính (linear search)* và *tìm kiếm nhị phân (binary search)*.

```
public class LinearSearch {  
    /** The method for finding a key in the list */  
    public static int linearSearch(int[] list, int key) {  
        for (int i = 0; i < list.length; i++)  
            if (key == list[i])  
                return i;  
        return -1;  
    }  
}
```

[0] [1] [2] ...  
list 

--	--	--	--	--	--

  
key Compare key with list[i] for i = 0, 1, ...

# Tìm kiếm tuyến tính

Key

List

3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8





# Tìm kiếm tuyến tính

```
/** The method for finding a key in the list */  
public static int linearSearch(int[] list, int key) {  
    for (int i = 0; i < list.length; i++)  
        if (key == list[i])  
            return i;  
    return -1;  
}
```

Ví dụ:

```
int[] list = {1, 4, 4, 2, 5, -3, 6, 2};  
int i = linearSearch(list, 4); // returns 1  
int j = linearSearch(list, -4); // returns -1  
int k = linearSearch(list, -3); // returns 5
```



# Tìm kiếm nhị phân

Key

List

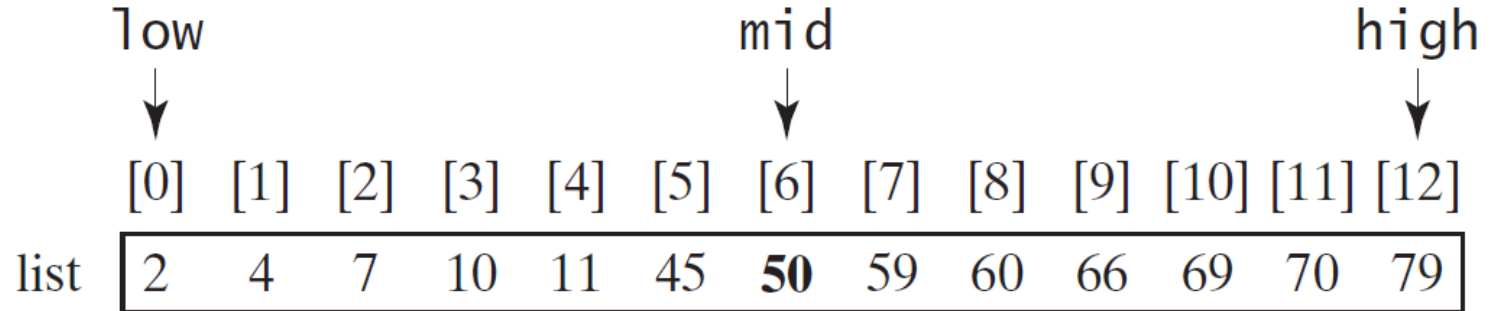
8	1	2	3	4	6	7	8	9
8	1	2	3	4	6	7	8	9
8	1	2	3	4	6	7	8	9



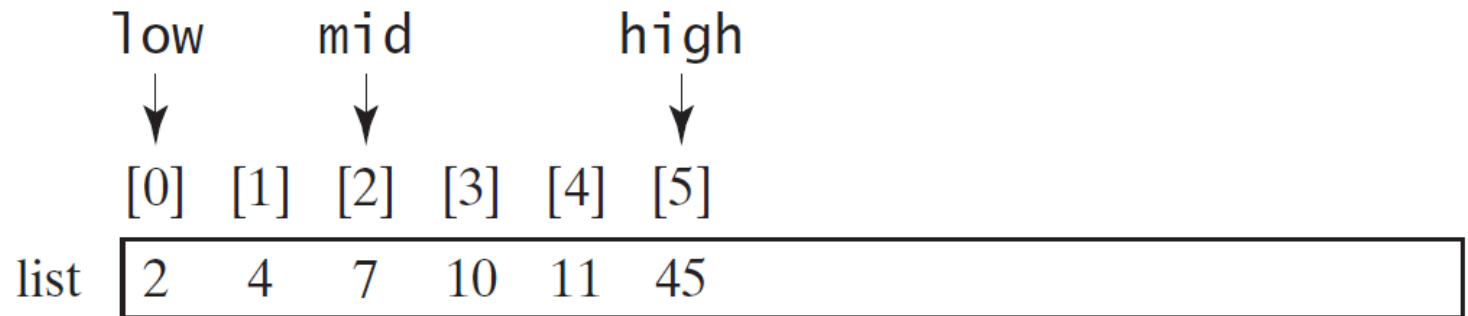
# Tìm kiếm nhị phân

key is 11

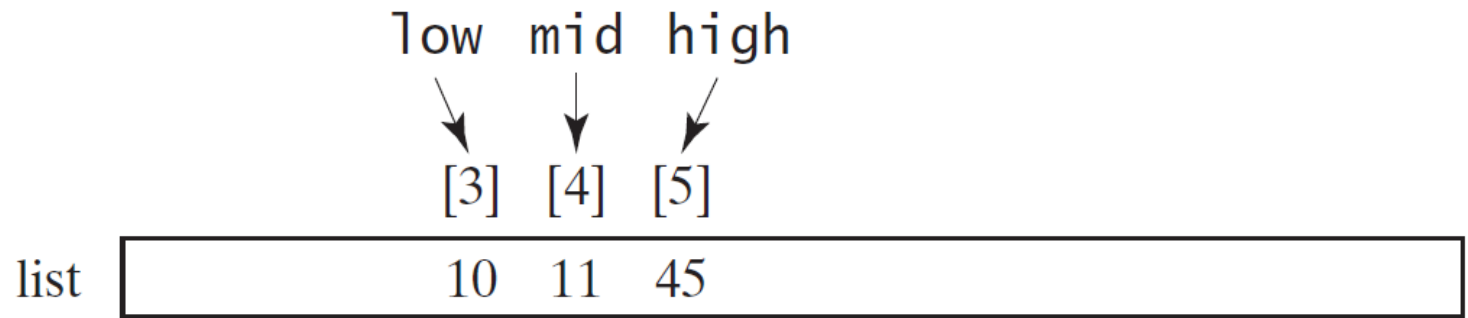
key < 50



key > 7



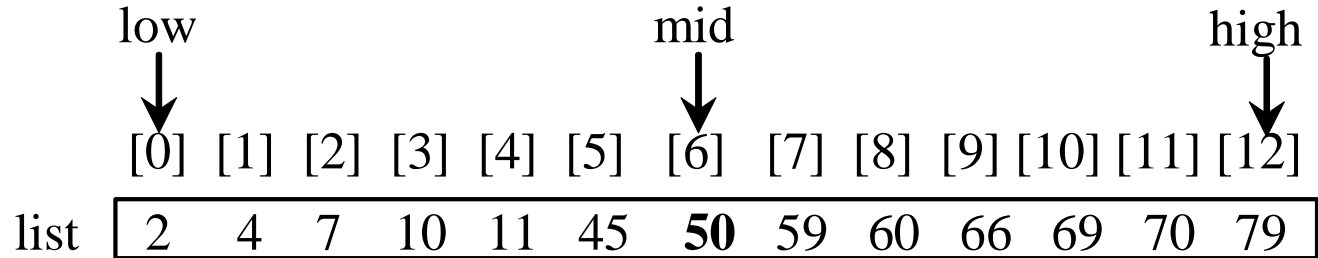
key == 11



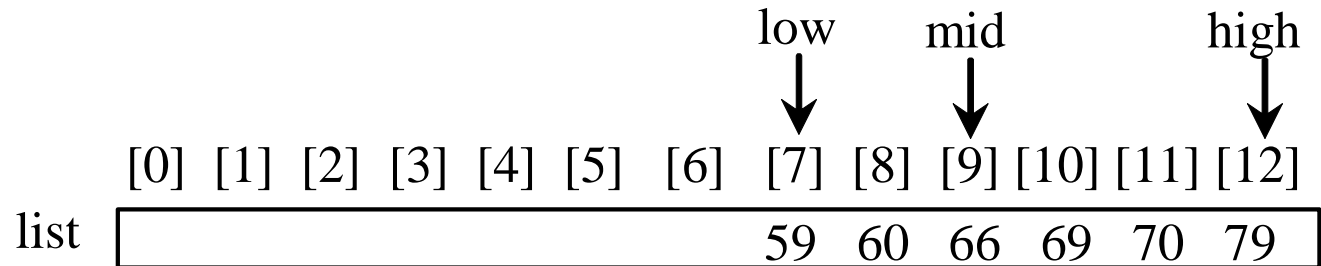
# Tìm kiếm nhị phân

key is 54

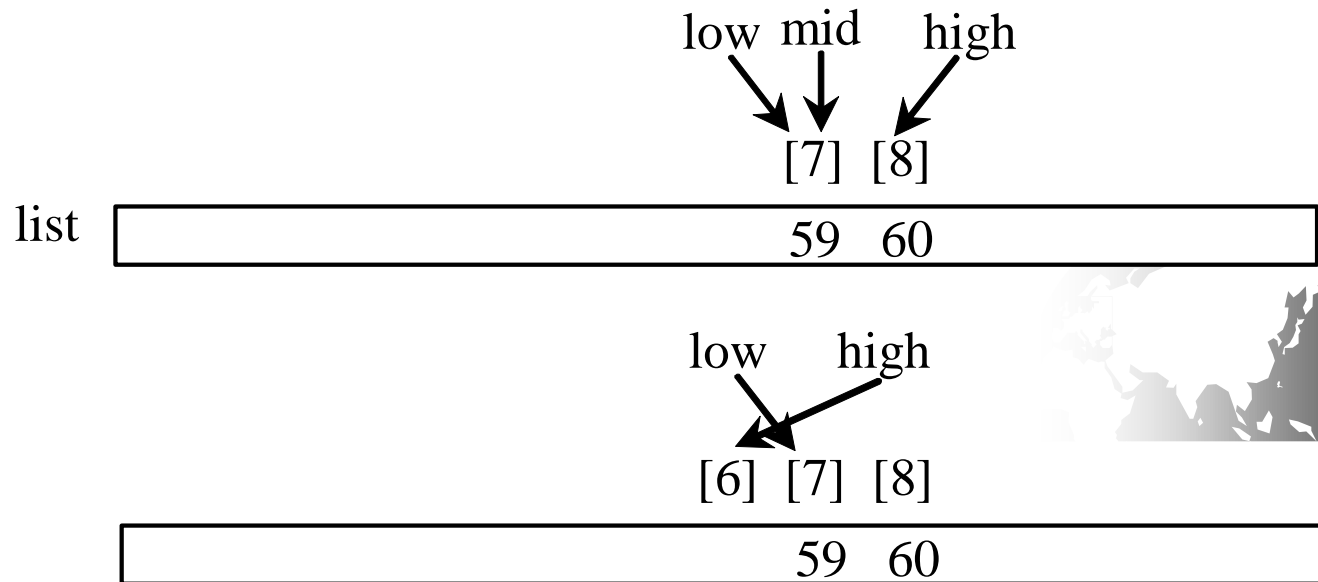
key > 50



key < 66



key < 59



# Tìm kiếm nhị phân

Phương thức `binarySearch` trả về chỉ số của phần tử được tìm thấy trong danh sách.

Nếu không tìm thấy sẽ trả về

-**insertion point** - 1.

Với **insertion point** là điểm mà phần tử cần tìm sẽ được chèn vào danh sách.



# Tìm kiếm nhị phân

```
/** Use binary search to find the key in the list */
public static int binarySearch(int[] list, int key) {
    int low = 0;
    int high = list.length - 1;

    while (high >= low) {
        int mid = (low + high) / 2;
        if (key < list[mid])
            high = mid - 1;
        else if (key == list[mid])
            return mid;
        else
            low = mid + 1;
    }

    return -1 - low;
}
```



# Phương thức `Arrays.binarySearch`

Java cung cấp nhiều nạp chồng phương thức `binarySearch` để tìm kiếm một giá trị trong mảng `int`, `double`, `char`, `short`, `long`, and `float`.

Phương thức `binarySearch` nằm trong lớp `java.util.Arrays`.

```
int[] list = {2, 4, 7, 10, 11, 45, 50, 59, 60, 66, 69, 70, 79};
```

```
System.out.println("Index is " +  
    java.util.Arrays.binarySearch(list, 11));
```

Trả về 4

```
char[] chars = {'a', 'c', 'g', 'x', 'y', 'z'};
```

```
System.out.println("Index is " +  
    java.util.Arrays.binarySearch(chars, 't'));
```

Trả về -4 (insertion point là 3,  
nên sẽ trả về -3-1)

Cần sắp xếp mảng theo thứ tự tăng dần trước khi sử dụng phương thức `binarySearch`.

# Phương thức `Arrays.sort`

Java cung cấp nhiều nạp chồng phương thức `binarySearch` để tìm kiếm một giá trị trong mảng `int`, `double`, `char`, `short`, `long`, and `float`. Phương thức `binarySearch` nằm trong lớp `java.util.Arrays`.

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

```
char[] chars = {'a', 'A', '4', 'F', 'D', 'P'};  
java.util.Arrays.sort(chars);
```

Java 8 cung cấp phương thức `Arrays.parallelSort(list)` để tìm kiếm nhanh hơn.





# Phương thức `Arrays.toString(list)`

Phương thức `Arrays.toString(list)` được sử dụng để trả về 1 chuỗi đại diện cho danh sách (list)

