

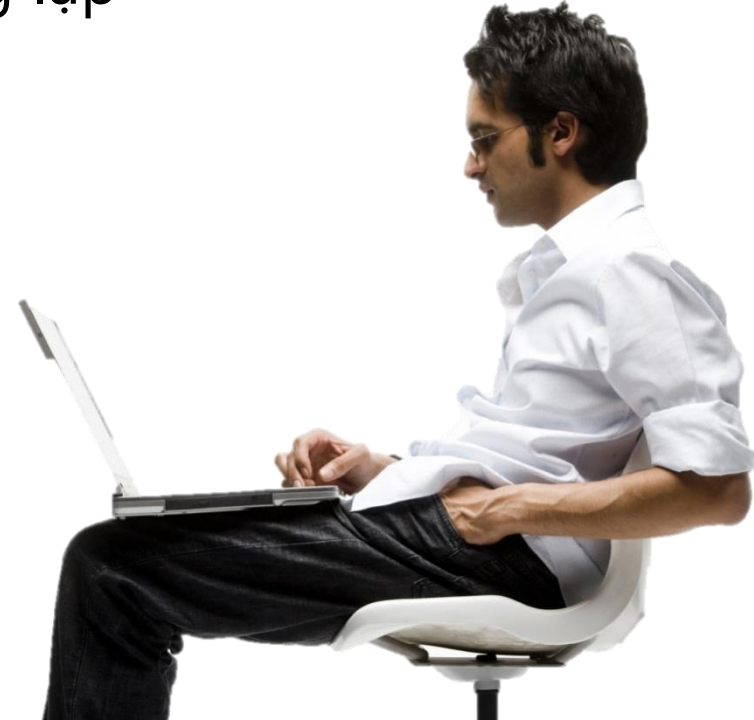


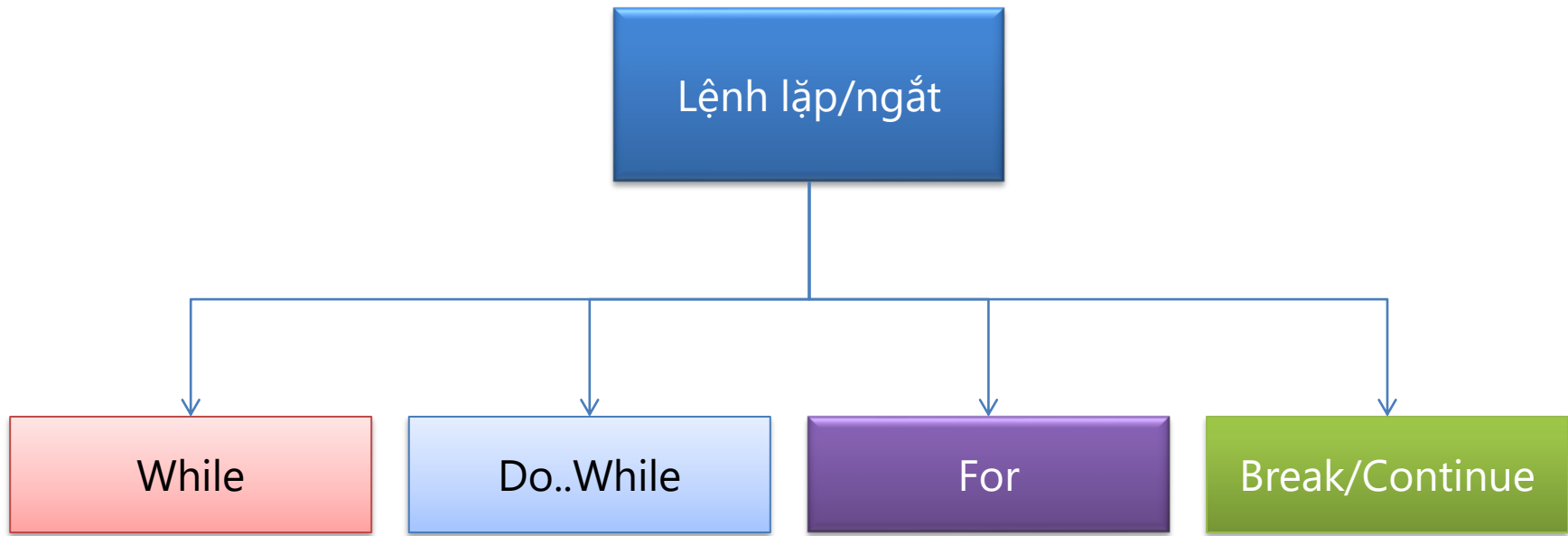
LẬP TRÌNH JAVA 1

BÀI 3: MẢNG VÀ LỆNH LẶP

PHẦN 1

- ❑ Kết thúc bài học này bạn có khả năng
 - ❖ Hiểu cấu trúc lệnh lặp và sử dụng các lệnh lặp
 - While
 - Do...while
 - For
 - ❖ Hiểu và áp dụng lệnh ngắt vòng lặp
 - Break
 - Continue
 - ❖ Hiểu và sử dụng mảng



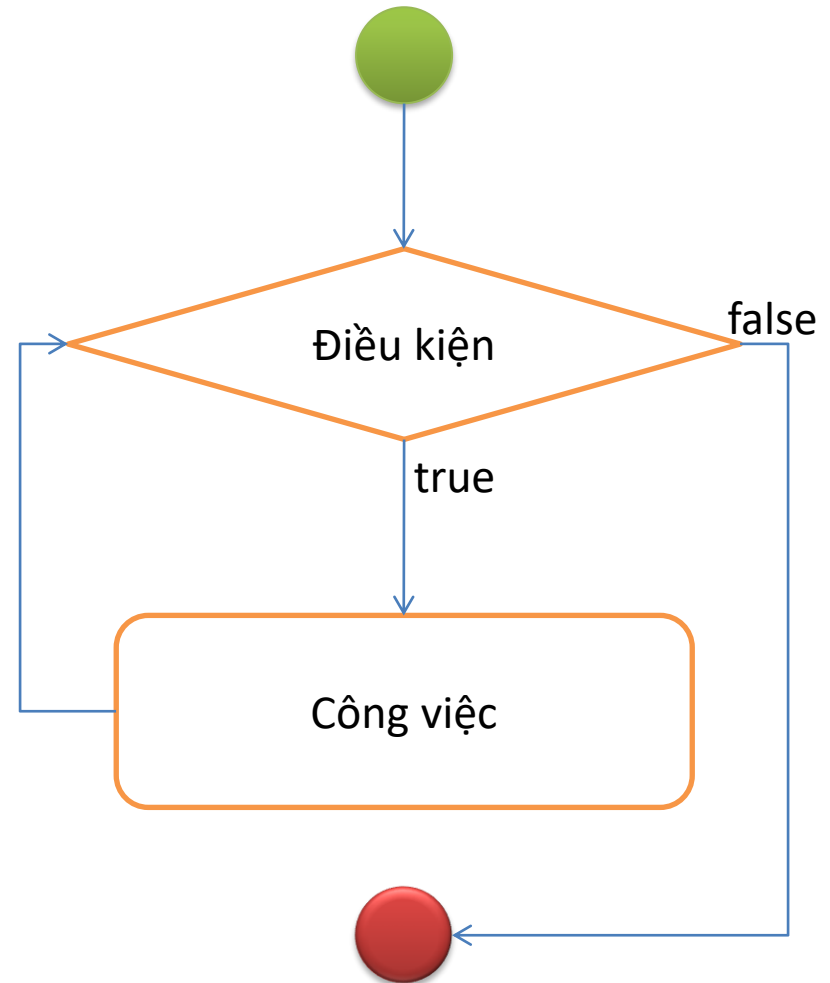


□ Cú pháp

```
while (<<điều kiện>>) {  
    // công việc  
}
```

□ Diễn giải:

- ❖ Thực hiện công việc trong khi biểu thức điều kiện có giá trị là true.

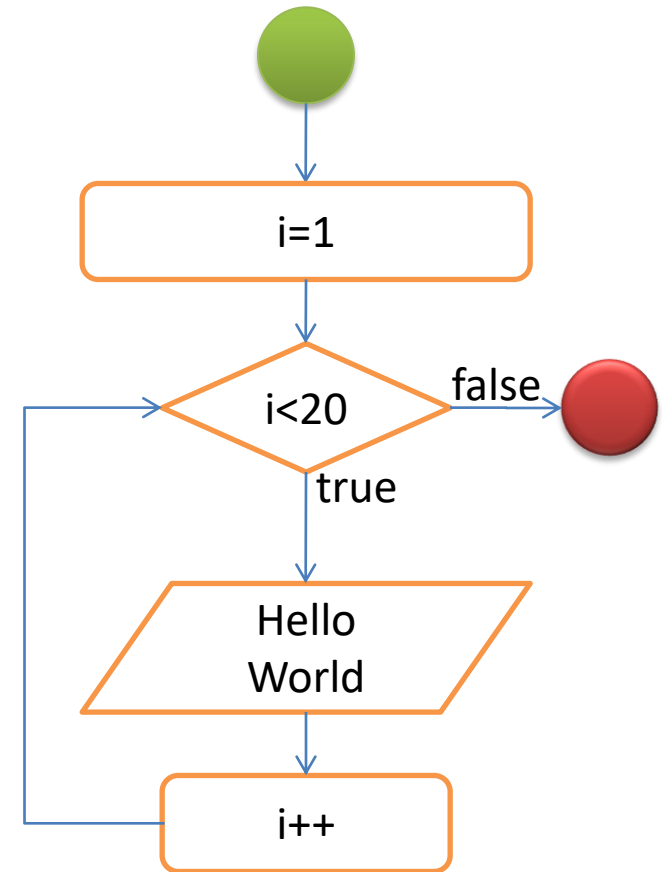


❑ Ví dụ

```
int i = 1;  
while (i < 20) {  
    System.out.println("Hello World !");  
    i++;  
}
```

❑ Diễn giải:

- ❖ Đoạn mã trên xuất 19 dòng Hello World ra màn hình





DEMO

1. Xuất bảng cửu chương 7
2. Tính trung bình cộng các số chia hết cho 3 từ 27 đến 250.



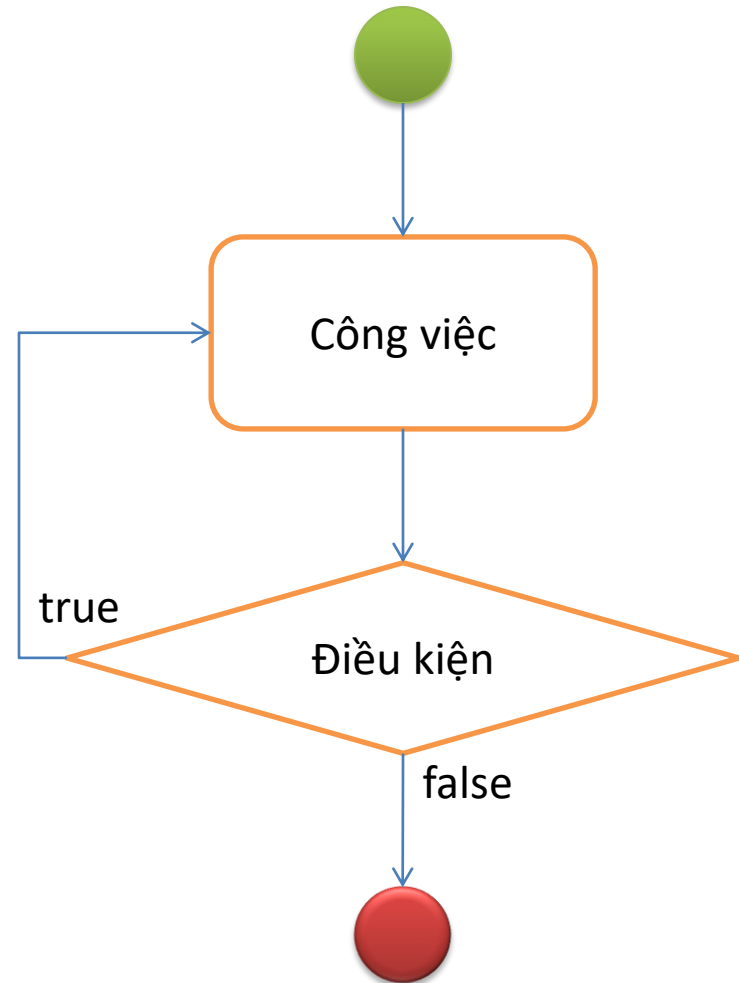
□ Cú pháp:

```
do {  
    // công việc  
}
```

```
while (<<điều kiện>>);
```

□ Diễn giải:

- ❖ Tương tự lệnh lặp while chỉ khác ở chỗ điều kiện được kiểm tra sau, nghĩa là công việc được thực hiện ít nhất 1 lần.

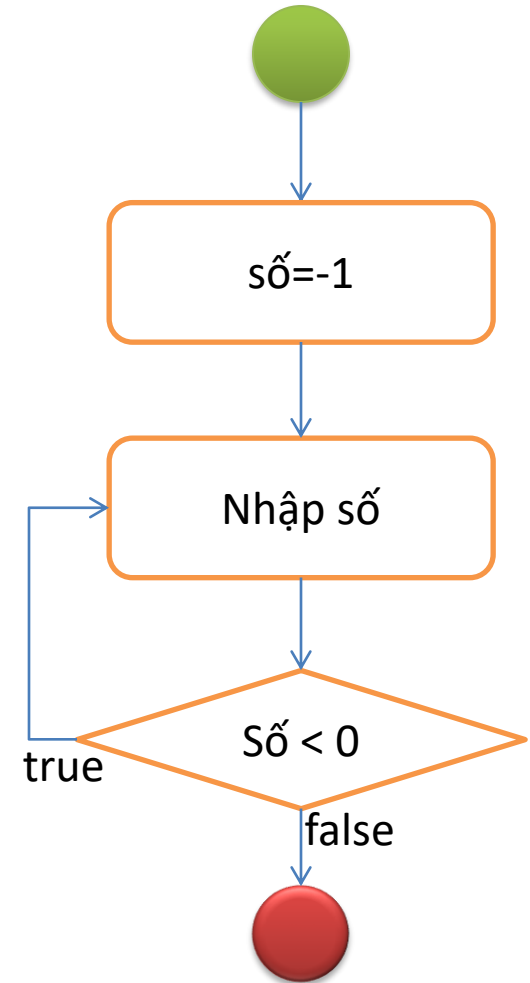


❑ Ví dụ

```
int so = -1;  
do {  
    so = scanner.nextDouble();  
}  
while (so < 0);
```

❑ Diễn giải:

- ❖ Đoạn mã trên chỉ cho phép nhập số nguyên dương từ bàn phím.





DEMO

Nhập điểm từ 0 đến 10

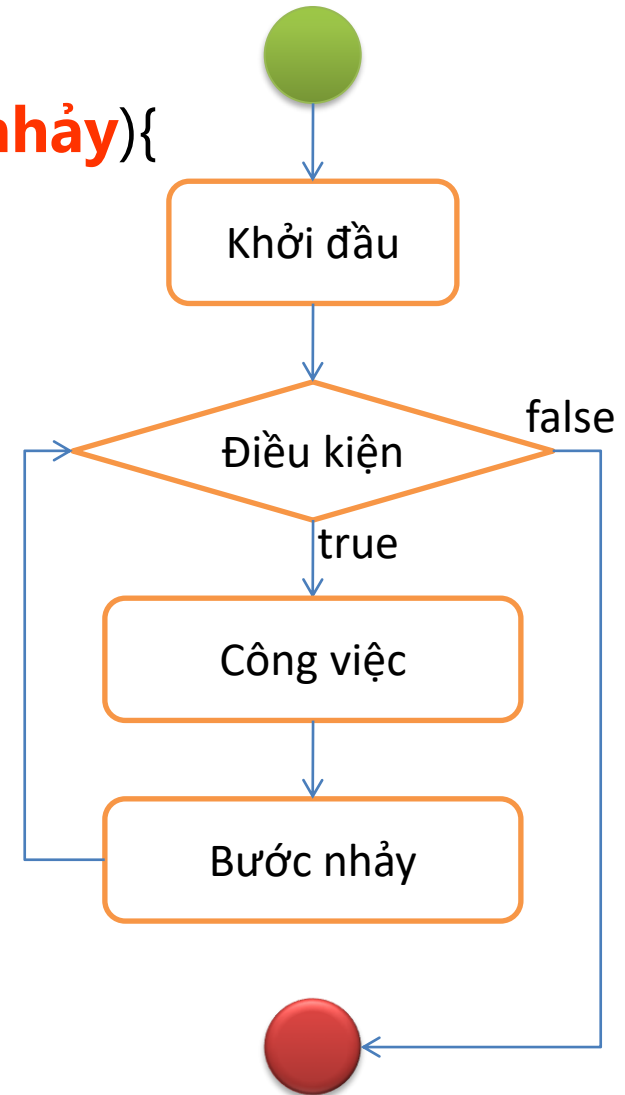


□ Cú pháp

```
for (khởi đầu ; điều kiện; bước nhảy){  
    // công việc  
}
```

□ Diễn giải

- ❖ B1: Thực hiện <<khởi đầu>>
- ❖ B2: Kiểm tra <<điều kiện>>
 - True: B3
 - False: kết thúc
- ❖ B3: Thực hiện << công việc >>
- ❖ B4: Thực hiện <<bước nhảy>>
- ❖ B5: Trở lại B2



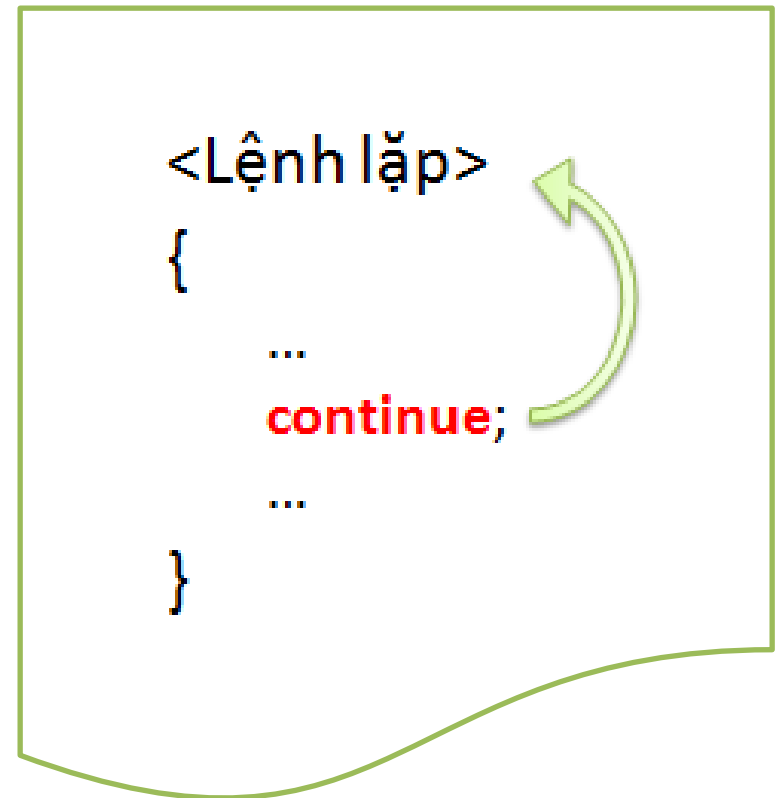
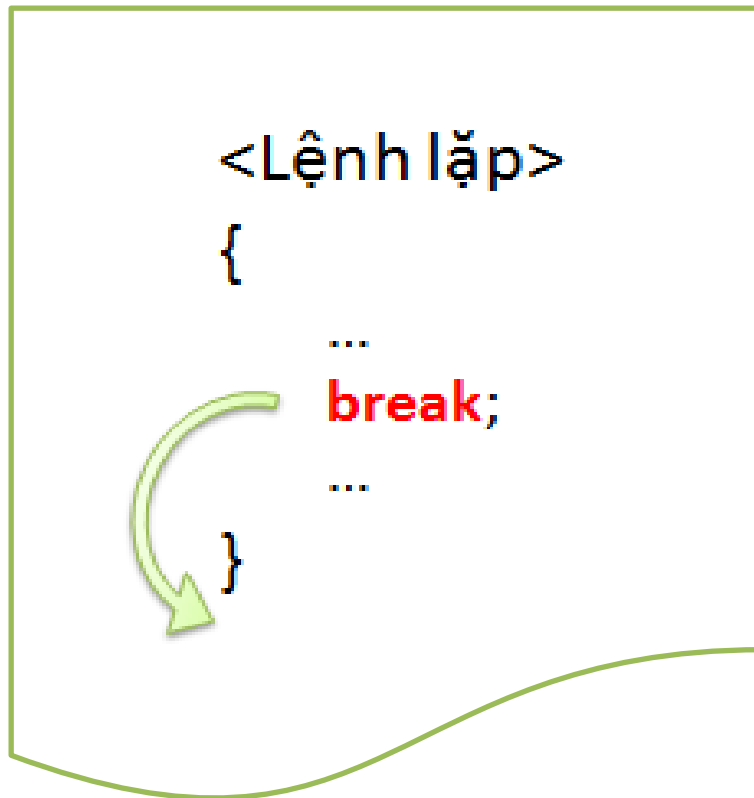


DEMO

Bảng cửu chương với lệnh lặp for



- ❑ **break** dùng để ngắt lệnh lặp
- ❑ **continue** dùng để thực hiện lần lặp tiếp theo ngay lập tức



□ Ví dụ:

```
int diem = 0;
while(true){
    diem = scanner.nextInt();
    if(diem >= 0 && diem <=10){
        break;
    }
    System.out.println("Điểm phải từ 0 đến 10");
}
```

□ Diễn giải:

- ❖ Nhập điểm hợp lệ (từ 0 đến 10)

- ☐ Lab 3 – bài 1
- ☐ Lab 3 – bài 2



LẬP TRÌNH JAVA 1

BÀI 3: MẢNG VÀ LỆNH LẶP

PHẦN 2

- ❑ Mảng là cấu trúc lưu trữ nhiều phần tử có cùng kiểu dữ liệu

0	1	2	3	4	5	6	7	8	← Indices
5	7	9	1	45	1	9	9	2	← Elements

- ❑ Để truy xuất các phần tử cần biết chỉ số (index). Chỉ số được đánh từ 0.
- ❑ Các thao tác mảng
 - ❖ Khai báo
 - ❖ Truy xuất (đọc/ghi) phần tử
 - ❖ Lấy số phần tử
 - ❖ Duyệt mảng
 - ❖ Sắp xếp các phần tử mảng

❑ Khai báo không khởi tạo

- ❖ `int[] a;` *// mảng số nguyên chưa biết số phần tử*
- ❖ `int b[];` *// mảng số nguyên chưa biết số phần tử*
- ❖ `String[] c = new String[5];` *// mảng chứa 5 chuỗi*

❑ Khai báo có khởi tạo

- ❖ `double[] d1 = new double[]{2, 3, 4, 5, 6};` *// mảng số thực, 5 phần tử, đã được khởi tạo*
- ❖ `double[] d2 = {2, 3, 4, 5, 6};` *// mảng số thực, 5 phần tử, đã được khởi tạo*

❑ Sử dụng chỉ số (**index**) để phân biệt các phần tử.
Chỉ số mảng tính từ 0.

❖ `int a[] = {4, 3, 5, 7};`

❖ `a[2] = a[1] * 4; // 4*4=16`

❖ Sau phép gán này mảng là {4, 3, 16, 7};

❑ Sử dụng thuộc tính **length** để lấy số phần tử của mảng

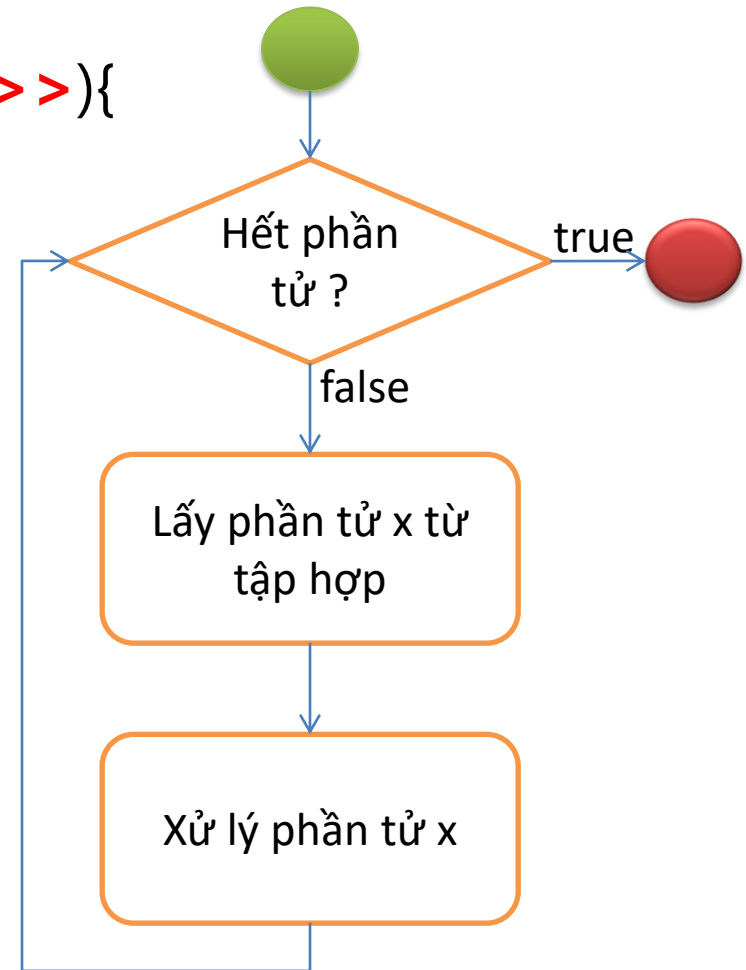
❖ `a.length` có giá trị là 4

□ Cú pháp

```
for (<<kiểu>> x : <<tập hợp>>){  
    // Xử lý phần tử x  
}
```

□ Diễn giải:

- ❖ For each được sử dụng để duyệt tập hợp. Mỗi lần lấy 1 phần tử từ tập hợp và xử lý phần tử đó.



- ❑ 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);  
}
```

❑ Ví dụ sau tính tổng các số chẵn của mảng.

- ❖ Lấy từng phần tử từ mảng với for-each
- ❖ Nếu là số chẵn thì cộng vào tổng

```
int[] a = {9, 3, 8, 7, 3, 9, 4, 2};  
  
double tong = 0;  
for(int x : a){  
    if(x % 2 == 0){  
        tong += x;  
    }  
}  
  
System.out.print("Tổng: " + tong);
```



DEMO

Nhập mảng số nguyên

+ Tính và xuất trung bình cộng

+ Xuất lập phương các phần tử



THAO TÁC MẢNG NÂNG CAO

```
int[] a = {9, 3, 8, 7, 3, 9, 4, 2};
```

```
System.out.println("Mảng gốc: " + Arrays.toString(a));
```

[9, 3, 8, 7, 3, 9, 4, 2]

```
Arrays.sort(a);
```

```
System.out.println("Sau sort: " + Arrays.toString(a));
```

[2, 3, 3, 4, 7, 8, 9, 9]

```
int i = Arrays.binarySearch(a, 8);
```

```
System.out.println("Vị trí của 8 là " + i);
```

Vị trí của 8 là 5

```
Arrays.fill(a, 0);
```

```
System.out.println("Sau fill: " + Arrays.toString(a));
```

[0, 0, 0, 0, 0, 0, 0, 0]

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]

```
Int[] a = {1, 9, 2, 8, 3, 7, 4, 6, 5};
```

Phương thức	Mô tả/ví dụ
<code><T> List<T> asList(T... a)</code>	Chuyển một mảng sang List với kiểu tương ứng. Ví dụ: List<Integer> b = Arrays.asList(a);
<code>int binarySearch(Object[] a, Object key)</code>	Tìm vị trí xuất hiện đầu tiên của một phần tử trong mảng. Ví dụ: int i = Arrays.binarySearch(a, 8);
<code>void sort(Object[] a)</code>	Sắp xếp các phần tử theo thứ tự tăng dần. Ví dụ: Arrays.sort(a);
<code>String toString(Object[] a)</code>	Chuyển mảng thành chuỗi được bọc giữ cặp dấu [] và các phần tử mảng cách nhau dấu phẩy. Ví dụ: String s = Arrays.toString(a);
<code>void fill(Object[] a, Object val)</code>	Gán 1 giá trị cho tất cả các phần tử mảng. Ví dụ: Arrays.fill(a, 9);



DEMO

Nhập mảng 5 SV và xuất tăng
dần theo alphabet



- ❑ Arrays.sort(mảng) không thể thực hiện
 - ❖ Sắp xếp giảm
 - ❖ Các kiểu không so sánh được
- ❑ Giải pháp: tự xây dựng thuật toán sắp xếp

```
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



□ Loop

- ❖ While
- ❖ Do...while
- ❖ For(;điều kiện;)
- ❖ For(phần tử: tập hợp)

□ Ngắt

- ❖ Break
- ❖ Continue

□ Mảng



- ☐ Lab 3 – bài 3
- ☐ Lab 3 – bài 4
- ☐ Lab 3 – bài 5 (giảng viên cho thêm)