

Bài 04

CÁC CẤU TRÚC ĐIỀU KHIỂN

CHƯƠNG TRÌNH TRONG JAVA

Giảng Viên: ThS. Giang Hào Côn

4.1/ Cấu trúc rẽ nhánh (Selection constructs)

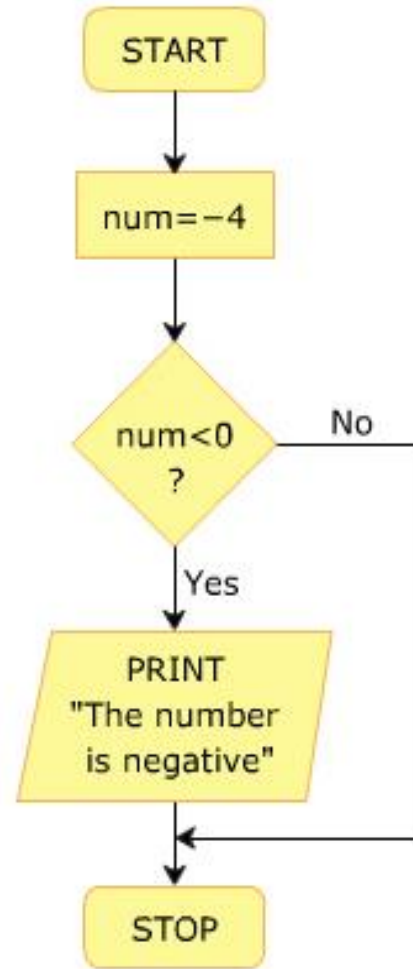
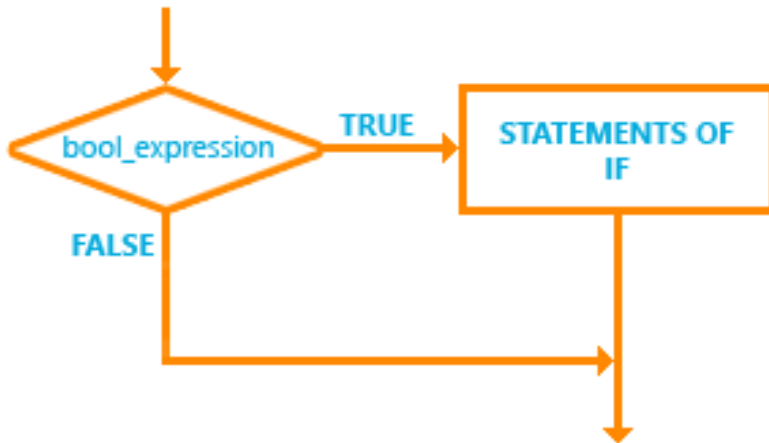
- Mặc nhiên, các lệnh trong chương trình được thực hiện một cách tuần tự, lên tục Cho đến khi hết lệnh thì kết thúc.
- Để chương trình xử lý “**thông minh hơn**”, ta có thể ứng dụng các cấu trúc rẽ nhánh trong chương trình để quyết định **khi nào thì hành tập lệnh này** ... và **khi nào thì hành tập lệnh khác** ...
- Ngôn ngữ Java cung cấp 2 dạng cấu trúc rẽ nhánh ở dạng:
 - ❖ **if ...**
 - ❖ **switch ... case**

4.2/ Cấu trúc rẽ nhánh if (if statement)

Cú pháp

```
if(conditional) {  
    statement1;  
    statement2;  
}
```

Mô tả thực thi



```
class Main {  
    public static void main(String[] args) {  
        int number = 10;  
        // checks if number is less than 0  
        if (number < 0) {  
            System.out.println("The number is negative.");  
        }  
        System.out.println("Statement outside if block");  
    }  
}
```

Kết quả

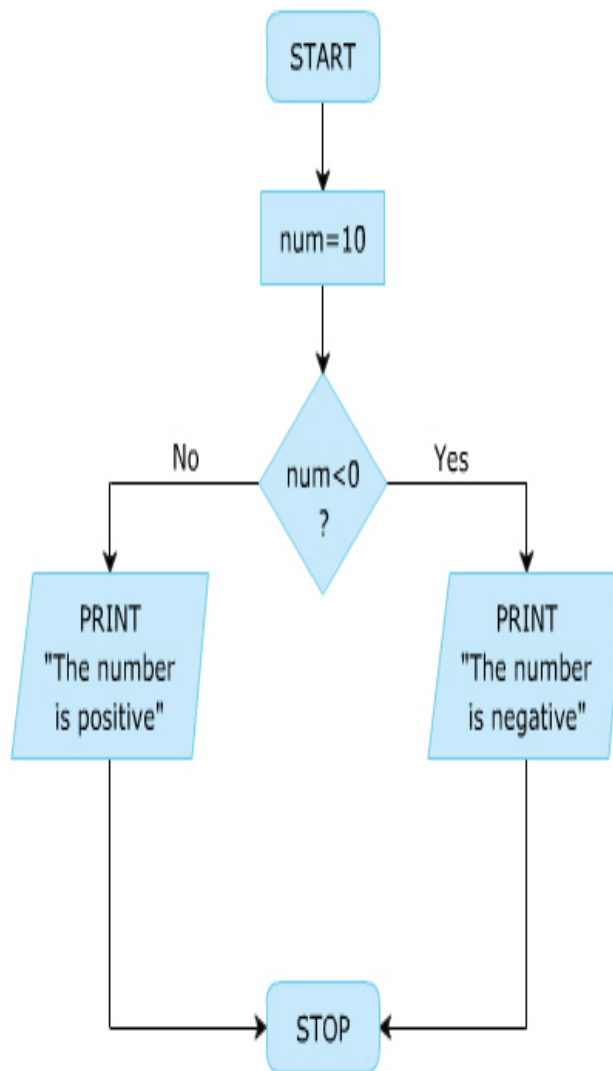
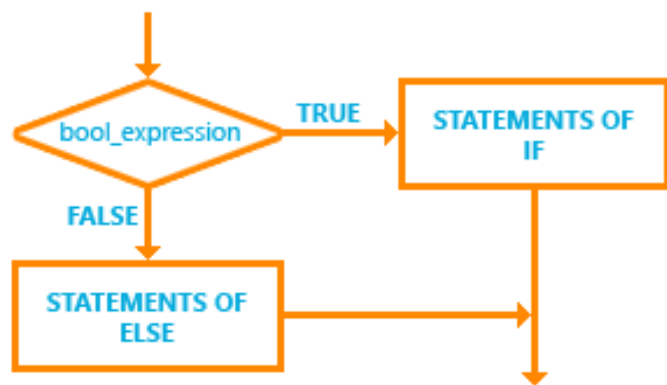
Statement outside if block

4.2/ Cấu trúc rẽ nhánh if ... else

Cú pháp

```
if(conditional) {
    statement1;
    statement2;
}
else {
    statement3;
    statement4;
}
```

Mô tả thực thi



```
class Main {
    public static void main(String[] args) {
        int number = 10;
        // checks if number is greater than 0
        if (number > 0) {
            System.out.println("The number is positive.");
        }
        // execute this block
        // if number is not greater than 0
        else {
            System.out.println("The number is not positive.");
        }
        System.out.println("Statement outside if...else block");
    }
}
```

Kết quả

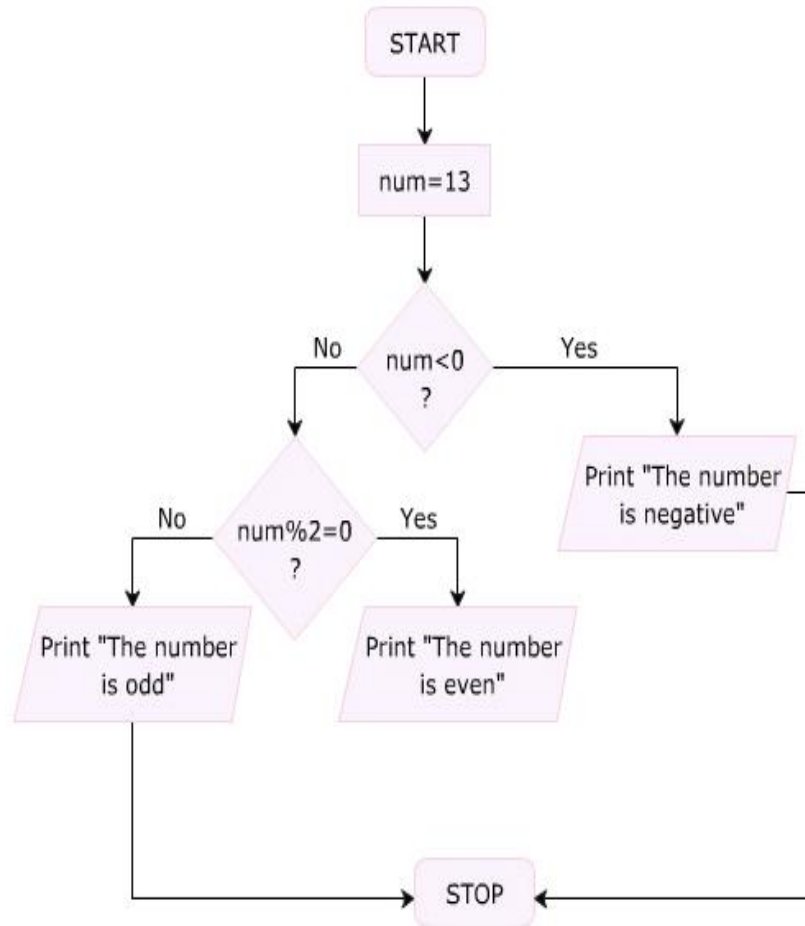
```
The number is positive.
Statement outside if...else block
```

4.2/ Cấu trúc rẽ nhánh if ... else if ...

Cú pháp

```
if (btdk)
{
    ...
}
else if (btdk)
{
    ...
}
else
{
    ...
}
```

Mô tả thực thi



Ví dụ

```
class Main {
    public static void main(String[] args) {
        int number = 0;

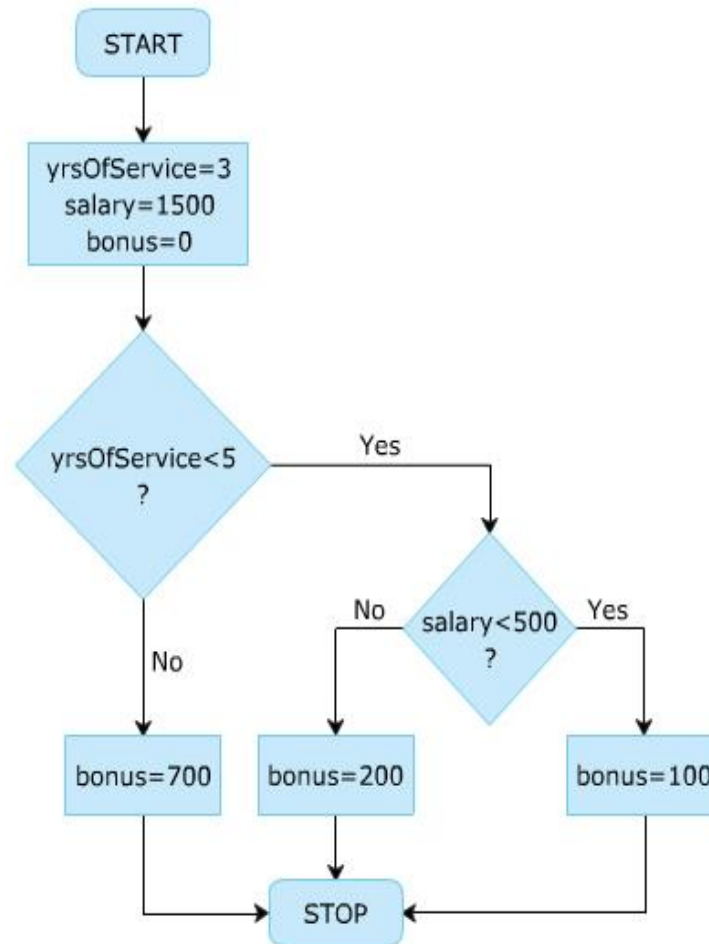
        // checks if number is greater than 0
        if (number > 0) {
            System.out.println("The number is positive.");
        }
        // checks if number is less than 0
        else if (number < 0) {
            System.out.println("The number is negative.");
        }
        // if both condition is false
        else {
            System.out.println("The number is 0.");
        }
    }
}
```

4.2/ Cấu trúc rẽ nhánh if lồng nhau

Cú pháp

```
if(conditional) {
    statement1;
}
else {
    if(conditional2) {
        statement2;
    }
    else {
        statement3;
    }
}
```

Mô tả thực thi



Ví dụ

```
class Main {
    public static void main(String[] args) {
        // declaring double type variables
        Double n1 = -1.0, n2 = 4.5, n3 = -5.3, largest;
        // checks if n1 is greater than or equal to n2
        if (n1 >= n2) {
            // if...else statement inside the if block
            // checks if n1 is greater than or equal to n3
            if (n1 >= n3) {
                largest = n1;
            }
            else {
                largest = n3;
            }
        } else {
            // if..else statement inside else block
            // checks if n2 is greater than or equal to n3
            if (n2 >= n3) {
                largest = n2;
            }
            else {
                largest = n3;
            }
        }
        System.out.println("Largest Number: " + largest);
    }
}
```

4.3/ Toán tử điều kiện (Conditional Operators)

Cú pháp

<Điều kiện> ? <Biểu thức 1> : <Biểu thức 2>

Ví Dụ:

```
int a=5, b=6;
```

```
String ketqua = a > b ? “số lớn là ” + a : “số lớn là: ” + b;
```

```
int a=10, b=8;
```

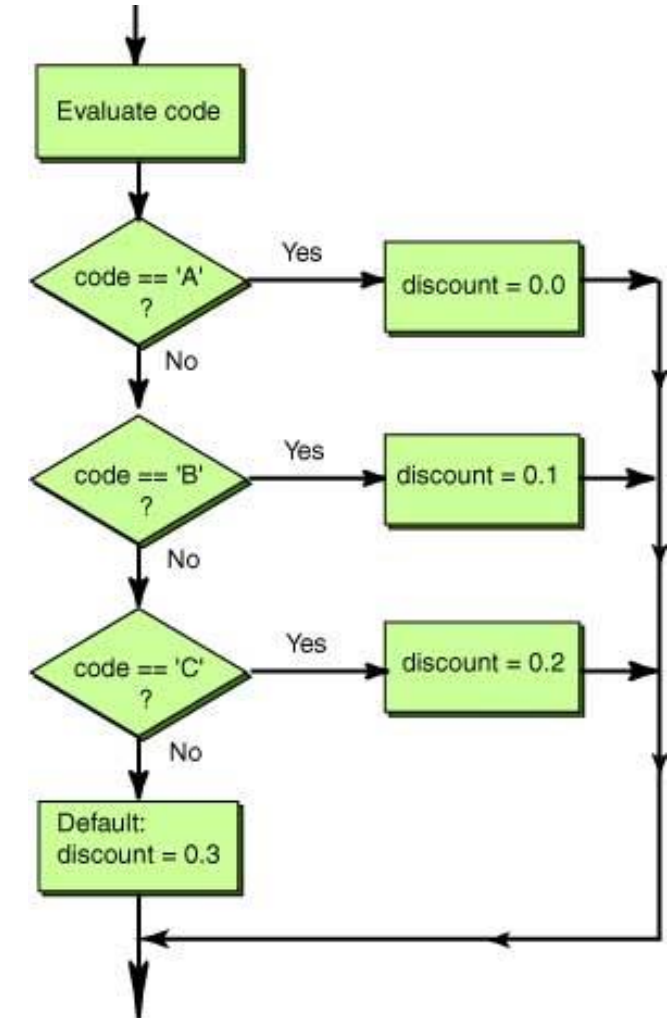
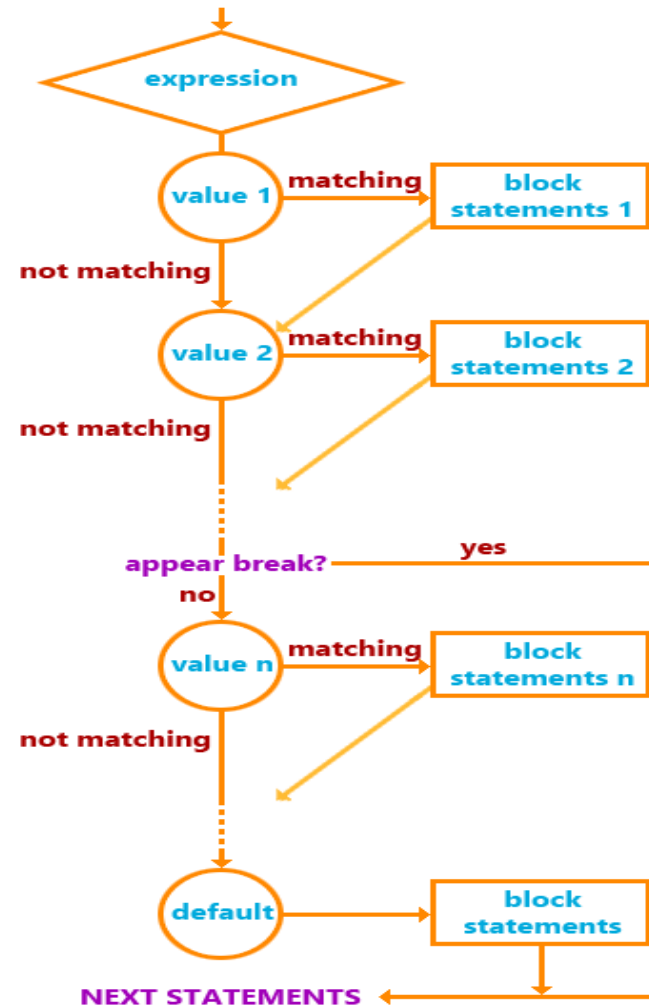
```
String ketqua = a > b ? “số lớn là ” + a : “số lớn là: ” + b;
```

4.4/ Cấu trúc switch ... case

Cú pháp

```
switch(expression) {
  case (value1):
    statement1;
    break;
  case (value2):
    statement2;
    break;
  default:
    statement4;
    break;
}
```

Mô tả thực thi



4.4/ Cấu trúc switch ... case

Ví dụ so sánh if ... else và switch case

Sử dụng cấu trúc dạng if-else

```
int month = 2;
if(month == 1)      System.out.println("January");
if(month == 2)      System.out.println("February");
if(month == 3)      System.out.println("March");
if(month == 4)      System.out.println("April");
if(month == 5)      System.out.println("May");
if(month == 6)      System.out.println("June");
if(month == 7)      System.out.println("July");
if(month == 8)      System.out.println("August");
if(month == 9)      System.out.println("September");
if(month == 10)     System.out.println("October");
if(month == 11)     System.out.println("November");
if(month == 12)     System.out.println("December");
if(month < 1 || month > 12) System.out.println("Error");
```

Sử dụng cấu trúc switch case

```
int month = 2;
switch(month) {
case 1:
    System.out.println("January"); break;
case 2:
    System.out.println("February"); break;
case 3:
    System.out.println("March"); break;
case 4:
    System.out.println("April"); break;
case 5:
    System.out.println("May"); break;
case 6:
    System.out.println("June"); break;
case 7:
    System.out.println("July"); break;
case 8:
    System.out.println("August"); break;
case 9:
    System.out.println("September"); break;
case 10:
    System.out.println("October"); break;
case 11:
    System.out.println("November"); break;
case 12:
    System.out.println("December"); break;
default:
    System.out.println("Error"); break;
}
```

4.4/ Cấu trúc switch ... case

Ví dụ 01

```
class Main {  
    public static void main(String[] args) {  
        int number = 44;  
        String size;  
        // switch statement to check size  
        switch (number) {  
            case 29:  
                size = "Small";  
                break;  
            case 42:  
                size = "Medium";  
                break;  
            // match the value of week  
            case 44:  
                size = "Large";  
                break;  
            case 48:  
                size = "Extra Large";  
                break;  
  
            default:  
                size = "Unknown";  
                break;  
        }  
        System.out.println("Size: " + size);  
    }  
}
```

Ví dụ 02

```
class Main {  
    public static void main(String[] args) {  
        int expression = 2;  
        // switch statement to check size  
        switch (expression) {  
            case 1:  
                System.out.println("Case 1");  
                // matching case  
            case 2:  
                System.out.println("Case 2");  
  
            case 3:  
                System.out.println("Case 3");  
            default:  
                System.out.println("Default case");  
        }  
    }  
}
```

Kết quả

Case 2

Case 3

Default case

4.4/ Cấu trúc switch ... case

Ví dụ 03

```
class Main {  
    public static void main(String[] args) {  
        int expression = 9;  
        switch(expression) {  
            case 2:  
                System.out.println("Small Size");  
                break;  
            case 3:  
                System.out.println("Large Size");  
                break;  
            // default case  
            default:  
                System.out.println("Unknown Size");  
        }  
    }  
}
```

Ví dụ 04

```
int abs = (a > 0) ? a : -a;
```

Thay vì sử dụng *if*

```
if (x > 0){  
    y = 1;  
}else{  
    y = -1;  
}
```

4.4/ Cấu trúc switch ... case

Ví dụ 05

Ví dụ sử dụng if:

```
if (so % 2 == 0){  
    System.out.println(so + " là số chẵn!");  
}else{  
    System.out.println(so + " là số lẻ!");  
}
```

Tương đương với:

```
System.out.println((so % 2 == 0) ? so + " là số chẵn!" : so + " là số lẻ!");
```

4.5/ Cấu trúc lặp (loop)

- Khác với cấu trúc rẽ nhánh, cấu trúc lặp được sử dụng trong chương trình để cho phép thực hiện lệnh (*Hay nhóm lệnh*) theo cách “**Lặp đi, lặp lại**”. Tức là những lệnh cùng phục vụ cho một mục đích xử lý nào đó trong chương trình sẽ được “**thi hành nhiều lần**” theo một “**chu kỳ xác định**”. Nhờ đó, việc thể hiện mã lệnh trong chương trình dựa trên cấu trúc lặp sẽ ngắn gọn, rõ ràng hơn.
- Có 3 cấu trúc lặp có thể sử dụng cho việc biểu diễn mã lệnh trong chương trình
 - ☐ **for** construct
 - ☐ **while** ... construct
 - ☐ **do ... while** construct

4.5.1/ Cấu trúc For

Vòng lặp **for** được sử dụng để chạy một khối code trong một số lần nhất định. Cú pháp:

```
for (<khởi đầu>; <điều kiện lặp>; <bước nhảy>) {  
    //code block  
}
```

Nguyên lý hoạt động của vòng lặp for như sau:

- **<khởi đầu>** khởi tạo giá trị của biến và chỉ thực thi một lần duy nhất. Sau đó, nếu **<điều kiện lặp>** đúng (true) thì thực thi các dòng lệnh trong vòng lặp for. Nếu **<điều kiện lặp>** sai (false) thì kết thúc vòng lặp.
- **<bước nhảy>** sẽ thay đổi giá trị của biến lúc **<khởi đầu>**. Giá trị của biến này sẽ được kiểm tra lại (kiểm tra **<điều kiện lặp>**) sau mỗi lần lặp.

4.5.1/ Cấu trúc For

Ví dụ bên dưới sẽ in ra “**Java is fun**” 5 lần

```
class Main {  
    public static void main(String[] args) {  
        int n = 5;  
        // for loop  
        for (int i = 1; i <= n; ++i) {  
            System.out.println("Java is fun");  
        }  
    }  
}
```

Kết quả

```
Java is fun  
Java is fun  
Java is fun  
Java is fun  
Java is fun
```

4.5.1/ Cấu trúc For

Giá trị khởi tạo cho vòng lặp for

<khởi đầu> trong vòng lặp for không hẳn lúc nào cũng là **int i = 0**, có thể thay đổi để phù hợp với yêu cầu hoặc quy ước trong lập trình.

```
class Main {  
    public static void main(String[] args) {  
        int n = 5;  
        // for loop  
        for (int i = 0; i < n; ++i) {  
            System.out.println("Java is fun");  
        }  
    }  
}
```

Trong chương trình hình bên, giá trị khởi tạo là **i=0**.

4.5.1/ Cấu trúc For

Vòng lặp for vô hạn

- Lưu ý **<điều kiện lặp>** để tránh trường hợp vòng lặp vô hạn.

```
class Main {  
    public static void main(String[] args) {  
        int sum = 0;  
        for (int i = 1; i <= 10; --i) {  
            System.out.println("Hello");  
        }  
    }  
}
```

<điều kiện lặp> ở ví dụ trên là $i \leq 10$, **<khởi đầu>** là $i = 1$, **<bước nhảy>** là $-i$ thì **<điều kiện lặp>** sẽ không bao giờ **false** và vòng lặp for sẽ lặp vô hạn.

4.5.1/ Cấu trúc For

Vòng lặp for each trong Java.

Vòng lặp for trong Java có một cú pháp thay thế giúp dễ dàng duyệt qua các mảng (array) và tập hợp (collection). Đó là vòng lặp for-each. **Cú pháp:**

Trong đó,

- **array** là một mảng hay một tập hợp.
- **item** là mỗi phần tử trong mảng hay tập hợp
- **dataType** là kiểu dữ liệu của các phần tử trong mảng hay tập hợp

```
for(dataType item : array) {  
    ...  
}
```

4.5.1/ Cấu trúc For

Vòng lặp for each trong Java.

Vòng lặp for trong Java có một cú pháp thay thế giúp dễ dàng duyệt qua các mảng (array) và tập hợp (collection). Đó là vòng lặp **for-each**. Ví dụ:

```
class Main {  
    public static void main(String[] args) {  
        // create an array  
        int[] numbers = {3, 9, 5, -5};  
  
        // for each loop  
        for (int number: numbers) {  
            System.out.println(number);  
        }  
    }  
}
```

Chương trình trên sử dụng for-each để duyệt mảng. Ở lần lặp thứ 1 thì item là 3, lần lặp thứ 2 thì item là 9, lần lặp thứ 3 thì item là 5, lần lặp thứ 4 thì item là -5.

Lưu ý: Vòng lặp for và for-each cho kết quả duyệt mảng như nhau, nhưng for-each ngắn gọn và dễ hiểu hơn.

4.5.1/ Cấu trúc For

Vòng lặp for lồng nhau.

Trong những tình huống đặc biệt, chúng ta cần sử dụng các **cấu trúc lặp lồng nhau** (*Nested loops*), quá trình thi hành sẽ được thực hiện như sau

- Ứng với mỗi lần lặp của cấu trúc **for** cấp 1 thì toàn bộ cấu trúc **for** chứa trong nó sẽ được gọi thi hành
- Số lần lặp được xác định là tích của số lần lặp cấp 1 nhân với số lần lặp của cấu trúc cấp 2

4.5.1/ Cấu trúc For

Vòng lặp for lồng nhau.

Ví dụ: ta có cấu trúc for lồng 2 cấp như minh họa sau

```
int s=0;
for( int i =1; i<=10; i++)           (Cấp 1)
{
    for (int j=1; j<=20; j++)         (Cấp 2)
        s +=5;
    printf("Hết lần lặp thứ %d, s = %d", i, s);
}
```

4.5.2/Cấu trúc while

Vòng lặp **while** trong Java được sử dụng để lặp lại một khối lệnh nếu thỏa điều kiện nào đó. **Cú pháp của vòng lặp while như sau:**

```
while (<điều kiện lặp>)  
{  
    //code block  
}
```

với **<điều kiện lặp>** thường là biểu thức với các toán tử quan hệ, kết quả trả về là **true (0)** hoặc **false (0)**.

Nguyên lý hoạt động của vòng lặp while như sau:

- Kiểm tra **<điều kiện lặp>**, nếu **<điều kiện lặp> đúng (true)** thì dòng lệnh trong while sẽ được thực thi.
- Quá trình kiểm tra **<điều kiện lặp>** và thực thi dòng lệnh trong **while** sẽ chấm dứt cho đến khi **<điều kiện lặp> sai (false)**. Tức là, nếu bất cứ khi nào **<điều kiện lặp> sai (false)** thì vòng lặp **while** sẽ chấm dứt.

4.5.2/Cấu trúc while

Ví dụ:

```
class Main {  
    public static void main(String[] args) {  
        // declare variables  
        int i = 1, n = 5;  
        // while loop from 1 to 5  
        while(i <= n) {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

Kết quả

1
2
3
4
5

4.5.3/ Cấu trúc do ... while

Vòng lặp **do while** cũng giống vòng lặp while nhưng **khởi lệnh trong vòng lặp thực hiện trước** rồi mới **kiểm tra điều kiện lặp sau**. Cú pháp:

```
do {  
    //code block  
}while (<điều kiện lặp>;
```

với **<điều kiện lặp>** thường là biểu thức với các toán tử quan hệ, kết quả trả về là **true (0)** hoặc **false (0)**.

Nguyên lý hoạt động của vòng lặp do while như sau:

- Thực thi dòng lệnh trong **do while** trước. Sau đó, kiểm tra **<điều kiện lặp>**, nếu **<điều kiện lặp>** đúng (**true**) thì thực thi dòng lệnh trong **do while** lần nữa.
- Quá trình kiểm tra **<điều kiện lặp>** và thực thi dòng lệnh trong **do while** sẽ chấm dứt cho đến khi **<điều kiện lặp>** sai (**false**). Tức là, nếu bất cứ khi nào **<điều kiện lặp>** sai (**false**) thì vòng lặp **do while** sẽ chấm dứt.

4.5.3/ Cấu trúc do ... while

Ví dụ:

```
class Main {  
    public static void main(String[] args) {  
        int i = 1, n = 5;  
        // do...while loop from 1 to 5  
        do {  
            System.out.println(i);  
            i++;  
        } while(i <= n);  
    }  
}
```

Kết Quả:

1
2
3
4
5

4.5.3/ Cấu trúc do ... while

Vòng lặp While vô tận

```
// infinite while loop
while(true){
    // body of loop
}

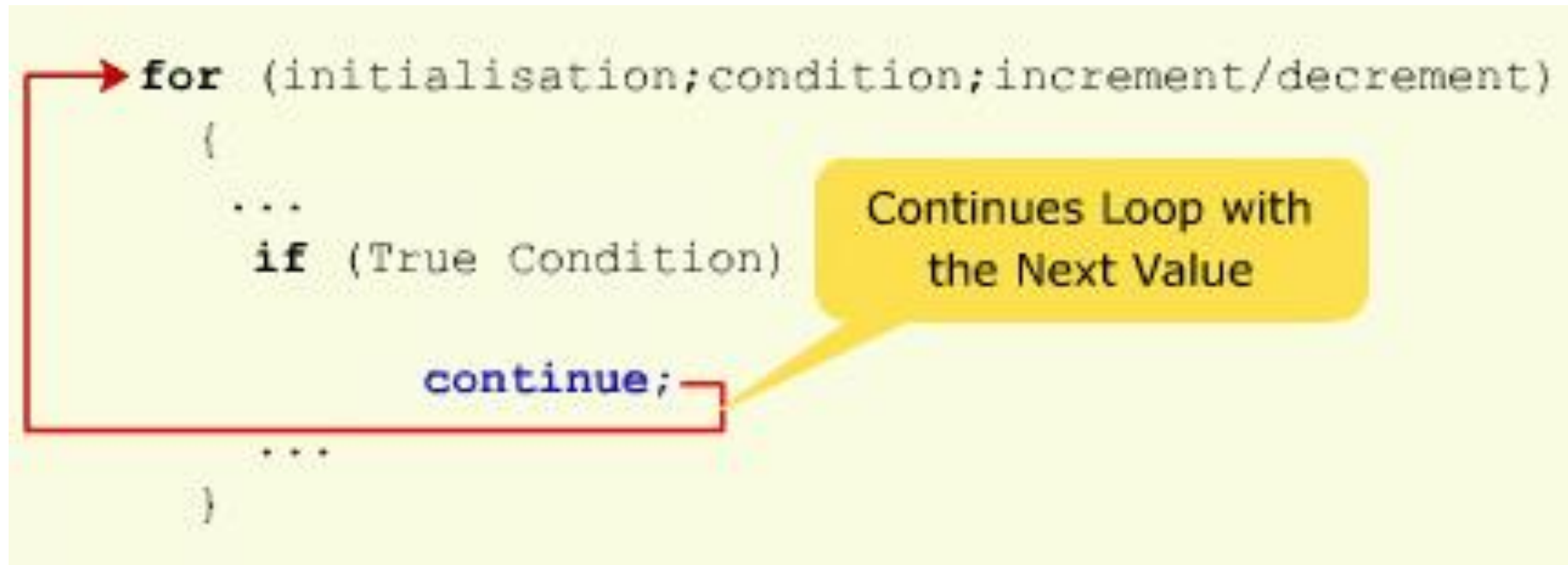
// infinite do...while loop
int count = 1;
do {
    // body of loop
} while(count == 1)
```

Nếu **<điều kiện lặp>** của vòng lặp **while** và **do while** luôn luôn là true thì đó là vòng lặp vô hạn.

Trường hợp nào nên sử dụng for hoặc while?

4.6/ Continue - Break

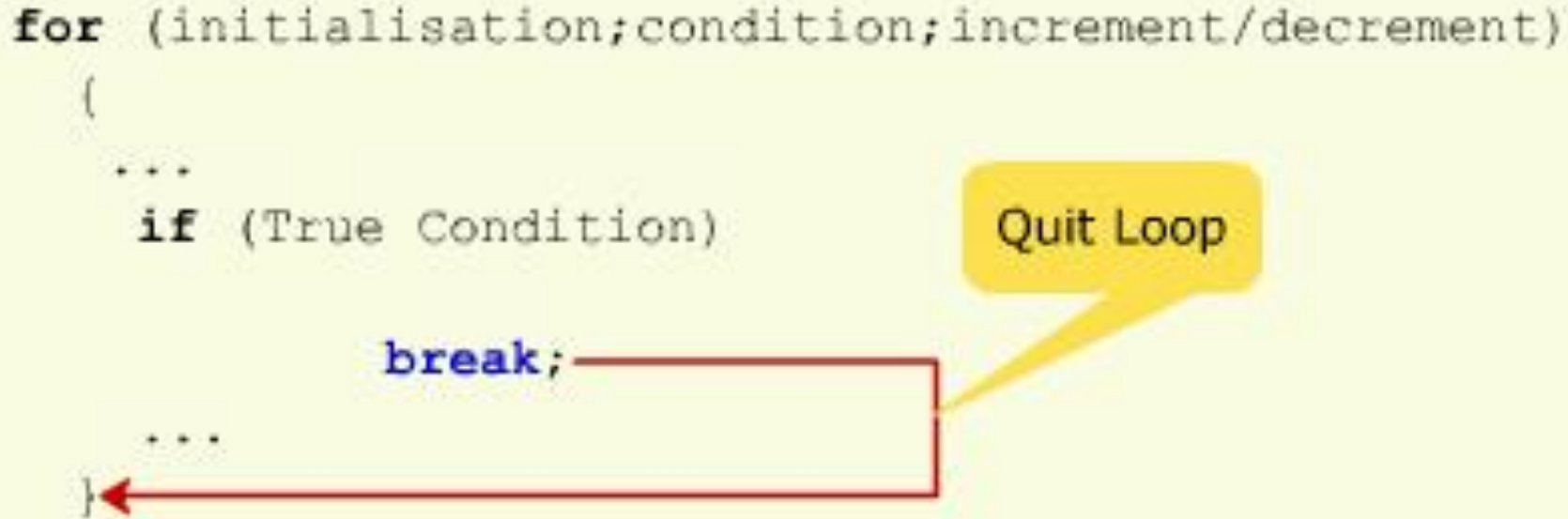
continue: Đây là lệnh thường được sử dụng kết hợp trong các cấu trúc lặp để bỏ qua các lệnh còn lại (*Của cấu trúc lặp*) và tiếp tục lần lặp tiếp theo khi thoả mãn 1 điều kiện nào đó trong chương trình.



4.6/ Continue - Break

break: gần giống như **continue**, tuy nhiên khác ở chỗ là **break** không những bỏ qua các lệnh còn lại trong cấu trúc có sử dụng nó mà còn kết thúc quá trình hoạt động của cấu trúc tương ứng luôn, không thực hiện nữa

```
for (initialisation; condition; increment/decrement)
{
    ...
    if (True Condition)
        break;
    ...
}
```



4.7/ label - goto

- **label**: là lệnh dùng để khai báo một nhãn (*cột mốc*) trong thân chương trình (*Nhằm hỗ trợ cho lệnh goto*). Nhãn chính là 1 định danh (*Identify*) trong chương trình luôn phải kết thúc bởi dấu “:”
- **goto**: dùng để cho phép chuyển điều khiển thực thi các lệnh trong chương trình tới một nhãn nào đó (*làm mất tính tuyến tính khi thực thi mã lệnh của chương trình*)

```
if (True condition)
```

```
{
```

```
    goto Display;
```

```
}
```

```
Display:
```

```
    Print "goto statement is executed"
```

Control Transferred
to Display

Lưu ý: Do lệnh goto kết hợp với label có nguy cơ phá vỡ tính cấu trúc của các giải thuật, chính vì thế, hiện nay, gần như rất ít được sử dụng khi viết chương trình

4.8/ return

return: được dùng để kết thúc chương trình con (*hàm hiện hành*), đồng thời có thể trả về giá trị (*hoặc không*) cho nơi đã gọi hàm. cú pháp: **return** [giá trị];

- Ghi chú: Lệnh **return** được sử dụng phổ biến trong các hàm (*function*) được định nghĩa trong chương trình

```
if (True Condition)
{
    ...
    return;
}
else
{
    ...
    ...
}
Print "return statement"
```

Program
Terminates

Ví dụ vòng lặp

■ Ví dụ 01: In các số tự nhiên từ 1 đến 1000

```
public class VongLap {
    public static void main(String[] args) {
        // in các số từ 1 đến 1000
        for(int i = 1; i <= 1000; i++) {
            System.out.println(i);
        }
    }
}

public class VongLapDoWhile {
    public static void main(String[] args) {
        int i = 1;
        do {
            System.out.println(i);
            i++;
        } while (i <= 1000);
    }
}

public class VongLapPWhile {
    public static void main(String[] args) {
        int i = 1;
        while (i <= 1000) {
            System.out.println(i);
            i++;
        }
    }
}
```

Ví dụ vòng lặp

- Ví dụ 02: Tính tổng các số tự nhiên từ 1 đến 1000

```
public class VongLapFor {
    public static void main(String[] args) {
        // Tổng các số từ 1 đến 1000
        int tong = 0;
        for(int i = 1; i <= 1000; i++) {
            tong = tong + i;
        }
        System.out.println(tong);
    }
}

public class VongLapDoWhile {
    public static void main(String[] args) {
        // in các số từ 1 đến 1000
        int i = 1;
        int tong = 0;
        do {
            tong = tong + i;
            i++;
        } while (i <= 1000);
        System.out.println(tong);
    }
}

public class VongLapWhile {
    public static void main(String[] args) {
        // in các số từ 1 đến 1000
        int i = 1;
        int tong = 0;
        while (i <= 1000) {
            tong = tong + i;
            i++;
        }
        System.out.println(tong);
    }
}
```


Ví dụ vòng lặp

■ Ví dụ 03: Duyệt mảng

```
public class VongLapFor {  
    public static void main(String[] args) {  
        // Khai báo một mảng  
        String[] niit = {  
            "1-Java",  
            "2-Python",  
            "3-PHP",  
            "4-JavaScript",  
            "5-CSS",  
            "6-HTML",  
            "7-Big Data"  
        };  
        // In ngược giá trị của mảng cho trước  
        for (int i = niit.length; i > 0 ; i--)  
            System.out.println(niit[i-1]);  
    }  
}
```

```
public class VongLapWhile {  
    public static void main(String[] args) {  
        // Khai báo một mảng  
        String[] niit = {  
            "1-Java",  
            "2-Python",  
            "3-PHP",  
            "4-JavaScript",  
            "5-CSS",  
            "6-HTML",  
            "7-Big Data"  
        };  
        int i = niit.length - 1;  
        // In ngược giá trị của mảng cho trước  
        while (i >= 0) {  
            System.out.println(niit[i]);  
            i--;  
        }  
    }  
}
```

```
public class VongLapDoWhile {  
    public static void main(String[] args) {  
        // Khai báo một mảng  
        String[] niit = {  
            "1-Java",  
            "2-Python",  
            "3-PHP",  
            "4-JavaScript",  
            "5-CSS",  
            "6-HTML",  
            "7-Big Data"  
        };  
        int i = niit.length - 1;  
        // In ngược giá trị của mảng cho trước  
        do {  
            System.out.println(niit[i]);  
            i--;  
        } while (i >= 0);  
    }  
}
```

Ví dụ vòng lặp

■ Ví dụ 03: Duyệt mảng – For each

```
public class VongLapForEach {  
    public static void main(String[] args) {  
        // Khai báo một mảng  
        String[] niit = {  
            "1-Java",  
            "2-Python",  
            "3-PHP",  
            "4-JavaScript",  
            "5-CSS",  
            "6-HTML",  
            "7-Big Data"  
        };  
        // In tất cả giá trị của mảng cho trước  
        for (String i : niit) {  
            System.out.println(i);  
        }  
    }  
}
```

- Mặc dù cú pháp và các sử dụng khác nhau nhưng các vòng lặp vẫn cho kết quả như nhau.
- Vấn đề là ta nên lựa chọn giải pháp sao cho phù hợp nhất để mang lại hiệu quả tốt nhất nhiệm vụ cần giải quyết.

Câu hỏi ôn tập

- 1) Cấu trúc rẽ nhánh **dùng cho mục đích gì** trong chương trình ?, Có những cấu trúc nào ?.
- 2) Toán tử điều kiện là gì ?
- 3) Hiểu thế nào về các cấu trúc lặp ?. Phân biệt các tình huống sử dụng cấu trúc lặp trong lập trình. Ví dụ: Khi nào dùng **for**, khi nào **while**, khi nào nên dùng **do ... while** ?.
- 4) Phân biệt **continue** – **break**.
- 5) Lệnh **return** dùng để làm gì ?

- **The for Statement**

(<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html>)

- **The while and do-while Statements**

(<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html>)

- **Java - Loop Control**

(https://www.tutorialspoint.com/java/java_loop_control.htm)