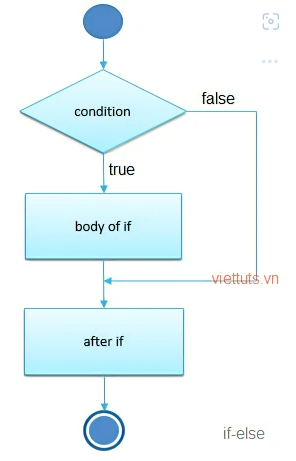
**Task4: Tìm hiểu các câu điều kiện và vòng lặp**

**1. Khái niệm khối lệnh(Block)**

Khối lệnh trong Java được biểu thị bằng cặp dấu “ { ” và “ } ”. Bạn sẽ thấy cặp ngoặc này đã xuất hiện trong khai báo class (bạn sẽ học đến class ở các bài viết về OOP sau). Trong trường hợp này cặp ngoặc đã tạo ra một khối lệnh đóng vai trò bao lấy code và cho biết tất cả các code bên trong đó đều là các code của class. Khi đó, chúng (các code trong cặp ngoặc đó) phải tuân theo các nguyên tắc của class (bạn sẽ biết các nguyên tắc này sau). Mọi dòng code nằm ngoài cặp ngoặc nhọn này sẽ không thuộc quyền quản lý của class đó. Hay cặp ngoặc nhọn xuất hiện ở khai báo phương thức (bạn cũng sẽ học đến phương thức ở bài sau), giúp tạo ra một khối lệnh đóng vai trò bao lấy code cho biết tất cả các code bên trong đó đều là code của phương thức đó. Cũng như trên, mọi dòng code nằm ngoài cặp ngoặc nhọn của phương thức này sẽ nằm ngoài xử lý logic của phương thức đó.

**2. Mệnh đề if**

Mệnh đề if trong java được sử dụng để kiểm tra giá trị dạng boolean của điều kiện. Mệnh đề này trả về giá trị True hoặc False . Có các kiểu của mệnh đề if-else trong java như sau:

1. Mệnh đề if
2. Mệnh đề if-else
3. Mệnh đề if-else-if

* Mệnh đề if: Mệnh đề if được sử dụng để kiểm tra giá trị dạng boolean của điều kiện. Khối lệnh sau if được thực thi nếu giá trị của điều kiện là **True**

cú pháp:

if (condition) {

    // khối lệnh này thực thi

    // nếu condition = true

}

Ví dụ:

public class Test {

    public static void main(String[] args) {

        int age = 20;

        if (age > 18) {

            System.out.print("Tuổi lớn hơn 18");

        }

    }

}

Kết quả:



* Mệnh đề if-else

Mệnh đề if-else cũng kiểm tra giá trị dạng boolean của điều kiện. Nếu giá trị điều kiện là **True**thì chỉ có khối lệnh sau if sẽ được thực hiện, nếu là **False**thì chỉ có khối lệnh sau else được thực hiện.

Cú pháp:

**if** (condition) {

    // khối lệnh này được thực thi

    // nếu condition = true

} **else** {

    // khối lệnh này được thực thi

    // nếu condition = false

}

Ví dụ:

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** number = 13;

**if** (number % 2 == 0) {

            System.out.println("Số " + number + " là số chẵn.");

        } **else** {

            System.out.println("Số " + number + " là số lẻ.");

        }

    }

Kết quả:



* Mệnh đề if-else-if

Mệnh đề if-else-if cũng kiểm tra giá trị dạng boolean của điều kiện. Nếu giá trị điều kiện if là **True**thì chỉ có khối lệnh sau if sẽ được thực hiện. Nếu giá trị điều kiện if else nào là **True**thì chỉ có khối lệnh sau else if đó sẽ được thực hiện... Nếu tất cả điều kiện của if và else if là **False**thì chỉ có khối lệnh sau else sẽ được thực hiện.

Cú pháp:

**if** (condition1) {

    // khối lệnh này được thực thi

    // nếu condition1 là true

} **else** **if** (condition2) {

    // khối lệnh này được thực thi

    // nếu condition2 là true

}  **else** **if** (condition3) {

    // khối lệnh này được thực thi

    // nếu condition3 là true

}

...

**else** {

    // khối lệnh này được thực thi

    // nếu tất cả những điều kiện trên là false

}

Ví dụ:

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** marks = 65;

**if** (marks < 50) {

            System.out.println("Tạch!");

        } **else** **if** (marks >= 50 && marks < 60) {

            System.out.println("Xếp loại D");

        } **else** **if** (marks >= 60 && marks < 70) {

            System.out.println("Xếp loại C");

        } **else** **if** (marks >= 70 && marks < 80) {

            System.out.println("Xếp loại B");

        } **else** **if** (marks >= 80 && marks < 90) {

            System.out.println("Xếp loại A");

        } **else** **if** (marks >= 90 && marks < 100) {

            System.out.println("Xếp loại A+");

        } **else** {

            System.out.println("Giá trị không hợp lệ!");

        }

    }

}

Kết quả:



**3. Mệnh đề Switch-case**

Mệnh đề switch-case trong java được sử dụng để thực thi 1 hoặc nhiều khối lệnh từ nhiều điều kiện.

Cú pháp:

**switch** (bieu\_thuc) {

**case** gia\_tri\_1:

    // Khối lệnh 1

**break**;  //tùy chọn

**case** gia\_tri\_2:

    // Khối lệnh 2

**break**;  //tùy chọn

......

**case** gia\_tri\_n:

    // Khối lệnh n

**break**;  //tùy chọn

**default**:

    // Khối lệnh này được thực thi

    // nếu tất cả các điều kiện trên không thỏa mãn

}

Ví dụ:

**public** **class** SwitchExample {

**public** **static** **void** main(String[] args) {

**int** number = 20;

**switch** (number) {

**case** 10:

            System.out.println("10");

**break**;

**case** 20:

            System.out.println("20");

**break**;

**case** 30:

            System.out.println("30");

**break**;

**default**:

            System.out.println("Not in 10, 20 or 30");

        }

    }

}

Kết quả:



**3.1 Mệnh đề Switch-case khi không sử dụng “break”**

Khi không sử dụng từ khóa 'break' trong mệnh đề switch-case. Điều này có nghĩa là các khối lệnh sau case có giá trị phù hợp sẽ được thực thi.

Ví dụ về mệnh đề switch-case:

**ublic** **class** SwitchExample2 {

**public** **static** **void** main(String[] args) {

**int** number = 20;

**switch** (number) {

**case** 10:

            System.out.println("10");

**case** 20:

            System.out.println("20");

**case** 30:

            System.out.println("30");

**default**:

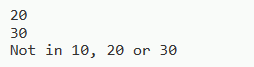
            System.out.println("Not in 10, 20 or 30");

        }

    }

}

Kết quả:



**4. Vòng lặp For**

Vòng lặp for trong java được sử dụng để lặp một phần của chương trình nhiều lần. Nếu số lần lặp là cố định thì vong lặp for được khuyến khích sử dụng, còn nếu số lần lặp không cố định thì nên sử dụng vòng lặp while hoặc do while.

Có 3 kiểu của vòng lặp for trong java:

1. Vòng lặp for đơn giản
2. Vòng lặp for cải tiến
3. Vòng lặp for gán nhãn

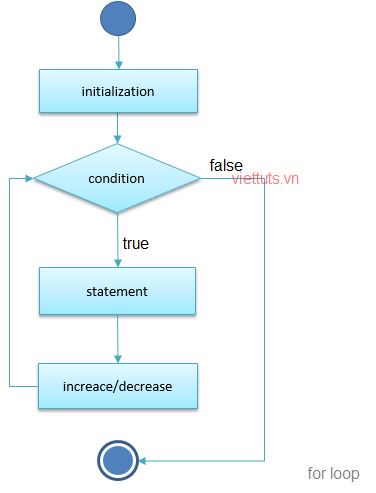
* Vòng lặp for đơn giản:

Cú pháp:

**for** (khoi\_tao\_bien ; check\_dieu\_kien ; tang/giam\_bien) {

    // Khối lệnh được thực thi

}



Ví dụ:

**public** **class** ForExample {

**public** **static** **void** main(String[] args) {

**for** (**int** i = 1; i <= 10; i++) {

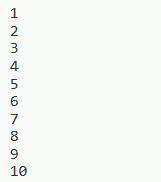
            System.out.println(i);

        }

    }

}

Kết quả:



* Vòng lặp for được cải tiến:

Vòng lặp for cải tiến được sử dụng để lặp mảng(array) hoặc collection trong java. Bạn có thể sử dụng nó dễ dàng, dễ hơn cả vòng lặp for đơn giản. Bởi vì bạn không cần phải tăng hay giảm giá trị của biến rồi check điều kiện, bạn chỉ cần sử dụng ký hiệu hai chấm ":"

Cú pháp:

**for** (Type var : array) {

    // Khối lệnh được thực thi

}

Ví dụ:

**public** **class** ForEachExample {

**public** **static** **void** main(String[] args) {

**int** arr[] = { 12, 23, 44, 56, 78 };

**for** (**int** i : arr) {

            System.out.println(i);

        }

    }

}

Kết quả:



* Vòng lặp for gán nhãn

Chúng ta có để đặt tên cho mỗi vòng lặp for bằng cách gán nhãn trước vòng lặp for. Điều này rất hữu dụng khi chúng ta muốn thoát/tiếp tục(break/continues) chạy vòng lặp for.

Cú pháp:

ten\_nhan:

**for** (khoi\_tao\_bien ; check\_dieu\_kien ; tang/giam\_bien) {

    // Khối lệnh được thực thi

}

Ví dụ:

**public** **class** LabeledForExample {

**public** **static** **void** main(String[] args) {

        aa: **for** (**int** i = 1; i <= 3; i++) {

            bb: **for** (**int** j = 1; j <= 3; j++) {

**if** (i == 2 && j == 2) {

**break** aa;

                }

                System.out.println(i + " " + j);

            }

        }

    }

}

Kết quả:



**5. Vòng lặp While**

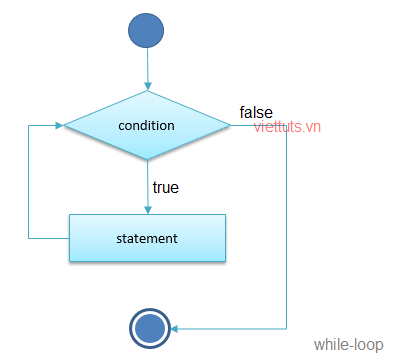
**Vòng lặp while trong java** được sử dụng để lặp một phần của chương trình một vài lần. Nếu số lần lặp không được xác định trước thì vòng lặp lặp while được khuyến khích sử dụng trong trường hợp này.

Cú pháp:

**while**(condition) {

    // Khối lệnh được lặp lại cho đến khi condition = False

}



Ví dụ:

**public** **class** WhileExample1 {

**public** **static** **void** main(String[] args) {

**int** i = 1;

**while** (i <= 10) {

   System.out.println(i);

   i++;

  }

 }

}

Kết quả:



**5.1. Vòng lặp While vô tận**

Nếu bạn để điều kiện lặp là True thì vòng lặp while sẽ chạy đến vô tận... Đến khi bạn stop chương trình đối với mỗi IDE(Eclipse, Netbean...) hoặc bấm Ctrl + C khi chạy bằng command.

Ví dụ về vòng lặp while vô tận:

**public** **class** WhileExample2 {

**public** **static** **void** main(String[] args) {

**while** (**true**) {

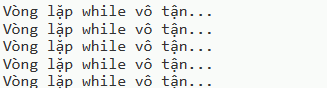
   System.out.println("Vòng lặp while vô tận...");

  }

 }

}

Kết quả:



**5.2 Vòng lặp Do-While**

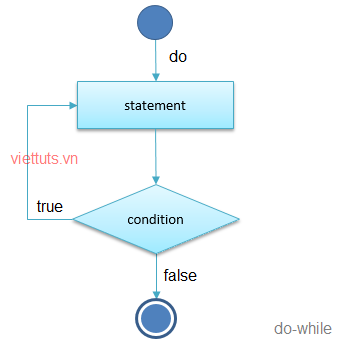
**Vòng lặp do-while trong java** được sử dụng để lặp một phần của chương trình một vài lần. Tương tự như vòng lặp while, ngoại trừ do-while thực hiện lệnh ít nhất một lần ngay cả khi điều kiện là False.

Cú pháp:

**do** {

    // Khối lệnh được thực thi

} **while**(condition);



Ví dụ sau tính tổng của 5 số tự nhiên đầu tiên dùng cấu trúc do-while:

**public** **class** DoWhileExample1 {

**public** **static** **void** main(String[] args) {

**int** a = 1, sum = 0;

**do** {

            sum += a;

            a++;

        } **while** (a <= 5);

        System.out.println("Sum of 1 to 5  is " + sum);

    }

}

(Biến a được khởi tạo với giá trị 1, sau đó nó vừa được dùng làm biến chạy (tăng lên 1 sau mỗi lần lặp) vừa được dùng để cộng dồn vào biến sum. Tại thời điểm kết thúc, chương trình sẽ in ra Sum of 1 to 5 is 15.)

Kết quả:



**5.3 Vòng lặp Do-While vô tận**

Ví dụ:

**public** **class** DoWhileExample2 {

**public** **static** **void** main(String[] args) {

**do** {

            System.out.println("Vòng lặp do-while vô tận...");

        } **while** (**true**);

    }

}

Kết quả:

