**Vòng lặp vô hạn**

* Vòng lặp mà lặp vô tận, không có điểm dừng gọi là vòng lặp vô hạn
* Vòng lặp vô hạn xảy ra khi điều kiện lặp luôn đúng
* Vòng lặp vô hạn có cả hai mặt lợi và hại. Tuy vậy khi biết ứng dụng vào từng trường hợp cụ thể thì bạn sẽ tận dụng được những lợi thế của vòng lặp này.
  + Mặt lợi là nó cho phép ta chạy chương trình đến vô thời hạn. Điều này sẽ cho ta có cơ hội giám sát được thời điểm kết thúc của vòng lặp dựa trên một dữ kiện nào đó.
  + Mặt hại là nếu cứ chạy vô thời hạn mà không dừng, thì sẽ chiếm CPU, RAM, các thành phần phần cứng liên quan nào đó. Dẫn đến không có đủ tài nguyên để phục vụ các chương trình khác.
* Các vòng lặp vô hạn thường sử dụng kèm với lệnh break để kết thúc lặp khi điều kiện nào đó xảy đến.
* Loại vòng lặp này thường được sử dụng để kiểm soát dữ liệu đầu vào hoặc chờ đợi một sự kiện nào đó.
* Ví dụ: Nhập các giá trị vào từ bàn phím, việc nhập sẽ kết thúc nếu nhập giá trị 0. Trong trường hợp này vòng lặp sẽ là vòng lặp vô hạn vì chưa biết khi nào mới kết thúc. Nhưng biết điều kiện để kết thúc là khi người dùng ấn phím 0.

var input = new Scanner(System.*in*);

var n = 0;

while (true) {

    System.*out*.println("Nhập 0 để kết thúc. Nhập khác 0 để tiếp tục");

    n = input.nextInt();

*// ... làm gì đó ở đây*

if(n == 0) { *// nếu n == 0*

break; *// thoát khỏi vòng lặp*

}

}

**Lệnh break**

* Khi nào cần chủ động thoát khỏi một cấu trúc, ví dụ cấu trúc lặp, cấu trúc switch, ta dùng break. Lệnh break sẽ cho phép ta kết thúc sớm và bỏ qua các lần lặp hoặc đánh giá kế tiếp vì lúc này ta đã đạt được mục đích. Do việc lặp tiếp hoặc đánh giá tiếp là dư thừa, không cần thiết.
* Lệnh break sẽ thoát khỏi cấu trúc lặp hoặc switch gần nhất chứa nó. Không thoát hoàn toàn khỏi chương trình.
* Khi gặp lệnh break, các lệnh phía dưới break(nếu có) và trong cùng khối với break sẽ không được thực hiện nữa.
* Ví dụ 1:

import java.util.Scanner;

public class Lesson16 {

    public static void main(String[] args) {

        var input = new Scanner(System.*in*);

        var n = 0;

        while (true) {

            System.*out*.println("Nhập 0 để kết thúc. Nhập khác 0 để tiếp tục");

            n = input.nextInt();

*// ... làm gì đó ở đây*

if(n == 0) { *// nếu n == 0*

**break**; *// thoát khỏi vòng lặp*

}

        }

    }

}

Trong ví dụ trên lệnh break chỉ thoát việc thực thi khỏi vòng lặp while. Không làm kết thúc chương trình. Sau khi thoát vòng lặp, các lệnh phía sau thân vòng lặp sẽ lần lượt được thực hiện theo logic của mã nguồn.

* Ví dụ 2:

var choice = input.nextInt(); *// nhập lựa chọn*

*// tiến hành đánh giá và thực hiện chương trình*

switch (choice) {

    case 0:

        System.*out*.println("Xin chào và hẹn gặp lại!");

**break**;

    case 1: *// nếu giá trị của choice = 1*

var sum = a + b; *// tính tổng hai số*

System.*out*.println(a + " + " + b + " = " + sum);

**break**;

    case 2: *// nếu giá trị của choice = 2*

var dif = a - b;

        System.*out*.println(a + " - " + b + " = " + dif);

**break**;

    case 3: *// nếu giá trị của choice = 3*

var prod = a \* b;

        System.*out*.println(a + " \* " + b + " = " + prod);

**break**;

    case 4: *// nếu giá trị của choice = 4*

if(b != 0) {

            var quot = a / b;

            System.*out*.println(a + " / " + b + " = " + quot);

        } else {

            System.*out*.println("Mẫu số = 0. Không thực hiện được phép chia.");

        }

**break**;

    default:

        System.*out*.println("Sai chức năng. Vui lòng chọn lại.");

**break**;

}

Mỗi case trong cấu trúc switch trên đều chứa break ở cuối. Sở dĩ break ở cuối là sau khi thực hiện xong case tương ứng thì sẽ thoát logic khỏi cấu trúc switch. Các case khác sẽ không được đánh giá hay xem xét tiếp nữa vì ta đã tìm đc thứ mong muốn ở case vừa thực hiện rồi. Nếu break đặt đầu mỗi case thì các lệnh trong case đó sẽ không được thực hiện vì logic chương trình đã kết thúc tại thời điểm nó gặp break.

* Ví dụ 3: Tìm giá trị đầu tiên trong đoạn [a, b] chia hết cho k != 0.

var input = new Scanner(System.*in*);

System.*out*.println("Nhập số nguyên a < b và giá trị k: ");

var a = input.nextInt();

var b = input.nextInt();

var k = input.nextInt();

if(k != 0) {

    for (int i = a; i <= b; i++) {

        if(i % k == 0) {

            System.*out*.println("giá trị đầu tiên thỏa: " + i);

**break**; *// kết thúc vòng lặp*

}

    }

*// làm gì đó tiếp ở đây*

} else {

    System.*out*.println("Nhập k != 0.");

}

Trong ví dụ trên, nếu ta tìm thấy một giá trị i nào đó trong đoạn [a, b] chia hết cho k != 0, ta thoát khỏi vòng lặp. Vì lúc này ta đã tìm đc lời giải. Vấn đề đã được xử lý. Logic chương trình chỉ thoát khỏi vòng lặp nếu gặp break. Chương trình sau đó sẽ tiếp tục thực hiện các lệnh phía dưới vòng lặp(nếu có).

**Lệnh continue**

* Khác với break, continue không làm chương trình thoát khỏi một khối nào khi gặp nó. Thực tế continue cho phép logic chương trình bỏ qua các lệnh ở phía dưới và cùng nằm trong thân vòng lặp với continue.
* Continue thường áp dụng kết hợp với các cấu trúc điều khiển như ra quyết định hoặc lặp.
* Ví dụ 4: Liệt kê các giá trị trong đoạn [a, b] chia hết cho k!=0.

if(k != 0) {

    for (int i = a; i <= b; i++) {

        if(i % k != 0) { *// nếu i không chia hết cho k*

continue; *// bỏ qua các lệnh phía dưới*

}

*// ngược lại, tức i mà chia hết cho k, thì thực hiện:*

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

    }

*// làm gì đó tiếp ở đây*

} else {

    System.*out*.println("Nhập k != 0.");

}

**Ví dụ minh họa**

Ví dụ 3 đầy đủ:

import java.util.Scanner;

public class Lesson16 {

    public static void main(String[] args) {

        var input = new Scanner(System.*in*);

        System.*out*.println("Nhập số nguyên a < b và giá trị k: ");

        var a = input.nextInt();

        var b = input.nextInt();

        var k = input.nextInt();

        if(k != 0) {

            for (int i = a; i <= b; i++) {

                if(i % k == 0) {

                    System.*out*.println("giá trị đầu tiên thỏa: " + i);

                    break; *// kết thúc vòng lặp*

}

            }

*// làm gì đó tiếp ở đây*

} else {

            System.*out*.println("Nhập k != 0.");

        }

    }

}

Ví dụ 4 đầy đủ:

import java.util.Scanner;

public class Lesson16 {

    public static void main(String[] args) {

        var input = new Scanner(System.*in*);

        System.*out*.println("Nhập số nguyên a < b và giá trị k: ");

        var a = input.nextInt();

        var b = input.nextInt();

        var k = input.nextInt();

        if(k != 0) {

            for (int i = a; i <= b; i++) {

                if(i % k != 0) { *// nếu i không chia hết cho k*

continue; *// bỏ qua các lệnh phía dưới*

}

*// ngược lại, tức i mà chia hết cho k, thì thực hiện:*

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

            }

*// làm gì đó tiếp ở đây*

} else {

            System.*out*.println("Nhập k != 0.");

        }

    }

}

Nhập số nguyên a < b và giá trị k:

3 50 7

7

14

21

28

35

42

49