به نام خدا

برنامه‌سازی پیشرفته

دانشگاه شهید بهشتی · دانشکده مهندسی کامپیوتر

دکتر مجتبی وحیدی اصل

دستورات کنترلی - بخش دوم

آریا زریاب

فهرست مطالب

1. مثالی از switch-case
2. دستور while
3. دستور do-while
4. دستور for
5. دستور break-continue
6. نوشتن چند برنامه
7. مثال های بیشتر

مثالی از switch-case

- برنامه‌ای بنویسید که شماره ماه و سال خورشیدی را از کاربر دریافت کرده و سپس تعداد روزهای ماه را در خروجی چاپ کند. کبیسه یا غیر کبیسه بودن سال را در نظر بگیرید.

1

```java
import java.util.Scanner;

class MonthDays{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Please enter a month number:");
        int month = in.nextInt();
        System.out.println("Please enter a year number:");
        int year = in.nextInt();
        int numDays = 0;

        switch (month) {
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
            case 6:
                numDays = 31;
                break;
            case 7:
            case 8:
            case 9:
            case 10:
            case 11:
                numDays = 30;
                break;
```

2

```
            case 11:
                numDays = 30;
                break;
            case 12:
                if (((year % 4 == 0) && !(year % 100 == 0)) || (year % 400 == 0) )
                    numDays = 30;
                else
                    numDays = 29;
                break;
            default:
                System.out.println("Invalid month.");
                break;
        }
        System.out.println("Number of Days = " + numDays);
    }
}
```

مقدمه

فرض کنید میخواهیم یک رشته مثل **"Welcome to Java!"** را ۱۰۰ مرتبه چاپ کنیم.

مسلماً کسالت بار است که بخواهیم دستور زیر را ۱۰۰ بار تکرار کنیم :

```
System.out.println("Welcome to Java!");
```

یا اینکه معلمی خطاب به دانش آموزش بگوید ۵۰۰ بار بنویس "من موشک های کاغذی را در کلاس پرتاب نخواهم کرد!"

100 times

```
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");

...

...

...
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
System.out.println("Welcome to Java!");
```

برای حل این مشکل چه باید کرد ؟

در برنامه‌نویسی، خیلی وقت‌ها نیاز داریم یک کار تکراری را بارها انجام دهیم. اگر بخواهیم این کار را با کپی‌پیست تکرار کنیم:

- کد طولانی و خسته‌کننده می‌شود،
- اگر جایی تغییر بخواهد، باید در همه‌جا تغییر بدهیم،
- و احتمال خطا بالا می‌رود.

اینجاست که حلقه‌ها وارد عمل می‌شوند! آن‌ها کمک می‌کنند:

- یک دستور یا مجموعه‌ای از دستورات را به‌طور خودکار تکرار کنیم،
- شرطی برای ادامه یا توقف تعیین کنیم،
- و کدی کوتاه‌تر، خواناتر و قابل نگهداری‌تر داشته باشیم.

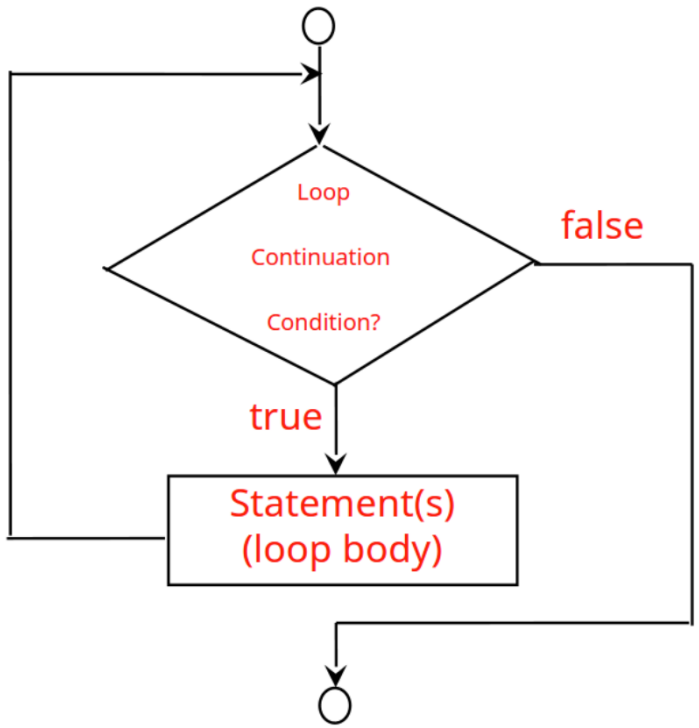
معرفی حلقه while

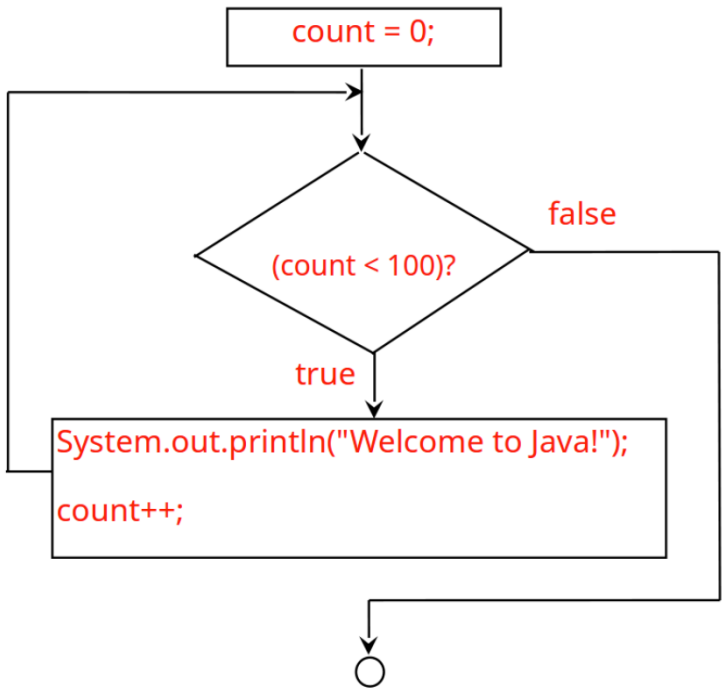
```
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java!");
    count++;
}
```

فلوچارت حلقه while

```
while (loop-continuation-condition) {
    // loop-body
    Statement(S);
}
```

```
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java!");
    count++;
}
```





ردگیری حلقه while

```
int count = 0;     Initialize count
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}
```

```
int count = 0;
while (count < 2) {     count < 2  is True
    System.out.println("Welcome to Java!");
    count++;
}
```

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");     Print Welcome to Java
    count++;
}
```

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;     Increment count by 1, count is 1 now
}
```

ردگیری حلقه while

```
int count = 0;
while (count < 2) {    count < 2  is still True since count is 1
    System.out.println("Welcome to Java!");
    count++;
}
```

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");    Print Welcome to Java
    count++;
}
```

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;    Increment count by 1, count is 2 now
}
```

ردگیری حلقه while

```
int count = 0;
while (count < 2) {    count < 2  is False since count is 2
    System.out.println("Welcome to Java!");
    count++;
}
```

```
int count = 0;
while (count < 2) {
    System.out.println("Welcome to Java!");
    count++;
}

The loop exits. Execute the next statement after the loop.
```

مثال : حدس اعداد

- برنامه ای بنویسید که به طور تصادفی یک عدد صحیح بین ۰ تا ۱۰۰ را تولید کند.
- برنامه به کاربر می گوید تا پشت سر هم اعدادی را وارد کند تا زمانی که عدد وارد شده با عدد تصادفی تولید شده برابر شود.
- به ازای هر ورودی کاربر، برنامه به او می گوید آیا مقدار وارد شده کاربر از عدد مربوطه کوچکتر است یا بزرگتر. در نتیجه کاربر میتواند ورودی بعدی را هوشمندانه تر تولید کند.

```java
import java.util.Scanner;

public class GuessNumber {
    public static void main(String[] args) {
        // Generate a random number to be guessed
        int number = (int)(Math.random() * 101);

        Scanner input = new Scanner(System.in);
        System.out.println("Guess a magic number between 0 and 100");

        int guess = -1;
        while (guess != number) {
            // Prompt the user to guess the number
            System.out.print("\nEnter your guess: ");
            guess = input.nextInt();

            if (guess == number)
                System.out.println("Yes, the number is " + number);
            else if (guess > number)
                System.out.println("Your guess (" + guess + ") is too high");
            else
                System.out.println("Your guess (" + guess + ") is too low");
        } // End of loop
    }
}
```

مثال : ابزار پیشرفته آموزش ریاضی

ابزار یادگیری تفریق در هر اجرا تنها یک سوال مطرح میکرد.

با استفاده از **loop** میتوانیم سوالاتی را پشت سر هم مطرح کنیم.

- برنامه ای بنویسید که ۵ سوال از کاربر بپرسد و تعداد پاسخ های صحیح را پس از پایان سوالات اعلام کند.

همچنین لیست سوالات به همراه صحیح یا غلط بودن پاسخ ارائه شود.

زمان سپری شده برای پاسخ دانش آموز نیز محاسبه شود.

- دستور زیر زمان فعلی را به میلی ثانیه بر می گرداند:

```
System.currentTimeMillis();
```

همچنین این متد در پکیج java.lang قرار دارد .

1

```java
import java.util.Scanner;

public class SubtractionQuizLoop {
    public static void main(String[] args) {
        final int NUMBER_OF_QUESTIONS = 5; // Number of questions

        int correctCount = 0; // Count the number of correct answers
        int count = 0; // Count the number of questions
        long startTime = System.currentTimeMillis();
        String output = ""; // output string is initially empty
        Scanner input = new Scanner(System.in);

        while (count < NUMBER_OF_QUESTIONS) {
        // 1. Generate two random single-digit integers
            int number1 = (int)(Math.random() * 10);
            int number2 = (int)(Math.random() * 10);

            // 2. If number1 < number2, swap number1 with number2
            if (number1 < number2) {
                int temp = number1;
                number1 = number2;
                number2 = temp;
            }

            // 3. Prompt the student to answer “What is number1 – number2?”
            System.out.print("What is " + number1 + " - " + number2 + "? ");
            int answer = input.nextInt();
```

2

```java
            // 4. Grade the answer and display the result
            if (number1 - number2 == answer) {
                System.out.println("You are correct!");
                correctCount++;
            }
            else
                System.out.println("Your answer is wrong.\n" + number1 + " - " + number2 + " should be " + (number1 - number2));

            // Increase the count
            count++;

            output += "\n" + number1 + "-" + number2 + "=" + answer + ((number1 - number2 == answer) ? " correct" : " wrong");
        }

        long endTime = System.currentTimeMillis();
        long testTime = endTime - startTime;

        System.out.println("Correct count is " + correctCount + "\nTest time is " + testTime / 1000 + " seconds\n" + output);
    }
}
```

پایان دادن به حلقه با یک مقدار کنترلی

در بیشتر مواقع تعداد دفعات اجرای حلقه از قبل مشخص نمی باشد.

در چنین مواقعی از یک مقدار ورودی برای پایان دادن به حلقه استفاده میکنیم.

- به این مقدار اصطلاحاً **نگهبان (sentinel)** گفته میشود.

- حال برنامه ای بنویسید که مجموع تعداد نامشخصی عدد را حساب کند.

مقدار 0 نگهبان یا پایان دهنده ورودی باشد.

```java
import java.util.Scanner;

public class SentinelValue {
    /** Main method */
    public static void main(String[] args) {
        // Create a Scanner
        Scanner input = new Scanner(System.in);

        // Read an initial data
        System.out.println("Enter an int value (the program exits if the input is 0): ");
        int data = input.nextInt();

        // Keep reading data until the input is 0
        int sum = 0;
        while (data != 0) {
            sum += data;

            // Read the next data
            System.out.println( "Enter an int value (the program exits if the input is 0): ");
            data = input.nextInt();
        }

        System.out.println("The sum is " + sum);
    }
}
```

حلقه های while تو در تو

می توان چندین حلقه **while** را به صورت تو در تو نوشت!

- تمرین : برنامه ای بنویسید که یک هرم از * چاپ کند.

```
*
**
***
****
*****
```

نکته مهم

- از **مقادیر ممیز شناور** (مانند float و double) برای بررسی **مساوی بودن** در شرط کنترلی حلقه استفاده **نکنید**.

چون برای برخی مقادیر، تقریبی و نه دقیق هستند.

در نتیجه ممکن است منجر به خطا و نتایج نادرست شوند.

- فرض کنید می خواهیم حاصل عبارت **0.1 + 0.2 + ... + 0.9 + 1** را محاسبه نماییم:

```
double item = 1;
double sum = 0;
while (item != 0) { // No guarantee item will be 0
    sum += item;
    item -= 0.1;
}
System.out.println(sum);
```

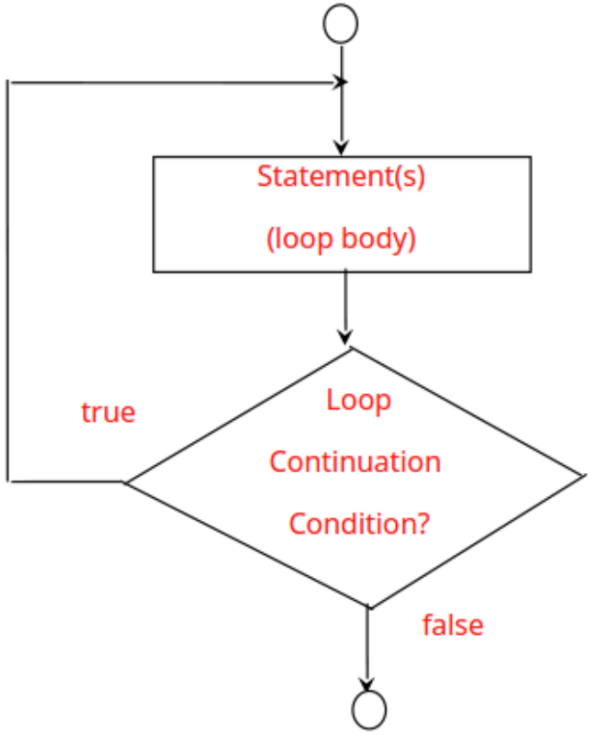
- متغیر item از ۱ شروع به کم شدن ۰.۱ در هر تکرار حلقه میشود. حلقه باید وقتی مقدار صفر میشود متوقف شود. با این حال، هیچ ضمانتی وجود ندارد که مقدار item دقیقا ۰ شود، چون محاسبه ممیز شناور تقریبی است.

در عمل ممکن است حلقه بی نهایت بار تکرار شود!

```java
public class checkFloatingPoint {
    public static void main(String[] args) {
        double item = 1;
        while (item > 0) {
            item -= 0.1;
            System.out.println(item);
        }
    }
}
```

معرفی حلقه do-while

```
do  {
    // Loop Body
    Statement(S);
}  while (loop-continuation-condition);
```



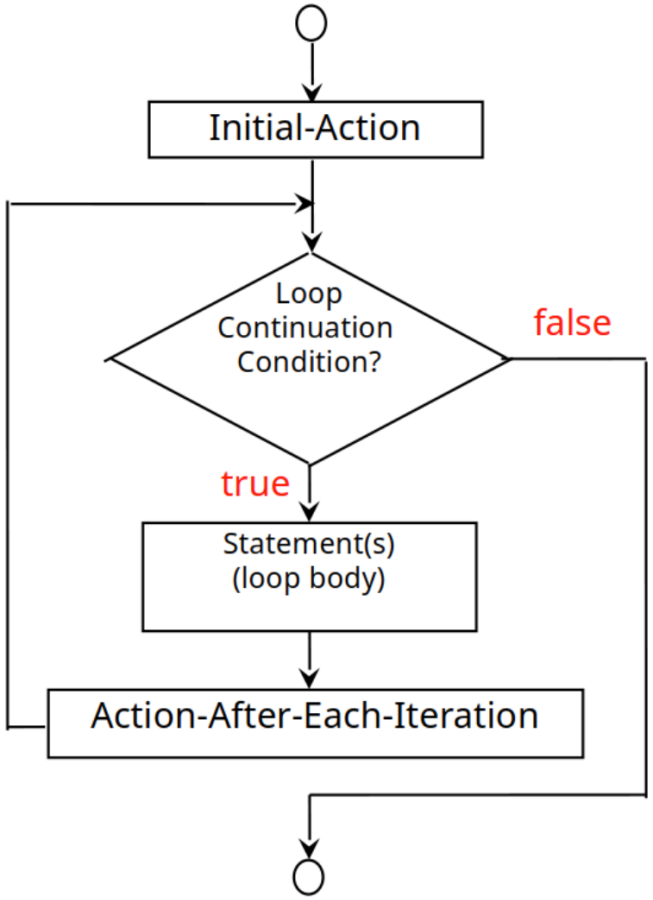
مثال

- برنامه ای بنویسید که بدون استفاده از آرایه و با استفاده از **do-while** دنباله فیبوناچی را از ۰ تا ۱۰۰ تولید کند.

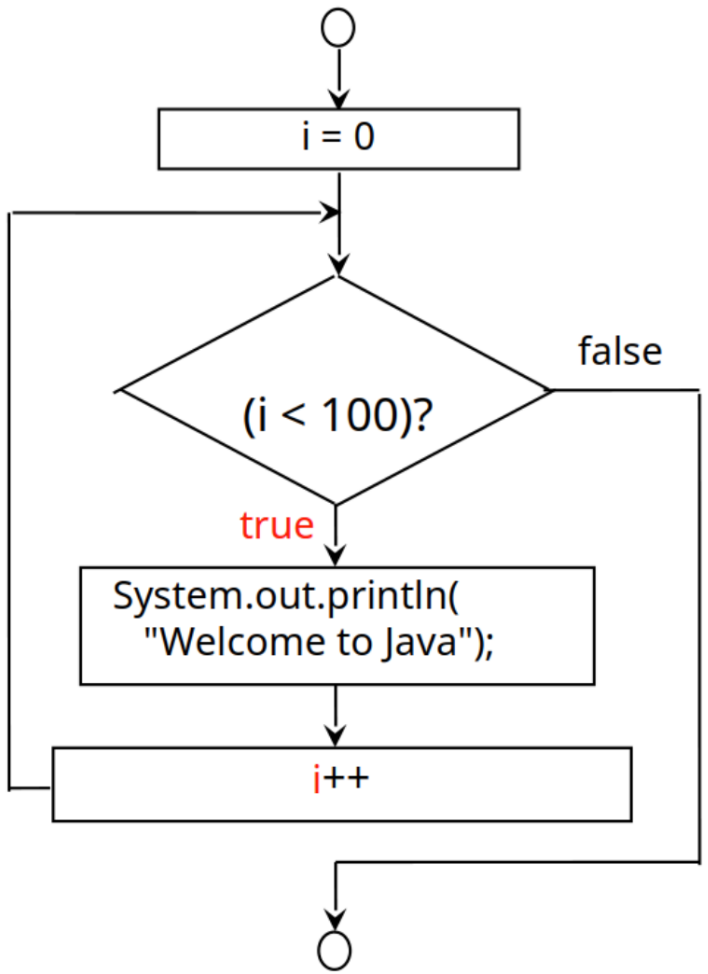
```java
public class Fibonacci {
    public static void main(String [] args) {
        int prevPrevVal = 0;
        int prevVal = 1;
        int curVal;
        System.out.print(prevPrevVal + " ");
        System.out.print(prevVal + " ");

        do {
            curVal = prevPrevVal + prevVal;
            System.out.print(curVal + " ");

            prevPrevVal = prevVal;
            prevVal = curVal;
        } while (prevVal + prevPrevVal <= 100);
    }
}
```

معرفی حلقه for

```
for (initial-action; loop-continuation-condition; action-after-each-iteration) {
    // loop-body
    Statement(S);
}
```



```java
int i = 0;
for (i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```



ردگیری حلقه for

```
int i;    Declare i
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}
```

```
int i;
for (i = 0; i < 2; i++) {     Execute initializer, i is 0 now
    System.out.println("Welcome to Java!");
}
```

```
int i;
for (i = 0; i < 2; i++) {     (i < 2) is True since i is 0
    System.out.println("Welcome to Java!");
}
```

```
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");     Print Welcome to Java
}
```

```
int i;
for (i = 0; i < 2; i++;) {      Execute adjustment statement, i is 1 now
    System.out.println("Welcome to Java!");
}
```

```
int i;
for (i = 0; i < 2; i++) {     (i < 2) is still True since i is 1
    System.out.println("Welcome to Java!");
}
```

```
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");     Print Welcome to Java
}
```

```
int i;
for (i = 0; i < 2; i++;) {      Execute adjustment statement, i is 2 now
    System.out.println("Welcome to Java!");
}
```

```
int i;
for (i = 0; i < 2; i++) {      (i < 2) is False since i is 2
    System.out.println("Welcome to Java!");
}
```

```
int i;
for (i = 0; i < 2; i++) {
    System.out.println("Welcome to Java!");
}

 Exit the loop. Execute the next statement after the loop.
```

نکته

- مقداردهی اولیه حلقه for ممکن است حاوی لیستی از صفر یا بیشتر عبارت باشد که با کاما (,) از هم جدا شده اند.
- عملیات بعدی در داخل تعریف حلقه نیز می تواند شامل صفر یا بیشتر عبارت باشد که با کاما (,) از هم جدا شده اند.
- دستورات زیر درست هستند، اما در عمل کمتر استفاده می شوند:

```
for (int i = 1; i < 100; System.out.println(i++));
```

```
for (int i = 0, j = 0; i + j < 10; i++, j++) {
    // Do something
 }
```

```java
public class DeclaringMultipleVariablesInForLoopExample {

    public static void main(String[] args) {

        /*
         * Multiple variables can be declared in declaration block of for loop.
         */

        for (int i = 0, j = 1, k = 2; i < 5; i++) {
            System.out.println("i : " + i + ", j : " + j + ", k : " + k);
        }

        /*
         * Please note that the variables which are declared, should be of same type
         * as in this example int.
         */

        // THIS WILL NOT COMPILE
        // for (int i = 0, float j; i < 5; i++);
    }
}
```

نکته

- اگر شرط تکرار حلقه در داخل for حذف شود، به معنای آن است که شرط همواره درست بوده و یک حلقه بی نهایت ایجاد خواهد شد.

```
for ( ; ; ) {
  // Do something
}
```

(a)

Equivalent

```
while (true) {
  // Do something
}
```

(b)

هشدار

- اضافه نمودن ; بعد از دستور **for** پیش از آغاز بدنه حلقه یک خطای منطقی متداول است !
- برنامه لزوما دچار خطای نحوی یا زمان اجرا نمی شود، اما نتیجه مورد انتظار تولید نخواهد شد، زیرا بدنه حلقه فقط یک بار اجرا خواهد شد.

```
for (int i = 0; i < 2; i++);      Logic Error
{
    System.out.println("Welcome to Java!");
}
```

- به دلیل مشابه حلقه زیر نادرست است !

```
int i = 0;
while (i < 10);   Logic Error
{
    System.out.println("i is" + i);
    i++;
}
```

- در مورد حلقه‌های **do-while** به **;** در انتهای حلقه نیاز داریم.

```
int i = 0;
do {
    System.out.println("i is" + i);
    i++;
} while (i < 10);   Correct
```

از کدام نوع حلقه استفاده کنیم ؟

- سه ساختار حلقه بیان شده، **while do-while** و **for** عملاً معادل یکدیگر هستند.
- برای مثال، یک حلقه **while** در (a) در شکل زیر را همیشه می توان با **for** به صورت (b) نوشت.

```
while (loop-continuation-condition) {
  // Loop body
}
```

(a)

Equivalent

```
for ( ; loop-continuation-condition; )
  // Loop body
}
```

(b)

- یک حلقه **for** در (a) در شکل زیر می تواند جز در موارد معدودی به فرم (b) نوشته شود.

```
for (initial-action;
     loop-continuation-condition;
     action-after-each-iteration) {
  // Loop body;
}
```

(a)

Equivalent

```
initial-action;
while (loop-continuation-condition) {
  // Loop body;
  action-after-each-iteration;
}
```

(b)

توصیه ها

- از دستور حلقه ای استفاده کنید که استفاده از آن آسانتر و بدیهی تر است.
- عموماً یک حلقه **for** در مواقعی استفاده می شود که تعداد تکرارها از قبل معلوم باشد.

برای مثال، بخواهید 100 پیغام را چاپ کنید.

- یک حلقه **while** معمولاً مواقعی استفاده می شود که تعداد تکرارها مشخص نیست و به مقدار نگهبان نیاز می باشد.
- یک حلقه **do-while** در مواقعی به جای **while** استفاده می شود که بخواهیم بدنه حلقه را پیش از چک کردن شرط تکرار حلقه اجرا کنیم.

دستور break

- استفاده از دستور **break** در داخل یک حلقه سبب می شود به کل از حلقه خارج شویم.
- در مثال زیر پس از اینکه مقدار **var** برابر 3 شد، از حلقه خارج می شویم.

```
for (int var = 0; var < 5; var++)
{
    System.out.println("Var is : " + var);

    if(var == 3)
        break;
}
```

- بنابراین برنامه خروجی زیر را چاپ خواهد کرد:

Var is : 0

Var is : 1

Var is : 2

Var is : 3

مثال

- برنامه حدس اعداد را این بار به کمک دستور **break** بنویسید!

```java
import java.util.Scanner;

public class GuessNumberUsingBreak {
    public static void main(String[] args) {
        // Generate a random number to be guessed
        int number = (int)(Math.random() * 101);

        Scanner input = new Scanner(System.in);
        System.out.println("Guess a magic number between 0 and 100");

        while (true) {
        // Prompt the user to guess the number
            System.out.print("\nEnter your guess: ");
            int guess = input.nextInt();

            if (guess == number) {
                System.out.println("Yes, the number is " + number);
                break;
            }
            else if (guess > number)
                System.out.println("Your guess (" + guess + ") is too high");
            else
                System.out.println("Your guess (" + guess + ") is too low");
        } // End of loop
    }
}
```

مثالی دیگر

تحلیل کنید که خروجی کد زیر چیست؟

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

        while (number < 20) {
            number++;
            sum += number;
            if (sum >= 100)
                break;
        }

        System.out.println("The number is " + number);
        System.out.println("The sum is " + sum);
    }
}
```

دستور continue

- استفاده از دستور **continue** در داخل یک حلقه سبب می شود سایر دستورات حلقه در آن تکرار مشخص اجرا نشوند و تکرار بعدی آغاز شود.
- در مثال زیر پس از اینکه مقدار **var2** برابر 1 شد، دستور **println** اجرا نشده و تکرار بعدی از 2 آغاز می شود.

```
for (int var1 = 0; var1 < 3; var1++)
{
    for (int var2 = 0; var2 < 3; var2++)
    {
        if(var2 == 1)
            continue;

        System.out.println("var1:" + var1 + ", var2:" + var2);
    }
}
```

var1:0, var2:0

var1:0, var2:2

var1:1, var2:0

var1:1, var2:2

var1:2, var2:0

var1:2, var2:2

مثال

خروجی برنامه زیر چیست؟

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

        while (number < 20) {
            number++;
            if (number == 10 || number == 11) continue;
            sum += number;
        }

        System.out.println("The sum is " + sum);
    }
}
```

مثال

- برنامه ای بنویسید که از حلقه های تو در تو برای چاپ جدول ضرب استفاده کند.

```java
public class MultiplicationTable {
    /** Main method */
    public static void main(String[] args) {
        // Display the table heading
        System.out.println("           Multiplication Table");

        // Display the number title
        System.out.print("    ");
        for (int j = 1; j <= 9; j++)
            System.out.print("   " + j);

        System.out.println("\n----------------------------------------");

        // Print table body
        for (int i = 1; i <= 9; i++) {
            System.out.print(i + " | ");
            for (int j = 1; j <= 9; j++) {
                // Display the product and align properly
                System.out.printf("%4d", i * j);
            }
            System.out.println();
        }
    }
}
```

مثالی دیگر

- برنامه ای بنویسید که از کاربر دو مقدار صحیح مثبت گرفته و بزرگترین مقسیوم الیه مشترک آنها را محاسبه نماید.

```java
import java.util.Scanner;

public class GreatestCommonDivisor {
    /** Main method */
    public static void main(String[] args) {
        // Create a Scanner
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter two integers
        System.out.println("Enter first integer: ");
        int n1 = input.nextInt();
        System.out.println("Enter second integer: ");
        int n2 = input.nextInt();

        int gcd = 1;
        int k = 2;
        while (k <= n1 && k <= n2) {
            if (n1 % k == 0 && n2 % k == 0)
                gcd = k;
            k++;
        }

        System.out.println("The greatest common divisor for " + n1 + " and " + n2 + " is " + gcd);
    }
}
```

مثالی دیگر

- فرض کنید درآمد کارمندی امسال 10,000$ است و افزایش حقوق هر سال 7% می باشد.

حساب کنید در چند سال حقوق کارمند دوبرابر اولین سال خواهد شد.

```
double tuition = 10000;   int year = 1  // Year 1
tuition = tuition * 1.07; year++;       // Year 2
tuition = tuition * 1.07; year++;       // Year 3
tuition = tuition * 1.07; year++;       // Year 4
...
```

```java
public class FutureTuition {
    public static void main(String[] args) {
        double tuition = 10000;   // Year 1
        int year = 1;
        while (tuition < 20000) {
            tuition = tuition * 1.07;
            year++;
        }

        System.out.println("Tuition will be doubled in " + year + " years");
    }
}
```

کنترل حلقه با Confirmation Dialog (GUI)

یک حلقه کنترل شده با مقدار نگهبان با استفاده از فرم گرافیکی confirmation dialog نیز قابل پیاده سازی است.

پاسخهای کاربر به صورت Yes یا No ادامه یافتن یا خاتمه یافتن حلقه را مشخص می کند.

```
import javax.swing.JOptionPane;

public class SentinelValueUsingConfirmationDialog {
    public static void main(String[] args) {
        int sum = 0;

        // Keep reading data until the user answers No
        int option = 0;
        while (option == JOptionPane.YES_OPTION) {
            // Read the next data
            String dataString = JOptionPane.showInputDialog(
                "Enter an int value: ");
            int data = Integer.parseInt(dataString);

            sum += data;

            option = JOptionPane.showConfirmDialog(null, "Continue?");
        }

        JOptionPane.showMessageDialog(null, "The sum is " + sum);
    }
}
```

خودمون رو بسنجیم

این بخش برای این طراحی شده که در پایان مطالعه این اسلاید، بتونی خودت رو محک بزنی و ببینی آیا مفاهیم رو به خوبی یاد گرفتی یا نه. سوالات زیر رو مرور کن و سعی کن بدون نگاه کردن به متن درس، به اون ها پاسخ بدی.

- چرا مهم هست که حلقه ها پایان یابند ؟ اگر هیچگاه پایان نیابند چه می‌شود؟
- بین حلقه هایی که آموختیم کدامشان تضمین میکند که حداقل یک بار بدنه حلقه اجرا شود؟
- آیا می‌توان چندین متغیر را در بخش مقداردهی اولیه حلقه for تعریف کرد؟ اگر بله، چه محدودیتی برای آنها اعمال می‌شود؟
- سعی کنید با استفاده از حلقه های تو در تو شکل هایی مانند لوزی، هرم ،الماس و .. بسازید.

پایان

در صورت هرگونه سوال یا پیشنهاد میتونید با من در ارتباط باشید :)

gmail: ariazaryab@gmail.com

telegram: @Arriaw