Java Basic for Tester

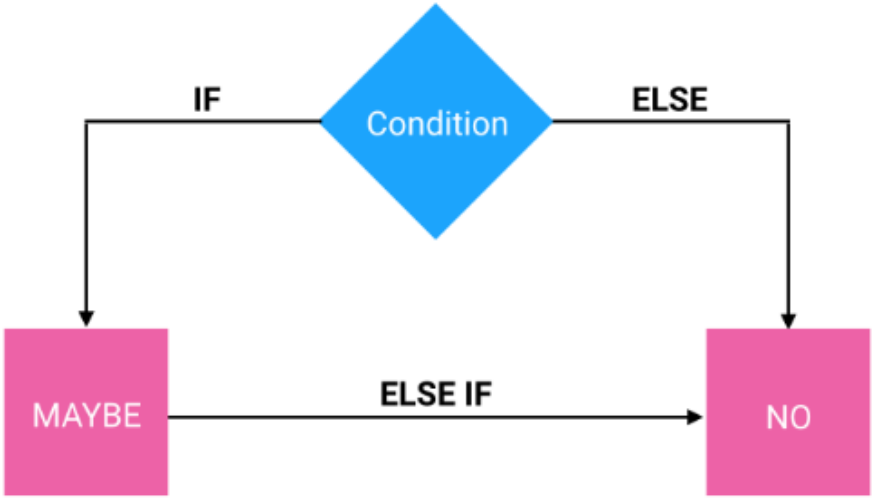
Java Flow Control

Agenda

- *Java if...else*
- *Java switch Statement*
- *Java for Loop*
- *Java for-each Loop*
- *Java while Loop*
- *Java break Statement*
- *Java continue Statement*

Java if...else

In computer programming, it's often desirable to execute a certain section of code based upon whether the specified condition is true or false (which is known only during the run time).



Java if (if-then) Statement

In Java, the syntax of the **if-then** statement is:

```
if (expression) {
    // statements
}
```

Here expression is a **boolean** expression. A boolean expression returns either **true** or **false**.

- if the expression is evaluated to **true**, statement(s) inside the body of **if** (statements inside parenthesis) are executed.
- if the expression is evaluated to **false**, statement(s) inside the body of **if** are skipped from execution.

Java if (if-then) Statement

```java
package JavaFlowControl;

public class IfStatement {
    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.");
        }
        System.out.println("This statement is always executed.");
    }
}
```

```
Run: IfStatement
"C:\Program Files\Java\jdk1.8.0_211
The number is positive.
This statement is always executed.

Process finished with exit code 0
```

Java if...else

Java if...else (if-then-else) Statement

In Java, the syntax of the **if-then-else** statement is:

```
if (expression) {
   // codes
}
else {
  // some other code
}
```

Expression is true.

```
int test = 5;

if (test < 10)
{
   // body of if
}
else
{
   // body of else
}
```

Expression is false.

```
int test = 5;

if (test > 10)
{
   // body of if
}
else
{
   // body of else
}
```

Java if...else (if-then-else) Statement

```java
package JavaFlowControl;

public class IfElse {
    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.");
        }
        else {
            System.out.println("The number is not positive.");
        }

        System.out.println("This statement is always executed.");
    }
}
```

```
Run: IfElse
"C:\Program Files\Java\jdk1.8.0_211\b
The number is positive.
This statement is always executed.

Process finished with exit code 0
```

Java if..else..if Statement

In Java, we have an **if...else...if** ladder, that can be used to execute one block of code among multiple other blocks.

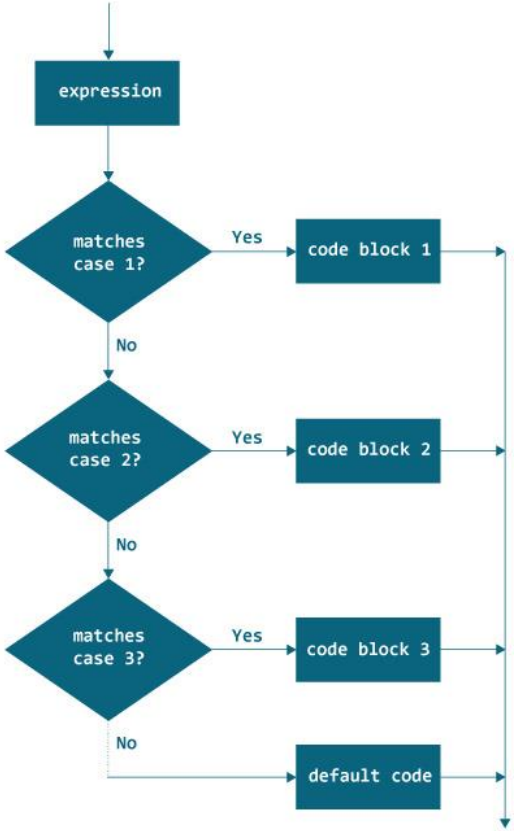
```
if (expression1) {
   // codes
}
else if(expression2) {
   // codes
}.
.
else {
   // codes
}
```

Java if..else..if Statement

```java
package JavaFlowControl;

public class Ladder {
    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.");
        }
        else {
            System.out.println("The number is 0.");
        }
    }
}
```

```
Run:  Ladder
"C:\Program Files\Java\jdk1.8.0_211
The number is 0.

Process finished with exit code 0
```

Java switch Statement




```
switch (variable/expression) {
case value1:
   // statements of case1
   break;

case value2:
   // statements of case2
   break;
   .. .. ...
   .. .. ...
default:
   // default statements
}
```

Java switch Statement

The Java switch statement only works with

- Java Primitive data types: byte, short, char, and int
- Java Enumerated types
- Java String Class
- Java Wrapper Classes: Character, Byte, Short, and Integer.

Java switch Statement

```java
package JavaFlowControl;

public class SwitchStatement {
    public static void main(String[] args) {
        int week = 4;
        String day;
        // switch statement to check day
        switch (week) {
            case 1:
                day = "Sunday";
                break;
            case 2:
                day = "Monday";
                break;
            case 3:
                day = "Tuesday";
                break;
            case 4:
                day = "Wednesday";
                break;
            case 5:
                day = "Thursday";
                break;
            default:
                day = "Invalid day";
                break;
        }
        System.out.println("The day is " + day);
    }
}
```

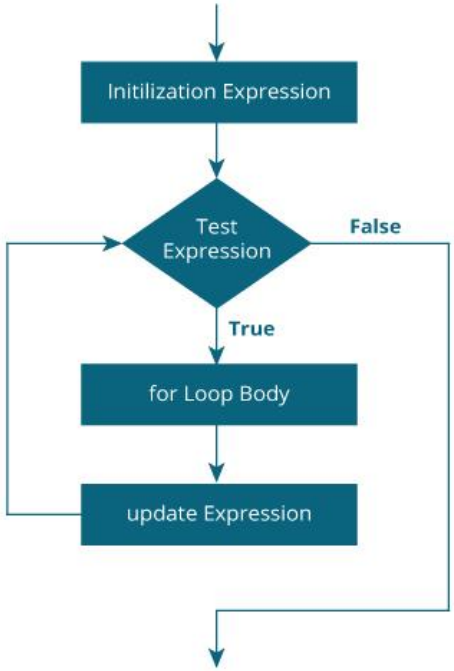
```
Run:  SwitchStatement
"C:\Program Files\Java\jdk1.8.0_211
The day is Wednesday

Process finished with exit code 0
```

Java for Loop

In computer programming, loops are used to repeat a specific block of code until a certain condition is met (test expression is false).

The syntax of **for loop** in Java is:

```
for (initialization; testExpression; update)
{
    // codes inside for loop's body
}
```



Java for Loop

```java
package JavaFlowControl;

public class ForLoop {
    public static void main(String[] args) {

        for (int i = 1; i <= 10; ++i) {
            System.out.println("Line " + i);
        }
    }
}
```

```
Run: ForLoop
"C:\Program Files\Java\jdk1.8.0_21
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10

Process finished with exit code 0
```

- **initialization expression**: int i = 1.e
- **test expression**: i <=10
- **update expression**: ++i

Java for-each Loop

```
class ForLoop {
    public static void main(String[] args) {

        char[] vowels = {'a', 'e', 'i', 'o', 'u'};

        for (int i = 0; i < vowels.length; ++ i) {
            System.out.println(vowels[i]);
        }
    }
}
```

```
class AssignmentOperator {
    public static void main(String[] args) {

        char[] vowels = {'a', 'e', 'i', 'o', 'u'};

        for (char item: vowels) {
            System.out.println(item);
        }
    }
}
```

Java for-each Loop

The syntax of **for each loop** in Java is:

```
for(data_type item : collections) {
    ...
}
```

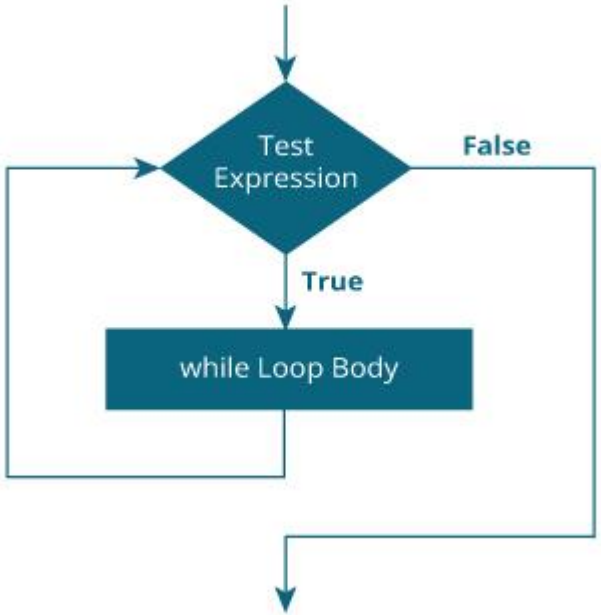
- **collection** - a collection or array that you have to loop through.
- **item** - a single item from the collections.

Java while and do...while Loop

Java while Loop

The syntax of **while** loop in Java is:

```
while (testExpression) {
    // codes inside the body of while loop
}
```



Java while Loop

```java
package JavaFlowControl;

public class WhileLoop {
    public static void main(String[] args) {

        int i = 1;

        while (i <= 10) {
            System.out.println("Line " + i);
            ++i;
        }
    }
}
```

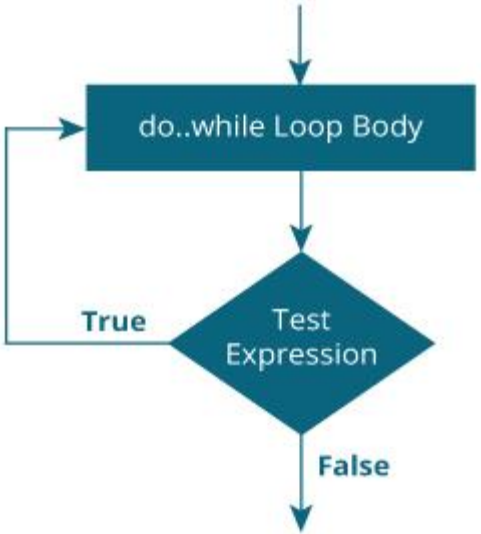
```
Run: WhileLoop
"C:\Program Files\Java\jdk1.8.0_21
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10

Process finished with exit code 0
```

Java while and do...while Loop

Java do...while Loop

The syntax of the do...while loop.

```
do {
   // codes inside body of do while loop
} while (testExpression);
```



Java do...while Loop

```java
package JavaFlowControl;

import java.util.Scanner;

public class DoWhile {
    public static void main(String[] args) {

        Double number, sum = 0.0;
        // creates an object of Scanner class
        Scanner input = new Scanner(System.in);

        do {

            // takes input from the user
            System.out.print("Enter a number: ");
            number = input.nextDouble();
            sum += number;
        } while (number != 0.0);  // test expression

        System.out.println("Sum = " + sum);
    }
}
```

```
Run: DoWhile
"C:\Program Files\Java\jdk1.8.0_211\
Enter a number: 6
Enter a number: 0
Sum = 6.0

Process finished with exit code 0
```

Java break Statement

While working with loops, it is sometimes desirable to skip some statements inside the loop or terminate the loop immediately without checking the test expression.

The **break statement** in Java terminates the loop immediately, and the control of the program moves to the next statement following the loop.

```
while (testExpression) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

```
do {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

Java break Statement

```java
package JavaFlowControl;

public class BreakStatement {
    public static void main(String[] args) {

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

            // if the value of i is 5 the loop terminates
            if (i == 5) {
                break;
            }
            System.out.println(i);
        }
    }
}
```

Run: BreakStatement

```
"C:\Program Files\Java\jdk1.8.0_211
1
2
3
4

Process finished with exit code 0
```

Java continue Statement

The continue statement in Java skips the current iteration of a loop (for, while, do...while, etc) and the control of the program moves to the end of the loop. And, the test expression of a loop is evaluated.

```
while (testExpression) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

```
do {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

Java continue Statement

```java
package JavaFlowControl;

public class ContinueStatement {
    public static void main(String[] args) {

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

            // if value of i is between 4 and 9, continue is executed
            if (i > 4 && i < 9) {
                continue;
            }
            System.out.println(i);
        }
    }
}
```

Run: ContinueStatement

```
"C:\Program Files\Java\jdk1.8.0_21
1
2
3
4
9
10

Process finished with exit code 0
```

Thank you