While &Do-While and For statements

Talent Accelerator
powered by START

Agenda

- While loop
- Do-while loop
- For loop
- Examples & Exercises
- Questions

zalando

If statement

Talent Accelerator
powered by START

What is “While loop”?

Similar to selection structure, a decision is made based on truth or falsity of condition: if condition is fulfilled, then statement(s) is (are) executed. If not, there is no execution.But in this case this execution repeats until the condition becomes false.

The repetition structures can be achieved using three predefined keyword constructions:while, for and do/while.

The while construct, consists of a block of code and a condition. The condition is evaluated first, and if condition set in expression is true, the code within the block is executed until the condition turns into false.

Syntax of while Statement

```
while (condition) {
   // Code to be executed
}
```

zalando

For example, if we want to find the first power of 3 that is greater than 100, we write this part of code:

```
public class Main {
    public static void main(String[] args) {
        int p = 1;
        while (p <= 100) {
            p = p * 3;
        }
        System.out.println(p);
    }
}
```

Usually we have block of statements to repeat the execution. The general construction for this is:

```
while (condition) {
    statement1;
    statement2;
    // ...
    statementN;
}
```

To print out all numbers between 1 and 100, an integer counter should be used. This counter is initialized to 1 and then increased in loop by 1:

```
public class Main {
    public static void main(String[] args) {
        int counter = 1;
        while (counter <= 100) {
            System.out.print(counter + " ");
            counter = counter + 1;
        }
    }
}
```

- When generating a loop, we should take into account that it has to end at some point.Loop Termination: Always ensure a mechanism to break out of the loop to avoid infinite loops. In the example below, the condition counter <= 100 and the increment counter++ work together to make sure the loop terminates.

- Increment/Decrement Operators: Java also supports the increment (++) and decrement (--) unary operators for integers and some other types like char, byte, short, and long. These can be used as either prefix (++x) or postfix (x++) forms.

- Nested Structures: Like if/else nested structures, you can also nest different control structures within a while loop.

For example, for ten values entered, check and print out those that is positive:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int counter = 1, value;

        while (counter <= 10) {
            System.out.print("Enter a value: ");
            value = scanner.nextInt();

            if (value > 0) {
                System.out.println(value + " is a positive number");
            }

            counter++;
        }

        scanner.close();
    }
}
```

Here counter is used only to count the needed number of iterations.

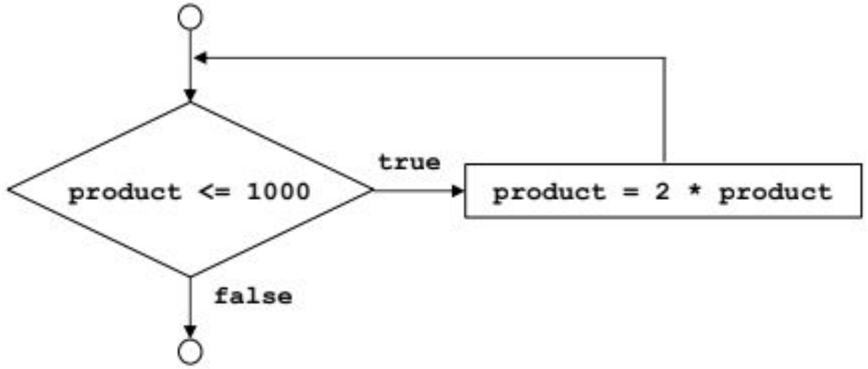
Counter-Controlled Loop

In a counter-controlled loop, you use a counter variable to control the number of loop iterations. The loop will execute as long as the counter variable meets a specified condition. The Java for loop is commonly used for this type of control.

Event-Controlled Loop

In an event-controlled loop, the loop will continue running based on some event or condition, often unrelated to a counter variable. The Java while loop or do-while loop is commonly used for this type of control.

While Structure FlowChart



Talent Accelerator powered by START

"While" exercises

- Problem Statement:
  Write a Java program to find the sum of the first N natural numbers, where N is entered by the user.
  - Solution :

    -
      ```
      int sum = 0;
       int count = 1;
       int N;
       while(count <= N){
       sum += count;
       Count++;
       }
      ```

zalando

“While” exercises

- Problem Statement:
  Create a program that keeps asking for a username and password until the correct ones are entered.
  - Solution :
    -

```
 String correctUsername = "admin";
String correctPassword = "1234";
String username, password;

while(true) {
    if(username.equals(correctUsername) &&
password.equals(correctPassword)) {
        break;
    }
}
```

Do-while loop

What is “do-while”?

The do...while repetition statement is similar to the while statement. In the while statement, the loop-continuation condition test occurs at the beginning of the loop before the body of the loop executes. The do...while statement tests the loop-continuation condition after the loop body executes; therefore, the loop body always executes at least once.

When a do...while terminates, execution continues with the statement after the while clause.

Syntax of if-else Statement

```
do {

   // statement or block of code

} while (condition);
```

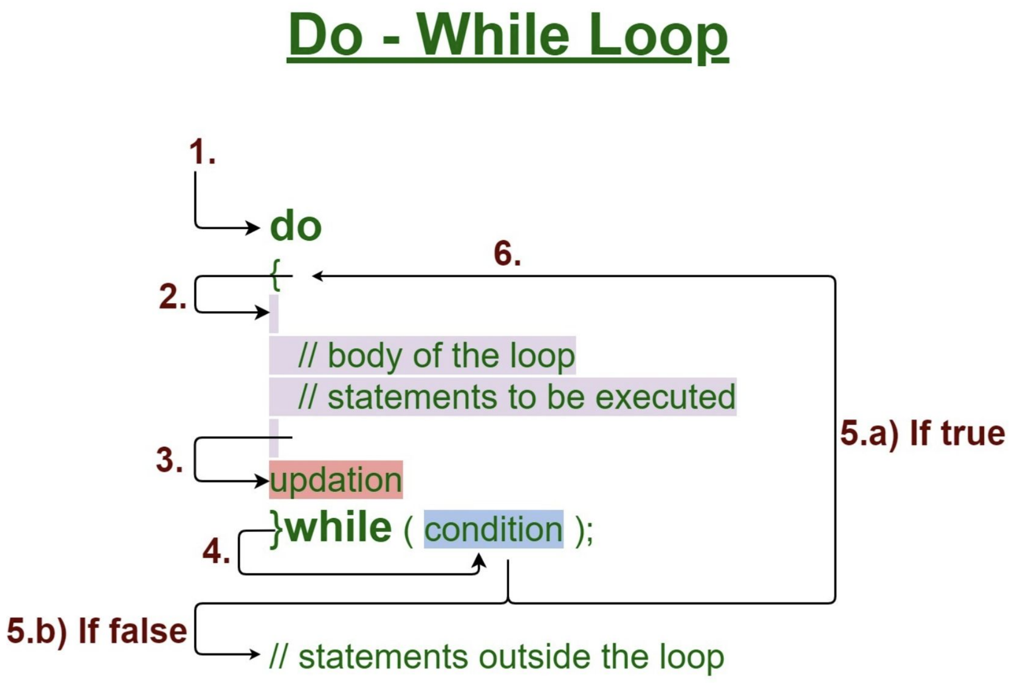
For example, if we use do...while statement to print the numbers 1–10 we will have this code below:

```
public class DoWhileExample {
    public static void main(String[] args) {
        int counter = 1; // Initialize counter to 1

        do {
            System.out.println(counter); // Output the current value of counter (line 12)
            counter++; // Increment counter (line 13)
        } while (counter <= 10); // Loop-continuation condition (line 14)

        // The next statement after the loop (line 16)
        System.out.println("Loop has ended.");
    }
}
```

Do-While Structure FlowChart



Talent Accelerator
powered by START

Break 10 minutes

zalando

For loop in Java

REPETITION STRUCTURE: FOR LOOP IN JAVA

The for statement is designed to simplify the implementation of count-controlled loops. In other words, a counter-based while loop can often be replaced by a for loop for more straightforward code.

Requirements

- A variable to act as a loop counter
- An initial value for the loop counter
- A condition for loop termination
- An increment/decrement statement for the loop counter

The general for construction when we have only one statement to execute is:

```
for (initialization; loopContinuationTest; increment) {
statement;
}
```

To print numbers from 1 to 100 in Java, you should:

Declare a variable that acts as the loop counter: In Java, you would declare a variable of type int. For example: int count.
Initial value for this variable: Set the initial value of the count to 1 (count = 1).
Condition to test for loop termination: The loop should continue as long as count <= 100.
Increment the variable in the loop: After each iteration, you should increment the value of count by 1 (count++).

In Java, you can do the declaration, initialization, loop continuation test, and increment all within the for loop itself.

Considering the above points, the Java for loop would look like:

```
for (int count = 1; count <= 100; count++) {
   System.out.println(count);
}
```

To print numbers from 100 to 1 in Java, you should:

Declare a variable that acts as the loop counter: Like before, this would be a variable of type int. For example: int count.
Initial value for this variable: Set the initial value of count to 100 (count = 100).
Condition to test for loop termination: The loop should continue as long as count >= 1.
Decrement the variable in the loop: After each iteration, you should decrement the value of count by 1 (count--).

Given these points, the Java for loop to accomplish this would look like:

```
for (int count = 1; count <= 100; count += 2) {
    System.out.println(count);
}
```

the following equivalences apply:

- counter += 2 is equivalent to counter = counter + 2
- counter -= 2 is equivalent to counter = counter - 2
- counter *= 2 is equivalent to counter = counter * 2
- counter /= 2 is equivalent to counter = counter / 2
- counter %= 2 is equivalent to counter = counter % 2

This is the same for other mathematical operations. Statements of the form:

```
variable = variable operator expression;
```

Can be written as

```
variable operator = expression;
```

Every for loop can be written as counter based while loop.

Our example for printing numbers from 100 to 1 can be rewritten as:

```
public class WhileLoopExample {
    public static void main(String[] args) {
        int count = 100;
        while (count >= 1) {
            System.out.println(count);
            count--;
        }
    }
}
```

You can use the break and continue keywords in a for loop in Java as well, just as you can with while and do-while loops. Here are some examples to demonstrate their usage:

Using break to exit a for loop:

The break statement will terminate the loop immediately, skipping any remaining iterations.

```
public class BreakExample {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if (i == 5) {
                break;
            }
            System.out.println(i);
        }
    }
}
```

Talent Accelerator
powered by START

Using continue in a for loop:

The continue statement skips the current iteration and jumps to the next iteration of the loop.

```
public class ContinueExample {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            if (i == 5) {
                continue;
            }
            System.out.println(i);
        }
    }
}
```

zalando

Nested Loops and Conditional Structures:

You can also nest loops and conditional structures within a `for` loop.

```
public class NestedExample {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                if (j == 2) {
                    continue;
                }
                System.out.println("i: " + i + ", j: " + j);
            }
        }
    }
}
```

Talent Accelerator
powered by

zalando

Questions?

Thank you for your attention!