Object Oriented Programming – Java 1

Lecture 6

Control Structures: Looping

Dr. Obuhuma James

Description

This topic concludes the concept of control structures by exploring looping as one of the flow control mechanisms. Unlike branching, covered in the previous topic, looping creates a repetitive effect until a certain condition is met. Three statements will be covered to demonstrate the implementation of looping in computer programming. These are the for loop, the while loop, and the do … while loop. The foreach loop, which is best suited for arrays, will be covered in the next topic.

Learning Outcomes

By the end of this topic, you will be able to:

- Comprehend on the concept of looping control structures.
- Demonstrate the various options used to implement looping in programming.
- Differentiate between decision-making and looping control structures.

Overview of Looping Control Structures

Looping is a control structure that allows for the order of execution to take one of two options available based on the outcome of a given condition. Unlike for the case of decision-making, looping allows for repetitive execution as long as the condition holds. Hence, looping control structures are also referred to as repetition or iterative flow control mechanisms. The conditional expression returns a Boolean value that can be either TRUE or FALSE as a determinant of the direction to be taken as shown in Figure 1. There are many situations that require similar statements to be executed several times [2]. For instance, execution of a formula to populate the mean grade for students in a class.

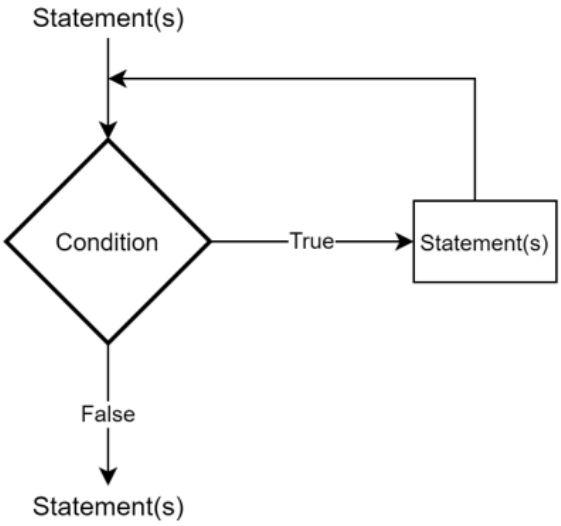

Figure 1. The Concept of Looping Control Structure

There are four ways that can be used to implement looping in Java, just like in the case of other programming languages. These are the for loop, the while loop, the do … while loop, and the foreach loop [1]. The following subsections will explore each of the loop options with an example used for purposes of demonstration. However, the foreach loop will be reserved for coverage in the topic on arrays since it is well suited to work hand in hand with array data.

The for Loop

The for loop is made up of three critical parameters, namely, an initialization, a condition, and a loop counter. The initialization is used to define and initialize a variable whose value serves as the starting point for loop execution. The variable is then used in the conditional expression to set a condition used to determine the number of times the loop should execute and the termination point. Moreover, the same variable is used in the loop counter where it gets updated every time an execution cycle occurs. The updated value for the variable is what is subjected to the conditional expression every time a check is made to determine the next move. The for loop offers a very concise format for execution of loops [1].

Syntax

```
for(initialization; condition; loopcounter){
        Statement(s) to be executed if condition is TRUE;
}
```

Example

Consider the following simple example of a Java program that prints numbers 1 to 5 in a vertical manner.

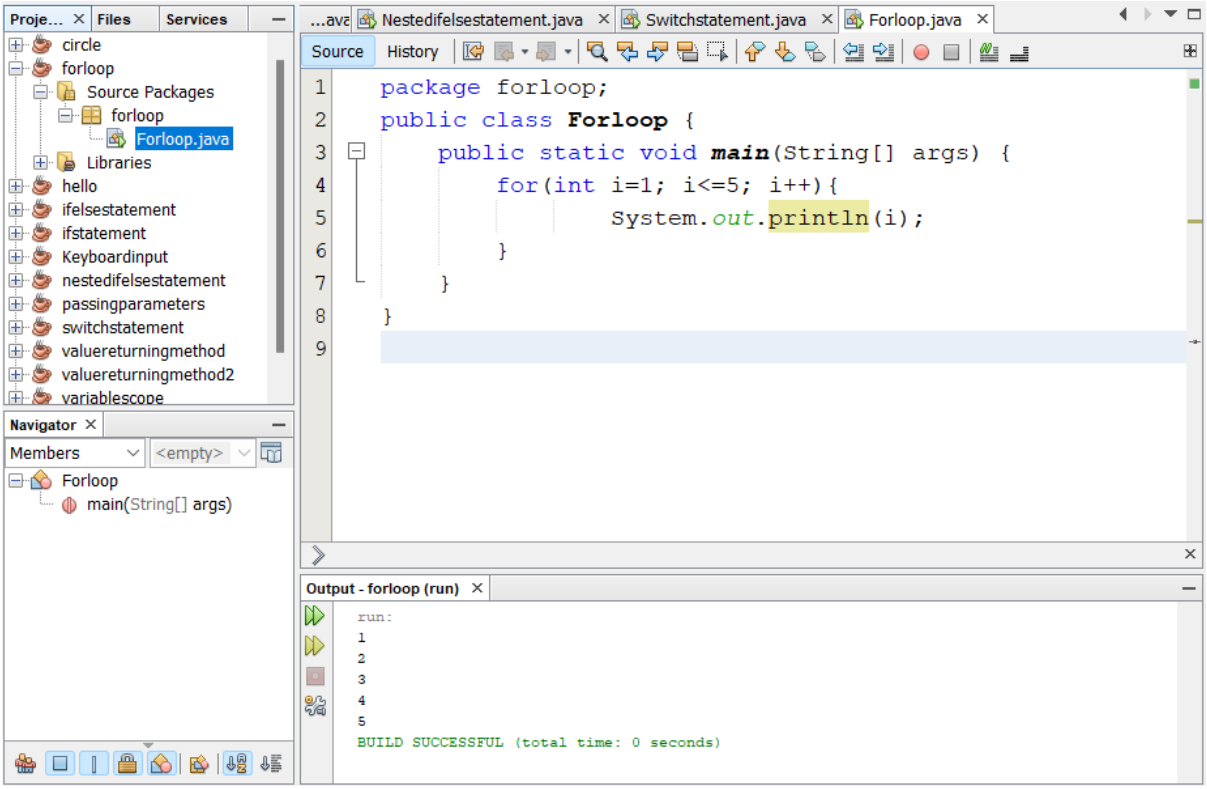

Figure 2. Example of the for Loop

Note that whatever is enclosed in the System.out.print() statement is what is desired for display every time the loop executes. Hence, in this case, the variable i is what is displayed. Since the variable gets updated upon every execution cycle, in this case, an increment by 1, the code segment ends up printing 1 – 5.

The while Loop

The while loop is made up of three critical parameters, namely, an initialization, a condition, and a loop counter, just like the case of the for loop. The initialization is used to define and initialize a variable whose value serves as the starting point for loop execution. The variable is then used in the conditional expression to set a condition to be used to determine the number of times the loop executes and the termination point. Moreover, the same variable is used in the loop counter where it gets updated every time an execution cycle occurs. The updated value for the variable is what is subjected to the conditional expression every time a

check is made to determine the next move. The while loop defines and sets the initialization variable before the loop starts. This approach is similar to the do … while, but it is done differently in the for loop. In the while loop, the condition which is the loop-controlling Boolean expression is the first statement to be executed [1].

Syntax

Initialization;

while(condition){

Statement(s) to be executed if condition is TRUE;

Loopcounter;

}

Example

The following example will print numbers 1 to 5 in a similar manner as was done by the for loop described in the previous subsection.

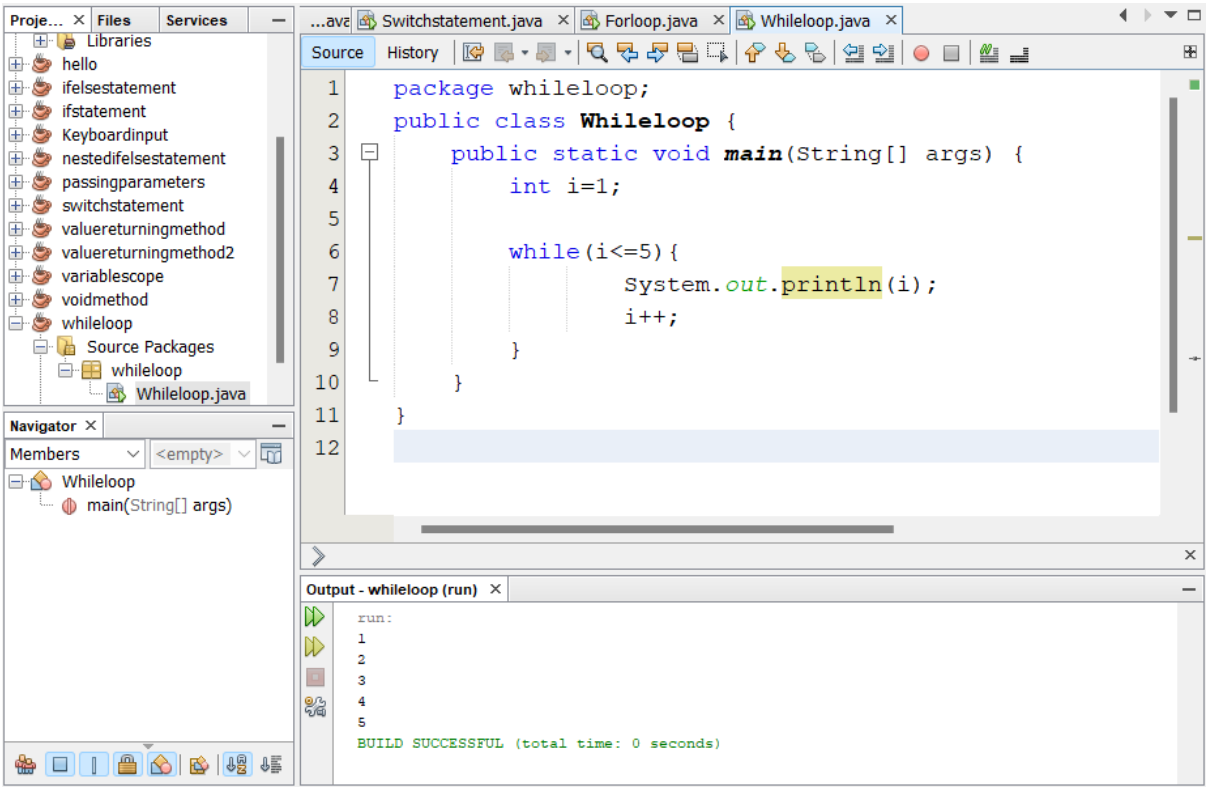

Figure 3. Example of the while Loop

The do … while Loop

The do … while loop is made up of three critical parameters, namely, an initialization, a condition, and a loop counter, just like for the case of the for loop and the while loop. The initialization is used to define and initialize a variable whose value serves as the starting point for loop execution. The variable is then used in the conditional expression to set a condition to be used to determine the number of times the loop executes and the termination point. Moreover, the same variable is used in the loop counter where it gets updated every time an execution cycle occurs. The updated value for the variable is what is subjected to the conditional expression every time a check is made to determine the next move. The unique approach used by the do … while loop compared to the other loops is the fact that the condition which is the loop-controlling Boolean expression is tested at the end after execution of the statements in the loop's body [1].

Syntax

```
Initialization;
do{
        Statement(s) to be executed if condition is TRUE;
        Loopcounter;
} while(condition);
```

Example

The following example will print numbers 1 to 5 in a similar manner as was done by the for loop and while loop described in the previous subsection.

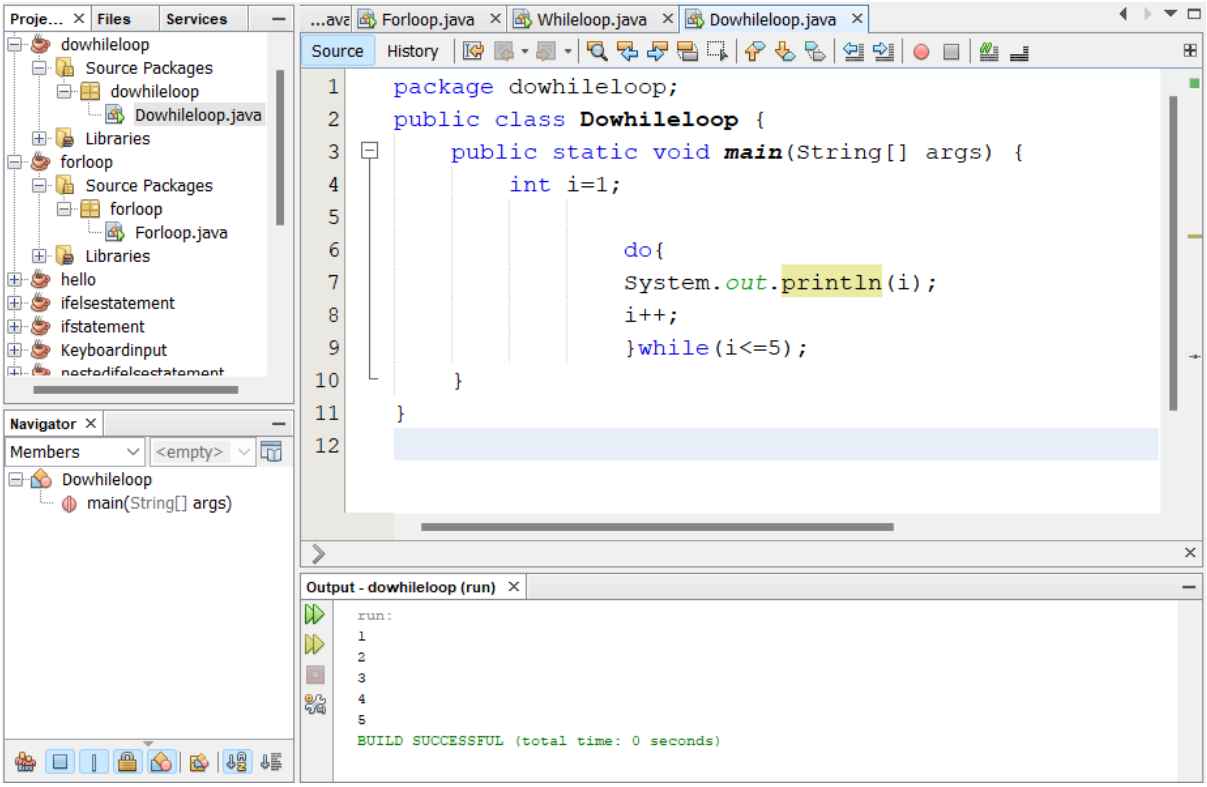

Figure 4. Example of the do … while Loop

Based on the example used to demonstrate the use of the for loop, the while loop, and the do … while loop, it is clear that each of the loops can perform a similar task. It is thus the choice of the programmer to decide which of the loop options to use based on his/her preference.

Application of Comparison and Logical Operators

The conditional expressions used in decision-making and looping statements can be made up of both comparison operators and logical operators. Under all circumstances, the outcome of the evaluation of the conditional expression should be a Boolean value that is either TRUE or FALSE. All the examples demonstrated in previous subsections have employed the use of comparison operators like equals (==) and greater than or equals to (>=). Logical operators are normally applicable in cases where two or more conditions are to be tested at once. For instance, consider the temperature program example used to demonstrate the if statement in the previous topic. Supposing we wish to check for temperatures falling between some range, say 20 to 40, then a logical AND operator will be applied as follows.

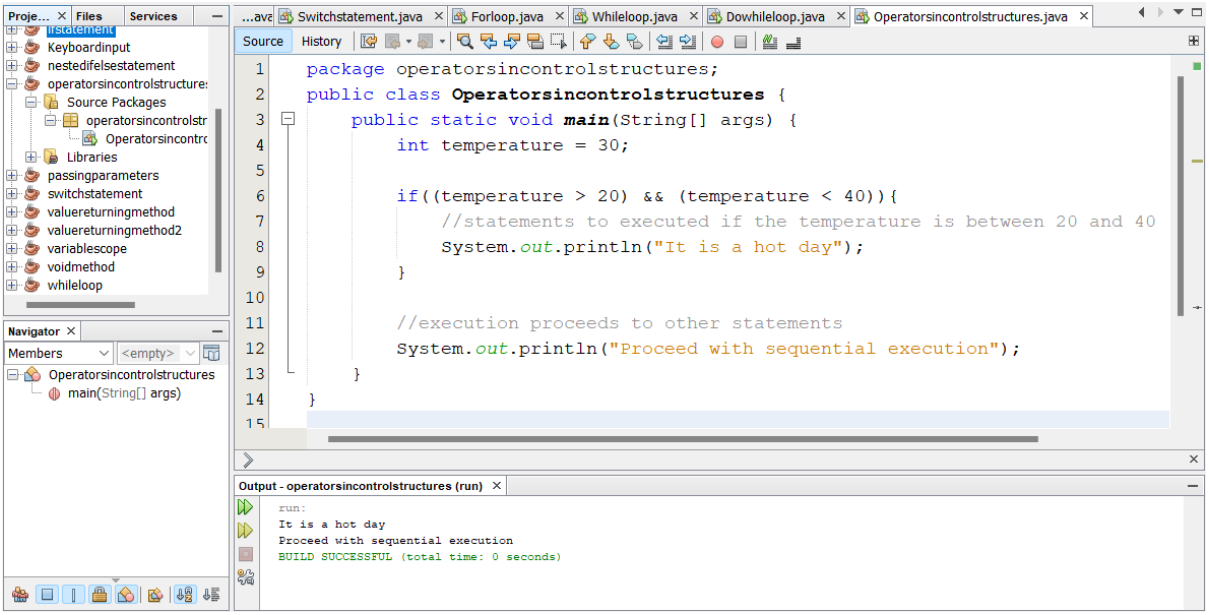

Figure 5. Example Application of Comparison and Logical Operators

Any other logical operator can be applied in a more or less similar manner based on conditions to be tested in any of the control structures covered in the two topics.

Summary

In this topic, the concept of control structures has been expounded and finalised. Looping as the third type of execution of program statements has been covered. Three statements that implement looping have been demonstrated. These are the for loop, the while loop, and the do … while loop. Despite the variance in the structure of the for, while, and do … while loops, they all perform similar tasks and can be applied interchangeably. It is hence the choice of the programmer on which of the three to use in any given case scenario. The foreach loop, not covered in this topic, is limited on its application, making it best suited for arrays. Decision-making and looping control structures covered in the course give programmers a more flexible approach for flow control as they write programs.

Check Points

1. Discuss the concepts behind looping as a type of control structure.
2. Differentiate between decision-making and looping control structures.
3. Demonstrate the concept of the for loop control structure.
4. Difference between the while loop and the do … while loop control structure.
5. Using an example, explain the relationship among the three main loop control structures.

Core Textbooks

1. Joyce Farrell, Java Programming, 7th Edition. Course Technology, Cengage Learning, 2014, ISBN-13 978-1-285-08195-3.
2. Malik, Davender S. JavaTM Programming: From Problem Analysis to Program Design, International Edition, 5th Edition, Cengage Learning.

Other Resources

3. Daniel Liang, Y. "Introduction to Java Programming, Comprehensive." (2011).
4. Malik, Davender S. JavaTM Programming: From Problem Analysis to Program Design, International Edition, 4th Edition, Cengage Learning, 2011.
5. Shelly, Gary B., et al. Java programming: comprehensive concepts and techniques. Cengage Learning, 2012.

References

[1] Farrell, J., Java Programming, 7th Edition. Course Technology, Cengage Learning, 2014, ISBN-13 978-1-285-08195-3.

[2] Malik, D. S., JavaTM Programming: From Problem Analysis to Program Design, International Edition, 5th Edition, Cengage Learning.

[3] Sebester, R. W., Concepts of Programming Languages, 12th Edition, Pearson, 2018, ISBN 0-321-49362-1.