FLOW OF CONTROL: LOOPS

Chapter 4

Java Loop Statements: Outline

- The **`while`** statement
- The **`do-while`** statement
- The **`for`** Statement

Java Loop Statements

- A portion of a program that repeats a statement or a group of statements is called a loop.
- The statement or group of statements to be repeated is called the body of the loop.
- Each repetition of the body is called an iteration.
- There must be a means of exiting the loop.

Why do we need loops ?

- There are many situations in which the same statements need to be executed several times.
- Example:
  - The sum of numbers from 1 to 100
    - compute 1 + 2 + 3 + 4 …… + 100
  - Reading grades for 50 students
    - Read grade of a student
    - repeat the operation 50 times
  - Entering the prices of some items until I am done
    - Read the price of an item
    - Check if the user is done, otherwise repeat the operation

The `while` Statement

- Also called a `while` loop
- A `while` statement repeats while a controlling boolean expression remains true
- The loop body typically contains an action that ultimately causes the controlling boolean expression to become false.

Why do we need that?

A loop that continues to execute **endlessly** is called an **infinite loop,** i.e., when the controlling expression remains always true.

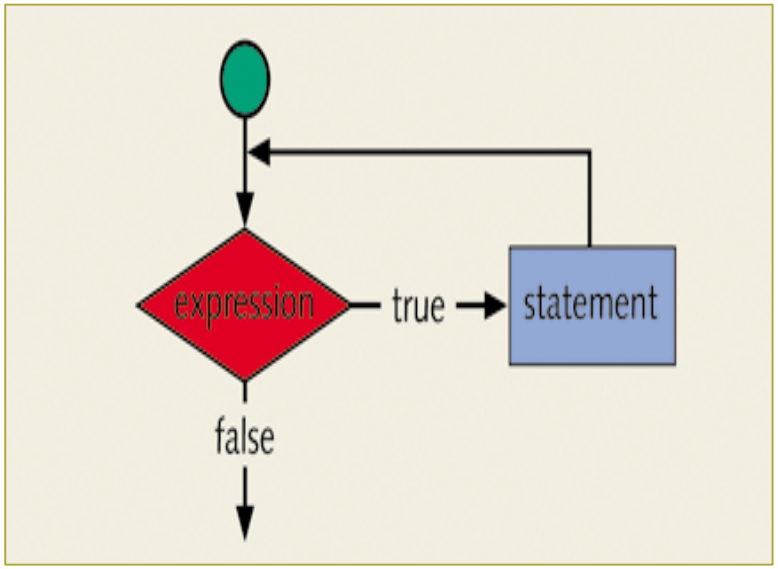
The `while` Statement – Syntax

SYNTAX 1

```
while (logical expression)
  statement 1;
```

SYNTAX 2

```
while (logical expression)
  {
    statement 1;
    statement 2;
    ---
    statement n;
  }
```



sample program, Listing 4.1

```
import java.util.Scanner;
public class WhileDemo
{
    public static void main (String [] args)
    {
        int count, number;
        System.out.println ("Enter a number");
        Scanner keyboard = new Scanner (System.in);
        number = keyboard.nextInt ();
        count = 1;

        while (count <= number)
        {
            System.out.print (count + ", ");
            count++;
        }
        System.out.println ();
        System.out.println ("Buckle my shoe.");
    }
}
```

Sample run – `class WhileDemo`

```
count = 1;
while (count <= number)
{    System.out.print (count + ", ");
     count++;
}
System.out.println ();
System.out.println ("Buckle my shoe.");
```

```
Enter a number:
2
1, 2,
Buckle my shoe.
```

```
Enter a number:
3
1, 2, 3,
Buckle my shoe.
```

```
Enter a number:
0

Buckle my shoe.
```

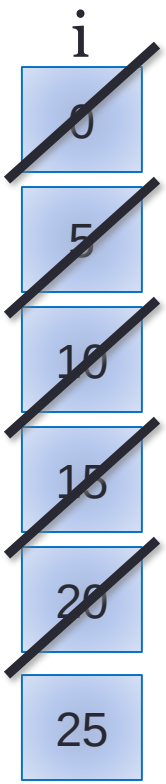
The loop body is iterated zero times.

Sample screen output

The **while** Statement – Tracing example

```
i = 0; //initialize
while (i <= 20) //condition
{
    System.out.println(i + " ");
    i = i + 5; //update
}
System.out.println();
```

Can you **trace** this?



Output

0
5
10
15
20

The `do-while` Statement – Syntax

SYNTAX 1

```
do
   statement;
while (logical expression);
```

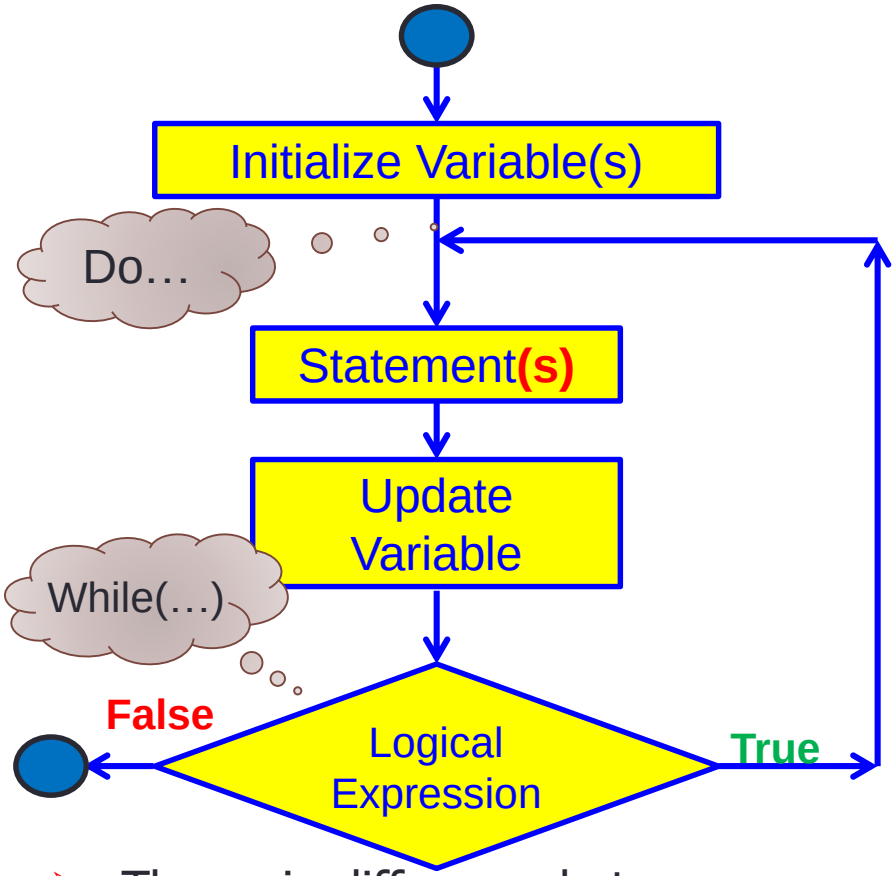
SYNTAX 2

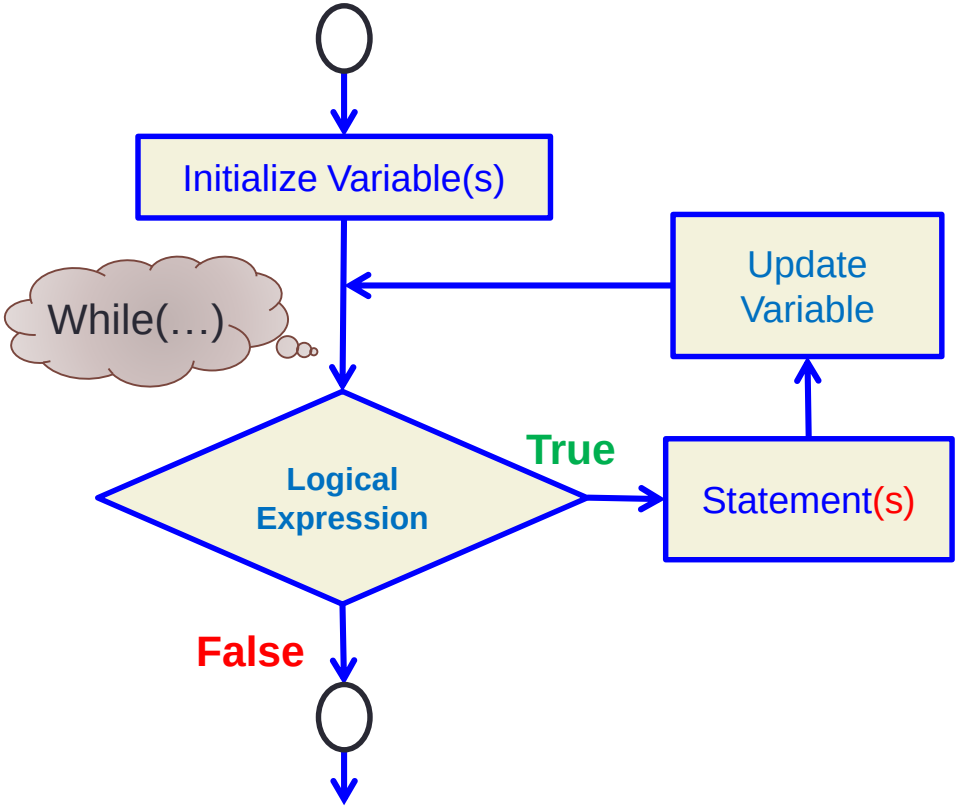
```
do
   {
     statement 1;
     statement 2;
     ---
     statement n;
   }
while (logical expression);
```

Don't forget the semicolon ;

- Also called a `do-while` loop
- Similar to a `while` statement, except that the loop body is executed at least once

The `do-while` vs the `while` Statement





- The main difference between `while` and `do…while` is that:
  - `while` first tests the logical expression, then executes the statements accordingly
  - `do…while` first executes the statements then tests the logical expression. If the condition is `true`, another iteration takes place.

sample program, Listing 4.1

```
import java.util.Scanner;
public class DoWhileDemo
{
    public static void main (String [] args)
    {
        int count, number;
        System.out.println ("Enter a number");
        Scanner keyboard = new Scanner (System.in);
        number = keyboard.nextInt ();
        count = 1;
        do
        {
            System.out.print (count + ", ");
            count++;
        } while (count <= number);
        System.out.println ();
        System.out.println ("Buckle my shoe.");
    }
}
```

The do-while Statement

```
count = 1;
do
{   System.out.print (count + ", ");
    count++;
} while (count <= number);
System.out.println ();
System.out.println ("Buckle my shoe.");
```

```
Enter a number:
2
1, 2,
Buckle my shoe.
```

```
Enter a number:
3
1, 2, 3,
Buckle my shoe.
```

Sample screen output

```
Enter a number:
0
1,
Buckle my shoe.
```

The loop body always executes at least once.

The do-while Statement – Example

Example 1 - do-while

```
int iteration = 0;      //initialize the LCV
do
{
   System.out.println (“Iteration = “ + iteration);
   iteration = iteration + 5;  //update the LCV
}
while (iteration <= 20); //test at the END of the loop
System.out.println (“After loop, iteration = “ + iteration);
```

Example 1 - while

```
int iteration = 0;
while (iteration <= 20)
{
  System.out.println (…);
  iteration = iteration + 5;
}

System.out.println(“After..”);
```

Output

```
Iteration = 0
Iteration = 5
Iteration = 10
Iteration = 15
Iteration = 20
After the loop, iteration = 25
```

Output

```
Iteration = 0
Iteration = 5
Iteration = 10
Iteration = 15
Iteration = 20
After the loop, iteration = 25
```

The do-while Statement – Example

Example 2 - do-while

```
int iteration = 0;      //initialize the LCV
do
{
   System.out.println ("Iteration = " + iteration);
   iteration = iteration + 5;  //update the LCV
}
while (iteration <= -1); //test at the END of the loop
System.out.println ("After loop, iteration = " + iteration);
```

Output

```
Iteration = 0
After the loop, iteration = 5
```

Example 2 - while

```
int iteration = 0;
while (iteration <= -1)
{
  System.out.println (…);
  iteration = iteration + 5;
}

System.out.println("After..");
```

Output

```
After the loop, iteration = 0
```

- The statements in the loop body of a **do-while** are executed at least once.
- Displaying a menu for the user and taking action according to the user's input is better implemented using **do-while**. (see example at the end)

do-while for input validation

- The do…while statement is convenient to use to validate the user input.
- The following code segment forces the user to enter a number between 0 and 100.

```
int score;
do
   {
      System.out.println ("Enter the student's score");
      score = read.nextInt();
   }  while ((score <0) || (score > 100));
```

- The above loop ends only when the two conditions are false; i.e.,
  - score is not less than zero ➔ when score >= 0

    and
  - score is not greater than 100 ➔ when score <=100

Infinite Loops

- A loop which repeats without ever ending is called an infinite loop.
- If the controlling boolean expression never becomes false, a **`while`** loop or a **`do-while`** loop will repeat without ending.

sample program, listing 4.4

```
import java.util.Scanner;
/**
Computes the average of a list of (nonnegative) exam scores.
Repeats computation for more exams until the user says to stop.
*/
public class ExamAverager
{
  public static void main (String [] args)
  {
    System.out.println ("This program computes the average of");
    System.out.println ("a list of (nonnegative) exam scores.");
    double sum;
    int numberOfStudents;
    double next;
    String answer;
    Scanner keyboard = new Scanner (System.in);
```

```
do
{ System.out.println ();
  System.out.println ("Enter all the scores to be averaged.");
  System.out.println ("Enter a negative number after");
  System.out.println ("you have entered all the scores.");
  sum = 0;
  numberOfStudents = 0;
  next = keyboard.nextDouble ();
  while (next >= 0)
  {   sum = sum + next;
      numberOfStudents++;
      next = keyboard.nextDouble ();
  }
  if (numberOfStudents > 0)
     System.out.println("The average is " + (sum/numberOfStudents));
  else
      System.out.println ("No scores to average.");
  System.out.println ("Want to average another exam?");
  System.out.println ("Enter yes or no.");
  answer = keyboard.next ();
} while (answer.equalsIgnoreCase ("yes"));
}}
```

Nested Loops

class ExamAverager

```
Want to average another exam?
Enter yes or no.
yes

Enter all the scores to be averaged.
Enter a negative number after
you have entered all the scores.
90
70
80
-1
The average is 80.0
Want to average another exam?
Enter yes or no.
no
```

Sample screen output

Nested Loops

- The body of a loop can contain any kind of statements, including another loop.
- In the previous example
  - The average score was computed using a **`while`** loop.
  - This **`while`** loop was placed inside a **`do-while`** loop so the process could be repeated for other sets of exam scores.

do...while Programming Problem

Write a complete program that displays a menu to perform an arithmetic operation between two non-integer numbers. The user should select one of the following symbols: +, -, *, /, and %. The menu should contain an option to exit from the program.

INPUT	User's selection (variable: selection, type: char) Two numbers (variable: num1, num2, type: double)
OUTPUT	Result of the operation (variable: result, type: double)

```
// import necessary libraries
import java.util.*;
public class doWhile
{
   static Scanner read = new Scanner (System.in);
   public static void main (String[] args)
     {
          // Declaration section
              double num1, num2, result=0;
              char selection;

          do
            {
              //Display Menu
              System.out.println ("+: addition");
              System.out.println ("-: subtraction");
              System.out.println ("*: multiplication");
              System.out.println ("/: division");
              System.out.println ("%: modulus");
              System.out.println ("x: exit");
              //Get user's selection
              System.out.print ("Enter selection ");   //prompt
              selection = read.next().charAt(0);
```

The menu is displayed at least once

```
22        if ((selection != 'x') && (selection !='X'))
23        {
24          System.out.println ("Enter two double numbers"),
25          num1 = read.nextDouble();
26          num2 = read.nextDouble();
27        switch (selection)
28        {
29          case '+': result = num1 + num2;        break;
30          case '-': result = num1 – num2;        break;
31          case '*': result = num1 * num2;        break;
32          case '/': if (num2 != 0)    result = num1 / num2;
33                    else                  System.out.println ("Invalid divisor");
34                    break;
35          case '%': if (num2 !=0)    result = (num1) % (num2);
36                    else                  System.out.println ("Invalid divisor");
37                    break;
38          default: System.out.println ("Invalid Input");
39        } //end switch
40        System.out.printf ("Result = %.2f", result);
41        } //end if
42      } while ((selection != 'x') && (selection != 'X'));
43      System.out.println ("End of program");
44    } // end main
45  } // end class
```

No need to proceed if the user wants to exit

The condition is tested at the end of the loop