Flow of control – Part 2

Repetition Statement or (Looping Statement) → Allow a program to repeat an action while Boolean statement remains true.

1. **for Repetition Statement**

   Java provides a structure that specifies a number of times to repeat an action, simply called **for loop**. The example below shows a reimplementation of the "Positive, Negative and Zero" application from the part 1 of selection statement by including a for loop.

```
1  import java.util.Scanner;
2  public class Example {
3    public static void main(String [] args) {
4      Scanner input = new Scanner(System.in);
5      /* for loop statement header includes the follwings:
6         1. a variable to count
7         2. a loop-continuation condition
8         3. an increment
9      */
10     for(int counter = 1; counter <= 5; counter++) { // beginnig of loop
11       System.out.print("Enter an integer:");
12       int number = input.nextInt();
13       if(number > 0) {
14         System.out.printf("%d is positive.\n", number);
15       }
16       else if(number < 0) {
17         System.out.printf("%d is negative.\n", number);
18       }
19       else {
20         System.out.printf("%d is zero.\n", number);
21       }
22     } // end of loop
23   }
24 }
```

When the for loop statement begins, the control variable counter is declared and initialized to 1. Next, the program checks the Boolean statement which is a loop-continuation condition, (counter <= 5), whether it is true of false. The two parts, variable creation and loop-continuation condition (Boolean statement) is separated by a semi-colon. If it is false, the program stops repetition. If it is true, the body of loop is executed. In this case, the loop body is from line 11 to 21 and enclosed in the curly brackets on line 10 and 22 respectively. After the loop's body is executed, the counter variable in then incremented by 1 because of the expression counter++; which appears after the second semi-colon.

Sample run:

```
> run Example
Enter an integer: 2

2 is positive.
Enter an integer: -1

-1 is negative.
Enter an integer: 0

0 is zero.
Enter an integer: -4

-4 is negative.
Enter an integer: 5

5 is positive.
```

The program now allows a user to repeat the process of entering an integer and displaying the output for five times.

2. **do…while Repetition Statement**
   Java provides a structure that repeats an action without a counter-controlled loop, simply called **do…while loop**. The loop allows repetition of an action until a false condition is reached or a break statement is executed.

   a. Repeat until a condition is reached.
      The example below shows a reimplementation of the "Positive, Negative and Zero" application from the part 1 of selection statement by including a do…while loop with a specific condition to exit. In the example, the loop repeats until a number entered is 0.

```
1  import java.util.Scanner;
2  public class Example {
3    public static void main(String [] args) {
4      Scanner input = new Scanner(System.in);
5
6      /* declare int number outside the loop so that it can be used
7         both inside and outside the loop at line 24.
8      */
9      int number;
10
11     // do...while loop statement header includes do keyword
12     do { // beginnig of loop
13       System.out.print("Enter an integer or 0 to exit:");
14       number = input.nextInt();
15       if(number > 0) {
16         System.out.printf("%d is positive.\n", number);
17       }
18       else if(number < 0) {
19         System.out.printf("%d is negative.\n", number);
20       }
21       else {
22         System.out.printf("%d is zero.\n", number);
23       }
24     } while (number != 0);// end of loop
25     // end of loop includes while keyword and a Boolean expression to exit the loop
26   }
27 }
```

Sample Run:

```
> run Example
Enter an integer or 0 to exit: 1

1 is positive.
Enter an integer or 0 to exit: -3

-3 is negative.
Enter an integer or 0 to exit: 2

2 is positive.
Enter an integer or 0 to exit: 0

0 is zero.
```

The program now allows a user to repeat the process of entering an integer and displaying the output until a 0 is entered. The do statement on line 12 starts the first iteration of executing the loop's body line 13 – 23. The while statement on line 24 checks the Boolean expression whether the integer entered is 0 or not. The statement literally says that the loop's body must repeat while the integer is not 0, otherwise stop repetition.

b. **Repeat until a break statement is executed.**
The example below shows a reimplementation of the "Positive, Negative and Zero" application from the part 1 of selection statement by including a do…while loop to repeat until break statement is executed to exit. In the example, the program prompts a user to enter 1 or 0 to whether repeat or exit respectively.

```
1  import java.util.Scanner;
2  public class Example {
3    public static void main(String [] args) {
4      Scanner input = new Scanner(System.in);
5
6      /* declare int number outside the loop so that it can be used
7         both inside and outside the loop.
8      */
9      int number;
10
11     // do...while loop statement header includes do keyword
12     do { // beginnig of loop
13       System.out.print("Enter an integer:");
14       number = input.nextInt();
15       if(number > 0) {
16         System.out.printf("%d is positive.\n", number);
17       }
18       else if(number < 0) {
19         System.out.printf("%d is negative.\n", number);
20       }
21       else {
22         System.out.printf("%d is zero.\n", number);
23       }
24
25       // prompt for an input used to define repetition or exit
26       System.out.print("\nDo you want to repeat?\nEnter 1 to repeat : 0 to exit");
27       int con = input.nextInt();
28       if(con == 1) {
29         continue; // repeat
30       }
31       else if(con == 0) {
32         break; // exit
33       }
34     } while(true);// end of loop
35     // end of loop includes while keyword and a Boolean expression to exit the loop
36   }
37 }
```

Sample Run:

```
> run Example
Enter an integer:  23

23 is positive.

Do you want to repeat?
Enter 1 to repeat : 0 to exit  1
Enter an integer:  -100

-100 is negative.

Do you want to repeat?
Enter 1 to repeat : 0 to exit  1
Enter an integer:  0

0 is zero.

Do you want to repeat?
Enter 1 to repeat : 0 to exit  0
```

The program now allows a user to repeat the process of entering either 1 or 0 to repeat or exit respectively. On line28, "if statement" is used to check the "con" variable whether it is 1 or not. If "con" is 1, then "continue" is used to repeat the loop's body. On line 31, "else if statement" is used to check the "con" variable whether it is 0 or not. If "con" is 0, then "break" statement is used to stop repetition.

3. **while Repetition Statement**

Java provides another structure that repeats an action without a counter-controlled loop, simply called **while loop**. The loop also allows repetition of an action until a false condition is reached or a break statement is executed.

The example below shows a reimplementation of the "Positive, Negative and Zero" application from the part 1 of selection statement by including a while loop to repeat until break statement is executed to exit. This program is similarly to the "do…while" loop above except it begins with "while" statement instead of "do" statement. In the example, the program prompts a user to enter 1 or 0 to whether repeat or exit respectively.

a. Repeat until a condition is reached.

```
1  import java.util.Scanner;
2  public class Example {
3    public static void main(String [] args) {
4      Scanner input = new Scanner(System.in);
5
6      /* declare int number outside the loop so that it can be used
7         both inside and outside the loop.
8      */
9      int number;
10     System.out.print("Enter an integer or 0 to exit:");
11     number = input.nextInt();
12
13     // while loop statement header includes do keyword
14     while(number != 0) { // beginnig of loop
15
16       if(number > 0) {
17         System.out.printf("%d is positive.\n", number);
18       }
19       else if(number < 0) {
20         System.out.printf("%d is negative.\n", number);
21       }
22       else {
23         System.out.printf("%d is zero.\n", number);
24       }
25
26       System.out.print("Enter an integer or 0 to exit:");
27       number = input.nextInt();
28     } // end of loop
29   }
30 }
```

b. **Repeat until a break statement is executed.**

```
1  import java.util.Scanner;
2  public class Example {
3    public static void main(String [] args) {
4      Scanner input = new Scanner(System.in);
5
6      /* declare int number outside the loop so that it can be used
7         both inside and outside the loop.
8      */
9      int number;
10
11     // while loop statement header includes do keyword
12     while(true) { // beginnig of loop
13       System.out.print("Enter an integer:");
14       number = input.nextInt();
15       if(number > 0) {
16         System.out.printf("%d is positive.\n", number);
17       }
18       else if(number < 0) {
19         System.out.printf("%d is negative.\n", number);
20       }
21       else {
22         System.out.printf("%d is zero.\n", number);
23       }
24
25       System.out.print("\nDo you want to repeat?\nEnter 1 to repeat : 0 to exit");
26       int  con = input.nextInt();
27
28       if(con == 1) {
29          continue; // repeat
30       }
31       else if(con == 0) {
32          break; // exit
33       }
34     } // end of loop
35   }
36 }
```