Repetition Control Structures

Module 5 CISS-110

Muhammad Rashid Bhatti

Repetition Control Structures

Certain actions in computer programs are done again and again till the complete work is done. For example, if we are asked to display numbers from 1 to 10, one way of this will be to do it in sequence 1 to 10. The other smarter way of performing the same task is to rerun steps again and again in a loop and change the number values from 1 to 10.

There are three commonly used loops in Java:

1) **WHILE Loop:**

The WHILE loop tests the loop entry condition first and if the condition is true then program enters the loop. After that the task mentioned inside loop is performed repeatedly till the loop entry condition becomes false. Then the program enters the next statement after the loop. In above example, we shall design following algorithm

```java
WHILE (number is greater than 0 and less than 11) {
    Display number
    number = number + 1
}
```

2) **DO-WHILE Loop:**

DO-WHILE loop executes the loop first and then checks the condition to see if it should execute again. Thus, a DO-WHILE will always execute at least one time.

```java
DO {
    Display number
    number = number + 1
} WHILE (number is greater than 0 and less than 11)
```
3) **FOR Loop:**

The FOR loop has a general format of:

FOR (initialize, test, change) {

}

In above example to generate table, we can use something like this

For(number=1, number<11, number= number+1) {

    Display number

}

**ALL THREE LOOPS COMBINED IN ONE PROGRAM:**

The practice program 5.1 is a good example of all three loop types combined in one Java program. We are displaying numbers from 1 to 10 through all three loops. This should help you to understand that all loops can do same task and replace each other.

Program 5.1

```java
// for, do-while, while loops together
public class Count {

    public static void main(String[] args) {

        int count;

        // for loop
        System.out.println(" ");
        System.out.println("For Loop Example ");
        System.out.println("------------- ");
        for(count=1; count<11; count=count+1){
            System.out.println("Count, using for loop, is: " + count);
        }

        // while statement
        System.out.println(" ");
        System.out.println("While Loop Example ");
        System.out.println("------------- ");
        // initialize count again to 1
        count = 1;
        while (count < 11) {
            System.out.println("Count, using while loop, is : " + count);
            count++;
        }
    }
}
```
// do-while
System.out.println(" ");
System.out.println("Do-While Loop Example ");
System.out.println("--------------------- ");
// initialize count again to 1
count = 1;
do {
    System.out.println("Count, using do-while loop, is: " + count);
count++;
} while (count < 11);


Output:

For Loop Example
------------------
Count, using for loop, is: 1
Count, using for loop, is: 2
Count, using for loop, is: 3
Count, using for loop, is: 4
Count, using for loop, is: 5
Count, using for loop, is: 6
Count, using for loop, is: 7
Count, using for loop, is: 8
Count, using for loop, is: 9
Count, using for loop, is: 10

While Loop Example
-------------------
Count, using while loop, is : 1
Count, using while loop, is : 2
Count, using while loop, is : 3
Count, using while loop, is : 4
Count, using while loop, is : 5
Count, using while loop, is : 6
Count, using while loop, is : 7
Count, using while loop, is : 8
Count, using while loop, is : 9
Count, using while loop, is : 10

Do-While Loop Example
---------------------
Count, using do-while loop, is: 1
Count, using do-while loop, is: 2
Count, using do-while loop, is: 3
Count, using do-while loop, is: 4
Count, using do-while loop, is: 5
Count, using do-while loop, is: 6
Count, using do-while loop, is: 7
Count, using do-while loop, is: 8
Count, using do-while loop, is: 9
Count, using do-while loop, is: 10

**FOR LOOP EXAMPLE:**

In practice program 5.2 we are using for loop to generate the table. The program starts by asking user to give number input. The program uses user input to generate multiplication table for that number. The program uses FOR loop and the loop starts from 1 and ensures that loop entry condition is valid, which is number is less than 11. Thus it continues repeatedly to generate the table rows one by one. The increment of count ensures that table is generated from 1 to 10. As soon as the variable count reaches 11, then loop entry condition become false and program execution stops.

Program 5.2:

```java
import java.util.Scanner;

// for loop generate a multiplication table
public class MultiplicationTable {
    static Scanner kb = new Scanner(System.in);
    
    public static void main(String[] args) {
        int n;
        // generate message for user to input an integer
        System.out.print("Please enter an integer: ");
        // take input of int
        n = kb.nextInt();
        // use for loop to generate the table
        for (int count = 1; count<11; count++) {
            System.out.println(n + "*" + count + " = " + n*count);
        }
    }
}
```

Output:

```
Please enter an integer: 4
4*1 = 4
4*2 = 8
4*3 = 12
```
\[
\begin{array}{l}
4 \times 4 = 16 \\
4 \times 5 = 20 \\
4 \times 6 = 24 \\
4 \times 7 = 28 \\
4 \times 8 = 32 \\
4 \times 9 = 36 \\
4 \times 10 = 40 \\
\end{array}
\]

**WHILE LOOP EXAMPLE:**

Practice program 5.3 is a good example of while loop. It also teaches you the basic concept of taking user input and providing output graphically using Java swing technology. Java swing is part of Graphical User Interface technology. Java swing will be discussed in detail in later modules. It is used to generate user friendly input and output. For this module, it is important to understand that we are importing javax.swing package. Then, we shall use JOptionPane class in javax.swing package.

If you go to java doc API at [http://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html](http://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html) you shall be able to get more information about some useful classes that java swing technology uses. For this, module please read the details about JOptionPane class.

Practice program 5.3 is providing familiar output as you saw in practice program 5.2 but using while loop and with a better Graphic User Interface. We are also using StringBuffer class. Java uses two classes to manipulate character strings:

- **String Class**: is used to manipulate character strings that cannot be changed
- **The StringBuffer class** is better in treatment of character strings because it works better with strings that could be appended or modified.

As a good programming practice, always use StringBuffer class, when you want to append characters to the original string.

Program 5.3:

```java
import javax.swing.*;

public class MultiplicationTableGUI
{
    public static void main(String[] args)
    {
        // while loop counter variable initialized to 1
```
```java
int count = 1;
int inputNumberInteger;
// output buffer is a special type of string variable that
// we can use to concatenate string values
StringBuffer outputMessage = new StringBuffer();

// generate input message for user through graphic user
// interface to give input of number between 1 and 20
String inputNumberStringForm = JOptionPane.showMessageDialog("\n Please enter an integer between 1 and 20");
// Convert that number from string to integer
inputNumberInteger = Integer.parseInt(inputNumberStringForm);

// check that count is less than 11 since count is
// initialized to 1 so this will generate table from 1 to 10
while ( count < 11 )
{
    // generate each row of table by first displaying input
    // number then count and then result of multiplication
    outputMessage.append(inputNumberStringForm + " * " + count + " = " + inputNumberInteger*count + "\n");
    count++;
}
// Show output in graphical user interface
JOptionPane.showMessageDialog(null, outputMessage, "Results", JOptionPane.PLAIN_MESSAGE);
```

Output:

![Image of GUI window showing the input dialog](image_url)
DO-WHILE LOOP EXAMPLE:

Practice program 5.4 is a good example of DO-WHILE loop. A good use of DO-WHILE loop is to get menu input from user in a loop. Please note that because of DO-WHILE loop, the input user interface does not go away till user enters the correct input.

Program 5.4:

```java
import javax.swing.*;

public class BankMenuGUILoop {
    public static void main(String[] args) {
        int menuItemNumber;
        // This loop will continue to execute and take input from user till
        // they give correct input in the range (1-4). Thus DO-WHILE is a
        // great way to ensure correct input from user.
        do {
            // User input message screen is generated by following line of code
            String inputNumberStringForm = JOptionPane.showInputDialog("Menu
            \n1. Open an account\n2. Deposit\n3. Withdraw\n4. Close an account\n0. Quit\n\n");
            // convert user string input to integer form
            menuItemNumber = Integer.parseInt(inputNumberStringForm);
            while (menuItemNumber < 0 || menuItemNumber > 4); // Based on user choice the correct piece of code shall be selected.
            switch(menuItemNumber) {
```
```java
    case 1:
        JOptionPane.showMessageDialog(null, "You have selected menu item: \n 1. Open an account", "Results", JOptionPane.PLAIN_MESSAGE);
        break;

    case 2:
        JOptionPane.showMessageDialog(null, "You have selected menu item: \n 2. Deposit", "Results", JOptionPane.PLAIN_MESSAGE);
        break;

    case 3:
        JOptionPane.showMessageDialog(null, "You have selected menu item: \n 3. Withdrawl", "Results", JOptionPane.PLAIN_MESSAGE);
        break;

    case 4:
        JOptionPane.showMessageDialog(null, "You have selected menu item: \n 4. Close an account", "Results", JOptionPane.PLAIN_MESSAGE);
        break;

    case 0:
        JOptionPane.showMessageDialog(null, "Good Bye", "Results", JOptionPane.PLAIN_MESSAGE);
        System.exit(0);
    }
}

Output:

![Image of dialog box with menu options and selected option 3]
**BREAK statement**

BREAK statements are:

- used to exit early from a loop
- used to skip remainder of switch structure

If condition is met, loop is exited immediately

**CONTINUE statement:**

CONTINUE statements are commonly used in all three loops (FOR, DO-WHILE, WHILE) to skip remaining statements in that loop iteration. The program control proceeds with the next iteration of the loop. When CONTINUE statement is executed in DO-WHILE loop, WHILE expression is evaluated after continue.

**Difference between BREAK and CONTINUE Statements:**

The difference between BREAK and CONTINUE statements with respect to loop is that Break leaves a loop, CONTINUE jumps to the next iteration but stays within the loop.