

Repetition using Java – Loops

Repetition is a programming concept that allows the computer to run one or more commands multiple times, without having to write them out each time. Instead, the commands are placed inside a *looping structure*, where they will be executed over and over, until some *condition* is met (i.e., becomes true).

Consider an example from cooking, where we have to stir a gravy until all of the lumps are gone.

Using *pseudocode*, the instructions might look like

```
while (there are lumps in the gravy)
  give the gravy a stir
end while
```

The idea is to keep checking for lumps. If there are still lumps, stir the gravy. If the lumps are gone, the loop ends.

The While Loop – Decision at the Beginning

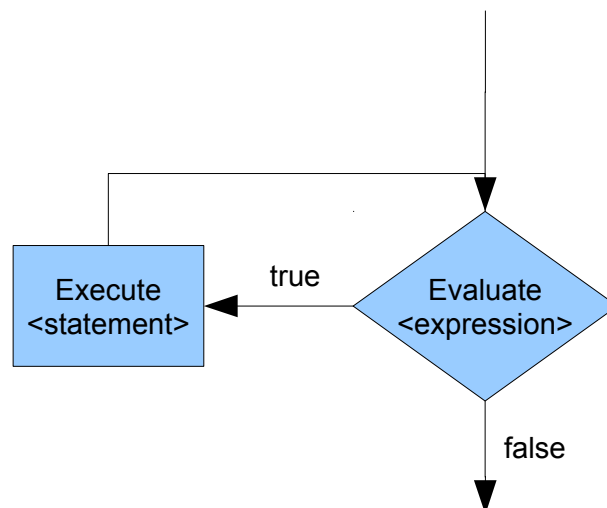
The execution of a *while loop* is similar to the execution of a *simple if statement*. A condition is checked on the first line (`true` or `false`). If the condition is `true`, we enter the loop and execute any commands found within the *body of the loop*. If the condition is `false`, we exit the loop without executing any commands.

```
if (<boolean expression>)
{
  <statements>
}
```

```
while (<boolean expression>)
{
  <statements>
}
```

In other words, the `while` loop is like an `if` statement that keeps repeating while the `boolean expression` is `true`.

We can also represent a `while` loop using a *flowchart*, as we did with `if-else` statements.



Repetition using Java – Loops

Example 1 – ask the user for integer values, and then outputs the square of the value. Continues to do this until the user enters zero, which causes the loop to exit.

```
class PrintSquares
{
    public static void main (String[] args)
    {
        System.out.print("Give an integer (or 0 to quit): ");
        int value = In.getInt();
        while (value != 0)
        {
            System.out.println(value + " squared is " + value*value);
            System.out.print("Next integer (or 0 to quit): ");
            value = In.getInt();
        }
    }
}
```

Some important notes about this program:

1. It is necessary to **initialize** the variable `value` before testing the condition of the while loop. Otherwise, `value` would have been undefined when the loop was first checked, and an error would occur.
2. There must be some way within the loop that will eventually cause the loop to exit. In this case, we keep asking the user for a new `value`, and we will exit when `value` is zero. We say zero acts as the **sentinel** value for this loop.
3. If there is no *sentinel* value, or if there is no way to set the sentinel value, the loop will never end. This results in an **infinite loop**, which must be terminated from the operating system of your computer. *This is almost always a programmer error.*
4. If the first value entered in the program were zero (the *sentinel* value), the contents of the loop would never execute.

Repetition using Java – Loops

Example 2 – Find the sum of a set of marks as they are entered one at a time.

```
class AddMarks
{
    public static void main (String [] args)
    {
        /**
         * This program reads the marks obtained for a
         * student and adds them together.
         **/

        System.out.println("Submit marks (value < 0 to stop");
        int totalMarks = 0;           // the sum of all marks entered
        int nextMark = In.getInt();

        while (nextMark >= 0)
        {
            totalMarks = totalMarks + nextMark;
            System.out.println("Next mark:");
            nextMark = In.getInt();
        }
    }
}
```

In this program, the *initialization* includes both setting the variable `totalMarks` to zero, as well as reading the initial value of `nextMark`. The value of `nextMark` is also read inside the loop, since exiting the loop depends on `nextMark` being less than zero (the *sentinel value*).

Repetition using Java – Loops

Example 3 – Find the average mark from several marks entered by the user. This builds on the previous example. Changes to the previous program are in **bold**.

```
class AverageMark
{
    public static void main (String [] args)
    {
        /**
         * This program reads the marks obtained for a
         * student and finds the average mark.
         * It prints the average mark, rounded to the
         * nearest integer.
         */

        System.out.println("Submit marks (value < 0 to stop");
        int totalMarks = 0;           // the sum of all marks entered
        int numberOfMarks = 0;         // the number of marks entered so far
        int nextMark = In.getInt();

        while (nextMark >= 0)
        {
            totalMarks = totalMarks + nextMark;
            numberOfMarks = numberOfMarks + 1; // total marks entered
            System.out.println("Next mark:");
            nextMark = In.getInt();
        }

        // only calculate average if number of marks is more than zero
        if (numberOfMarks > 0)
        {
            int average = Math.round((float)totalMarks/numberOfMarks);
            System.out.println("Average for " + numberOfMarks +
                           " students is " + average);
        }
    }
}
```

In this program, the *initialization* includes both setting the variables `totalMarks` and `numberOfMarks` to zero, as well as reading the initial value of `nextMark`. The value of `nextMark` is also read inside the loop, since exiting the loop depends on `nextMark` being less than zero (the *sentinel value*). Notice that the program only attempts to find the average if at least one mark has been entered (which prevents dividing by zero if no marks are entered).

Repetition using Java – Loops

Exercises

1. What is the minimum number of times that the body of a while statement can be executed?
2. What is the maximum number of times that the body of a while statement can be executed?
3. What will be printed by this fragment? Determine your answer on paper before you check the answer by running a program (Note: you can test this code by copying and pasting into the *Interactions Pane* in DrJava).

```
int m = 10;
int n = 0;
while (m > n)
{
    System.out.println(m + " " + n);
    m = m - 1;
    n = n + 2;
}
```

4. Write a program that asks the user for an `integer` *sentinel value* which will be used to end the program. Once you have the sentinel value, repeatedly ask the user for `integer` values until they give the sentinel value, then stop the program.
5. Write a program that will calculate the sum of the first N positive integers, where N is a value input by the user. For example, if the user were to input 3, the answer would be 6 ($1 + 2 + 3 = 6$). The output should be something like:
"The sum of the first 3 integers is 6"
6. Write a program that prints the square root of each (double) number input by the user. It will continue to ask for new numbers until a negative value is input by the user, which should end the program. While the program is running, the output and input might look as follows:

Sample Output	Input
Enter a positive number (or a negative number to quit):	49
The square root of 49 is 7	
Enter another positive number (or a negative to quit):	3.14159
The square root of 3.14159 is 1.77245	
Enter another positive number (or a negative to quit):	-1

Repetition using Java – Loops

7. Write a program that prompts the user for a sequence of integers using zero as a sentinel. The program should count the number of times that consecutive values are equal. For example, if the input is

3 6 7 7 4 4 4 6 0

then the program should determine that there are three cases in which consecutive values are equal.

Repetition using Java – Loops

Solutions

1. Zero, if the condition is false before the loop is ever entered
2. Infinite, if the condition is always true (an infinite loop is almost always a programmer error)
3. This example gives a good opportunity to perform a *manual walkthrough* of the code. A walkthrough is where you go through the code by hand (i.e., pencil and paper), keeping track of the important variables and other important events. For example:

m	n	m > n	output
10	0	TRUE	10 0
9	2	TRUE	9 2
8	4	TRUE	8 4
7	6	TRUE	7 6
6	8	FALSE	

Therefore, the output would be:

```
10 0
9 2
8 4
7 6
```

4.

```
class WhileLoopsExercise4
{
    public static void main (String [] args)
    {
        int sentinel;
        int input;

        System.out.println("What is the sentinel value?");
        sentinel = In.getInt();

        System.out.println("Enter a value (" + sentinel + " to stop)");
        input = In.getInt();

        while (input != sentinel)
        {
            System.out.println("Enter a value (" + sentinel + " to stop)");
            input = In.getInt();
        }
        System.out.println("End of program...");
    }
}
```

Repetition using Java – Loops

```
5. class WhileLoopsExercise5
{
    public static void main (String [] args)
    {
        int maxN;
        int sum = 0;
        int currentN = 1; // start from 1

        System.out.println("Enter a positive integer value");
        maxN = In.getInt();

        while (currentN <= maxN)
        {
            sum = sum + currentN;
            currentN = currentN + 1;
        }

        System.out.println("The sum of the first " + maxN
                           + " integers is " + sum);
    }
}
```

If the user enters a negative number, or zero, the program will enter an infinite loop. It is possible to improve the program in several ways, one of which is shown below.

```
class WhileLoopsExercise5
{
    public static void main (String [] args)
    {
        int maxN;
        int sum = 0;
        int currentN = 1; // start from 1

        System.out.println("Enter a positive integer value");
        maxN = In.getInt();

        if (maxN >= 1)
        {
            while (currentN <= maxN)
            {
                sum = sum + currentN;
                currentN = currentN + 1;
            }

            System.out.println("The sum of the first " + maxN
                               + " integers is " + sum);
        }
        else
        {
            System.out.println("Error: " + maxN +
                               " is not a positive integer!");
        }
    }
}
```

Repetition using Java – Loops

6. class WhileLoopsExercise6

```
{
    public static void main (String [] args)
    {
        double number;    // user input
        double root;      // calculated square root

        System.out.println("Enter a positive number (or negative to quit)");
        number = In.getFloat();

        while (number >= 0)
        {
            root = Math.sqrt(number);
            System.out.println("The square root of " + number + " is " + root);
            System.out.println("Enter a positive number (or negative to quit)");
            number = In.getDouble();
        }
    }
}
```

7. class WhileLoopsExercise7

```
{
    public static void main (String [] args)
    {
        int number;    // user input
        int previousNumber = 0;
        int count = 0;    // track number of consecutive occurrences

        System.out.println("Enter an integer value (or zero to quit)");
        number = In.getInt();

        // if the user enters zero right away, we never enter the loop
        while (number != 0)
        {
            previousNumber = number;
            System.out.println("Enter an integer value (or zero to quit)");
            number = In.getInt();
            if (number == previousNumber)
            {
                count = count + 1;
            }
        }
        System.out.println("There were " + count +
                           " occurrences of consecutive numbers");
    }
}
```