JAVA PROGRAMMING – EXERCISE 4

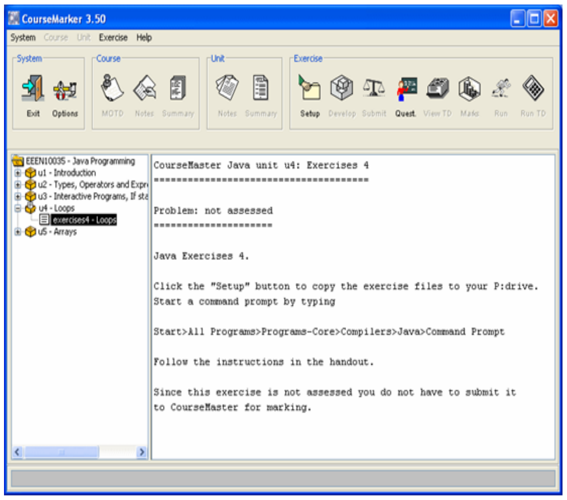
In this lab, we'll be dealing with for loops, while loops and do-while loops.

COURSE MARKER

1) Start Course Marker by selecting
2) **Start>All Programs> CourseMarker >CourseMarker**

After a few moments you will see the login window –
NOTE: User Name is your usual username (mch…), and the password is your student ID number:

3) Once you're in, access the relevant assignment by selecting:
   EEEN10035 – Java Programming>u4 – Loops



```
CourseMaster Java unit u4: Exercises 4
======================================


Problem: not assessed
=====================


Java Exercises 4.

Click the "Setup" button to copy the exercise files to your P:drive.
Start a command prompt by typing

Start>All Programs>Programs-Core>Compilers>Java>Command Prompt

Follow the instructions in the handout.

Since this exercise is not assessed you do not have to submit it
to CourseMaster for marking.
```

4) Click the Setup button on top, and then click OK.
   Clicking the Setup button copies the files needed for this exercise to the folder P:>mchXXYY> EEEN10035u4exercises4 (where mchXXYY is replaced with your username).

   The files copied are:
   Loop1.java, Loop2.java, Loop3.java, ComputeFactorial.java and ConvertFtoC.java.

   These are the java files you'll be working on this time.

5) That’s all we need from Course Marker, so now you can close it.

6) Everything we do from here on out will involve *notepad* and *command prompt*.

7) Open command prompt with,
start >All Programs and scroll down until you see the folder **JDK** –click it and then click **cmd.exe**

In the resulting window type
P: *<press enter>*
and then type
cd mchXXYY\EEEN10035u4exercises4 *<press enter>*
(We’re just navigating to the folder where course marker copied all the files.)

8) Now, type the command **dir** and press enter.
You should see the following files in the folder: Loop1.java, Loop2.java, Loop3.java, ComputeFactorial.java and ConvertFtoC.java.

9) Before we move on ahead, we need to understand for-loops.

<u>FOR LOOPS</u>

All a loop does is help you avoid repetition.
Look at these two programs,

Program A – Without any loops

```
public class A {

 public static void main(String [ ] args)  {
          int age=10;
         System.out.println(age++);
         System.out.println(age++);
         System.out.println(age++);
         System.out.println(age++);
         System.out.println(age++);
         System.out.println(age++);
         System.out.println(age++);
         System.out.println(age++);
   }
```

```
Output:
10
11
12
13
14
15
16
17
```

```
}
```

Program B – With a ‘for-loop’

```
public class B{

 public static void main(String [ ] args)  {
          int age=10;
          for (int i=0; i<8; i++) {
             System.out.println(age++);
          }
 }
}
```

Output, same as Program A:

```
10
11
12
13
14
15
16
17
```

As you see the for-loop saves up a lot of time when you want to execute a section of code multiple times. How does it work?
For-loop Syntax:

```
for (initialization;  condition;  post-loop action) {
     // statements to execute go here
}
```

Initialisation – A counter is initialized, and given a starting value, such as 0.
Condition – The for-loop will keep running, as long as this condition is true.
Post-loop action– The counter needs to be updated, so as to affect the condition, and eventually stop the loop.

Take a look at another example;

```
public class ForExample {
   public static void main(String [ ] args) {
        int i;
        for (i = 0; i < 5; i++)       {
                System.out.println("i = " + i);
        }
   }
}
```

- In the above **i** is the loop counter, initialized to 0. The condition is **i < 5,** and as long as this is true, the loop statement will be executed.
- Once the loop statements have finished, the loop variable i is incremented with **i++**, and the condition is tested again.
- If **i < 5** is false, the loop exits and the program finishes.
  Note that the change could be something *other* than i++ (remember this is equivalent to i = i + 1), e.g the change could be i = i + 2, or i = i - 3, etc.

Video Help?
http://stream.manchester.ac.uk/Play.aspx?VideoId=9639

Return to the command window that you opened earlier on.
In each of the following, use a for-loop in the implementation:

Task 1:
Using notepad, develop the file **Loop1.java** that prints out the numbers 1-100. Compile and test the program.

Task 2:
Develop the file **Loop2.java** that prints out the square root of the numbers 1-100 *(hint: use Math.sqrt())*. Use System.out.printf to format the numbers to 3 decimal places. Compile and test the program.

Task 3:
Develop the file **Loop3.java** that prompts the user for an integer less than 100 and prints out the all numbers between in steps of 5. E.g. if the user enters 67, the program prints 72, 77, 82, 87, 92, 97. Use the Scanner class to read input form the keyboard.

Task 4:
Complete the class **ComputeFactorial.java** using a for loop, as follows:
computeFactorial returns the factorial, n!, of a number, such that
n! = 1   if n=0
n! = 1 x 2 x ... x n if n>=1

WHILE LOOPS

The while loop is simpler than the for-loop and operates as follows:

while (*loop condition*) {
        loop body statement(s)
}

The while loop begins with the **while** keyword followed by a boolean expression, i.e. the *loop condition* above. As long as the expression is true, the statements that make up the *loop body* are executed: the loop terminates when the expression becomes false.

Example: Compute the sum of the numbers 1 to 100:

```
public class Anything {

  public static void main(String [ ] args)  {
          int sum=0, i=0;
          while (i<100) {
              sum += i;
              i++;
           }
  }
}
```

In the above example, i is a loop counter and sum stores the accumulated result. The loop body is only executed if the loop condition (**i<100**) is true. Each time in the loop, i is added to sum and incremented, and eventually the loop condition becomes false.

Task 5:
Complete the class ConvertFtoC using a while loop, as follows:
ConvertFtoC compares the approximation

Capprox = (F - 30)/2

for the conversion of Fahrenheit(F) to Centigrade(C), with the more familiar version:

C = (F - 32) * 5 / 9

Use a while loop to compare Capprox against C, for different values of F.
F, C and Capprox are of type double and are all 0 initially. F should be incremented by 1 each time round the loop, which should finish when the absolute* difference between C and Capprox is greater than or equal to 4.0 (use Math.abs(C -Capprox)). Finally, the program prints out the value of F when the condition is met.

DO-WHILE LOOPS

A do-while is exactly like a while loop, except for one major difference.

A while loop checks if the condition you provided is true or false, and then runs the code.
A do-while loop runs the code first, and then checks if the condition you provided is true or false.

The syntax of a do-while, is as follows,

```
do {
        loop body statement(s)
} while (loop condition);
```

The do-while loop begins with the **do** keyword followed by the loop body; the loop condition comes last. The statements that make up the loop body are executed as long as the condition is true. This is similar to the while loop except that the loop body executes at least once before the loop condition is tested. Notice again that there is no counter to keep track of the number of iterations of the loop; the loop simply terminates when the expression becomes false.

Example program: Reverse the digits of a positive integer

```
public class Anything {

  public static void main(String [ ] args)  {
            int number = 1234;
            int units, result=0;
            do {
              units = number % 10;
              result = (result*10) + units;
              number /= 10;
            } while (number > 0);
  }
}
```

The do loop strips off the last digit of number (obtained as number % 10), which becomes the first digit of the reversed number (result). The reversed number is then multiplied by 10 and number divided by 10, in preparation for the next iteration.

This line is just another way of writing number=number/10;

Task 6:
Using a do-while loop, complete the class ValidGrade. The program should request a grade continuously as long as an *invalid* grade is entered. An invalid grade is any grade less than 0 or greater than 100. After a valid grade has been entered the program should display the value of the grade entered.

When you have finished, save your work and logout.

Task 7:
There are some additional , “advanced”, java exercises, see “Week 4” under Further Exercises on Blackboard. See how many you can complete.

END