# *Programming I (420-B10-HR)*

# *Lab 10 – Formatting Output and Introduction to the while Statement*

Date assigned: Tuesday, October 27, 2015

Date due: **Tuesday, October 27, 2015**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to:

1. use the methods in the **NumberFormat** class to format output numeric data as fixed, currency and percentages;
2. use the **printf()** method to format output;
3. use a **while** loop to execute a body of statements a fixed number of times.

Review test cases and debugging.

**Commands Used:**

**while** **(***condition***)**

*statement***;**

**while** **(***condition***)**

**{**

*block of statements to be executed while the condition is true***;**

**}**

**To Be Handed In:**

1. The **Lab 10 Terminology and if Review Quiz** should be completed on **Moodle**.
2. The files ***username*\_B10\_L10\_while** folder should be zipped and uploaded to **Moodle**. Make sure that you have reformatted all your Java files to make them properly indented.

**To Start:**

1. Log on to **Moodle**, go to the **Programming I** course page and complete the **Lab 10 Terminology and if Review Quiz**.
2. Download and unzip the **B10\_L10\_while** folder from **Moodle** to your **H:\420-B10\Labs** folder. Rename it to ***username*\_B10\_ L10\_while**.
3. Start **Eclipse**. Use your **H:\420-B10\Labs** folder as the workspace.
4. Create a **New Java Project** called ***username*\_B10\_ L10\_while**.

# Debugger Review

***Purpose:*** Review using the debugger to find and correct program errors.

***To Do:***

## Open the **TestSorter** and **Sorter** classes in the **sorter** package. The **Sorter** class contains three integer instance variables with associated mutators and accessors and the **sortNumber()** method. The **sortNumber()** method is supposed to rearrange a, b and c so that they are in ascending order. The **TestSorter** class contains a **main()** method to test the **sortNumber()** methodon the following test cases:

| **Test Case** | **Object State** | | | **Expected Results** | | |
| --- | --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **a** | **b** | **c** |
| 1. Numbers are in order | 1 | 2 | 3 | 1 | 2 | 3 |
| 1. First number is smallest, second number is largest | 1 | 3 | 2 | 1 | 2 | 3 |
| 1. Second number is smallest, third number is largest | 2 | 1 | 3 | 1 | 2 | 3 |
| 1. Third number is smallest, second number is largest | 2 | 3 | 1 | 1 | 2 | 3 |
| 1. Second number is smallest, first number is largest | 3 | 1 | 2 | 1 | 2 | 3 |
| 1. Third number is smallest, first number is largest | 3 | 2 | 1 | 1 | 2 | 3 |
| 1. Two of the numbers are the same | 1 | 2 | 1 | 1 | 1 | 2 |

Which test cases work? \_\_1 and 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Which test cases don't work? \_all of the others\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Set a breakpoint in **TestSorter** on the **sortNumber()** call for the first test case that doesn't work. Run the program with the debugger. Step into the **sortNumber()** method. Step through the method and watch what it does. When you see where it goes wrong, stop the debugger and correct the mistake. Run all the test cases again. Keep using the debugger to find the error and correct it until all the test cases pass.

# The NumberFormat Class for Currency

***Purpose*:** Learn how to use the **NumberFormat** class to output currency amounts.

***New Statements:***

import java.text.NumberFormat;

NumberFormat currency = NumberFormat.getCurrencyInstance();

**Meaning:** *Create a new instance of the* ***NumberFormat*** *class called* ***currency****. Since* ***NumberFormat*** *is an abstract class, you cannot directly instantiate an object belonging to the class. Instead you must use one of its* ***get…Instance()*** *methods. You can now use the* ***format()*** *method with this object to output data in the correct format for the type of instance you have created.*

currency.format(sale.calcTax())

**Meaning**: *Format the output of the* ***calcTax()*** *method for the* ***sale*** *object as a* ***CurrencyInstance*** *(i.e. with a leading $, exactly two digits to the right of the decimal point and commas between each three digits on the left of the decimal point.)*

***To Do:***

## The **TaxCalculator** program calculates the GST and PST on a sales item. Run the **TaxCalculator** program using an amount purchased of 25.99. What is the output?

The provincial sales tax is 1.8193000000000001

The goods and services sales tax is 1.2995

The total bill is 29.108800000000002

## We want to modify the output so that it displays as dollars and cents:

### Add the following line after the import **Scanner** statement in the **TaxCalculator** class:

**import java.text.NumberFormat;**

### Add the following object instantiation after the declaration for the **Scanner** object in the **TaxCalculator** class:

**NumberFormat currency = NumberFormat.getCurrencyInstance();**

### Change the **System.out.println** statement for the sales tax to the following:

**System.out.println("The provincial sales tax is "**

**+ currency.format(sale.calcPST()));**

## Run the program again using an amount purchased of. What is the output for the provincial sales tax now?

The provincial sales tax is $1.82

The goods and services sales tax is 1.2995

The total bill is 29.108800000000002

## Fix the **System.out.println** statements for the GST and the total so that they print in dollars and cents.

# The NumberFormat Class for Percentage

***Purpose*:** Learn how to use the **NumberFormat** class to output percentages.

***New Statements:***

percent.setMaximumFractionDigits(3);

**Meaning:** *Set the maximum number of digits to the right of the decimal point to 3 for the percent object.*

***To Do:***

## Open **BankGICUser** and **BankGIC**. The **BankGIC** class calculates the amount of interest earned at a given interest rate on a Guaranteed Investment Certificate (GIC) over a fixed number of years.

## Run the **BankGICUser** program for an initial principal of 10000, an interest rate of 1.768% and 10 years to maturity. What is the value of the interest rate that is displayed? 1.7680000000000002%

## We want to modify the output so that the rate displays as a percentage with at most three decimal places.

### Add the following line after the import **Scanner** statement in the **BankGICUser** class:

**import java.text.NumberFormat;**

### Add the following local object instantiation at the beginning of the **printResults()** method:

**NumberFormat percent = NumberFormat.getPercentInstance();**

### Change the **System.out.println** statement to the following:

**System.out.println("The resulting principal compounded at "**

**+ percent.format(cd.getInterestRate()) + " is "**

**+ cd.getAccumulatedPrincipal());**

## Run the program again for an initial principal of 10000, an interest rate of 1.768 and 10 years to maturity. What is the value of the interest rate that is displayed now? \_\_\_2%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What has changed from the first output? \_\_\_\_it got rounded up\_\_\_\_\_\_\_\_\_

## We need to specify that we want three decimal places to be displayed. To do this, add the following lines after the creation of the **percent** instance:

**percent.setMaximumFractionDigits(3);**

## Run the program again for the same initial principal, interest rate and years to maturity. What is the value of the interest rate that is displayed now? \_\_\_1.768%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What has changed from the previous output? \_it uses the interest rate inputted instead of rounding it\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Run the program again for the same initial principal, an interest rate of 2.5 and 10 years to maturity. What is the value of the interest rate that is displayed now? \_\_2.5%\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Add a currency instance to the **printResults()** method and use it to output all the money amounts.

# The NumberFormat Class for Fixed Numbers

***Purpose*:** Learn how to use the **NumberFormat** class to output real numbers with a fixed number of decimal digits.

***New Statements:***

fixed.setMinimumFractionDigits(3);

**Meaning:** *Set the minimum number of digits to the right of the decimal point to 3 for the fixed object.*

***To Do:***

## Open **TemperatureTest**. This program converts Fahrenheit temperatures to Celsius and vice versa.

## Run the program for a temperature of 100 degrees Celsius. What is the equivalent Fahrenheit temperature? \_100.0 degress C is 212.0 degrees F\_\_\_\_\_

## Run the program for a temperature of 17 degrees Fahrenheit. What is the equivalent Celsius temperature? 17.0 degrees F is -8.333333333333334 degrees C\_\_\_\_\_\_\_\_

## We want to modify the output so that the temperatures display with exactly two decimal places.

### Import the **NumberFormat** class before the class header in the **TemperatureTest** class:

### Add the following local object instantiation after the declaration of the **Scanner** object:

**NumberFormat fixed = NumberFormat.getInstance();**

### Change the **System.out.println** statement to the following:

**System.out.println(fixed.format(tempIn) + " degrees "**

**+ tempType + " is " + fixed.format(tempResult)**

**+ " degrees " + newType); // Report the result**

## Run the program again for 17 degrees Fahrenheit and then for 100 degrees Celsius. What is the Celsius temperature for 17 degrees Fahrenheit now? 17 degrees F is -8.333 degrees C

100 degrees C is 212 degrees F

What is the Fahrenheit temperature for 100 degrees Celsius now? \_\_\_\_\_\_\_\_

What has changed from the first output? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## We need to specify that we want exactly two decimal places to be displayed. To do this, add the following line after the creation of the **fixed** instance:

**fixed.setMaximumFractionDigits(2);**

## Run the program again for 17 degrees Fahrenheit and 100 degrees Celsius. What is the Celsius temperature for 17 degrees Fahrenheit now? \_\_\_\_\_\_\_\_\_\_\_

What is the Fahrenheit temperature for 100 degrees Celsius now? \_\_\_\_\_\_\_\_

What has changed from the previous output? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## We have limited the number of decimal places to 2, but 2 decimal places still aren't displayed for an integer result. To do this, add the following lines after the creation of the **fixed** instance:

**fixed.setMinimumFractionDigits(2);**

## Run the program again for 17 degrees Fahrenheit and 100 degrees Celsius. What is the Celsius temperature for 17 degrees Fahrenheit now? \_\_\_\_\_\_\_\_\_\_\_

What is the Fahrenheit temperature for 100 degrees Celsius now? \_\_\_\_\_\_\_\_

# The printf() method

***Purpose:*** Learn how to format output using the **printf()** method.

***To Do:***

## Add the following **printf()** statement in the if statement to print the final values of **TemperatureTest**. You will get a syntax error that you will have to fix before you can run the program.

**System.out.printf("%6.2f degrees %c is %6.2f degrees %c\n",tempIn, tempType, tempResult, newType);**

## Run the program using 17 F as the input temperature. Compare the two lines of output – one using **NumberFormat**, the other using **printf()**.

## Add a new method called **getTemperatureScale()** to the **Temperature** class. It should have a single character parameter and should return a string. If the parameter is 'F' or 'f' return "Fahrenheit". If it is 'C' or 'c' return "Celsius". Otherwise return "Unknown". Use a **switch** statement to determine what value to return.

## Add the following statement to print the original temperature after the statement you added in step 1:

**System.out.printf("\n%12s%7.2f\n", temperature.getTemperatureScale(tempType), tempIn);**

## Change the **printf()** statement to display the temperature with 3 decimal places.

## Add another **printf()** statement to print the new temperature in the same way you printed the original one in the previous step.

## Run the program. Notice that the words Fahrenheit and Celsius are staggered. We want them to be lined up along the left-hand margin. To do this, add a negative sign before the field size. Replace the "%12s" with %-12s" in the **printf()** statements.

## Run the program to see the results. If you test it using 17F, the output of the last two **printf()** statements you added should be:

Fahrenheit 17.000

Celsius -8.333

# A counting while loop

***Purpose:*** Use a counting loop to input a fixed number of lines of data.

***To Do:***

## Open **WhileExample** and run it. *Use any 5 values as input*.

***Questions:***

How many numbers are read in? \_\_\_\_\_\_\_\_\_\_

How many times is each of the following lines of **WhileExample** repeated?

|  |  |
| --- | --- |
| System.out.println("Enter 5 values, one per line "); | \_\_\_\_\_\_ |
| System.out.println("Count is " + count + " Number is " + number); | \_\_\_\_\_\_ |
| System.out.println("The value of count after the loop is " + count); | \_\_\_\_\_\_ |

## Change the while statement to:

**while (count < 2)**

and run the program again.

***Questions:***

How many numbers are read in this time? \_\_\_\_\_\_\_\_\_\_

What line tells the program when to stop repeating?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Change the while statement to:

**while (count < 10)**

and change the increment statement (++count) to:

**count += 2;**

Run the program again.

***Questions:***

How many numbers are read in this time? \_\_\_\_\_\_\_\_\_\_

Why?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Now we want to add up all the numbers we input. Change the while statement back to **while (count < 5)** and change the **count += 2** statement back to **++count**.

## Add an integer called **sum** to the **main()** method and initialize it to **0**.

## After reading in **number**, add it to **sum**.

## Change the **println()** statement to print **sum** each time as well as the number and the count.

## Add a **println()** statement at the end of the program to display the final value of **sum**.

## Test the program on the following test case:

| **Test Case** | **Input** | **Object State** | | **Expected Output** |
| --- | --- | --- | --- | --- |
|  | **Number** | **Count** | **Sum** |  |
| All numbers are positive |  | 0 | 0 |  |
| 5 | 0 | 5 | Count is 0 Number is 5 Sum is 5 |
| 7 | 1 | 12 | Count is 1 Number is 7 Sum is 12 |
| 12 | 2 | 24 | Count is 2 Number is 12 Sum is 24 |
| 103 | 3 | 127 | Count is 3 Number is 103 Sum is 127 |
| 6 | 4 | 133 | Count is 4 Number is 6 Sum is 133 |
|  |  |  | The value of count after the loop is 5  The value of sum after the loop is 133 |

***To Do:***

## Add any statements necessary to print and calculate the average. Test your program on the following test case:

| **Test Case** | **Input** | **Object State** | | **Calculations** | **Expected Output** |
| --- | --- | --- | --- | --- | --- |
|  | **Number** | **Count** | **Sum** |  |  |
| All numbers are positive |  | 0 | 0 |  |  |
| 5 | 0 | 5 |  | Count is 0 Number is 5 Sum is 5 |
| 7 | 1 | 12 |  | Count is 1 Number is 7 Sum is 12 |
| 12 | 2 | 24 |  | Count is 2 Number is 12 Sum is 24 |
| 103 | 3 | 127 |  | Count is 3 Number is 103 Sum is 127 |
| 6 | 4 | 133 |  | Count is 4 Number is 6 Sum is 133 |
|  | 5 | 133 | average = 133/5 = 26.6 | The value of count after the loop is 5  The value of sum after the loop is 133  The average is 26.6 |

## Modify the program to work for **N** numbers, where **N** is input before the loop is executed. Test your changes. A sample dialog at the beginning of the program might look like this:

**How many numbers do you want to enter? 3**

**Enter 3 values, one per line**

## Use the following test cases to test your changes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Input** | | **Object State** | | | **Calculations** | **Expected Output** |
| **num** | **number** | **num** | **count** | **sum** |
| Find the average of 4 numbers – not all positive | 4 |  | 4 | 0 | 0 |  |  |
|  | 15 |  | 0 | 15 |  | Count is 0 Number is 15 Sum is 15 |
|  | -3 |  | 1 | 12 | 15-3 = 12 | Count is 1 Number is 12 |
|  | 40 |  | 2 | 52 | 12 + 40 = 52 | Count is 2 Number is 52 |
|  | 11 |  | 3 | 63 | 52 + 11 = 63  63 / 4 = 15.75 | Count is 3 Number is 63  The final value of sum is 63  The average is 15.75 |