COMP 1020
Lab 1

MATERIAL COVERED

- Introduction to basic Java

Notes:

- These exercises are intended to give you basic practice in using Java (as opposed to Processing or Python).
- You should definitely do at least the Bronze and Silver exercises. You can do these
  - before the lab, on your own, using any Java programming environment you like, or
  - during the lab, using DrJava, with assistance from the lab TA, or
  - after the lab, on your own.
- For credit, you must demonstrate at least one working exercise, in the lab, using DrJava.
- Make sure your TA has recorded your mark before leaving.
- The three questions are sequential – each builds on the previous one.
- Each exercise should require roughly 15 lines of actual code (in addition to the code from the previous exercises). This does not include blank lines, } lines, or comments.
- Always try to complete as many exercises as you can on every lab. For Lab 1, everyone should try to complete at least the Bronze and Silver exercises.

Input, output, IF, and loops

1. Download the file **`TemplateLab1Bronze.java`** from the course website.
2. Complete the **`readData( )`** method so that it operates as follows:
   a. It should repeatedly prompt the user to enter an integer value, and then react as follows:
      i. Values between 1 and 100 are "valid". It should print out a message which echoes the value, and indicates that it is valid, as shown below. It should also keep track of the number of such valid values that were entered.
      ii. The value 0 is special, and should cause the method to quit. Nothing should be printed out in this case.
      iii. All other values should print an error message, as shown below.
   b. It should return the number of valid values that were entered.
3. The output from the program should look like this. Note that the last line is printed by the supplied main method, not by your code. User input is shown in red.

```
Enter an integer from 1 to 100 (0 to quit):50
Entry 50 accepted.
Enter an integer from 1 to 100 (0 to quit):99
Entry 99 accepted.
Enter an integer from 1 to 100 (0 to quit):123
Invalid entry rejected.
Enter an integer from 1 to 100 (0 to quit):-42
```

```
Invalid entry rejected.
Enter an integer from 1 to 100 (0 to quit):1
Entry 1 accepted.
Enter an integer from 1 to 100 (0 to quit):0
3 valid entries were read in.
```

Arrays and methods

---

1. Modify your program as follows:
    a. Add a parameter of type **`int[]`** (array of integers) to the existing **`readData`** method. The method should now store all of the valid values (but not the 0 or the invalid ones) into this array.
    b. Write a method **`void printArray(int[] a, int n)`** which will print out the first **`n`** elements in the array **`a`**, in the format shown in the example below. The numbers should be separated by commas, but there should be no comma after the last one. There should be no blanks.
    c. Write a method **`double average(int[] a, int n)`** which will find and return the average of the first **`n`** values in the array **`a`**. Be careful to return an accurate value calculated as a **`double`**, not as an **`int`**. In the example below, the average should be 54.3333 not 54.0.
    d. Modify the **`main`** method so that it creates an array capable of holding up to 100 **`int`** values. Use the **`readData`** method to place the input values into this array. Then use the **`printArray`** and **`average`** methods to print the elements from the array, and their average, as shown below.
2. The output from the program should now look like this:

```
Enter an integer from 1 to 100 (0 to quit):50
Entry 50 accepted.
Enter an integer from 1 to 100 (0 to quit):99
Entry 99 accepted.
Enter an integer from 1 to 100 (0 to quit):203
Invalid entry rejected.
Enter an integer from 1 to 100 (0 to quit):14
Entry 14 accepted.
Enter an integer from 1 to 100 (0 to quit):0
3 valid entries were read in:
50,99,14
Their average is 54.333333333333336
```

Make the following further modifications to the program:

1. Modify the **`readData`** method so that it keeps the values in the array ***sorted*** into ascending order at all times. Each new value must be placed in the proper position, with all larger values shifted up by one position.
2. Write a method **`double median(int[] a, int n)`** which will find the median of the first **`n`** elements of the array **`a`**. As long as the array is in ascending order (which it should be), the median is defined as follows: if **`n`** is odd, then the median is the middle value, and if **`n`** is even the median is the average of the two middle values. For example, the median of 1,4,19,45,199 is 19 and the median of 1,4,19,22,45,99 is 20.5 (the average of 19 and 22). *Try to write this method in one line, without using an if statement. Think about it.*
3. Modify the main program so that it prints the median as well as the average.
4. The output of the program should now look like this:

```
Enter an integer from 1 to 100 (0 to quit):45
Entry 45 accepted.
Enter an integer from 1 to 100 (0 to quit):19
Entry 19 accepted.
Enter an integer from 1 to 100 (0 to quit):99
Entry 99 accepted.
Enter an integer from 1 to 100 (0 to quit):4
Entry 4 accepted.
Enter an integer from 1 to 100 (0 to quit):1
Entry 1 accepted.
Enter an integer from 1 to 100 (0 to quit):22
Entry 22 accepted.
Enter an integer from 1 to 100 (0 to quit):0
6 valid entries were read in:
1,4,19,22,45,99
Their average is 31.666666666666668
Their median is 20.5
```
