MAT 2170: Laboratory 12

Key Concepts

1. `char` and the `Character` class of static methods
2. `String` Class and its messages
3. Writing a class from scratch

Exercises

Design first, then implement. Come to lab with an outlined solution for each exercise, along with test cases and expected results for the Palindrome problem. Plan ahead.

1. Exercise 7, page 287. Sentence Palindrome. Create as a project in a `lab12` folder.
   Create and utilize these methods:
   (a) `cleanString()` — removes all non-alphabetic characters
   (b) `Reverse()` — reverses a string
   (c) `isPalindrome()` — determines whether a string is a palindrome
   Notes:
   - In order to receive credit, your program must be modular — i.e., decomposed into logical sections and **methods**.
   - The program is to process multiple inputs — use an empty string to signal end of input.
   - When the user indicates they are finished, thank them for using your program and note that it has completed execution.
2. Suppose we are doing research and need to keep track of temperature data where some of our data will be in degrees Celsius and some will be in degrees Fahrenheit. You are to write a `Temperature` class that can store, compare, add, and display such data. In a turn–about, we are providing the program, you are to write the class (from scratch — not derived from another class).

☐ In the `lab12` folder, create a `TestTemperature` project. Add the `acmLibrary.jar` and the `myLibrary` project to the libraries.

☐ Create an empty java file named `TestTemperature.java` in this project, then copy and paste the file contents from the website into this file and save it. You will find that much of this program is commented out to help you test your methods as you add them to the `Temperature` class, rather than all at once. As you complete methods, move the start of the comment (`/*`) further down the program.

☐ Next, create a `util` package in the `myLibrary` project. In this package create a new empty Java file, `Temperature.java`. This file is where you'll place the implementation of your `Temperature` class.

Your class should have *three data members*: a temperature `value`, a temperature `scale`, and a numeric formatter (to be used in the `toString()` method). Store the `scale` as a `char`, using '`C`' for Celsius and '`F`' for Fahrenheit. Do not allow any other characters to be used as the scale – default to 'C' if the scale

☐ isn't 'C' or 'F'. Use a `double` to store `value`. Copy the `DecimalFormat` (line #117 in the appendix) from `TestTemperature.java` into your class, just after you declare the other data members. Be sure to add `import java.text.DecimalFormat;` at the beginning of the file.

The `Temperature` class **must** include:

☐
- *Four constructor methods*, all of which must be completed before the test program will execute correctly:
  - one for each instance variable (assume zero degrees if no value is specified and Celsius if no scale is specified)
  - one with two parameters for the two instance variables
  - a default constructor (no parameters) set to zero degrees Celsius.

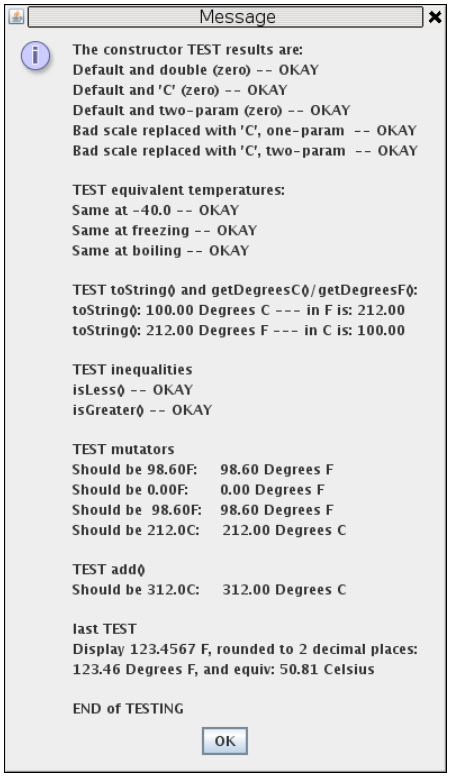
☐
- *Two inspector methods* to return the temperature as either Celsius or Fahrenheit:
  - `getDegreesC()` to return the temperature in degrees Celsius
  - `getDegreesF()` to return the temperate in degrees Fahrenheit.

Use the following formulas to write these methods:

$$\text{degreesC} = \frac{5\,(\text{degreesF} - 32)}{9} \qquad \text{degreesF} = \left(\frac{9 * \text{degreesC}}{5}\right) + 32$$

- *Three mutator methods*: one to set the value, `setValue()`, one to set the scale ('`F`' or '`C`' *only* – default to '`C`' if anything else is provided), `setScale()`, and one to set both, `setTemperature()`;
- Three comparison methods:
  - `isEqual()` — tests whether the parameter and receiver temperatures are equivalent
  - `isLess()` — tests whether the receiver's temperature is less than that of the parameter
  - `isGreater()` — tests whether the receiver's temperature is larger than that of the parameter
- A `toString()` method which returns a `String` with the value, `" Degrees "`, and its scale. The value is to be rounded to hundredths – two numbers after the decimal point. For example, `123.46 Degrees F`.
- And finally, an `add()` method which returns the sum of the `Temperature` object receiving the `add()` message and a `Temperature` parameter — resulting in a `Temperature` object (in Celsius).

Be sure to check that your output is correct by comparing it to the following:



Finishing Up

1. Publish both projects on your web site, updating the `myLibrary.jar` file in `www`.
2. Electronically submit the lab folder and your `myLibrary` project.
3. Turn in a printout of your sentence palindrome program and your Temperature class (no need to print the test file).

Appendix

Contents of TestTemperature.java

```
1  import util.*;
2  import acm.program.*;
3  import java.lang.*;
4  import java.text.DecimalFormat;
5  public class TestTemperature extends DialogProgram  {
6    public void run()  {
7      //create a single string to output to a dialog box
8      String results = "The constructor TEST results are:\n";
9
10     // Testing constructors and setting up for various tests
11     Temperature T0 = new Temperature();
12     Temperature T1 = new Temperature(0.0);
13     Temperature T2 = new Temperature('C');
14     Temperature T3 = new Temperature(0.0, 'C');
15     Temperature T4 = new Temperature(-40.0, 'C');
16     Temperature T5 = new Temperature(-40.0, 'F');
17     Temperature T6 = new Temperature(0.0, 'C');
18     Temperature T7 = new Temperature(32.0, 'F');
19     Temperature T8 = new Temperature(100.0, 'C');
20     Temperature T9 = new Temperature(212.0, 'F');
21     Temperature Bad1 = new Temperature(100.0,'K');
22     Temperature Bad2 = new Temperature('M');
23
24     // Test that 0 degrees Celsius same no matter how created;  also tests isEquals()
25     if (T0.isEqual(T1))
26     results += "Default and double (zero) -- OKAY\n";
27     else
28     results += "Default and double (zero) failed equals\n";
29
30 /*  if (T0.isEqual(T2))
31     results += "Default and 'C' (zero) -- OKAY\n";
32     else
33     results += "Default and 'C' (zero) failed equals\n";
34
35     if (T0.isEqual(T3))
36     results += "Default and two-param (zero) -- OKAY\n";
37     else
38     results += "Default and two-param (zero) failed equals\n";
39
40    if (Bad2.isEqual(T3))
41      results += "Bad scale replaced with 'C', one-param  -- OKAY\n";
42     else
43      results += "Bad scale in constructor, one-param  -- FAILED\n";
44
45     if (Bad1.isEqual(T8))
46      results += "Bad scale replaced with 'C', two-param  -- OKAY\n";
47     else
48      results += "Bad scale in constructor, two-param  -- FAILED\n";
49
50     // Test equivalent values in Celsius and Fahrenheit
51     results += "\nTEST equivalent temperatures:\n";
52     if (T4.isEqual(T5))
53     results += "Same at -40.0 -- OKAY\n";
54     else
55     results += "Same at -40.0 failed equals\n";
56
57     if (T6.isEqual(T7))
58     results += "Same at freezing -- OKAY\n";
59     else
```

```
60      results += "Same at freezing failed equals\n";
61
62      if (T8.isEqual(T9))
63      results += "Same at boiling -- OKAY\n";
64      else
65      results += "Same at boiling failed equals\n";
66
67      // Test toString() and getDegreesF()/getDegreesC()
68      results += "\nTEST toString() and getDegreesC()/getDegreesF(): \n";
69      results += "toString(): " +T8.toString() + " --- in F is: " +
70      patternDot2.format(T8.getDegreesF()) + "\n";
71
72      results += "toString(): " + T9.toString() + " --- in C is: " +
73      patternDot2.format(T9.getDegreesC()) + "\n";
74
75      // Test isGreater() / isLess()
76      results += "\nTEST inequalities\n";
77      if (T7.isLess(T9))
78      results += "isLess() -- OKAY\n";
79      else
80      results += "isLess() -- FAILED\n";
81
82      if (T9.isGreater(T7))
83      results += "isGreater() -- OKAY\n";
84      else
85      results += "isGreater() -- FAILED\n";
86
87      // TEST mutators
88      results += "\nTEST mutators\n";
89      T5.setValue(98.6);
90      T1.setScale('F');
91      T2.setTemperature(98.6, 'F');
92      T9.setScale('W');
93      results += "Should be 98.60F:     " + T5.toString() +
94            "\nShould be 0.00F:       "  + T1.toString() +
95            "\nShould be  98.60F:    " + T2.toString() +
96            "\nShould be 212.0C:     " + T9.toString();
97
98      // TEST add()
99      results += "\n\nTEST add()";
100     Temperature adder = Bad1.add(T9);
101     results += "\nShould be 312.0C:     " + adder.toString();
102
103     // Last test
104     results += "\n\nlast TEST \nDisplay 123.4567 F, rounded to 2 decimal places:   \n";
105     Temperature T10 = new Temperature(123.4567, 'F');
106     results += T10.toString() + ", and equiv: " +
107     patternDot2.format(T10.getDegreesC()) + " Celsius \n";
108
109 */  results += "\nEND of TESTING";
110     println(results);
111   }
112
113    public static void main(String args[]){
114     (new TestTemperature()).start();  }
115
116     // create a global floating-point formatter
117   DecimalFormat patternDot2 = new DecimalFormat("####0.00");
118 }
```