# CS 1400-03 Introduction to Programming and Problem Solving
**Project #4**
**(Due: 11:59 PM, Monday, 3/15/2021)**

Write a `Temperature` class that has two private instance variables:

- `degrees`: a `double` that holds the temperature value
- `scale`: a `char` either 'C' for Celsius or 'F' for Fahrenheit (either in uppercase or lowercase)

The class should have
(1) four constructor methods:

- a no-argument constructor that sets the default temperature to zero degrees Celsius
- a one-parameter constructor that accepts initial degrees as parameter and assume Celsius
- a one-parameter constructor that accepts initial scale as parameter and assume zero degrees (be sure that the scale is 'F', 'f', 'C', or 'c', if any other invalid character is given, display an error message and terminate your program)
- a two-parameter constructor that accepts initial values for the two instance variables (also need to validate the scale)

(2) two getter methods:

- `getDegreesC` that returns the degrees Celsius
- `getDegreesF` that returns the degrees Fahrenheit

Use the following formulas when appropriate,

```
degreesC = (degreesF – 32) *5 /9
degreesF = (degreesC *9 /5) + 32
```

Do not change the internal state of the object in these accessor methods. For example,

```
Temperature newYorkTemperature = new Temperature(32, 'F'); //Line 1
System.out.println(newYorkTemperature.getDegreesC());       //Line 2
```

After line 1, the variable `newYorkTemperature` will reference a temperature object containing 32 degrees Fahrenheit. When `getDegreesC` is called on `newYorkTemperature` in line 2, your program should convert 32 degrees Fahrenheit to 0 degrees Celsius and return 0. However, do not overwrite the `degrees` value of `newYorkTemperature` with the converted degrees value.

(3) three setter methods:

- `setDegrees` that modifies the temperature value
- `setScale` that modifies the scale (scale validation is required)
- `setTemperature` that modifies both (scale validation is required)

Please note that we need to validate the scale in several constructors and setter methods, you should implement validation in one method, and then use this method any number of times elsewhere in the program. Do not implement validation in every method separately.

(4) three comparison methods:

- `public boolean equals (Temperature other)` – This method will take the other temperature as a parameter and return true if the calling object has the same temperature as `other`, false otherwise. Note that a Celsius temperature can be equal to a Fahrenheit temperature as indicated by the above formulas.

- `public boolean lessThan (Temperature other)` – This method will return true if the calling object's temperature is less than the other, false otherwise.
- `public boolean greaterThan (Temperature other)` – This method will return true if the calling object's temperature is greater than the other, false otherwise.

Write a driver program named `TemperatureDemo.java` that tests each of the constructors, getters, setters, and include at least one true and one false case for each of the comparison methods.

Below is the skeleton of the driver program. Please fill in the omitted code. Use the comments as hints for the omitted code. Your driver program must generate the same output as shown on next page, including round to the nearest tenth of a degree.

```
public class TemperatureDemo
{
   public static void main(String[] args)
   {
      System.out.println("\nTest 4 constructors:");
      System.out.println("t1: created by no-arg constructor");
      Temperature t1 = new Temperature();
      // create t2 here

      // create t3 here

      // create t4 here

      System.out.println("\nTest 2 getter methods:");
      System.out.printf("t1 is %5.1f %s = %5.1f %s\n", t1.getDegreesC(), "C",
                                                       t1.getDegreesF(), "F");
      // display t2 here

      // display t3 here

      // display t4 here

      System.out.println("\nTest 3 comparison methods:");
      System.out.printf("is t1 (<, ==, >) t2? results are (%s, %s, %s)\n",
                    t1.lessThan(t2), t1.equals(t2), t1.greaterThan(t2));
      // compare t2 with t3 here

      // compare t1 with t4 here

      System.out.println("\nTest 3 setter methods:");
      // use the method setDegrees() to change t1 from 0 C to -40 C
      System.out.println("changing t1 from 0 C to -40 C...");
      t1.setDegrees(-40);
      System.out.printf("t1 is %5.1f %s = %5.1f %s\n", t1.getDegreesC(), "C",
                                                       t1.getDegreesF(), "F");

      // use the method setTemperature() to change t3 from 0 F to 100 C

      // use the method setScale() to change t4 from 32 F to 32 C

      // use the method setScale() to change t2's scale to 'G'
   }
}
```

The following is **required output** when running the program:

```
Test 4 constructors:
t1: created by no-arg constructor
t2: created by 1-arg constructor with initial degree -40
t3: created by 1-arg constructor with initial scale 'F'
t4: created by 2-arg constructor with initial temperature 32 F

Test 2 getter methods:
t1 is   0.0 C =  32.0 F
t2 is -40.0 C = -40.0 F
t3 is -17.8 C =   0.0 F
t4 is   0.0 C =  32.0 F

Test 3 comparison methods:
is t1 (<, ==, >) t2? results are (false, false, true)
is t2 (<, ==, >) t3? results are (true, false, false)
is t1 (<, ==, >) t4? results are (false, true, false)

Test 3 setter methods:
changing t1 from 0 C to -40 C...
t1 is -40.0 C = -40.0 F

changing t3 from 0 F to 100 C...
t3 is 100.0 C = 212.0 F

changing t4 from 32 F to 32 C...
t4 is  32.0 C =  89.6 F

set t2’s scale to 'G'...
invalid scale
```

In your `cs1400/project` directory, create two programs named `Temperature.java` and `TemperatureDemo.java`. Your Java programs must begin with the comments below and follow the naming and coding conventions posted on Blackboard.

```
// your name
// CS1400, section 03
// Project 4 – Temperature Conversion Object
// date
```

Generate a `script` file `pj4.txt` with appropriate time stamps and the following steps visible:

1. a `pwd` to show the current working directory
2. an `ls -l` to show in long format the files in your `cs1400/project` directory
3. display both `Temperature.java` and `TemperatureDemo.java`
4. compile `TemperatureDemo.java`
5. run `TemperatureDemo` (make sure you have hardcoded all test cases in your driver program)

Submit the file `pj4.txt` on Gradescope.