Assignment 3 – Part 2

Comparable, Abstract, Enums

Due: Friday, March 11th at 11:59pm

Submit a single zip file called **A3-Part2.zip**.
Part 2 of the assignment has 10 marks.

Q3 – Comparable [9 marks]

Consider the following abstract class.

```
public abstract class Box implements Comparable<Box>{
   String label;
   String location;
   int size;
   public String getLabel(){ return this.label;}
   public int getSize(){ return this.size;}
   public String getLocation(){ return this.location;}
}
```

You will complete the provided `SpecificBox` class which extends the Box class. The SpecificBox class must be a **concrete** class. The SpecificBox class must be implemented so that a total ordering is imposed on SpecificBox objects as follows:

Let X and Y be SpecificBox objects, then (in this order)

a) X < Y whenever X's location is alphabetically "less than" Y's location, or
b) X < Y whenever X and Y have the same location and length(X's label) < length(Y's label), or
c) X < Y whenever X and Y have the same location, the lengths of their labels are the same, and A's size is larger than Y's size, or
d) X = Y whenever X and Y have the same location, their label lengths are the same and their sizes are the same,
e) Otherwise, Y < X

Another way of specifying the ordering would be to consider a sorted list of SpecificBox objects. They would first be sorted by location (alphabetically), then for ties (that is, having the same location) sorted by length of labels, and finally for ties in both sorted by size from largest to smallest.

When you implement your SpecificBox class, the compareTo() method should only return one of seven different numbers. If the locations of the boxes are different it should return ±1, if the locations are the same, but the lengths of the labels are different it should return ±2, if the locations and label lengths are the same, but the size is different it should return ±3. If they are the same box (by state) then it should return 0.

Add your SpecificBox.java file to your submission zip.

Q4: Enums [1 mark]

In Assignment 2, you implemented a Temperature class.

The scales used were fixed (final) strings from a special **Scale** class. This was done so that there could be no chance of spelling mistakes when working with temperatures. This was also a bit awkward to work with.

Java provides another way of doing this. **Enum** classes provide us with a different way to make a new data type. The data in these (Enum) data types are constants (like the Scale names) but we can also have constructors and other behaviour defined.

Assigned reading: https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html

Consider redoing the Temperature class using enums instead of the final strings that we used. Create an enum class **ScalesEnum** that we could use instead of the Scale class that we did use. Your enum class must be in a file called **ScalesEnum.java**..

Add your Scales.java file to your zip file.

Recap [A3-Part2.zip]

Submit a single zip file called **`A3-Part2.zip`**. Your zip file should have two (or three) files in it.

- **SpecificBox.java**
- **ScalesEnum.java**

Note: as with the previous assignments, there is a 48-hour grace period. You can submit up until Sunday, March 13th at 11:59pm without penalty. Remember that there are no guarantees of any help over the weekend though, so it is really encouraged that you try to submit without using the grace period.