CSCI 2120
Homework 3: Recursion

Introduction

For this assignment you will implement two recursive methods and write JUnit tests for each one. You may write all three methods in the same class file.

You will be required to write Javadoc-style documentation for all of your methods, including the test methods.

Procedure

1) Write a *recursive method* to compare two Strings using **alphabetical order** as the natural order (case insensitive), as you might implement in a Comparator<String> object. DO NOT use the String class' built-in compareTo method!! **(50 points)**

```
public static int compare(final String s1, final String s2)
```

The function should return an integer with the following convention:

s1 comes before s2	returns an integer less than 0
s1 == (or indistinguisable from) s2	returns 0
s1 comes after s2	returns an integer greater than 0

Hint: look into the methods charAt(int index) and substring(int beginIndex) at

https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

2) Write a *recursive method* that uses **your compare(String, String)** method to find the minimum (**i.e. first by alphabetical order**) String in an array of Strings, given the array and the number of strings in the array: **50 points**

```
public static String findMinimum(final ArrayList<String> stringArray)
    OR
public static String findMinimum(final String[] stringArray, int numStrings)
```

Hint: Use the above method signature to call a private, helper method that is recursive - you can add extra parameters to this helper method!

3) Write a JUnit test class that fully tests each of these two methods.

(Bonus and submission/grading guidelines on next page)

BONUS) Structural Recursion – A train. Write a set of classes that model a train – starting with an abstract class called TrainCar as follows:

```
public abstract class TrainCar {

        protected double     itsDistanceFromHome;
        protected TrainCar    itsNextConnectedCar;

        public TrainCar (TrainCar nextCar) {
           itsNextConnectedCar = nextCar;
           itsDistanceFromHome = 0.0;

        public abstract void advance( double howFar);
    public abstract Boolean isMemberOfValidTrain();
}
```

Using TrainCar, extend it into Engine, BoxCar, and Caboose with their own constructors and implementation of the advance and isMemberOfValidTrain methods. Then write a class Train that only knows about the Engine. A valid train consists of an engine followed by some number of BoxCars ending with a Caboose. Calling isMemberOfValidTrain on the Engine should return true if this actually the setup and false if it is not. Calling advance(double) on the engine should advance the entire train. **30 points**

Submission

You will add, commit, and push your program to Gitlab. Label your homework folder "HW3" in your repository. The bonus will count as bonus points for Honors students normally in this homework and is not required.

Grading

Each of the recursive methods will be worth 35% of your grade. Each tester will be worth 15% of your grade.