CPSC 1181 - Assignment 2 [50 marks]

Objectives:

- Implement a class and instantiate objects
- Explain how programming instructions are executed when several classes are involved
- Describe the following terms: variable scope, instance variable/attribute, constructor, accessor/getter, mutator/setter

Submission:

- Submit a **zip file** containing **only** Train.java and TrainTest.java to Brightspace prior to the **due date set in Brightspace**. Do not include class or other files
- Submissions that are less than 24 hours late receive a 1% per hour late penalty. Submission that are more than 24 hours late will not be accepted.
- You can use one of your 2 day extensions.
    - If you submit more than 24 hours late, one will be used automatically
    - **Add a comment to your submission in Brightspace IF**
        - You want to use an extension for a submission less than 24 hours late
        - You are submitting more than 24 hours late, but **don't** want to use an extension…you just want feedback.
- Submissions that are unzipped or that contain .class or other unneeded files will be penalized.Submissions that are unzipped or that contain .class or other unneeded files will be penalized.
- **IMPORTANT:** The submission process might be slightly different in Brightspace than before because I'm using group submissions so I can group partners together. **If you are unsure of your submission for any reason, submit it AND email it to me.** This is true in general as well.
-

Exercise 1 [15 marks]

Create a class called Train that simulates a freight train.

Each Train object should have a name, a number representing the power of its locomotive(s), and an array of freight cars. Each freight car is represented by an integer representing how many tons that it weighs. **There are no locomotives in the cars array.**

Create the constructor so that it accepts 2 parameters, the train's name and the total power of the locomotive(s). The train initially has no cars.

Create the getters and setters necessary for the instance data of this class. Only create a getter for the array of freight cars, **no setter.** Freight cars can be added/removed with other methods. Create a toString method that returns a string summary of the Train object.

Create the method `getTotalWeightOfCars()` that returns the sum of the weights of all of the cars and `getNumberOfCars()` that returns how many cars the train has (does not include locomotives).

Create a method to compute the maximum speed of the train. Subtract the total weight of the train from the pulling power of the locomotive(s) This gives the maximum speed in km/h. The highest possible speed is 150 km/h. So, if a train has locomotive power 150 and [20, 30, 20] as its cars, its maximum

speed would be (150 – (20 + 30 + 20)) = 80. If the locomotive power was 1000, the top speed would be 150 as that is the maximum speed possible.

Exercise 2 [9 marks]

Create a method `void removeAllCars()` that will remove all cars from the train.

Create a method `void addCars(int... weights)` that will add cars to the end of the train. Add the passed in weights to the collection of cars.

Create a method `void mergeTrains(Train other)` that will add all of the locomotive power and cars from the other train to the current train. So after calling `train1.mergeTrains(train2)` the parameter train will have no cars and 0 locomotive power when done. They will all have been added to train1.

Exercise 3 [16 marks]
Create the JUnit test class called TestTrain to ensure that all of the methods of your class work correctly. Use assertArrayEquals when testing the methods that return arrays.

Notes

- Train methods should reuse other Train methods when it is useful
- assertArrayEquals is useful for testing the functions related to the freight cars array

Style and Conventions +5 marks
Documentation +5 marks
Exercise 1:Train class basics 15 marks

Exercise 2: removeCars/addCars/merge 9 marks

Exercise 3: Junit Testing 16 marks