**CISC 181 Spring 2020**

**Lab 7 Part A– Individual Assignment**

Due: April 23 at 11:55PM on Canvas

*Labs are to be completed individually. You are free to consult other students to help complete the practice sets (see syllabus for collaboration policy). However, keep in mind that each practice set is designed to cover basic material on which you will be quizzed and tested.*

This practice set is intended to cover the following major topics:

* Linked Lists Data Structures

**Setup - You must do this first!**

* You can work on either your machine or on an eCalc machine in lab.
* Follow the instructions in the **CISC181 Setting Up For Labs** document using "**Lab7"** instead of "**LabX"** and "**lab\_7"** instead of "**lab\_X"**.
* Follow the instructions in **CISC181 Setting Up Your JUnit Tests** using "**Lab7"** instead of "**LabX"**.

**Lab 7 Problems**

To create a new class, right click on ‘cisc181.lab\_7’ and select ‘New’. Select ‘Java Class’. Name your class as described in each Problem. Each class will have its own ‘.java’ file.

In class we modeled connectable strands of light using a linked list. In this problem we will model a train using linked lists. Each train car is a node and a Train is a linked list of train cars.

**Problem 1: [12points]**

Create a class named ***TrainCar***:

1. Add a field named ***cargoType*** (String) that describes what the train is carrying
2. Add a field named **weight** that is an int type representing the pounds of the cargo
3. Add a field named ***next*** of type ***TrainCar*** that represents the ***next*** train car in the train
4. Add a constructor that accepts and sets these three properties in the order listed above
5. Add ***getCargoType*** and ***getWeight*** methods
6. Add a ***getNext*** method that returns the next ***TrainCar***
7. Add a ***setCargotype, setWeight*** method
8. Add a ***setNext*** method which will set the ***next*** property to the ***TrainCar*** passed to this method
9. Add an **unload** method that returns a String with the cargo type that the TrainCar held and removes the cargo from the TrainCar – (sets the cargoType to an empty string and the weight to 0 )
10. Override the ***toString*** method that returns the string “Empty Train Car” if the TrainCar is not carrying any cargo. It should return a string in the following format if the TrainCar has come cargo in it:

Cargo : <<weight>> pounds of <<cargoType>>

See test cases for details. <<weight>> should be replaced with the weight value and <<cargoType>> should be replaced with cargo type.

1. Download ‘TrainCarTest.java’ and put it in your Test folder. Test your **TrainCar** class. ***Your class should pass all the tests without changing anything in the test file.***

**Problem 2: [33 points]**

Create a class named ***Train:***

1. Add a property named **head** which represents the first ***TrainCar*** in the train
2. Add a constructor that doesn’t accept any parameters and sets ***head*** to ***null***
3. Add a method named ***attachAtEnd*** that accepts a String representing the cargo that the new ***TrainCar*** will carry, an int representing the weight and attaches a new ***TrainCar*** to the end of the Train. (Hint: When traversing the list, do you want to end on the last TrainCar or the do you want to go past the end of the list? See Linked List Part C lecture for more details.)
4. Add a method named ***attachAtFront*** that accepts a String representing the cargo that the new ***TrainCar*** will carry, an int representing the weight and attaches a new ***TrainCar*** to the front of the Train. (Hint: Draw it first before trying to write the code.)
5. Add a method name **countTrainCars** that returns the total number of TrainCars
6. Add a method named **countEmptyTrainCars** that returns the total number of TrainCars that are not carrying any cargo
7. Add a method named **totalWeight** the returns the total number of pounds of cargo that the Train is carrying.
8. Add a method named **unloadCargo** that accepts a String parameter representing cargoType and traverses the Train and unloads all the TrainCars that have that type of cargo. This method should return the number of TrainCars that were unloaded.
9. Add a method named **loadCargo** that accepts a String representing a cargo type and an int representing the number of pounds of cargo. This method loads the first empty TrainCar with the cargo. If there are no empty TrainCars, a new TrainCar object is created, loaded and attached to the end of the Train (Hint – use your attachAtEnd method).
10. Add a method named ***removeFromFront*** that removes the first ***TrainCar*** from the train and returns the ***cargo*** that it held. This method should return null if there are no ***TrainCars***. (Hint: Draw it first before trying to write the code.)
11. Add a method named **insertCargoAfter** that accepts a String representing the cargo that the new ***TrainCar*** will carry, an int representing the weight, and a String representing the cargo of the TrainCar that that the new TrainCar should follow.

For example:

***insertCargoAfter(“bananas”, 150, “melons”)***

should insert a new TrainCar carrying 150 pounds of bananas after the first TrainCar that is carrying melons.

If there is no TrainCar carrying melons, the new TrainCar should be inserted at the end of the train. (Hint: See Linked List Part B for how to insert a node in a linked list.)

1. Add a **toString** method that prints the toString values of each TrainCar in the Train (on a separate line) starting at the first train. If there are no TrainCars it should print “Empty Train”. (See test cases for details – you need to concatenate a newline after each TrainCar’s toString.)
2. Download ‘TrainTest.java’ and put it in your Test folder. Test your **Train** class. ***Your classes should pass all the tests without changing anything in the test file.***

**Submit Your Java files by uploading to Canvas on Lab 7 Part A assignment:**

***TrainCar.java***

***Train.java***