# CS1083 Assignment #4 - Winter 2021

**Due: Friday, February 12th before 4:00pm (Atlantic), submitted in the Assignment 4 dropbox in Desire2Learn. (Read the submission instructions at the end of this document carefully).**

The purpose of this assignment is:

- Review design and sorting.

Preparation:

- Review the programming example "The Firm" from Listings 10.1-10.7 of your textbook.

**This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during a help session in the "Faculty of Computer Science Student Success Centre" team or to your course instructor.**

---

## Part 1

You will design and implement a hierarchy of categories of train cars that can be coupled together to form a train. The train cars that can be part of the train include passenger cars, freight cars, and tank cars. The train car code, which consists of letters and numbers, and the year the train car was last inspected are recorded for each train car in the train. The train car code starts with P for passenger cars, F for freight cars, and T for tank cars. For passenger cars, the number of tickets sold for the car and if the car has food service (true) or not (false) is recorded. The income a passenger car generates is based on a ticket price of $120 if there is no food service on the car and $150 if there is. For freight cars, the weight of the freight it is hauling is recorded and the income the car generates is based on a hauling fee of $11,500 for loads of under 100,000lbs and $25,000 for loads of 100,000lbs or more. For tank cars, the volume of materials the tank contains in litres and if the materials are hazardous (true) or not (false) are both recorded. The income a tank car can generate is based on a hauling rate of $17/L for hazardous materials and $9.50/L for all other materials.

Include a **`TrainCar`** class in your design. The **`TrainCar`** class must comply with the Javadoc specifications given in TrainCar.pdf (posted in D2L).

Your design must follow OO design principles illustrated in Chapter 10.

Construct a **UML class diagram**. There are several examples of UML class diagrams in your textbook (ie: Figure 10.1). In your diagram there should be a box for each class (divided into 3 sections: name, attributes, methods) and appropriate connecting lines to indicate relationships between classes. All instance variables and methods should be shown including private data/methods and constructors. Remember, private items have a minus sign in front of them, public items have a plus sign in front of them, and abstract class and method names are italicized.

You can make your UML diagrams using a drawing application, or if you prefer you can instead draw it by hand on paper and take a photo to include in your report document. However, please make sure that your diagram is neat and easy to read when you take a photo of it. Pencil probably won't show up that well, so you may need to make a good copy using pen.

## Part 2

Design a **`Train`** class for keeping track of the company that owns the train and the train cars that make up the train. You must use arrays and not ArrayList to store the train cars in the **`Train`** class. The following operations must be supported by separate methods in the **`Train`** class:

1. Returns a **copy** of the list of train cars in the train (note: this will be the train car array you will sort in the driver).
2. Search for a given train car using the train car code passed in as a parameter. This method returns the target train car if found, null otherwise.
3. toString() that returns all the information about the train including the list of train cars and the income of each car (see example in the sample output).

Add the **`Train`** class to your UML diagram and include the appropriate connections.

**Continued on next page…**

## Part 3

**Implement the classes** described in Parts 1 and 2.

For testing these classes, create a driver program that creates a **`Train`** object representing a train owned by some company that has some train cars that form the train of each type (passenger, freight, tank). The name of the company and the list of train cars are passed in as parameters to the constructor.

- Print the **`Train`** object (using the toString() method); this will print the name of the company and the list of train cars in the original order (with the code, year of last inspection, and calculated income). See sample output provided at the end of the assignment document.
- Then, print the list of vehicles in sorted order (train cars are sorted alphabetically by type F, P, T and if they have the same type then sorted by income in ascending order (smallest income to greatest income)) using the **`selectionSort`** method provided in the Sorter class. Remember to sort a **copy** of the train car list. Include in the printout only the code and the income of the train. See sample output provided at the end of the assignment document.
- Then, search for a few specific vehicles' codes among the vehicles in the inventory; print an appropriate message for these searches. See sample output provided at the end of the assignment document.

## Part 4

**Test your program** with the test file (test1.txt) that is posted in D2L and shown on the next page. Also, create two additional test files of your own. Each test file will have the same structure and contents of the test file provided: train company's name (on a separate line), number of train cars (on a separate line), followed by lines with the code and inspection year, and then the information specific to each train car type. Include at the end some codes that you will search the inventory for. Note the code for passenger cars starts with 'P', the code for freight cars starts with 'F' and the code for tank cars starts with 'T'.

**Sample input and output continued on next page…**

**Sample Input:**

```
CN Railway
5
P714 2012 85 true
F019 2018 75000
F221 2016 105000
T102 2017 35000 true
P904 2018 78 false
T102
F220
P714
```

**Sample Output:**

```
CN Railway
P714    Inspection Year: 2012
        Income: $12,750.00
F019    Inspection Year: 2018
        Income: $11,500.00
F221    Inspection Year: 2016
        Income: $25,000.00
T102    Inspection Year: 2017
        Income: $61,250.00
P904    Inspection Year: 2018
        Income: $9,360.00

Sorted Data:

F019    $11,500.00
F221    $25,000.00
P904    $9,360.00
P714    $12,750.00
T102    $61,250.00

Search results:

Train Car T102  found
Train Car F220  NOT found
Train Car P714  found
```

**Submission instructions on next page…**

**Your electronic submission (submitted via Desire2Learn) will consist of two files:**

i. a written report. This should begin with a title page that includes: the course (CS 1083), your section (FR01B, FR02B, FR03B), the assignment number, your full name, and your UNB student number. That should be followed by two sections, with each part clearly identified with a section heading. Include:

   a. the final UML diagram (after completing parts 1 & 2)

   b. all the source code for your solution (with Javadoc comments included)

   c. the sample input and output from running your test driver (for all 3 input files).

   This written report should be prepared using a word processor; we recommend using Microsoft Word (i.e. create a .docx file for your report). Copy & paste your java source code & required output into the report document. Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code to maintain proper indentation.) Once the report is complete and you've checked it all over, save the .docx file for your own records, and then **save a second copy in pdf format for submission**. (Note: Be sure to open that file in a pdf viewer to verify that the pdf was generated correctly.) **The SINGLE pdf file containing your report will be submitted to the appropriate assignment drop box on Desire2Learn**. (It is important that you submit a pdf file and NOT the original Word document. This pdf will allow the marker to write comments directly on your work to give you better feedback.)

   Note: name this report as follows: **YourName_CS1083_As4_Report.pdf**

ii. an archive file (**.zip**) that contains all your Java source code and input files for this assignment. Make sure that your archive includes all .java files (in case the marker wishes to compile & run your code to test it). You should not include the report document or the .class files in your archive. This archive should be submitted as a **single file** to the appropriate drop box on Desire2Learn.

   Note: name this archive file as follows: **YourName_CS1083_As4_Archive.zip**