CS1083 Assignment #5 - Winter 2021

Due: Friday, February 19th before 4:00pm (Atlantic), submitted in the Assignment 5 dropbox in Desire2Learn. (Read the submission instructions at the end of this document carefully).

The purpose of this assignment is to provide you with practice with exceptions.

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.

---

1. Adding Exceptions to Assignment 1

In assignment 1 you implemented a program to keep track of vehicles on a ferry. The only error handling that was done was via boolean return values for some of the methods in the Ferry class. For example, the method to add a vehicle to the ferry returned false if the vehicle would put the ferry over its weight limit, passenger limit or axel limit, or if the vehicle was already on the ferry. This doesn't allow us to determine the cause of the failure for this method call. (If false is returned, we only know that the requested operation failed, but not why it failed.)

For this assignment you will add exception handling to a solution for Assignment #1 that is provided to you in D2L, consisting of the files **Vehicle.java**, **Ferry.java**, and **FerryDriver.java**. (Aside: Our original solution for Assignment #1 was modified a bit to make it work easier with a menu-based driver program for this assignment. Specifically, the removeVehicle, updatePassengers, and findVehicle methods in the Ferry class now accept a vehicle ID as a parameter rather than a reference to a Vehicle object.)

Read, compile and run the program that we have provided. Notice that if you try to add a vehicle to a ferry and it fails, the message given is "==> vehicle not added"; the reason isn't given. Also see what happens if you provide invalid input to a call to the Scanner's **nextInt()** or **nextDouble()** methods; e.g. entering "Bob" after the prompt "Enter weight:".

Your tasks are:

(a) Create three new exceptions, called **LimitExceededException**, **DuplicateVehicleException**, and **VehicleNotFoundException**. Each of these classes should have two (overloaded) constructors: a default constructor and one that accepts a message describing the exception.

(b) Modify **Ferry.java** so that certain methods throw exceptions. Specifically:

  i. **addVehicle** - Modify the method signature so that it has a void return type, and indicate that this method may throw a LimitExceededException or a DuplicateVehicleException. In the body of the method, you should throw LimitExceededException if adding the vehicle would exceed any of the three specified limits for this ferry, or throw DuplicateVehicleException if the specified vehicle is already on this ferry. When throwing an exception, be sure to include a message that further explains the problem; in the case of LimitExceededException, for example, the message should indicate which limit check failed. (If adding this vehicle would exceed multiple different limits, your exception message should simply describe the first limit test that failed.)

  ii. **removeVehicle** - Modify the method signature so that it has a void return type, and indicate that this method may throw a VehicleNotFoundException. In the body of the method, you should throw VehicleNotFoundException if the specified vehicle isn't found on this ferry.

  iii. **updatePass** - Modify the method signature so that it has a void return type, and indicate that this method may throw a LimitExceededException or VehicleNotFoundException. In the body of the method, you should throw LimitExceededException if changing the number of passengers in the specified vehicle would exceed the specified passenger limit for this ferry, or throw VehicleNotFoundException if vehicle isn't found.

(c) Modify **FerryDriver.java** to handle all exceptions thrown by methods in Ferry.java. In addition, add exception-handling code to the driver to cover any places where InputMismatchException could be thrown by a Scanner method. The completed program should handle any exceptions, no matter the input. Exceptions that occur inside the while (!done) loop should be handled by printing a message and allowing execution to proceed to the next iteration (presenting the user with the menu options once again). Exceptions that occur before the while loop should be handled by printing a message and terminating the program. Helpful hint: try-catch blocks can be nested.

Part 2

Test your program with interactive sessions in the terminal with test input cases that will execute "normally" (no exceptions thrown), and test cases that cause each exception to be thrown. Include different test cases for LimitExceededException that generate different messages. Copy and paste the contents of your terminal window into your report so that we can see your test input and corresponding output.

Your electronic submission (submitted via Desire2Learn) will consist of <u>two files</u>:

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:
   - The source code for the three exception classes that you wrote (LimitExceededException, DuplicateVehicleException, and VehicleNotFoundException).
   - The modified source code for Ferry.java and FerryDriver.java
   - The sample input & output for your test cases (copy & paste the contents of the terminal window into your report).

   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_As5_Report.pdf**

ii. an archive file (**.zip**) that contains all your Java source code 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_As5_Archive.zip**