Car Rental

1 Introduction

This assignment provides an introduction to polymorphism. We're giving you an application and asking you to modify it to make it easily extensible. You'll also gain practice in reading and understanding an existing code base.

2 Problem Description

You've been hired by a car rental company to update their leasing software. Right now the company only rents four-passenger Ford Taurus sedans, but they want to offer eight-passenger Dodge Caravans and other kinds of vehicles in the future. Your task is to update the company's leasing software so that it can easily be extended with new kinds of cars. While you're at it, the previous software consultants didn't provide any Javadoc docmentation for the existing code, so you'll also add Javadoc to the existing code in addition to any new code you write.

3 Solution Description

We provide the following classes:

- `Customer` represents a customer.
- `Lease` represents a rental, i.e., a car rented to a customer.
- `RentalCompany` represents the rental company.
- `FordTaurus` represents the only kind of car the company currently has.
- `Main` runs the leasing application.

You need to create a new abstract class `Car`, make `FordTaurus` a subclass of `Car`, add the `DodgeCaravan` class which holds eight passengers and should also be a subclass of `Car`, and update the other classes in the system to work with any `Car` subclass.

Once you've created your new classes and modified the provided classes, running the application should look something like this:

```
$ java Main
Options:
1 - Lease car
2 - Return car
3 - Quit
Enter option numer: 1
Customer name: Brian
How many passengers will Brian have? 2

Hertz's current leases:
Ford Taurus with license plate 0000 leased to Brian for $30.00 a day.
Hertz's available cars:
Ford Taurus with license plate 1111
Ford Taurus with license plate 2222
Dodge Caravan with license plate 3333
Dodge Caravan with license plate 4444

Options:
1 - Lease car
2 - Return car
3 - Quit
Enter option numer: 1
Customer name: Stewie
How many passengers will Stewie have? 7

Hertz's current leases:
Ford Taurus with license plate 0000 leased to Brian for $30.00 a day.
Dodge Caravan with license plate 3333 leased to Stewie for $40.00 a day.
Hertz's available cars:
Ford Taurus with license plate 1111
Ford Taurus with license plate 2222
Dodge Caravan with license plate 4444

Options:
1 - Lease car
2 - Return car
3 - Quit
Enter option numer: 2
Which car would you like to return?
0: Ford Taurus with license plate 0000 leased to Brian for $30.00 a day.
1: Dodge Caravan with license plate 3333 leased to Stewie for $40.00 a day.
Enter the number: 0

Hertz's current leases:
Dodge Caravan with license plate 3333 leased to Stewie for $40.00 a day.
Hertz's available cars:
Ford Taurus with license plate 1111
Ford Taurus with license plate 2222
Dodge Caravan with license plate 4444
Ford Taurus with license plate 0000

Options:
1 - Lease car
2 - Return car
3 - Quit
Enter option numer: 3
Bye!
Hertz's current leases:
Dodge Caravan with license plate 3333 leased to Stewie for $40.00 a day.
Hertz's available cars:
Ford Taurus with license plate 1111
Ford Taurus with license plate 2222
Dodge Caravan with license plate 4444
Ford Taurus with license plate 0000
```

4 Checkstyle

You have been provided with a Checkstyle Jar file and the CS 1331 Checkstyle configuration file (`cs1331-checkstyle.xml`). We will run checkstyle on all the Java source files you submit and deduct one point from your score for each style error found by Checkstyle. You can run Checkstyle on your code like this (in the directory containing all your Java source files):

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. You can easily count Checkstyle errors by piping the output of Checkstyle through `wc -l` and subtracting 2 for the two non-error lines printed above (which is how we will deduct points). For example:

```
$ java -jar checkstyle-5.6-all.jar -c cs1331-checkstyle.xml *.java | wc -l
     2
```

The Java source files we provide contain no Checkstyle errors. You are responsible for any Checkstyle errors you introduce when modifying these files.

5 Javadoc

Add Javadoc coments to all classes and *public* methods, both for the new classes you write and the classes that were provided to you. You'll need to read and understand the given code to do this.

6 Tips

- Download all the provided Java source files, as well as the Checkstyle jar and `cs1331-checkstyle.xml` files, into a directory you create for this project.
- You may assume that you always get valid input.
- You don't need to modify any of the logic in the existing classes, only type information.
- Remember to chain to superclass constructors where appropriate.

7 Turn-in Procedure

Submit all of the Java source files we provided (and you umodified) as well as all new Java source files you created to T-Square. Do not submit any compiled bytecode (`.class` files), the Checkstyle jar file, or the `cs1331-checkstyle.xml` file. When you're ready, double-check that you have submitted and not just saved a draft.

8 Verify the Success of Your Submission to T-Square

Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
   (a) It helps insure that you turn in the correct files.
   (b) It helps you realize if you omit a file or files.[1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
   (c) Helps find last minute causes of files not compiling and/or running.

[1]Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight Friday. Do not wait until the last minute!