COSC1295 Advanced Programming
Computer Science and Information Technology
School of Science, RMIT

Assignment 1 - Semester 1, 2019

Submission due date: 11:59pm 5 April 2019

Introduction

This assignment is worth 15% towards your final grade.

NOTE

- This assignment is of a size and scope that can be done as an individual assignment. Group work is not allowed.
- A more detailed marking rubric will be provided closer to submission.

You are required to implement a basic Java program using Java Standard Edition 8.0 or higher. This assignment is designed to:

- Practise your knowledge of design classes in Java
- Practise the implementation of various kinds of classes in Java
- Practise the use of polymorphism

Academic Integrity

The submitted assignment must be your own work. For more information, please visit http://www.rmit.edu.au/academicintegrity.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment even if you have very similar ideas.

Plagiarism-detection tools will be used for all submissions. Penalties may be applied in cases of plagiarism.

Overview

NOTE: Carefully read this document. In addition, regularly follow the Canvas assignment discussion board for assignment related clarifications and discussion.

For this assignment you need to write a console application in the Java programming language which allows a company called ThriftyRent to manage the renting and maintenance of two types of vehicles: cars and vans.

Vehicle

Each vehicle managed by ThriftyRent has the following attributes:

- Vehicle id: a string which uniquely identifies each vehicle. A vehicle id must start with C_ if the vehicle is a car and V_ if the vehicle is a van.
- Year: the year when the vehicle was manufactured.
- Make: for example Toyota, Honda, Nissan, Hyundai, Kia...
- Model: for example Corolla, Camry, Accord...
- Number of passenger seats of the vehicle (more details later)
- Vehicle type: ThriftyRent currently has two types of rental vehicles: car and van. The different characteristics of those types of vehicles are described in later sections.
- Vehicle status: employees of ThriftyRent will inspect this attribute to determine whether the vehicle is currently available for rent or being rented or under maintenance.

Furthermore, each vehicle also keeps its own collection of Rental Records, i.e information about the 10 most recent times that vehicle has been rented. Details of a rental record is shown below:

Rental Record

Each Rental Record has the following attributes:

- Record id: a string which uniquely identifies each rental record. A rental record id is constructed by concatenating the following three attributes

  vehicleId_ + customerId_ + rentDate (8 digit format: ddmmyyyy)

  Note: In Assignment 1, each customer is simply represented by a unique string of customer id of your choice. There is no need to implement a class to store customer information.

- Rent date: the date when a customer rents the vehicle

  Source code of the DateTime.java is provided. Please click here to access the code. Some examples of how to use the DateTime.java class is here.

- Estimated return date: the calculated date given the number of days a customer wants to rent the vehicle and the rent date shown above

  Example: a customer wants to rent a vehicle on 14/02/2019 for 3 days, hence the estimated return date will be 17/02/2019

- Actual return date: the date when the customer actually returns the vehicle
- Rental fee: the fee calculated based on the type of vehicle, the rent date and the estimated return date.

- Late fee: the additional fee which must be calculated when the actual return date is after the estimated return date

    Note: car and van have different formulae to calculate rental fee and late fee, which will be shown in the next sections

Car

As mentioned above, ThriftyRent has two types of vehicles for short-term rental. The first type is car, which has the following characteristics:

- A car can have either 4 or 7 passenger seats
- Each car can be rented for:
    - a minimum of 2 days if the rental day is between Sunday and Thursday inclusively
    - or a minimum of 3 days if the rental day is Friday or Saturday
    - and a maximum of 14 days
- Car type has the following rental rates:
    - $78 per day for a 4-seat car
    - $113 per day for a 7-seat car
- If a car is returned earlier than the estimated return date, there is no additional fee applied and the rental fee is calculated based on the rent date and the date the car is returned (actual return date)
- About late fee:
    - If a car is returned later than the estimated return date, then the rental rate of each late day is 125% of the normal daily rate for that car type. For example, a 4-seat car has a daily rate of $78 as shown above. Therefore the rental rate of each late day is 125% of 78 = 125/100 * 78 = $97.5
- A car has no fixed maintenance schedule. ThriftyRent can perform maintenance on a car at any time that there is no customer renting the car, i.e. when the car is available.

Van

The second type of vehicle ThriftyRent offers for short-term rent is vans. A van has the following characteristics:

- A van always has 15 passenger seats
- Each van can be rented for a minimum of 1 day ~~and a maximum of 30 days~~ (this condition is removed)
- The rental rate of a van is $235 per day
- About late fee: if a van is returned after the estimated return date, then the late fee is calculated at $299 per day
- Each van has a strict maintenance schedule because ThriftyRent wants all their vans to be in the best possible conditions. Therefore they specify the following requirements:

- All vans must have a maintenance interval of 12 days.

- Each van must keep its last maintenance date. Maintenance operations for a van must be done no more than 12 days (as specified by the maintenance interval above) after its last maintenance date.

- Customers will not be allowed to rent a van for a time period which exceeds the date on which maintenance operation must be done.

  Example: a van is available and last underwent maintenance on 15/01/2019. Maintenance must be done for that van no later than 27/01/2019. Therefore, if a customer wants to rent that van on 23/01/2019 for 5 days, ThriftyRent system will reject that request.

Implementation Requirements

General Implementation Requirements

- You are required to modularise classes properly. No method should be longer than 50 lines.
- You should aim to provide high cohesion and low coupling.
- You should aim for maximum encapsulation and information hiding.
- Your coding style should be consistent with Java coding conventions (http://www.oracle.com/technetwork/java/codeconventions-150003.pdf)
- You should comment important sections of your code remembering that clear and readily comprehensible code is preferable to a comment.
- Your programs will be marked with Java SE 8.0. Make sure you test your programs with this setting before you make the submission.

Main Implementation Requirements

Your Rental Record class must meet the following requirements:

- Override the **public String toString()** method to return a string containing the details of a rental record in the following format:

  `recordId:rentDate:estimatedReturnDate:actualReturnDate:rentalFee:lateFee`
  (notice how the colon is used as a separator)

  Example 1: When a vehicle is being rented, but hasn't yet been returned, calling the toString method of the latest Rental Record object for that vehicle will return a String as shown below:

  `C_108CRSB_s3147490_11022019:11/02/2019:16/02/2019:none:none:none`
  (notice how ***none*** strings are used for actualReturnDate, rentalFee and lateFee)

  Example 2: When a vehicle has been returned, the latest rental record of that vehicle will be updated with the actualReturnDate, the rentalFee and any lateFee. Therefore, calling the toString method of the latest Rental Record object of that vehicle will return a String as shown below:

`C_108CRSB_s3147490_11022019:11/02/2019:16/02/2019:16/02/2019:390.00:0.00`
(notice that now all attributes have value)

- Implement a **public String getDetails()** method. This method should build a string and return that string. The returned string should be formatted in a human readable form as shown below. This method MUST NOT do the actual printing to the console. Please refer to the following examples:

  Example 1: When a vehicle is being rented, but hasn't yet been returned, calling the getDetails method of the latest Rental Record object of that vehicle will return a String as shown below:

  ```
  Record ID:             C_108CRSB_s3147490_11022019
  Rent Date:             11/02/2019
  Estimated Return Date: 16/02/2019
  ```
  (the actualReturnDate, rentalFee and lateFee have no value yet and therefore are not included in the returned string)

  Example 2: When a vehicle has been returned, the latest rental record of that vehicle will be updated with the actualReturnDate, the rentalFee and any lateFee. Therefore, calling the getDetails method of the latest rental record object of that vehicle will return a String as shown below:

  ```
  Record ID:             C_108CRSB_s3147490_11022019
  Rent Date:             11/02/2019
  Estimated Return Date: 16/02/2019
  Actual Return Date:    16/02/2019
  Rental Fee:            390.00
  Late Fee:              0.00
  ```
  (notice that rentalFee and lateFee are printed with 2 decimal places)

Implementation requirements for all vehicle classes (car and van)

Each vehicle must maintain its own collection of rental records. These records store information about the 10 most recent times that vehicle has been rented.

The following methods can be called on any object of type car or van:

(Hint: implementing these methods is a good chance for you to apply inheritance and polymorphism in your code)

public boolean rent(String customerId, DateTime rentDate, int numOfRentDay)

This method is called on a vehicle object (either a car or a van) to perform the operations required to rent the vehicle.

This method should check for pre-conditions to determine if that vehicle can be rented. For example, this method will return false when that vehicle is currently being rented or is under maintenance. You should check any other possible conditions which would make this method return false.

If the vehicle is available for rent, this method will perform all necessary operations to update the information stored in this vehicle object based on the input parameters. For example,

updating the vehicle status, creating a new rental record, updating the rental record collection of the vehicle, and any other operations you consider necessary.

Finally, this method will return true if the vehicle can be rented successfully.

public boolean returnVehicle(DateTime returnDate)

This method is called on a vehicle object (either an car or a van) to perform the operations required to return the vehicle.

This method should check for pre-conditions to determine if that vehicle can be returned. For example, this method will return false when the given returnDate is prior to the rentDate stored in the rental record. You should check any other possible conditions which would make this method return false.

If the vehicle can be returned, this method will perform all necessary operations to update the information stored in this vehicle object based on the input parameters. For example, updating the vehicle status, updating the corresponding rental record with the rental fee, the late fee, and any other operations you consider necessary.

Finally, this method will return true if the vehicle can be returned successfully.

public boolean performMaintenance()

This method is called on a vehicle object (either a car or a van) to execute the operations required to perform the maintenance of the vehicle.

This method should check for pre-conditions to determine if maintenance operations can be performed in that vehicle. For example, this method will return false when the vehicle is currently being rented. You should check any other possible conditions which would make this method return false.

If the vehicle is ready for maintenance, this method will perform all necessary operations to update the information stored in this vehicle object when a maintenance happens. Finally, this method will return true if the vehicle is now under maintenance.

public boolean completeMaintenance(DateTime completionDate)

This method is called on a vehicle object (either an car or a van) to perform the operations required when the maintenance of that vehicle is finished.

This method should check for pre-conditions. For example, when this vehicle is currently being rented, it does not make sense to call completeMaintenance method on this vehicle object, and therefore this method should return false. You should check any other possible conditions which would make this method return false.

If it is possible to complete maintenance, this method will perform all necessary operations to update the information stored in this vehicle object now that maintenance has been finished. Finally, this method will return true to indicate that the maintenance of this vehicle has finished.

public String toString()

This method should build a string and return it to the calling method. The returned string should be formatted in a pre-defined format as shown below:

`vehicleId:year:make:model:numOfSeats:status`
(Notice how the colon is used as a separator. If that vehicle is a van, the attribute lastMaintenanceDate is appended.)

Example 1: A 2013 Toyota Corolla with ID `C_108CRSB` with 4 passenger seats is available for rent. Calling toString method of that vehicle object will return the following line:

```
C_108CRSB:2013:Toyota:Corolla:4:Available
```

Example 2: A van is currently being rented. It is a 2017 Mercedes-Benz Sprinter with ID V_63WMSB, 15 passenger seats and its last maintenance date 19/02/2019. Calling toString method of that vehicle object will return the following line:

`V_63WMSB:2017:Mercedes-Benz:Sprinter:15:Rented:19/02/2019`
(Notice: because this is a van, we need to keep track of its last maintenance date. Therefore the toString() output of a van should include it's last maintenance date at the end as shown above)

public String getDetails()

This method should build a string and return it to the calling method. This method SHOULD NOT do the actual printing. The returned string contains all information about the vehicle, including details about up to 10 most recent rental records of that vehicle. The returned string should be formatted in a pre-defined human readable format. See the examples below:

Example 1: a car is available for rent for the first time (no rental record)

```
Vehicle ID:           C_108CRSB
Year:                 2018
Make:                 Toyota
Model:                Corolla
Number of seats:           4
Status:               Available
RENTAL RECORD:        empty
```

(no need to keep track of maintenance date of a car)

Example 2: A van is currently being rented for the first time

```
Vehicle ID:           V_63WMSB
Year:                 2017
Make:                 Mercedes-Benz
Model:                Sprinter
Number of seats:           15
Status:               Rented
```

```
Last maintenance date: 19/02/2019
RENTAL RECORD
Record ID:             V_63WMSB_CUS001_20022019
Rent Date:             20/02/2019
Estimated Return Date: 23/02/2019
```

(notice that the last maintenance date and part of the first rental record are shown because the vehicle is being rented for the first time)

Example 3: a Van is currently being rented. It was rented and returned one time before.

```
Vehicle ID:           V_63WMSB
Year:                 2017
Make:                 Mercedes-Benz
Model:                Sprinter
Number of seats:           15
Status:               Rented
Last maintenance date: 27/02/2019
RENTAL RECORD
Record ID:             V_63WMSB_CUS003_01032019
Rent Date:             01/03/2019
Estimated Return Date: 05/03/2019
--------------------------------------
Record ID:             V_63WMSB_CUS001_20022019
Rent Date:             20/02/2019
Estimated Return Date: 23/02/2019
Actual Return Date:    23/02/2019
Rental Fee:           705.00
Late Fee:             0.00
```

(notice how the latest rental record is shown first, although it's still not complete until the vehicle is returned)

Implementing the ThriftyRentSystem application class

You are required to implement a class named ThriftyRentSystem which will contain a collection to store up to 50 objects of both types car and van. Objects of type car and van will be added during runtime by the user of your program in the command line using a menu system described below.

Your ThriftyRentSystem class should present the following menu for employees of ThriftyRent company to manage rental vehicles. The following menu should be presented to the users:

```
**** ThriftyRent SYSTEM MENU ****

Add vehicle:              1
Rent vehicle:             2
Return vehicle:           3
```

```
Vehicle Maintenance:       4
Complete Maintenance:      5
Display All Vehicles:      6
Exit Program:              7
Enter your choice:
```

User should enter a number from the menu above to select an option. If the input is outside of that range, an error message should be displayed and the menu should be re-displayed. When a valid number is entered, your program should execute the corresponding method and then return to the menu. Your program should exit if the user selects the Exit Program option.

You may use a sub menu under an option.

All output data should be printed to the standard output.

Following is a description of each feature provided by the menu above:

Add vehicle

The user (an employee of ThriftyRent) selects this option to create a new vehicle. The user can enter all relevant details of a new vehicle, such as vehicle id, vehicle type, year, make, model, number of seats, and last maintenance date if the vehicle is a van.

You should perform all necessary data validation, for example, invalid vehicle id or vehicle id already exists. If there is an error, your program should print an appropriate error message to the console and should go back to the menu immediately without creating or storing a new vehicle.

Rent vehicle

By selecting this option, an employee of ThriftyRent can then enter a vehicle ID and enter relevant information for a customer to rent that vehicle. Your implementation should be similar to the following example:

Example 1: rent a car which is available for rent

```
Enter your choice: 2
Vehicle id: C_108CRSB
Customer id: CUS0011
Rent date( dd/mm/yyyy): 19/02/2019
How many days?: 3
Vehicle C_108CRSB is now rented by customer CUS0011
****** Rental Company ******
Add Property:       1
Rent Property:      2
Return Property:    3
..............
(main menu is shown again)
```

Example 2: attempt to rent a vehicle which is not available for rent

```
Enter your choice: 2
Vehicle id: C_109CRSB
Vehicle C_109CRSB could not be rented
****** Rental Company ******
Add Property:       1
Rent Property:      2
Return Property:    3
..............
```

(main menu is shown again, without asking user further details)

Return vehicle

By selecting this option, an employee of ThriftyRent can then enter a vehicle ID and a return date to return that vehicle. If a return is successful, your program should print all relevant information about that vehicle, including information about the latest rental record. You should perform all necessary data validation, for example, if that vehicle is currently under maintenance, it's not reasonable to return that vehicle.

Vehicle Maintenance

By selecting this option, an employee of ThriftyRent can then enter a vehicle ID to put that vehicle under maintenance. You should perform all necessary data validation to avoid any unreasonable scenario.

Example: perform maintenance of a Vehicle which is not currently being rented

```
Enter your choice: 4
Vehicle id: V_63WMSB
Vehicle V_63WMSB is now under maintenance

****** Rental Company ******
Add Property:       1
Rent Property:      2
..............
```

(main menu is shown again)

Complete Maintenance

By selecting this option, an employee of ThriftyRent can then enter a vehicle ID to complete maintenance of that vehicle. From the ID, if that vehicle is a van, then the system will prompt the employee to enter a maintenance completion date.

Example: complete maintenance of a van

```
Enter your choice: 5
Enter property id: V_63WMSB
Maintenance completion date (dd/mm/yyyy): 19/02/2019
Vehicle V_63WMSB has all maintenance completed and
ready for rent

****** Rental Company ******
Add Property:       1
Rent Property:      2
Return Property:         3
..............
```
(main menu is shown again)

Display All vehicles

By selecting this option, an employee of ThriftyRent can see on the Console all information about all rental vehicles stored in the system, including details about up to 10 most recent rental records of each vehicle. (Hint: calling the getDetails method on each vehicle object).

Start-up Class.

You should create a startup class which contains a main method in which an object of the ThriftyRentSystem class is created and a single method is called on that object to run the entire ThriftyRentSystem application.

Design Requirements

You must work out an object design for the above task. You should take advantage of object-oriented concepts such as composition, inheritance, method overriding, abstract classes, interfaces wherever appropriate.

Class hierarchies, relationships and components must be conceptualised in a relevant manner, based on the problem description and special conditions listed in this document. Furthermore, you must be able to explain how your program design will perform in certain scenarios and circumstances.

It may be necessary for your design to provide more functionality, such as accessors and mutators, than is specified in the above sections here for the mechanics of your design to work.

Submission Details

You must submit a zip file of your project via Canvas.

You can submit your assignment as many times as you would like prior to the due date. The latest submission will be graded.

Please do NOT submit compiled files (*.class files), or you will get zero.
You will get zero if the submitted code cannot be compiled.

THE END