FACULTY OF COMPUTING
& INFORMATION TECHNOLOGY

KING ABDULAZIZ UNIVERSITY

FCIT
KAU

كـلـيــة الـحـاسـبــات
وتـقـنـيـة الـمـعـلـومـات
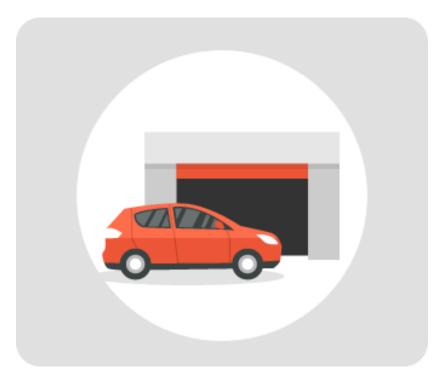جامعة الـمـلك عـبـدالـعـزيـز

Vehicle Rental System

Assignment 2

CPCS203 Programming-II Spring 2019

Assigned Date: Sunday, 10/2/2019
Delivery Date: Sunday, 3/3/2019

Instructions

- This program must ONLY be submitted on the Blackboard!
- This project worth 6% of the overall module marks (100%).
- NO assignment will be accepted after 11:59 pm for any reason
- Students can submit their assignment between 11 and 11:59 PM but in   this case it will be consider as late submission.
- For discussion schedule, check the captain name, date and time on the BlackBoard.
- *Further information is provided in the course syllabus.*

Objectives

- Performing procedure on Objects and classes.
- Learn how to use and implement Class and Object concepts.
- Learn to use File I/O (Reading/Writing from/to files).

How to submit your assignment?

- Submit your assignment on the Blackboard ONLY.
- Make sure to add your names / IDs / Section / Your name / Assignment number at the beginning of your program

Files provided with assignment

- Input file samples:
  - **inputCar.txt**: which contains all Cars/Services information that needs to be entered into the system.
  - **ReservationInput.txt**: contains all the commands for reservation requests. These commands are read from the file and processed by the system.
- Output files:
  - **CarsInfo.txt**: A file that displays all the cars information in the system (The information in this file is read from **inputCar.txt)**.
  - **ReservationStatus.txt**: which contains a log of all processed reservations including successful and unsuccessful reservation requests. For successful reservation requests, a detailed invoice is included.

Note: Please check the format of each of these files and make sure you follow this format in your assignment solution.

1.1 The EasyRent Software Description

You have been hired to develop a program software for a local Car Rental Company. Besides car rental, the company also offers customers additional services such as a navigation system devices, car seats, and full rental insurance.

The system you are requested to develop is called **EasyRent** and is expected assist the agent at the company in processing car reservation and additional services requests. At the beginning, your system will read the available cars information from **inputCar.txt.** Information read from **inputCar.txt** are written with all the details into the car information file, called **CarsInfo.txt**.

After that, the system will be ready to process reservation requests! For each reservation request, the cars' information including customer information, pick up and drop off location and date, and any services the customer would like to add are all read from the input file **ReservationInput.txt**. After processing each reservation request, the status of this reservation, whether it was successful or not, is written to the output file **ReservationStatus.txt**. If a reservation request is successful, then an invoice will be generated and written to this file. Otherwise, an appropriate message is displayed to show that the reservation was unsuccessful (Sorry! The requested car is not available).

For a more detailed description of the system and commands, please follow the next three steps which will explain how to develop the **EasyRent** System.

Step 1: Adding Car Information and Services

All Cars and services, which are available for renting, will be entered to the system before performing any reservation The first line read from *inputCar.txt* contains two integers, which determine the number of available cars and services in the system. For example, the company has (6) cars, and provide (3) services. The following commands are either for inputting the car information or inputting types of available services. In the following, we describe the format of each command.

1.1 Command: AddCar

This command is used to add all information of cars available for renting. Car information includes car brand (such as : Toyota, Mercedes, Nissan), model year, daily rental rate in S.R. transmission type (either false for Manual or true for Automatic), car type which indicates if it is Economy, Sport or Luxury. At last, if the car contains a convertible body style, the data field Convertible will be true, and false means not convertible. Check the following example and table.

Command Example
AddCar Toyota 2015 200 False Economy False

Field name	Type	Example
Brand	String	Toyota
year	int	2015
rate	double	200
Transmission	boolean	False
Type	String	Economy
Convertible	boolean	False

1.2 Command: AddService

This command will input the available additional services with the service's price into the system. There are three rental services: Navigation system, Full Insurance and Car Seat.

Command Example
AddService Navigation_system 150

Field name	Type	Example
Service	String	Navigation_system
Rate	double	150

1.3 Command: Quit

The command quit will exit the process of entering the system information.

Important Notes
• The transmission data field is Boolean: true means that the car transmission type is Manual and false means it is Automatic • The Convertible data field is Boolean: true means that the car transmission type is Convertible and false means that it is not.

Step 2: Reserve the Car

Car reservation information are found in ***Reservationinput.txt***. The first line of this file is an integer that determines the number of reservation requests to be processed. For example, in this attached file ***Reservationinput.txt***, three (3) reservations requests will be processed. In the following, a more detailed description of processing reservation requests is provided.

2.1 Command: Reserve

This command is used to reserve a car along with the desired additional services. The system should read all tokens of this command until the word **submit** is found. For each reservation, the user should indicate the car type/Transmission/Convertible, pickup/drop off location, pick up and drop off date. It also takes the customer information which are first name, last name, email, credit card number, and the customer code. At last, the system will read the requested service (**if any**).

Command Example
Reserve Sport Automatic Convertible Jeddah Jeddah 2018 11 19 2018 11 22 Saleem Omar Saleem@gmail.com 112233 786 Navigation_system submit

Field name	Type	Example
Car type	String	Sport
Transmission	String	Automatic
Convertible	String	Convertible
Pickup location	String	Jeddah
Drop off location	String	Jeddah
Pick up date	Date	2018 11 19
Drop off date	Date	2018 11 22
First name	String	Saleem
Last name	String	Omar
Email	String	Saleem@gmail.com
Credit Card	Long	112233
Customer Code	int	786
Service	String	Navigation_system

Consider the following notes when processing reservation requests:

Important Notes
• The system will read the **Transmission** type as a string. It is either "Manual" or "Automatic". Then the system should convert the string it into Boolean: True means Manual, False means Automatic.
• For each confirmed reservation, the system should generate a new **reservation reference** code: Rental initial letter + Random value (between 0 and 999) + the year of the car. Example: SO_998_2017 This code will be printed later in the **ReservationStatus.txt** (check step 3).

- **You must check for an available car based on the requested type, transmission and convertibility.** If the requested car is not available, the reservation cannot be completed, and an appropriate message is displayed. (see the ***ReservationStatus.txt)*** , for example:

SORRY: The reservation is NOT completed
There is no available Car

Step 3: Print all the information

3.1 Print the available car information

As mentioned earlier, the available cars information are read from **inputCar.txt** and are written to the file **CarInfo.txt.** Please refer to **inputCar.txt** for the specification of the **Add Car command. A**lso, refer to **CarInfo.txt** for the format of printing the car information in that file. Note that the field's transmission type and convertible are read as Boolean types from **inputCar.txt,** but they are written as strings in **CarInfo.txt**

3.2 Print the input status in the file ReservationStatus.txt.

The system should print a log of all processed reservation requests including successful and unsuccessful reservations. The invoice of successful reservations is printed, and also an appropriate message is printed for unsuccessful reservations. For example, in step 2, once the system process the following command from the ***Reservationinput.txt,*** the following information presented in the table below is written to the file ***ReservationStatus.txt:***

Command Example
Reserve Sport Automatic Convertible Jeddah Jeddah 2018 11 19 2018 11 22 Saleem Omar Saleem@gmail.com 112233 786 Navigation_system submit

Reservation Confirmation and Invoice
DONE: The reservation is completed ******Reservation Refrence number : SO_998_2017 ******Customer information : Customer Name: Saleem Omar, Email: Saleem@gmail.com, Code: 786 ******Pick up location : Jeddah   ******Drop of location : Jeddah ******Pick up date : 19-11-2018   ******Drop of date : 22-11-2018 ******Car information : The car Type: Mercedes Sport, Year: 2017, Transmission: Automatic and Convertible

```
******Additional services : Service Navigation_system
--------------- Invoice Details ---------------
 Number of reserved days: 3
 Intial Total: 1800.0
--------------- Additional Services Price ---------------
 Total After additional Services  : 1950.0
--------------- Final Payment after Discount ---------------
 Final Total  : 1560.0
```

For calculating the rental final cost, shown in the above invoice, you need to consider the following:

- o If the requested car is **"Luxury",** the daily rental rate of the car is increased by 10%.
- o Calculate the number of rental days and then use it for calculating the initial total price depending on the daily rental rate of the requested car.
- o The requested additional service cost is added to the initial cost.
- o The *customer code* determines the discount rate according to the following criteria:
  - If the code starts with 9 or 8, then the discount rate will be 20%.
  - If the code starts with 6, 5, or 4, then the discount rate will be 15%.
  - If the code starts with 3, 2,1, or 0, then the discount rate will be 10%.

1.2 UML Class Diagram for EasyRent

In addition to the main class, you should create five classes as shown in the following UML diagram. Note that you should write appropriate constructor, setter, and getter methods for all classes. (You don't need to follow the same given arguments). Be aware of the visibility (public-private) for each attribute/method.

Car

- brand : String
- carType : String
- year_of_construction : int
- car_rate : double
- transmission_Manual : boolean
- convertible : boolean

+ Car (brand : String, year : int, rate : double, transmission : boolean, type : String, convertible : boolean)
+ toString() : String
+ setBrand (brand : String) : void
+ getBrand ( ) : String
+ setCarType (type : String) : void
+ setCarType ( ) : String
+ setYearOfConstruction (year : int) : void
+ getYearOfConstruction ( ) : int
+ setCarrate (rate : double) : void
+ getCarrate ( ) : double
+ setTransmission (Transmission : boolean) : void
+ getTransmission( ) : boolean
+ setConvertible (Convertible : boolean) : void
+ isConvertible ( ) : boolean
+ calculateFinalPrice() : double

Customer

- first_name : String
- last_name : String
- email : String
- credit_Card : long
- discount_code : int

+ Customer(firstname : String, lastname : String, email : String, card : long, code : int)
+ getFirstName() : String
+ setFirstName(firstname : String) : void
+ getLastName() : String
+ setLastName(lastname : String) : void
+ getemail() : String
+ setemail(email : String) : void
+ getCreditCard() : long
+ setCreditCard(card : long) : void
+ getClientCode() : int
+ toString() : String

Service

- servicetype : String
- serviceprice : double

+ Service (type : String, price : double)
+ gettype() : String
+ settype(type : String) : void
+ getPrice() : double
+ setPrice(price : double) : void
+ toString() : String

Reservation

- reservation_code : String
- pick_up_location : String
- drop_of_location : String
- pick_up : Date
- drop_of : Date
- date_of_reservation : Date
- customer : Customer
- car : Car
- additional_services : Service

+ Reservation (pickup : String, dropof : String, dPickup : Date, dDropof : Date, customer : Customer, car : Car)
+ setReservationCode (code : String) : void
+ getReservationCode () : String
+ setPickUpLocation (pickup : String) : void
+ getPickUpLocation () : String
+ setDropofLocation (dropof : String) : void
+ getDropofLocation () : String
+ addService (currentservice : Service) : void
+ getServices () : Service
+ setPickup (pickup : Date) : void
+ getPickup () : Date
+ setDropOf (dropof : Date) : void
+ getDropOf () : Date
+ getDateOfReservation() : Date
+ getCustomer () : Customer
+ setCar (another : Car) : void
+ getCar () : Car
+ toString() : String

Important Notes:

- Use of class & object, arrays of Object, and passing object to method
- Use of Files, Reading/Writing from/on files
- Your program output must be exactly same as given sample output files.
- Your display should be in a readable form.
- Organize your code in separated methods.
- Document your code with comments.
- Use meaningful variables.
- Use dash lines between each method.
- **Delayed submission will not be accepted and there will not be any extension of the project.**

Deliverables:

- You should submit one zip file containing all java codes: BA1587412P2_EasyRent.java where BA is your section, 1587412 your ID and P2 is program 2.
- **NOTE: your name, ID, and section number should be included as comments in all files!**
  ### Input and Output Format
  Your program must generate output in a similar format to the sample run provided.
  Sample input: See sample input file.
  Sample output: See sample output files.

Good Luck!