**Kol 5ara kos emak ya 3ar9a**

**LAB Grade:**

|  |  |
| --- | --- |
| http://www.aus.edu/common/images/logo_small.gif | **American University of Sharjah**  **College of Engineering**  **Department of Computer Science and Engineering** |

**Faculty Details**

|  |  |
| --- | --- |
| Instructor: Lab Instructor :  Office:  Phone:  e-mail:  Semester: | Dr. Michel Pasquier Mr. Suresh Radder  EB2-126A  971-6-515-2924 [sradder@aus.edu](mailto:sradder@aus.edu) Spring 2017 |

**Course Details**

|  |  |
| --- | --- |
| Course:  Semester: | CMP 256 L – GUI Design and Programming  Spring 2017 |

**Lab and Assignment Details**

|  |  |
| --- | --- |
| Assignment No:  Assignment Topic:  Date:  Lab Location: | **4**  Inheritance  21/02/2017  EB2 -125 |

**Academic Integrity Pledge**

|  |
| --- |
| As a student of American University of Sharjah, I here by state that I will abide by the AUS Integrity Pledge that:   * I will hold myself accountable for all that I say and write. * I will hold myself responsible for the academic integrity of my work * I will not carry out unauthorized copying or printing of the work of others * I will not misrepresent my work nor give or receive unauthorized aid * I will behave in a manner that demonstrates concern for the personal dignity, rights and freedoms of all members of the community * I will respect university property and the property of others; and * I will not tolerate a lack of respect for these values.   **Student Name:**  **Student ID:** |

**CMP256 – GUI Design and Programming**

**Lab 4 – Inheritance**

***Objectives***

* Understand the notions of *super* classes and *sub* classes
* Understand *abstract* classes and *concrete* classes
* Understand and use generic array lists(*e.g.* *ArrayList(T)*)
* Understand and use *polymorphism*

***Exercise 1***

Implement the following class hierarchy for the Rent-a-Car Management Company, as follows:

* Car

This is an *abstract* class that represents cars to be *rented out*. The class defines the following attributes: *car id, brand, registration number, model number, status (*available, rented-out*),* and *rental charge* based on the *number of days the car is hired.*

* SmallCar

This is a concrete sub-class of the Car class. It defines the following additional attribute: *car body type* (hatchback, sedan, convertible). The rental charges are shown in the table:

|  |  |
| --- | --- |
| Brand/Usage Type | Daily |
| BMW | AED 300 |
| Mercedes | AED 400 |
| Toyota | AED 100 |

***Note:*** Must calculate the rental charges based on the number of days the car is used.

* MidRangeCar

This is a concrete sub-class of the Car class. It defines the following additional attributes: *safety* (number of air bags)and *luxury* (TV).The rental charges are shown in the table below, with no extra charge for safety but 5% extra on daily charges for luxury:

|  |  |
| --- | --- |
| Brand/Usage Type | Daily |
| BMW | AED 600 |
| Mercedes | AED 800 |
| Toyota | AED 200 |

***Note:*** Must calculate the rental charges based on the number of days the car is used.

* LargeCar

This is a concrete child class derived from the Car class. It defines additional attributes: *drive system* (two wheel or four wheel)and *passenger-cargo-volume.* The rental charges are shown in the table given below.

\

|  |  |
| --- | --- |
| Brand/Usage Type | Daily |
| BMW | AED 1000 |
| Mercedes | AED 1200 |
| Toyota | AED 500 |

***Note:*** Must calculate the rental charges based on the number of days the car is used.

* RentalContract

This is a concrete class which defines the attributes: *car id*, *customer name*, driving *license number*, *issue date* (date of renting out)*, return date* (expected/actual return date of the car that is rented out)

* RentalManager:

This class defines the attributes*: list of cars* and *list of rental contracts. This class* also defines *methods* to add/remove *a car* or *a rental contract* to/from the respective lists*.* Use an *ArrayList* class *provided* in *java.util* package for creating a list of *cars and rental contracts.*

Note: This class can be utility class and static methods can be provided.

***Additional requirements:***

* Must draw the UML diagram before starting the actual program development.

UML Link: <https://drive.draw.io/>.

* Member variables and member methods must have suitable access modifiers.
* Implement only required *setter* and *getter* methods for member attributes.
* Provide any additional *methods* as required.
* Override the *clone()* method for the classes *Car, Smallcar, MidRangeCar and LargeCar.````*
* Override the *equals()* method for the classes *Car, Smallcar, MidRangeCar and LargeCar.*
* Override the *toString()* for all the classes using a suitable print format.
* Make sure that *clone(),* *equals(),* and *toString()* use the *super* class’ equivalent method*.*
* The *issue date* and *return date* attributesin *the RentalContract* *class must* be of type *LocalDate.*
* The concept *of polymorphism* must be used wherever applicable.

Develop the main program such that the *Rent-a-Car* manager:

* + Can list all the cars available with all their details.
  + Can ask a customer to choose a car.
  + Can execute the rental contract i.e., create a rental contract and add it to the rental contract list.
  + Can calculate and display *rental charges* for a customer based on *license number*.
  + Can cancel a contract, which involves displaying the complete details of the customer (name and driving license number), details of the car rented out to the customer, and rental charges.

***Hand in:*** Solution code and sample input/output

***Due date:*** At the beginning of next lab session

***Grading policy:*** Mentioned in course outline posted on ilearn.

