OO Analysis & Design Project 2013 Instructions

- Please answer the following questions. Include any websites/books referenced at the **end** of each answer that requires a description.
- You will be required to submit solutions in hardcopy format on Thursday the 3rd of May.
- Please ensure that all questions that require a descriptive answer are written in your own words.
- Copying and pasting text is not acceptable and you will lose marks.
- The questions cover different areas of the course material.
- Include a Cover page and a table of contents. A student declaration of the following nature should be included at the back of the project. Plagiarised material will not be allocated a mark
- You may be required to attend an oral examination on Week 13th

Student declaration

I can confirm the following details:
Student ID/Registration number: 296338
Name: Mary Murphy
Module Name: OO Analysis and Design
Module Lecturer : Mary Davin
I confirm that this is my own work.
Due Date: Friday May 2nd

Student Signature:

Part A 80%

Please read the following specification and answer the questions below

Requirements Specification

"Davin's Rent a Vehicle" is a vehicle rental company which has a wide variety of makes and models available to hire 7 days a week. Davin's has branches in city locations in Ireland. The main branch office is located at Cork Airport. All branches are managed by a manager which is an employee of Davin's. Each employee works in one branch at any time. It is possible for an employee to be reassigned to a different branch on request. Each branch has its own email address and phone number.

You have been commissioned on behalf of CIT computing department to design a system that would support the needs of Davin's rental business.

The system needs to allow for the following

1. When a new vehicle is purchased for the rental company it is catalogued under a category. The following information is recoded when a vehicle is added to the system. Make, model, registration number, colour, mileage, date acquired ,image It is assigned to one branch when it is purchased.

2. Each branch uses a multi-storey car park. Each vehicle is parked in a parking space in the car park. Spaces are of two different sizes normal and large. Each space is identified by unique number which comprises of floor number and space number. This is to facilitate easy identification of the location of a vehicle. Large vehicles are parked in large spaces.
3. If a vehicle is no longer suitable for rental (involved in a serious crash or its age is too old) the system needs to be informed.
4. Customers can use the system to get a quotation for a category of vehicle they require for a time range.
5. Customers can reserve a vehicle in advance. In addition to reserving a vehicle the customer can reserve additional items such as child booster seat. (these additional items will incur an additional cost)To reserve a vehicle the customer must provide first name and last name and credit card details.

6. Vehicle rental is complicated by the following factors:
    a. The customer renting the vehicle may not be the driver, and there could be many drivers. For each driver the following information needs to be captured: full name, birth date, driver's license number, and country where license was issued.
    b. Information about the third party who may be paying the bill also needs to be captured.
    c. Extras such as collision damage waiver, liability insurance, and personal accident insurance may be added for an additional daily charge.
    d. A rental agreement consists of an agreement number, the customer's name, type of rate (daily, weekend, or weekly), vehicle identification (license number) and vehicle price category. It also includes the date, time, and mileage when vehicle leaves and expected date and time of return.
7. Returning a vehicle consists of dropping it off. Payment is a separate transaction. The following items need to be considered.
    a. The drop of location may be different to pick up location.
    b. The vehicle is rented with a full tank of fuel and the customer is responsible for refuelling the vehicle before returning it. When a vehicle is returned an amount for fuel is added if the tank is not fuel.
    c. The date, time and mileage when the vehicle is returned are recorded.
    d. The vehicle must be inspected for damage. Images are taken of the vehicle which needed to be recoded in case of future disputes. If the vehicle is damaged the details of damages needs to be recorded. If collision damage waiver is not accepted at the time of rental, the cost of repair is added to the bill.
    e. The vehicle is also inspected to determine the need of minor maintenance. If maintenance is needed, the vehicle is not cleaned up and returned to the rental lot. Instead it is sent to the rental garage for needed repairs and maintenance.
    f. If the vehicle passes inspection, it is sent to the vehicle wash and placed on the lot, ready to be rented again.
8. A bill is given to the customer if the customer is present. Otherwise, it is mailed to the customer. Often this is a third – party payer such an insurance company.
9. Payment is made by either cash or credit card. A €1000 deposit is collected at the time of rental if there is no damage waiver. The deposit is either paid in cash or charged to credit card.
10. Rates are charged as follows.
    a. A weekend is defined as the period from 6:00P.M Friday to 8:00 A.M. Monday.
    b. There is no mileage charge for the first 250 kilometres each day.
    c. Any time over the week is computed by prorating the weekly rate.
    d. The company has four price categories for the vehicles, although it should be possible for more categories to be added in the future.

    A vehicle can be either manual or automatic within a category. The rates for an automatic are a percentage more expensive than the automatic equivalent.

    Each category has the following information recorded about it. The name, engine size.

The four categories are:
Economy(E) (Ford Focus or similar.)
Budget (B) (Ford Ka or similar)
MidSize(M) (Opel Zafira or similar)
Full Size(F) (Ford Galaxy or similar)

11. A percentage discount is offered occasionally during the year. The system needs to be informed the dates when a discount applies and the rate of discount.
12. The current rates for manual vehicles are :

Class	Daily Charge	Weekly Charge	Weekend Charge	Mileage Charge
E	€24.99	€149.99	€39.99	€.20
B	€29.99	€169.99	€44.99	€.22
M	€34.99	€189.99	€49.99	€25
F	€39.99	€209.99	€54.99	€.27

**Note Theses rates can change.

Using the Rental Business System Specification Carry out the following tasks

1. *Draw up a list of all events that the car rental company encounters in making reservations, renting, accepting returned cars, inspecting, billing and receiving payment.*

2. *Create an event table using list of events identified in (i)*

3. *Create a Use Case Diagram to model functional requirements*

4. *Create a use case specification or an activity diagram for each use case.*

5. *Create a System Sequence Diagram for each Use Case.*

6. *Create a Domain Model.*

7. *Create a sequence or a Communication diagram to realize the happy scenario for each use case.*
8. *Create a Design Class Diagram to support interaction diagrams developed in (vii)*

9. *Extend Class Diagram to allow for the following additional requirements.*

a. The manager would like to determine the license numbers of all drivers for a specific vehicle that has been involved in a motorway accident while it was being rented.
b. The manager would like to determine which category of vehicle was the most popular during a given time period.

10. Develop a state chart Diagram (state transition diagram) for the different possible states of a vehicle based on information provided above.

11. Explain how the class diagram would be represented in a relational database

Important : Please note any assumptions you make

Part B 20 %

For the following questions on design patterns use UML class diagrams in your explanations.

I. What is the purpose of the pattern?
II. What problems does it solve ?
III. Reverse engineer the code in each question into a class diagram.

Q1 Explain the meaning of the MVC pattern using the following sample code.

Model.java

```
import java.awt.*;
import java.util.*;

public class Model extends Observable {

    public Color color = new Color(0x010000);
    private int count = 10;

    void doSomething() {
        if (--count <= 0) {
            color = color.brighter();
            count = 10;
            //
            setChanged();
            notifyObservers();
        }
    }
}
```

view.java

```
import java.awt.*;
import java.util.*;

public class View extends Canvas implements Observer {

    Color color;

    public void paint(Graphics g) {
        if (color ==  null) return;
        g.setColor(color);
        g.fillOval(0, 0, getWidth(), getHeight());
    }

    public void update(Observable observable, Object arg) {
        color = ((Model)observable).color;
        repaint();
    }
}
```

Controller.java

```
import java.awt.*;

public class Controller extends Frame {

    static Model model = new Model();
    static View view = new View();

    public static void main(String args[]) {
        Frame f = new Frame();
        f.add(view);
        f.setSize(200, 220);
        f.setVisible(true);

        model.addObserver(view);

        for (int i = 0; i < 200; i++) {
            model.doSomething();
            try { Thread.sleep(20); } catch (InterruptedException e) {}
        }
        f.dispose();
    }
}
```

Q2 Explain the meaning of the Observer Design Pattern using the following code

```
import java.util.Observable;

public class Subject extends Observable {
private String name;
private float price;

public Subject(String name, float price) {
this.name = name;
this.price = price;
```

```
}

public String getName() {return name;}
public float getPrice() {return price;}
public void setName(String name) {
this.name = name;
setChanged();
notifyObservers(name);
}
public void setPrice(float price) {
this.price = price;
setChanged();
notifyObservers(new Float(price));}}


import java.util.Observable;
import java.util.Observer;

public class NameObserver implements Observer {
private String name;
public NameObserver() {
name = null;
System.out.println("NameObserver created: Name is " + name);
}

public void update(Observable obj, Object arg) {

if (arg instanceof String) {
name = (String)arg;
System.out.println("NameObserver: Name changed to " + name);
}
}
}

import java.util.Observable;
import java.util.Observer;

public class PriceObserver implements Observer {
private float price;
public PriceObserver() {
price = 0;
System.out.println("PriceObserver created: Price is " + price);
}
public void update(Observable obj, Object arg) {
if (arg instanceof Float) {
price = ((Float)arg).floatValue();
System.out.println("PriceObserver: Price changed to " +
price);
}
}
}
```

```
public class TestObservers {

public static void main(String args[]) {

// Create the Subject and Observers.
Subject s = new Subject("Corn Pops", 20.5f);
NameObserver nameObs = new NameObserver();
PriceObserver priceObs = new PriceObserver();

// Add those Observers!
s.addObserver(nameObs);
s.addObserver(priceObs);

// Make changes to the Subject.
s.setName("Frosted Flakes"); // It prints NameObserver: Name changed to
Frosted Flakes

s.setPrice(30.6f); //It prints PriceObserver: Price changed to 30.6

s.setPrice(50.8f); //It prints PriceObserver: Price changed to 50.8

s.setName("Sugar Crispies"); // It prints NameObserver: Name changed to Sugar
Crispies
}
```

Q3 Explain the meaning of the façade design pattern using the following code..

```
class CPU {
    public void processData() { }
}

class Memory {
    public void load() { }
}

class HardDrive {
    public void readdata() { }
}

class Computer {
    private CPU cpu;
    private Memory memory;
    private HardDrive hardDrive;

    public Computer() {
        this.cpu = new CPU();
        this.memory = new Memory();
```

```
        this.hardDrive = new HardDrive();
    }

    public void run() {
        cpu.processData();
        memory.load();
        hardDrive.readdata();
    }
}


class User {
    public static void main(String[] args) {
        Computer computer = new Computer();
        computer.run();
    }
}
```