COMP 2920: Software Architecture & Design

Assignment #2

Max marks: 15 + 10 Weight: 4%

**Objectives:** To gain an understanding of the structural model

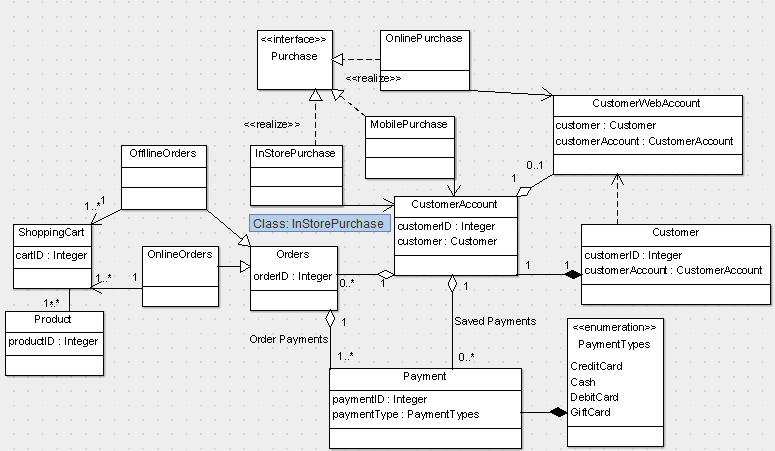
PART-A: Class Diagram

Q1. Design a shopping system that may have online or offline orders. Consider this as a common ground between business analysts and software developers.

Business rules are:

1. Each customer has unique id and is linked to exactly one **account**.
2. Each Account will own shopping cart and orders.
3. Customer could register as a web user to be able to buy items online, but is not necessary as purchases could also be made by phone or by ordering from catalogues.
4. Web user has login name which also serves as unique id and must be linked to a **shopping cart**. Shopping cart belongs to account.
5. Each order could refer to several **payments**, possibly none.
6. Every payment has unique id and is related to exactly one account.
7. Each order may have many products but at least 1
8. A product could be associated to many orders or no order at all.

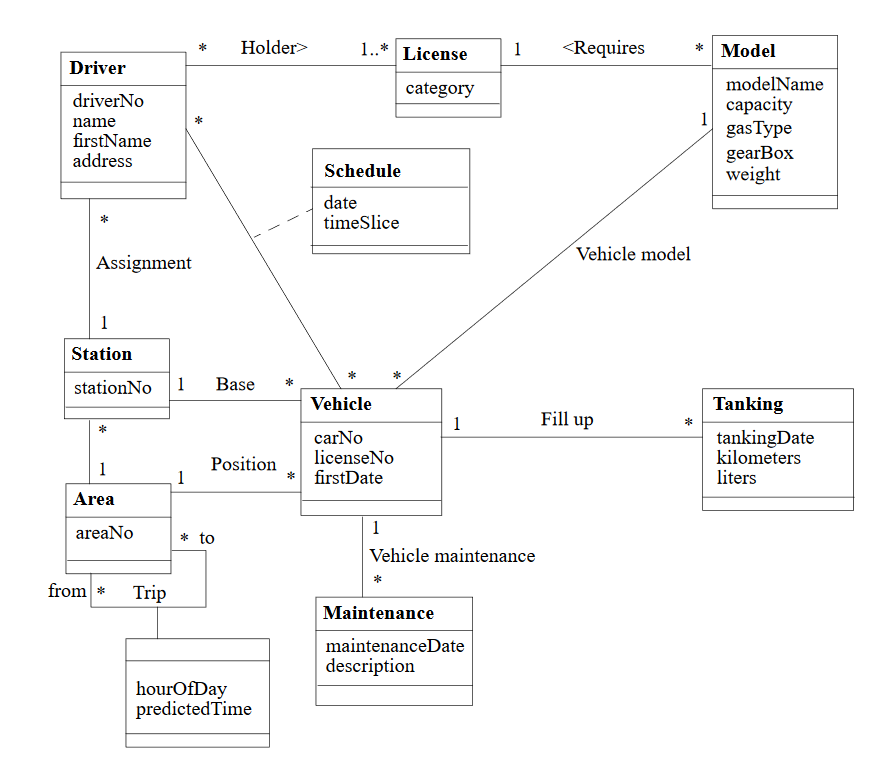
Generate the classes and make the relationships. You may submit the diagram drawn by hand or using ArgoUML



Classes and Interfaces:

1. Customer
2. CustomerAccount
3. CustomerWebLogin
4. Orders
5. ShoppingCarts
6. OrderItems
7. Purchase -> Interface
8. OnlinePurchase
9. PhonePurchase
10. InStorePurchase
11. Payment
12. PaymentTypes

Q2. Study the diagram below and explain its relationships.



* Each Area is identified by areaNo and can have many Vehicles and stations and can have many trips to and from different areas.
* Each station has a unique id as stationNo can have many vehicles at its base and drivers assigned to it
* A driver has a driverNo, name and address properties and can be assigned to multiple vehicles and also uses a Schedule with date and timeSlice properties.
* For a driver to be able to use a vehicle, one must hold a licence which permits to drive the specific vehicle models.
* A vehicle can be assigned 0 or many drivers and also follows the Schedule. Each vehicle has a Tanking data to record the tankingDate, kilometers and liters of fuel used. Also, each vehicle is subjected to vehicle inspection.
* A vehicle has a specific model with modelName, capacity, gearbox, gasType and weight data.

Part B: Relationships

Q1. Look at the way we may define the relationships between the classes so that we know how objects can relate to each other. Following are the common relationships between classes:

* aggregation
* inheritance
* using
* association
* instantiation
* Composition

To do: Explain the different relationships using examples. Make a comparison between aggregation vs composition relationships.

1. Aggregation: This is a relation in which one object contains another object but both can coexist without each other and have independent life spans.
   * Station and Vehicle
   * Vehicle and Driver
   * Driver and Licence
   * Area and Station
   * Area and Vehicle
2. Inheritance: In this, a child class contains all properties and functions from its parents and has others that are added to it.
   * Vehicle can Inherit Model Class
   * Maintenance can have sub classes to define repairs and special maintenance tasks to be done
3. Using: This would be when a class uses functions from another class without necessarily containing it
   * Driver Uses the Schedule
4. Association: This is when a class uses functionality of another class and contains an instance of the class in it as a variable
   * Vehicle needs a vehicle model
   * Vehicle fills up tanking
   * Driver holds a Licence
   * Driver is assigned to station
5. Instantiation: This is a type of relation where operations in one class create objects of another class.
6. Composition: This is when a class has instances of other classes and the lifecycle of the other objects depends on this class objects lifecycle
   * A Vehicle is composed of multiple parts eg Tanks
   * A driver is composed of body parts
   * A station is composed of structures
   * A model is composed of gearbox, gasType, weight

Q2. Booch proposes five metrics to measure the quality of classes:

* Coupling
* Cohesion
* Sufficiency
* Completeness
* Primitiveness

To do: Explain the above terms using examples and define the measure of these metrics important for a system.

* Coupling: This defines how closely classes rely on each other and how changes in one class cause changes in other classes. A high coupling is discouraged since if one class code is changed, it would need all classes using this class to also have changes made.
* Cohesion: measures how strongly classes work together. A high cohesion is desired where class functionalities ae clearly defined and implemented.
* Sufficiency: Measures how many of the characteristics a class covers of the entities that it represents.
* Completeness: Measures the behaviour and functionality implementation of the class compared with the entity that it represents.
* Primitiveness: This measure how the behaviour of the class functions depending on the state of the class object and not the state of other methods. In simple terms, it is how methods are independent of each other.