Assignment – 2

Name: Malika Hafiza Pasha
Student ID: 212238171
Course: CSC 581 (Advanced Software Engineering)

1. Who is an actor, and what is the difference between a primary and secondary actor? Explain your answer with an example.

Answer:

- A role or entity that communicates with the software system is called an **actor**.
- Actors might be external software components, systems, hardware, or human users who interact with the system under development.
- These actors engage with the system to accomplish certain objectives and have distinct roles or duties within it.

Primary Actor:

- A user or system component that interacts with the system primarily to accomplish a particular objective is considered the primary actor.
- For the feature to be successfully executed, they must start the use case or functionality.
- Primary actors have specified objectives that are related to the system and are the primary users of the system's services.

Example 1: **Student Information System**
Primary Actor: Student
Description: The student interacts with the system to access their grades, enroll in courses, and view academic information.

Example 2: **Social Media Platform**
Primary Actor: User (e.g., individual, business, organization)
Description: The user interacts with the platform to post content, connect with others, and engage in social interactions.

Secondary Actor:

- User or system component that helps the primary actor achieve their objectives or is impacted by the use case but is not a direct beneficiary is referred to as a secondary actor.
- Despite not being the primary users of the system for that specific functionality, they engage with it to offer input, assistance, or receive information.

Example 1: **Flight Booking System**
Secondary Actor: Airline Reservation System
Description: The airline reservation system interacts with the flight booking system to confirm flight availability and process bookings initiated by the primary actor (passenger).

Example 2: **Online Banking System**
Secondary Actor: Third-Party Verification Service
Description: A third-party verification service interacts with the online banking system to authenticate and verify transactions initiated by the primary actor (account holder).

2. How is a use case diagram different from a use case? (20 points)

Answer:

Use Case:

A use case describes a particular functionality or behavior that a system ought to display. It describes how a system's software interacts with its external components, or actors, to accomplish a specific objective. A use case often consists of a series of operations or steps that depict a sequence of actions, encompassing both the system's and external actors' actions.

Example of Use Case: **Making a purchase.**
Primary Actor: Customer
Steps:

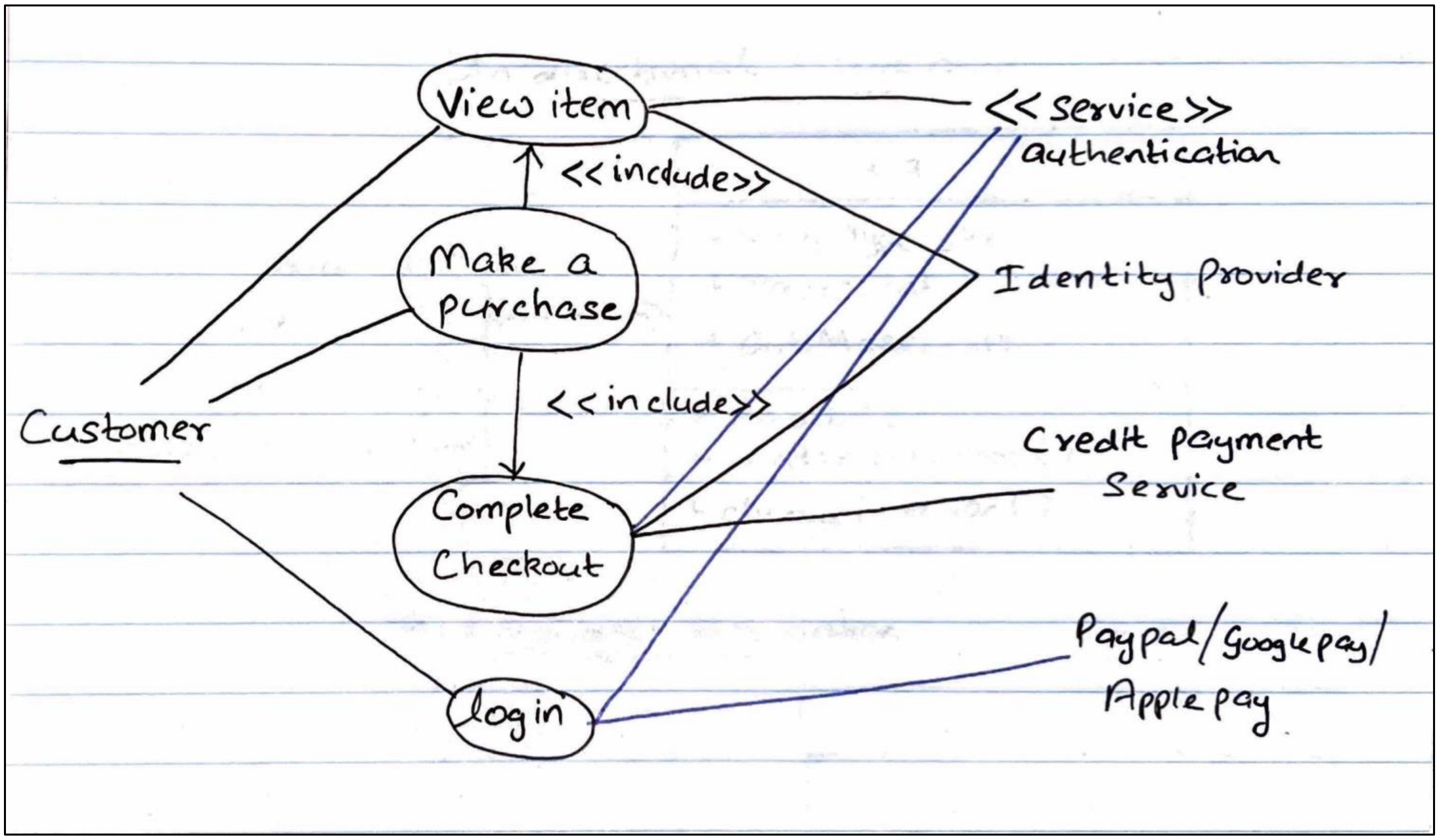
1. Customer selects items to purchase.
2. The customer provides payment information.
3. System processes the payment.
4. System updates the inventory.
5. System sends a purchase confirmation.

Use Case Diagram:

A use case diagram is a visual representation that shows the functioning of the system and how outside factors (actors) interact with it. Visualizing the many use cases and how they relate to the actors is useful. The diagram demonstrates the functional needs of the system in a clear and succinct manner by including use cases, actors, and the relationships between them.

Components of a Use Case Diagram:
Actors: Represent external entities interacting with the system.
Use cases: Represent particular system characteristics or functionalities.
Relationships: Indicate the interactions between the actors and the use cases.

Example of Use Case Diagram: **Online Shopping System**



3. Provide examples of the three use-case-to-use-case relationships. Explain your answers.

Answer:

Include relationship:

When one use case contains the actions of another use case, this is represented by the "include" relationship. Typically, this connection is used to demonstrate how a particular functionality (the included use case) is always included in the base use case's behavior.

Example of include relationship:

Take the e-commerce use case "Purchase Item" as an example. Since adding an item to the cart is a requirement for making a purchase, this use case may also include the "Add to Cart" use case. The actions of adding the item to the cart are part of the "Purchase Item" use case.

Extend relationship:

The relationship "extend" is used to represent optional or constrained behavior that could expand the scope of the base use case. The use case for extensions is optional and only runs under specific circumstances.

Example of extend relationship:

The fundamental use case of a "Online Payment" system could be "Make Payment." "Apply Discount," which expands the base use case if the user has a discount code, could be an extension to this. Only when the user has a working discount code does this use case—which is optional—occur.

Inherit (Generalized) relationship:
When one use case is a specialized form of another use case, inheritance is modeled using the "generalization or inherit" connection. It is comparable to object-oriented programming's inheritance.

Example of inherit relationship:
Take the "Manage User Profile" use case as an example. Specializations for this use case include "Edit User Profile" and "View User Profile." The "Edit User Profile" and "View User Profile" use cases may include additional or specialized behavior in addition to behavior that is inherited from the basic "Manage User Profile" use case.

4. What are the three key parts of the definition of a class, and what is a class diagram? Answer with an example.
Answer:
The three key parts of the definition of a class are:

1. **Class Name:**

The name that identifies the class. It is used to create objects based on the class blueprint.

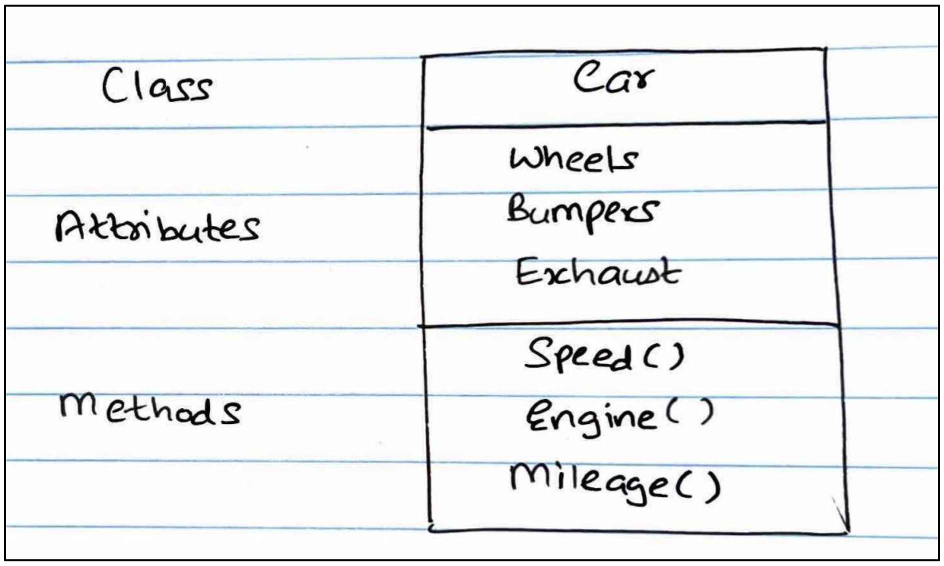
2. **Attributes (Properties, Fields):**

The data members or variables that represent the characteristics or properties of objects created from the class.

3. **Methods (Functions, Operations):**

The functions or operations that define the behavior or actions that objects created from the class can perform.

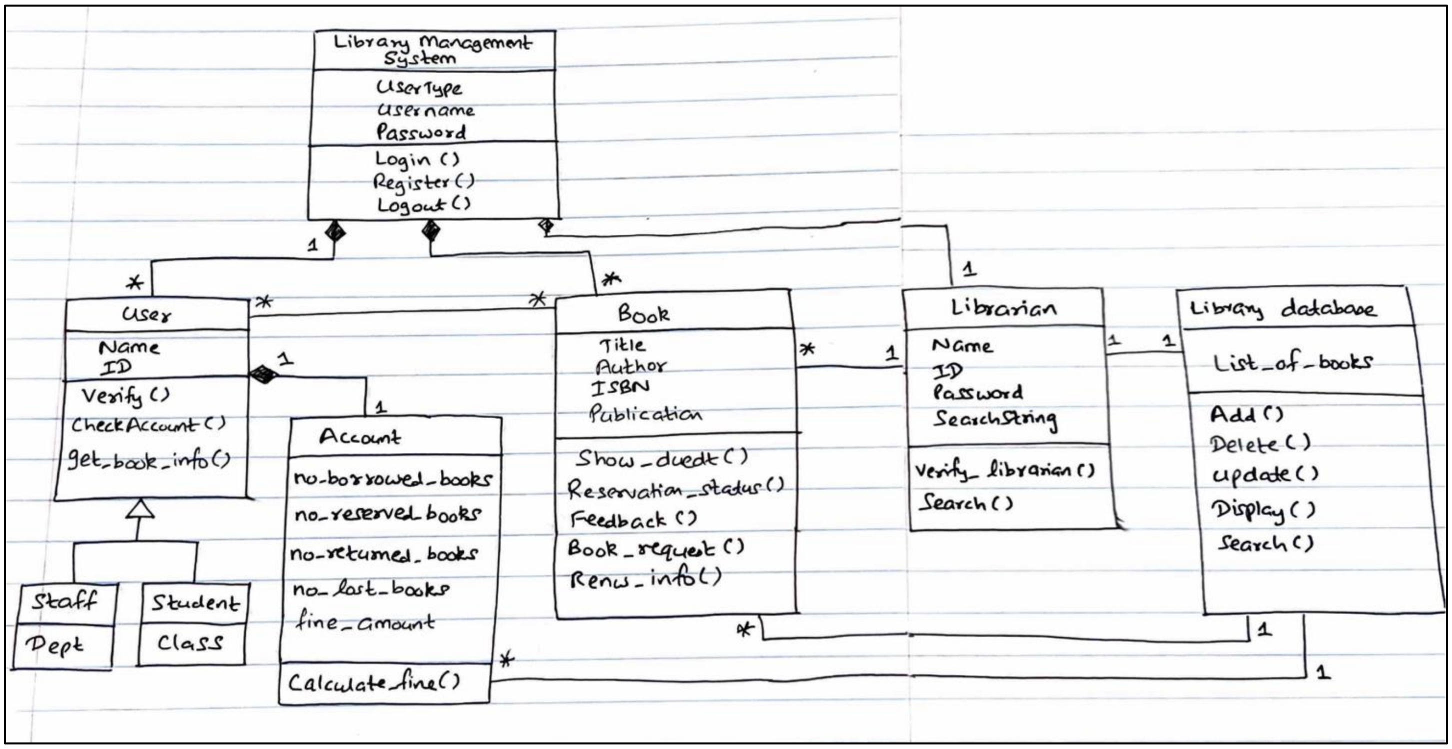
Example of Class Name, Attributes and Methods:



Class Diagram:

A class diagram is a particular kind of UML (Unified Modeling Language) diagram that shows the organization and connections between the classes in a system. A high-level view of the system's design is provided by the visual representation of the classes, attributes, methods, and their associations.

Example of Class Diagram: **Library Management System**



5. Explain the two main relationships in a class diagram with examples. (It is mandatory to create a sketch of a class diagram to answer this question.)

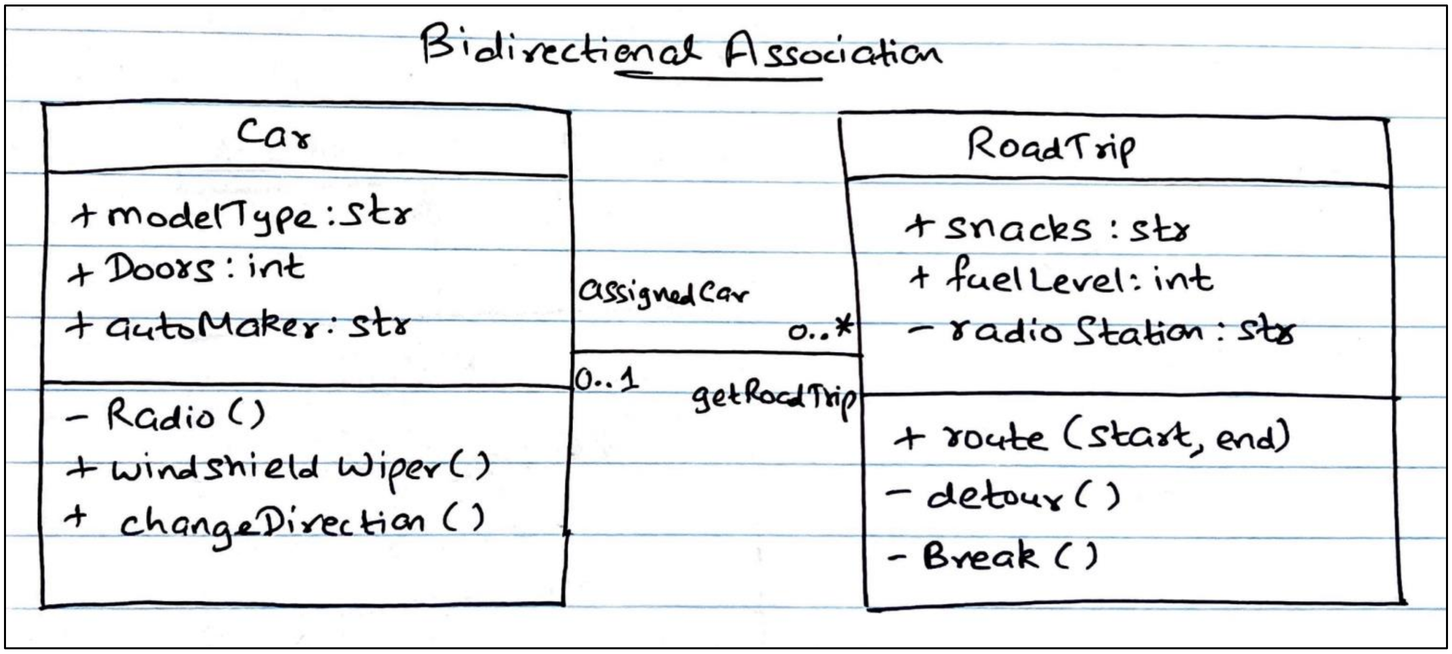
Answer:

Association Relationship

A relationship between two classes that is reciprocal and denotes a connection between them is called an association.
It serves to illustrate the relationships between classes in terms of their instances or objects.
One-to-one, one-to-many, or many-to-many associations are all possible.

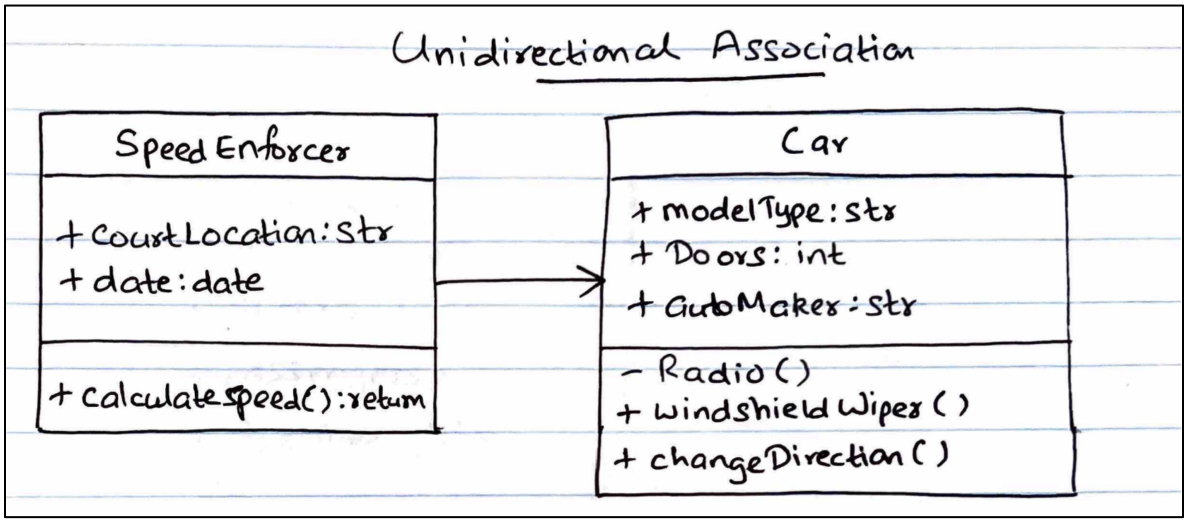
Example of Association Relationship:

Bidirectional Relationship



RoadTrip and the Car classes are connected. When an instance of RoadTrip exists, it can either have one instance of Car associated with it or not be connected with any Cars because the Car at one end of the line assumes the association of "assignedCar" with the multiplicity value of 0..1. To show that a RoadTrip could have numerous instances of Cars connected with it in this instance, a separate Caravan class with a multiplicity value of 0..* is required. The multiplicity number is set to 0 because one automobile instance could have several "getRoadTrip" associations, or in other words, one automobile could travel on multiple journeys.

Unidirectional Relationship

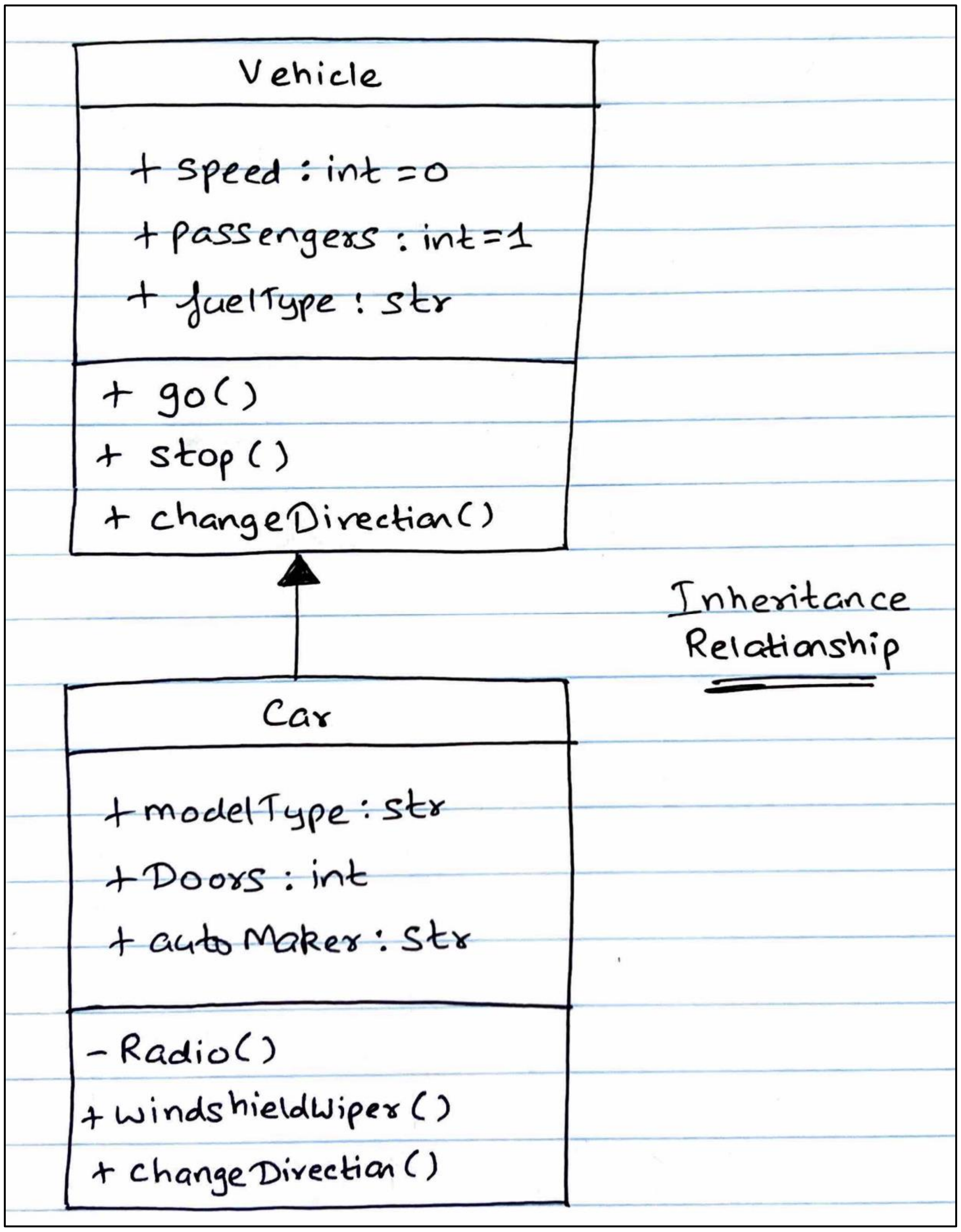


If we encounter a speed trap while traveling when a speed camera records your driving behavior, but you won't be aware of it until we receive a notice in the mail. Although it isn't depicted in the illustration, the multiplicity number in this scenario would be 0..* based on how frequently we pass the speed camera.

Inheritance Relationship

Inheritance is an “is-a" relationship between classes, representing an "is a kind of" or "is a type of" relationship.
It is used to simulate how properties and methods are passed down from the base.
The subclass inherits the properties and behaviors of the superclass.

Example:



In this example, the object "Car" would inherit not only the specific attributes (model type, number of doors, auto maker), but also the specific attributes (speed, numbers of passengers, fuel) and methods (go(), stop(), changeDirection()) of the parent class ("Vehicle"). In a class diagram, inheritance is represented by a solid line and a closed, hollow arrow.