MIT World Peace University
SET

Assignment 3 and 4

Naman Soni Roll No. 10

Contents

1 Aim

Object Oriented Analysis and design using UML diagrams: Draw Use case and Class Diagram using Open-Source Tool.

2 Objectives

The tasks we have to do are:

1. You will have to identify the main entities (objects) for this system.
2. You will have to find out the relationships between these objects.
3. You will have to find the necessary attributes and function that need to be associated with each object to implement the functionality mentioned above.
4. You will make a final comprehensive diagram show and all objects and their relations along with their at- tributes and functions.

3 Problem Statement

Draw UML Use Case Diagram and Class Diagram for the following problem statement: The restaurant industry is a fast-paced environment that requires efficient management of resources to ensure timely and quality service. In the traditional restaurant setting, the ordering process, food preparation, and inventory management are carried out manually, leading to errors, delays, and inefficiencies.

4 Theory

4.1 *Use Case Diagram*

4.1.1 What is a 'Use Case Diagram'?

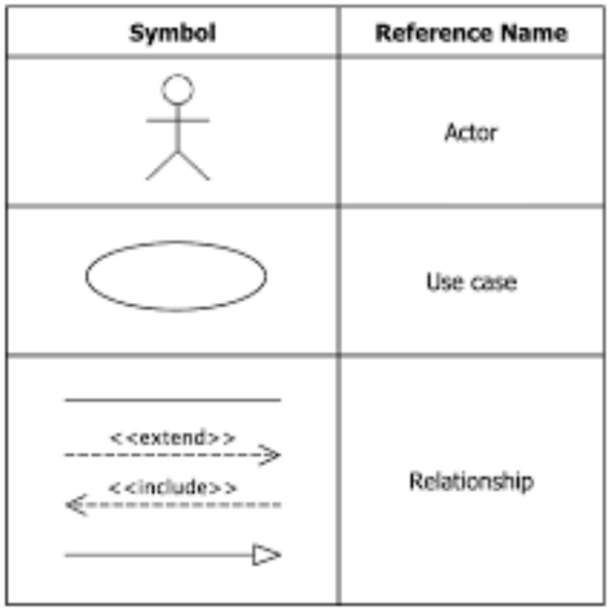
A use case diagram is a graphical representation of the interactions between a system. It is a behavioral diagram that shows the functionality of a system using actors and use cases. Use cases are the actions that can be performed on the system to achieve a goal. Actors are the entities that interact with the system to perform these use cases. Use cases are represented as ovals and actors are represented as stick figures.

4.1.2 What is the use of 'Use Case Diagram'?

- It is a graphical representation of the functional requirements of the system.
- It helps in understanding the system from the user's perspective.
- It is useful in understanding the functional requirements of the system.
- It helps in identifying the actors and use cases of the system.
- It helps in identifying the relationships between the actors and use cases.
- It helps in identifying the relationships between the use cases.

4.1.3 Elements of Use Case Diagram

1. **Use Case:** A use case is a set of actions that can be performed by an actor to achieve a goal. It is represented as an oval.
2. **Actor:** An actor is an entity that interacts with the system to perform a use case. It is represented as a stick figure.
3. **Relationship:** There are three types of relationships between actors and use cases.
   - **Association:** Association is a relationship between an actor and a use case. It represents that the actor can perform the use case. It is represented as a solid line between an actor and a use case.
   - **Generalization:** Generalization is a relationship between two use cases. It represents that the second use case is a more specific version of the first use case. It is represented as a solid line with an arrow pointing from the first use case to the second use case.
   - **Include:** Include is a relationship between two use cases. It represents that the second use case is a part of the first use case. It is represented as a dashed line with an arrow pointing from the first use case to the second use case.
   - **Extend:** Extend is a relationship between two use cases. It represents that the second use case is an alternative version of the first use case. It is represented as a dashed line with an arrow pointing from the second use case to the first use case.



Symbol	Reference Name
	Actor
	Use case
<<extend>> <<include>>	Relationship

4.2 Class Diagrams

4.2.1 What is a Class Diagram?

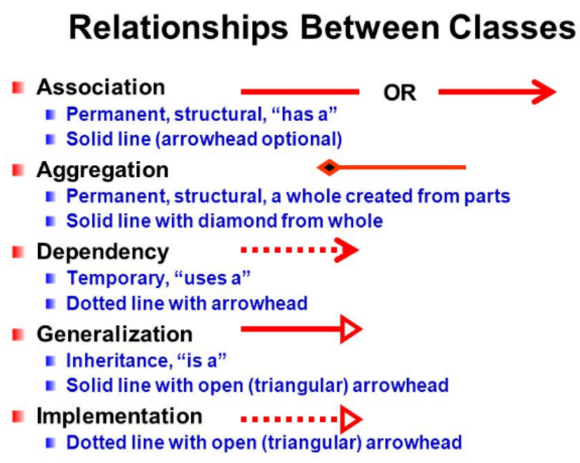
A class diagram is a static diagram that shows the structure of a system. It is a structural diagram that shows the classes and their relationships in a system. It describes the attributes and operations of a class. It shows the static structure of a system.

4.2.2 What is the use of Class Diagram

1. It is a graphical representation of the static structure of a system.
2. It helps in understanding the structure of a system.
3. It describes the classes and their relationships in a system.
4. It is useful in identify the classes of a system.
5. It is useful in identifying the relationships between the classes.

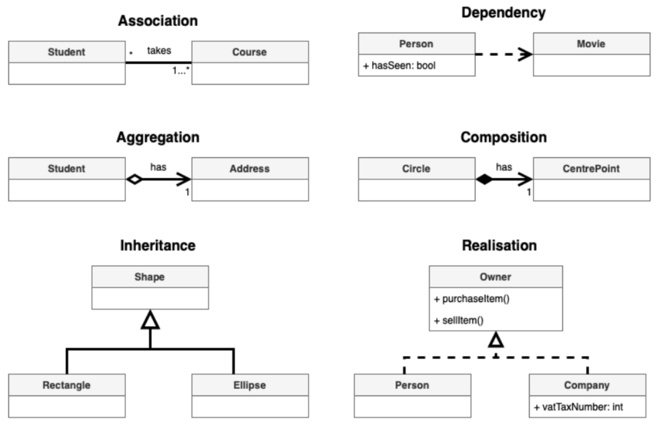
4.2.3 Elements of a Class Diagram

1. **Class:** A class is a set of objects that have similar attributes and operations. It is represented as a rectangle.
2. **Attributes:** Attributes are the properties of a class. It is represented as a variable inside a class.
3. **Operations:** Operations are the functions that can be performed on a class. It is represented as a function inside a class.
4. **Relationships:** There are three types of relationships between classes:
    - **Association:** Association is a relationship between two classes. It represents that the second class is a part of the first class. It is represented as a solid line between two classes.
    - **Generalization:** Generalization is a relationship between two classes. It represents that the second class is a more specific version of the first class. It is represented as a solid line with an arrow pointing from the first class to the second class.
    - **Dependency:** Dependency is a relationship between two classes. It represents that the second class is dependent on the first class. It is represented as a dashed line between two classes.



Relationships Between Classes

- Association — OR →
  - Permanent, structural, "has a"
  - Solid line (arrowhead optional)
- Aggregation
  - Permanent, structural, a whole created from parts
  - Solid line with diamond from whole
- Dependency
  - Temporary, "uses a"
  - Dotted line with arrowhead
- Generalization
  - Inheritance, "is a"
  - Solid line with open (triangular) arrowhead
- Implementation
  - Dotted line with open (triangular) arrowhead



5 Procedure

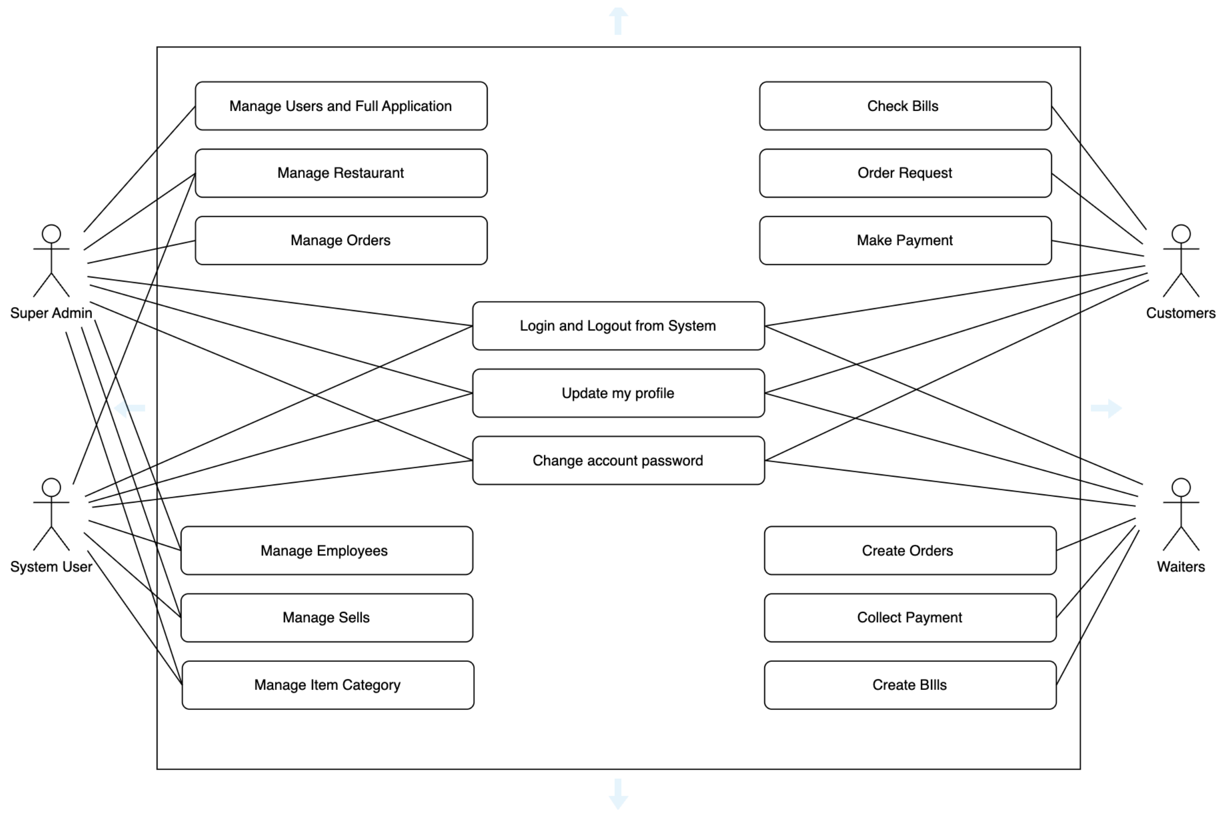
User Case Diagram

- Create Actors to represent classes of people, organizations, other systems, software or devices that interact with your system or subsystem.
- Create Use Cases for each of the goals that each actor seeks to achieve with the system.
- Use Associations to link actors to use cases.
- Use <<include >>, <<extend >>and generalization to show the relationships among the use cases

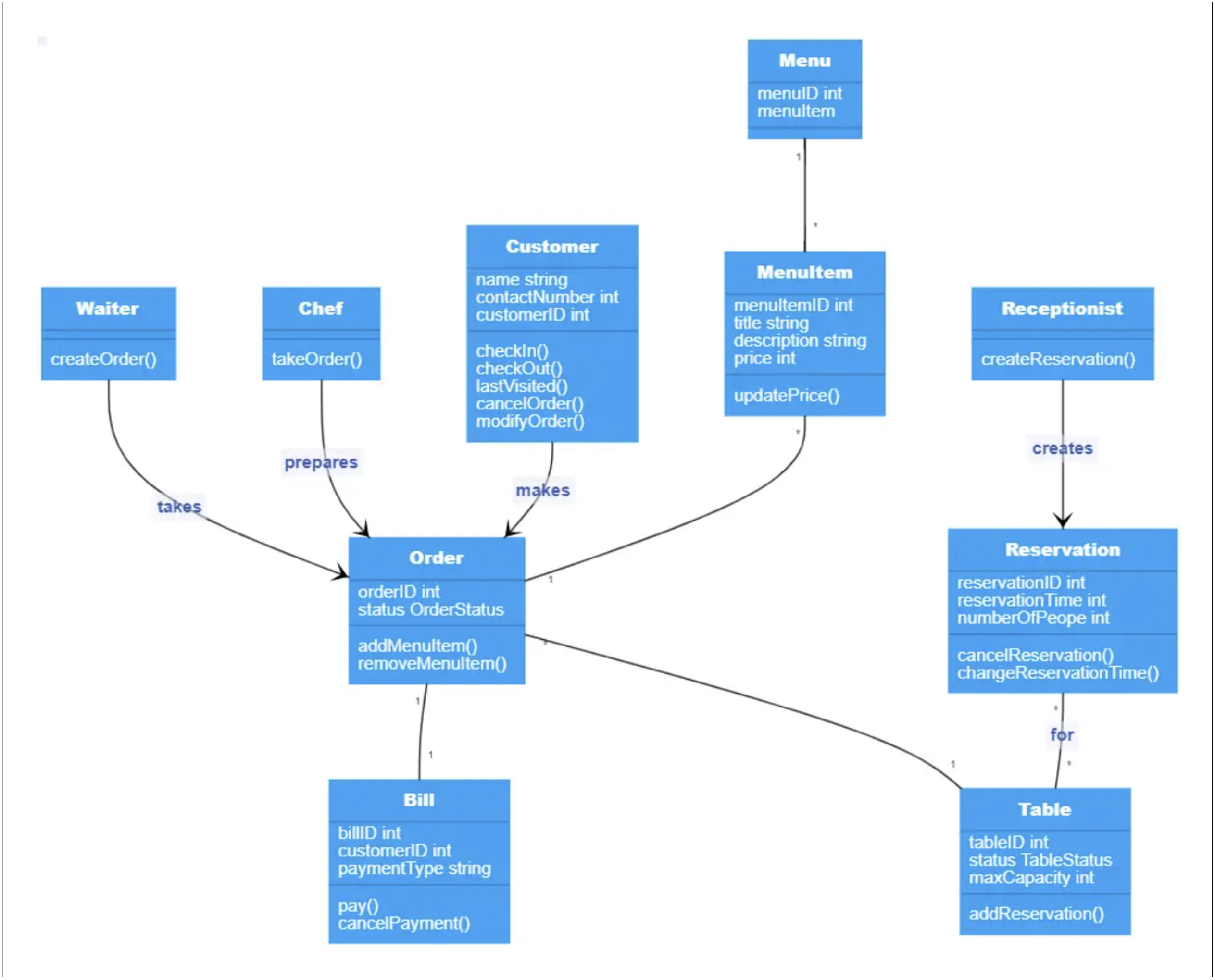
Class Diagram

- Write a problem statement, clearly defining the scope of the system.
- List out nouns as the probable classes from the problem statement.
- Draw the classes of the nouns focus should be on understanding relationships.
- Draw model of the system- the class and model. show the relationships between the classes.
- multiplicity in the class model. Data types not required.

6 Use Case Diagram:



7 Class Diagram



8 Platform

Operating System:MacOS 64-bit
IDE:Visual Studio Code
External Program for Diagrams:Draw.io

9 Conclusion

Thus, we learnt about Use Case Diagrams and UML Class Diagrams in Detail.

10 FAQ's

10.1 What kind of relationships classes have? Explain all relationships with examples.

Ans. There are several types of relationships between classes. Here are three main types of relationships:

1. **Inheritance:** Inheritance is a relationship between two classes where one class (the child or derived class) inherits properties and behavior from another class (the parent or base class). This relationship is used to reuse code and to create a hierarchy of classes with increasing levels of specialization.
   For Example: Consider a class hierarchy for different types of vehicles in software. The base class could be "Vehicle" which has methods like "start" and "stop". The derived classes could be "Car", "Truck", and "Motorcycle", each inheriting the methods of the base class while also having their own unique methods.

2. **Association:** Association is a relationship between two classes where one class uses an instance of another class. This is often referred to as a "has-a" relationship. This relationship is used to represent real-world relationships between objects.
   For Example: Consider a class for a Library Management System which has a relationship with a class for Books. The Library class might have a property called "books" which is an array of Book objects. This represents an association between the two classes, where the Library class "has-a" collection of Book objects.

3. **Dependency:** Dependency is a relationship between two classes where one class depends on the other class. This means that changes to one class may affect the behavior of the other class.
   For Example: Consider a class for a Payment Gateway which depends on a class for Credit Card Validation. If the Credit Card Validation class changes its behavior, it may affect the behavior of the Payment Gateway class. This relationship is often represented by an arrow pointing from the dependent class to the independent class.

10.2 Explain any 2 terminologies used in Use case diagrams.

Ans. Two terminologies used in Use Case Diagrams are:

1. **Actor:** An actor is an external entity that interacts with the system. Actors are represented by stick figures in use case diagrams. They represent the roles played by people, systems, or devices that interact with the system being developed.
   For Example: Consider a use case diagram for an online shopping system. The customer and the system administrator are actors in this system. The customer interacts with the system to browse and purchase products, while the system administrator manages the products, orders, and user accounts.

2. **Use Case:** A use case represents a specific functionality or behavior of the system. Use cases are represented by ovals in use case diagrams. They describe the actions that an actor can perform in the system. **For example:** In the online shopping system, a use case could be "Browse Products", "Add to Cart", "Checkout", "Manage Products", "Manage Orders", etc. Each of these use cases represents a specific behavior of the system that can be initiated by an actor.

10.3 Explain the aggregation and composition in diagram?

1. **Aggregation:** Aggregation is a relationship between two classes where one class is a part of another class. The part class can exist independently of the whole class, and can be part of other whole classes as well. The relationship is represented by a diamond shape on the side of the whole class, pointing to the part class.
   For example Consider a class hierarchy for a car manufacturing system. The Car class has an aggregation relationship with the Wheel class. The Wheel class can exist independently of the Car class and can be part of other classes, such as a Bicycle class. The Car class can also have a relationship with other part classes, such as the Engine class and the Seat class.

2. **Composition:** Composition is a relationship between two classes where one class is made up of another class, and the part class cannot exist independently of the whole class. The whole class owns the part class and is responsible for its creation and destruction. The relationship is represented by a filled diamond shape on the side of the whole class, pointing to the part class.
   For example Consider a class hierarchy for a music player application. The MusicPlayer class has a composition relationship with the Playlist class. The Playlist class cannot exist without the MusicPlayer class, and is created and destroyed by the MusicPlayer class. The MusicPlayer class can have other part classes as well, such as the Song class and the User class, but the Playlist class is an essential part of the MusicPlayer class and cannot exist independently.