MIT World Peace University

Software Engineering and Testing
Second Year B. Tech, Semester 4

UML Use Case and Class Diagrams

Assignment No. 3 - *UML Diagrams*
and Assignment No. 4 - *Class Diagrams*

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

March 3, 2023

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 functions 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 attributes and functions.

3 Problem Statement

Draw UML Use Case Diagram and Class Diagram for The Following Problem:

The Purpose of an Attandence Assistant App is to help reduce the time taken for recording the attendance of a classroom in a school or college. The app will be able to record the attendance of a class in a matter of a few Seconds with minimum Energy Expended. It will record data on cloud, and be accessible to all the Teachers.

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 the elements of 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. A use case diagram is a dynamic diagram because it shows the behavior of the system.

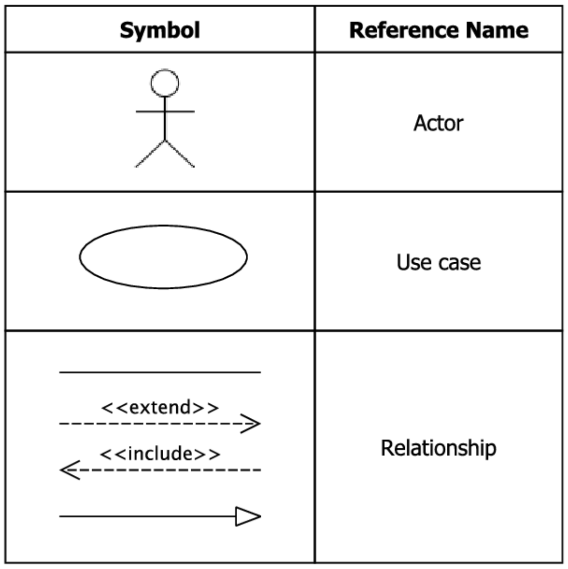
4.1.2 What is the use of a Use Case Diagram

1. It is a graphical representation of the functional requirements of the system.
2. It helps in understanding the system from the user's point of view.
3. It is useful in understanding the functional requirements of the system.
4. It helps in identifying the actors and use cases.

5. It helps in identifying the relationships between actors and use cases.
6. It helps in identifying the relationships between use cases.

4.1.3 Elements of a Use Case Diagram

1. **Use Case:** A use case is a set of actions that are performed by an actor to achieve a goal. It is a scenario that describes the interaction between a user and a system to achieve a particular goal. A use case is represented as an oval in a use case diagram.
2. **Actor:** An actor is an entity that interacts with the system to perform a use case. An actor can be a person, an organization, a software system, or a device. An actor is represented as a stick figure in a use case diagram.
3. **Relationships:** There are three types of relationships between actors and use cases.
   (a) **Association:** Association is a relationship between an actor and a use case. It shows that the actor is associated with the use case. It is represented as a line with a closed arrowhead. The arrowhead points from the actor to the use case.
   (b) **Generalization:** Generalization is a relationship between two use cases. It shows that one use case is a more general use case than the other use case. It is represented as a line with an open arrowhead. The arrowhead points from the more general use case to the more specific use case.
   (c) **Include:** Include is a relationship between two use cases. It shows that one use case is a part of the other use case. It is represented as a line with a solid arrowhead. The arrowhead points from the more general use case to the more specific use case.
   (d) **Extend:** Extend is a relationship between two use cases. It shows that one use case can be extended by the other use case. It is represented as a line with a dashed arrowhead. The arrowhead points from the more general use case to the more specific use case.



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

Figure 1: Symbols of a Use Case Diagram

4.2 Diagram

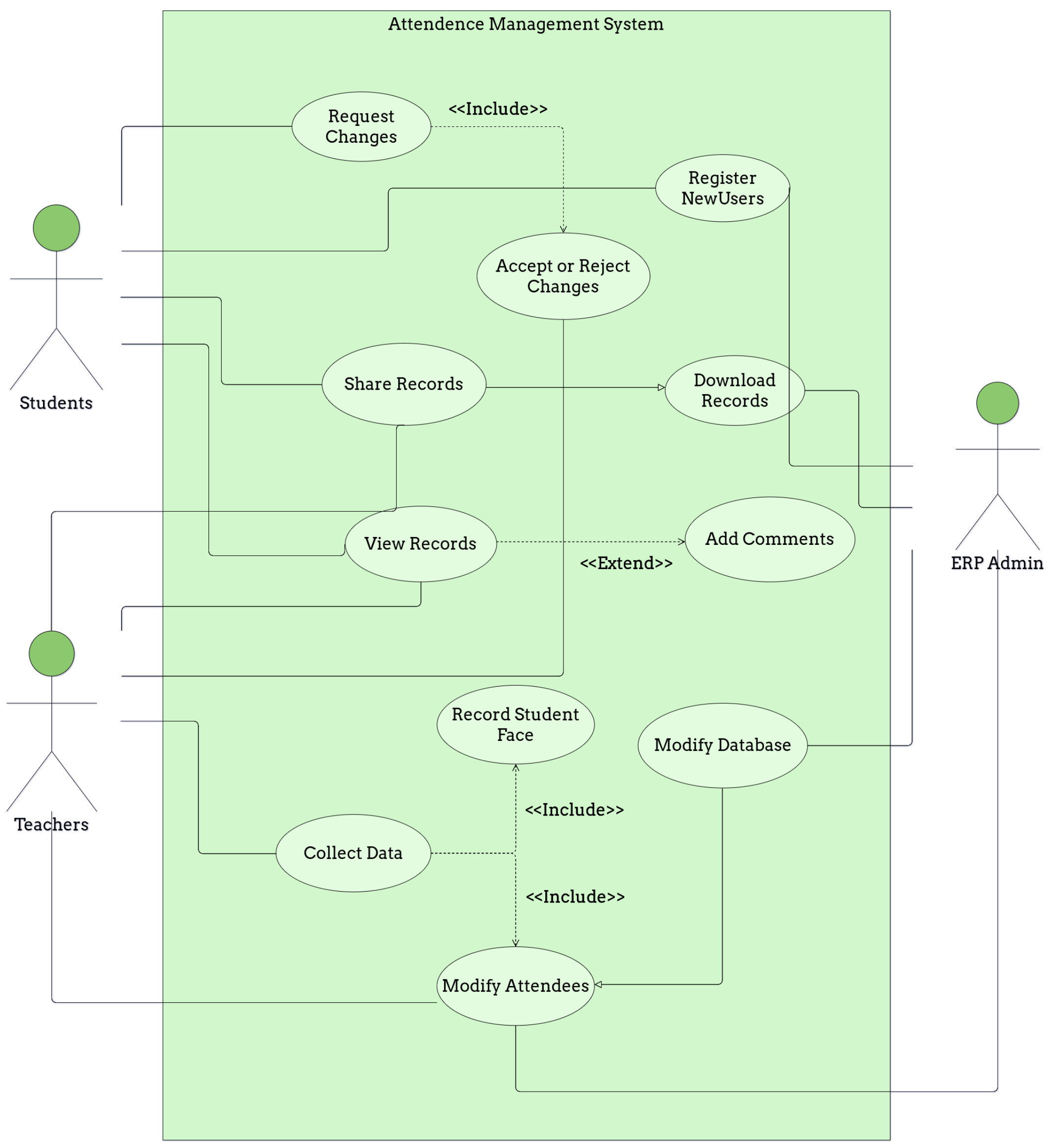


Figure 2: Use Case Diagram

4.3 Class Diagram

4.3.1 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.*

4.3.2 What is the use of a 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 describes the attributes and operations of a class.
5. It is useful in identifying the classes of a system.
6. It is useful in identifying the relationships between classes.

4.3.3 Elements of a Class Diagram

1. **Class:** A class is a collection of objects that have similar attributes and operations. It is the basic building block of object-oriented systems. A class is represented as a rectangle in a class diagram.
2. **Attributes:** An attribute is a variable that describes the state of a class. It is a data member of a class. An attribute is represented as a variable in a class diagram.
3. **Operations:** An operation is a behavior of a class. It is a function that describes the behavior of a class. An operation is represented as a function in a class diagram.
4. **Relationships:** There are four types of relationships between classes.
   (a) **Association:** Association is a relationship between two classes. It shows that two classes are associated with each other. It is represented as a line with a closed arrowhead. The arrowhead points from the class that has a whole part relationship to the class that has a part-whole relationship.
   (b) **Generalization:** Generalization is a relationship between two classes. It shows that one class is a more general class than the other class. It is represented as a line with an open arrowhead. The arrowhead points from the more general class to the more specific class.
   (c) **Dependency:** Dependency is a relationship between two classes. It shows that one class is dependent on another class.
   (d) **Composition:** The composition is a special type of aggregation which denotes strong ownership between two classes when one class is a part of another class. For example, if college is composed of classes student. The college could contain many students, while each student belongs to only one college.
   (e) **Aggregation:** Aggregation. refers to the formation of a particular class as a result of one class being aggregated or built as a collection. For example, the class "library" is made up of one or more books, among other materials. In aggregation, the contained classes are not strongly dependent on the lifecycle of the container.
   (f) **Multiplicity:** Multiplicity can be set for attributes, operations, and associations in a UML class diagram, and for associations in a use case diagram. The multiplicity is an indication of how many objects may participate in the given relationship or the allowable number of instances of the element.

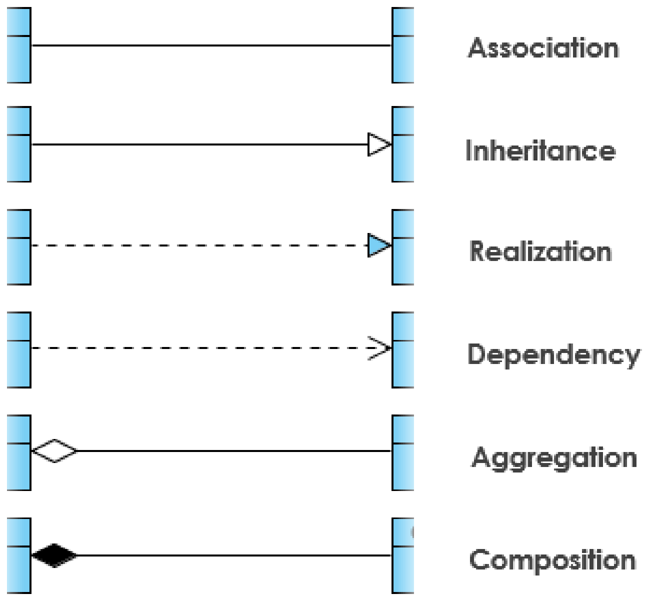


Figure 3: Relationships between Classes

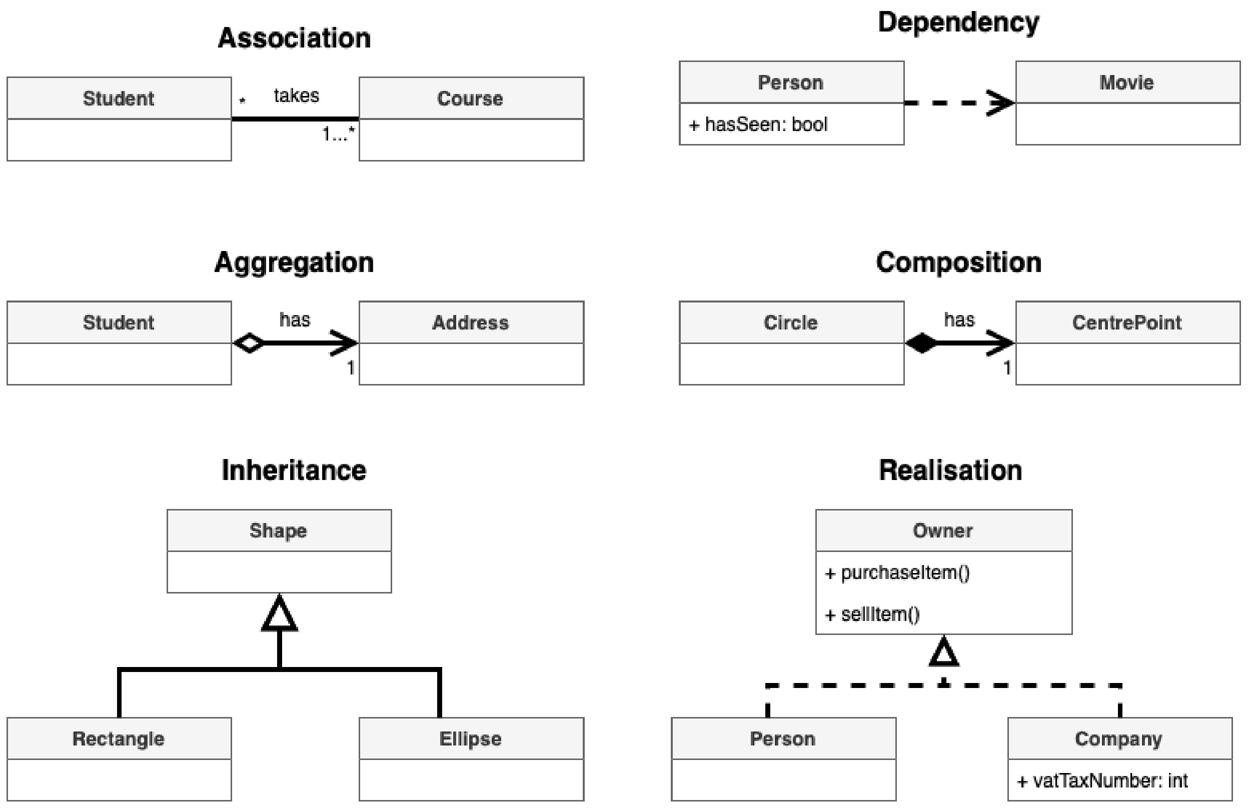


Figure 4: UML Connectors

5 Procedure

5.1 Use Case Diagram

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

5.2 Class Diagram

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

5.3 Diagram

6 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
External Programs for Diagrams : Draw.io

7 Conclusion

Thus, we learnt about Use case diagrams, and UML Class diagrams in detail.

8 FAQ

1. **What kind of relationships do classes have? Explain all relationships with examples.**

   (a) **Association**: An association is a relationship between two classes, where one class uses an instance of the other class as a property or method parameter. Associations can be one-way or two-way. For example, in a library system, the "Book" class could be associated with the "Author" class because each book has one or more authors.

   (b) **Aggregation**: Aggregation is a type of association where one class is part of another class. The part class is typically a smaller part of the whole, and it can belong to more than one whole class. For example, a "Car" class can have a "Wheel" class, and each wheel can belong to many cars.

   (c) **Composition**: Composition is similar to aggregation, but the part class cannot exist outside the whole class. The whole class is responsible for creating and destroying its part classes. For example, a "House" class can have a "Room" class, and each room can only belong to one house.

   (d) **Inheritance**: Inheritance is a relationship where one class is a subclass of another class, inheriting its properties and methods. The subclass can add additional properties and methods or override the superclass's methods. For example, a "Student" class can inherit from a "Person" class, inheriting the properties of "name" and "age."

   (e) **Dependency**: Dependency is a relationship where one class depends on another class. If the dependent class changes, the class that depends on it may need to be modified. For example, a "Payment" class may depend on a "CreditCard" class, but if the credit card system changes, the payment class may need to be updated.

2. **Explain any 2 terminologies used in Use case diagrams.**

   (a) **Actors**: An actor is an external entity that interacts with the system. Actors are usually represented by stick figures in use case diagrams. Actors can be human users, other software systems, or even hardware devices. Actors are used to represent the roles that different users play in the system. For example, in a banking system, the "Customer" could be an actor who interacts with the system to perform tasks like depositing money, withdrawing money, and checking account balances.

   (b) **Use cases**: A use case is a set of actions or steps that a system performs to achieve a specific goal. Use cases describe the functionality of the system from a user's perspective. Each use case describes a specific task or process that the system can perform. Use cases are often represented as ovals in use case diagrams. For example, in a banking system, the "Withdraw Money" use case could describe the process that a customer follows to withdraw money from their account. The "Deposit Money" use case could describe the process that a customer follows to deposit money into their account.

3. **Explain the aggregation and composition in diagram?**

   (a) **Aggregation**: Aggregation is a "has-a" relationship between classes, where one class is made up of one or more instances of another class. In other words, a class contains a reference to another class, which can exist independently. In an aggregation relationship, the part class can belong to multiple whole classes at the same time.

(b) **Composition**: Composition is a "part-of" relationship between classes, where one class is a part of another class. In other words, the part class cannot exist without the whole class, and a whole class can have only one instance of the part class. The part class's lifetime is controlled by the whole class, and if the whole class is destroyed, the part class is also destroyed.