Kishan Polekar
Submitted to : Sunae Shin
SE 3910
September 12, 2018

Unified Modeling Language (UML)

What is UML?

UML stands for Unified Modeling Language. It is used to get an idea of the workflow that we will be using when creating a software or a piece of program. In simple words, it is a blueprint of the project that can be referenced to keep on track, check what functionality is needed before starting on the project, and also to make changes beforehand. It is a diagrammatic representation of the project. It can be considered a pseudo code as it lists all the important functions that the program will perform, and also serves as a go-to document.

Types of UML

There are many types of UML diagrams that vary based on one's needs. Broadly, UML diagrams are divided into two parts:

- **Behavioral UML Diagram**
- **Structural UML Diagram**

As the names suggest, Behavioral diagrams analyze the behavior of a system/program. What will the program do? What components will be completed in a certain amount of time? What is the use of each component? All these are answered through different behavioral diagrams. The main sub-types of behavioral diagrams are:

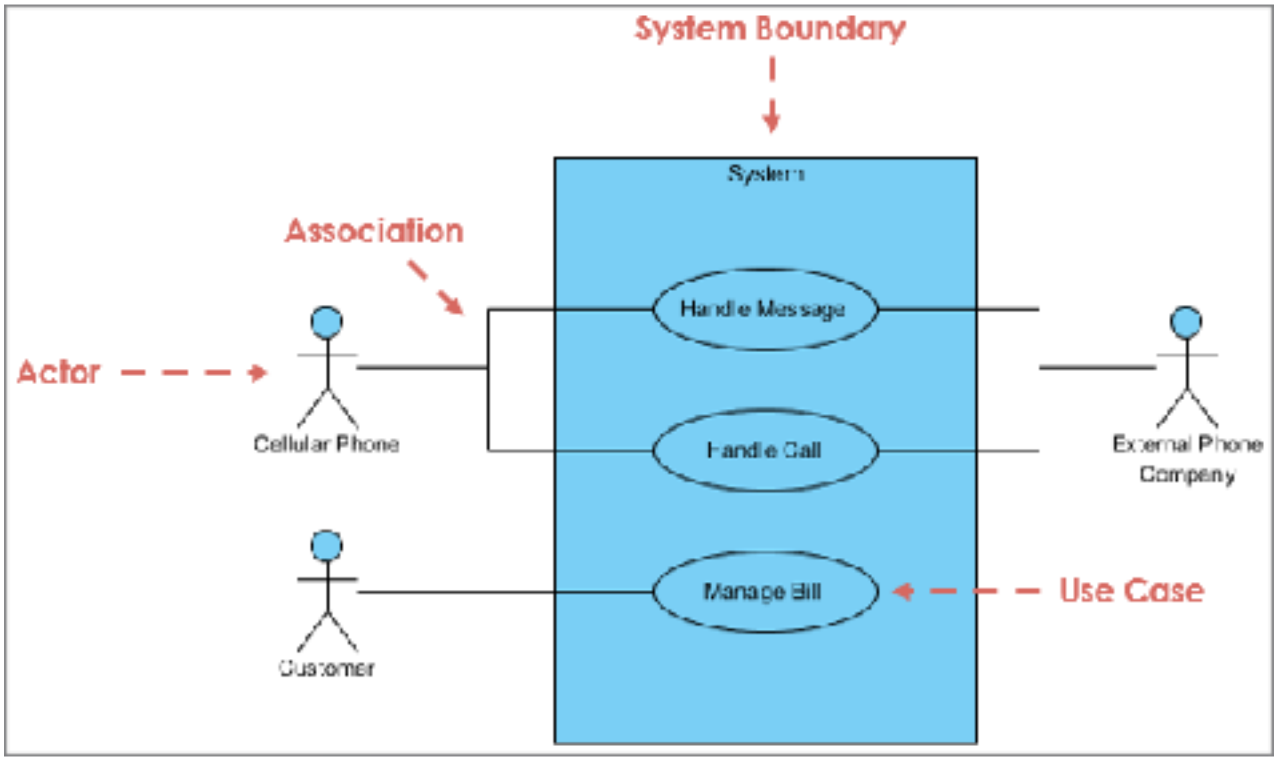
- Use Case Diagram
- Activity Diagram
- State Machine Diagram
- Sequence Diagram
- Communication Diagram
- Interaction Overview Diagram
- Timing Diagram

Structural diagrams deal with the layout of the program and its different components. These diagrams give us an overview of how the program will look, what features will be included, how will they interact with each other, and more. Some examples of structural UML's are:

- Class Diagram
- Object Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Package Diagram
- Profile Diagram

Use Case Diagram

It is one of the most important and widely used UML diagram in the industry. It describes the relation between different components of a system, the people involved, and their relations. These diagrams give us a complete picture of what components are included in the project, their actors or the people/internal/external factors that use these components, and the associativity of the two.

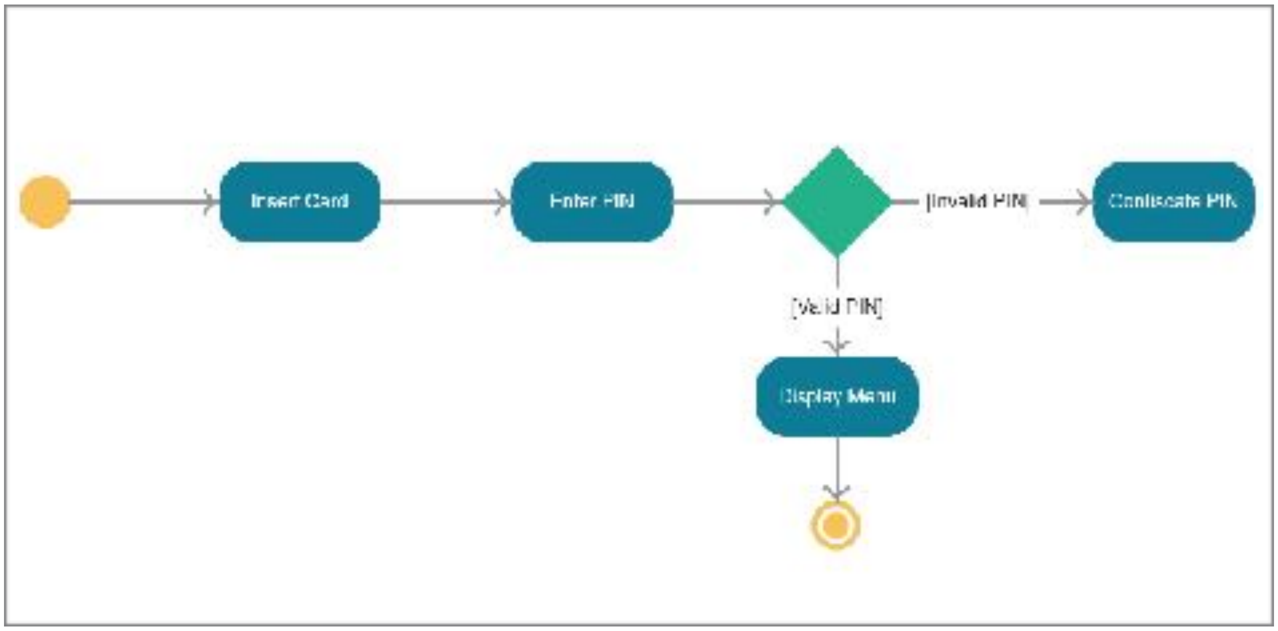


Use Case Diagram

In the above example, the box denotes the system boundary. All components included in the project are enclosed within this box. The functions are called use cases. They are acted upon by the "actors". These actors can be people, internal or external applications, or even another system itself. The relations and actions are shown using straight arrows.

Activity Diagram

Activity Diagrams represent a system in a graphical way. It uses many shapes to denote specific actions and components. These are pretty easy to understand since they use simple representations of each entity.

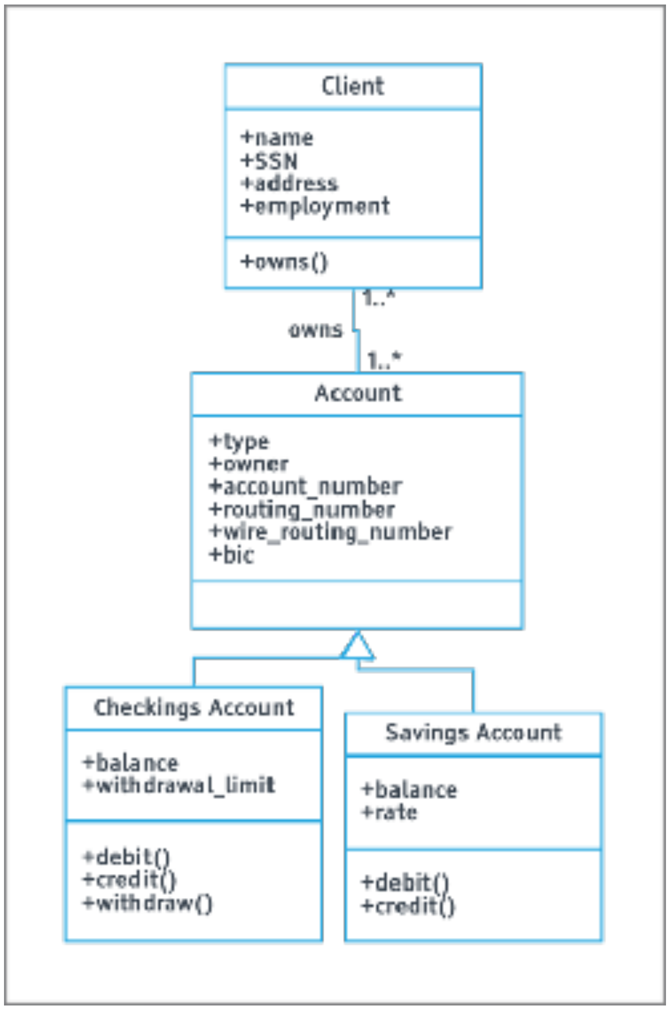


Activity Diagram

In the above example, the start of a process (inserting a card into the ATM machine) is denoted by a filled circle. Any action is enclosed in a rounded box. The diamond is a decision node which will have at least two diverging nodes. It is used to make a decision and follow a particular path if a certain decision has been made. The end of the process is denoted by a filled circle with another circle enclosing it. The entire diagram uses arrows that are showing the flow of events.

Class Diagram

Class diagrams are the most widely used diagrams in terms of software development. Since most modern programs are based on Object Oriented Programming, class diagrams are very useful in outlining what classes will be included in the project, the attributes of each class, the functions that each class will perform, and how all classes are related to each other.

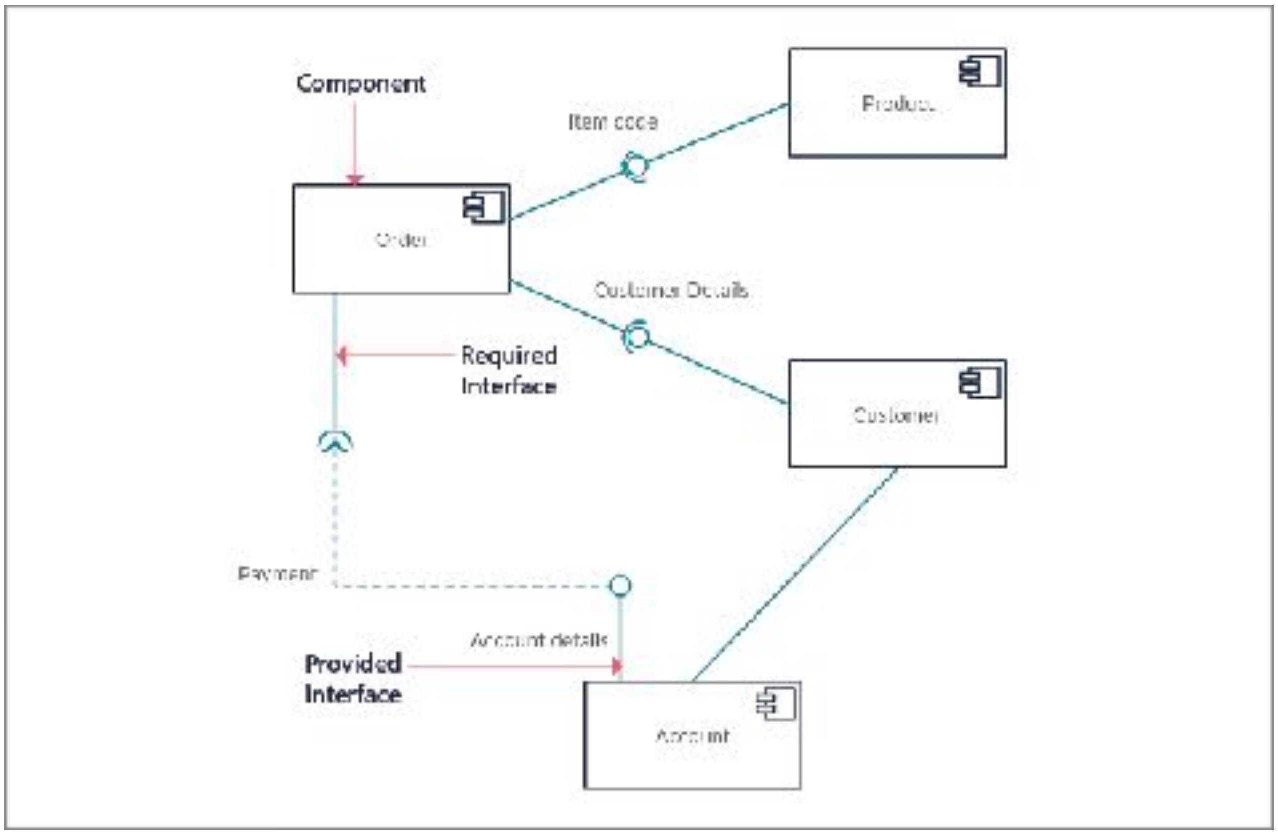


Class Diagram

Class diagrams are pretty self explanatory. In the above example, the class Account is the parent of two classes: Checking Account and Savings Account. This relation is denoted by an arrow pointed towards the parent class. Both the child classes will inherit all properties of the parent but will have some components of their own too. There are three components in any class diagram: class name, class attributes, and class functions. The plus and minus signs before each attribute and function depicts their score, viz., public and private.

Component Diagram

These diagrams denote the structural relationships between the components of a system. Each component in a system is linked to some other component using special symbols which can be understood and inferred for a better view of the project. These are mainly used when dealing with big or complex systems since component diagrams give a good idea about the component relations.

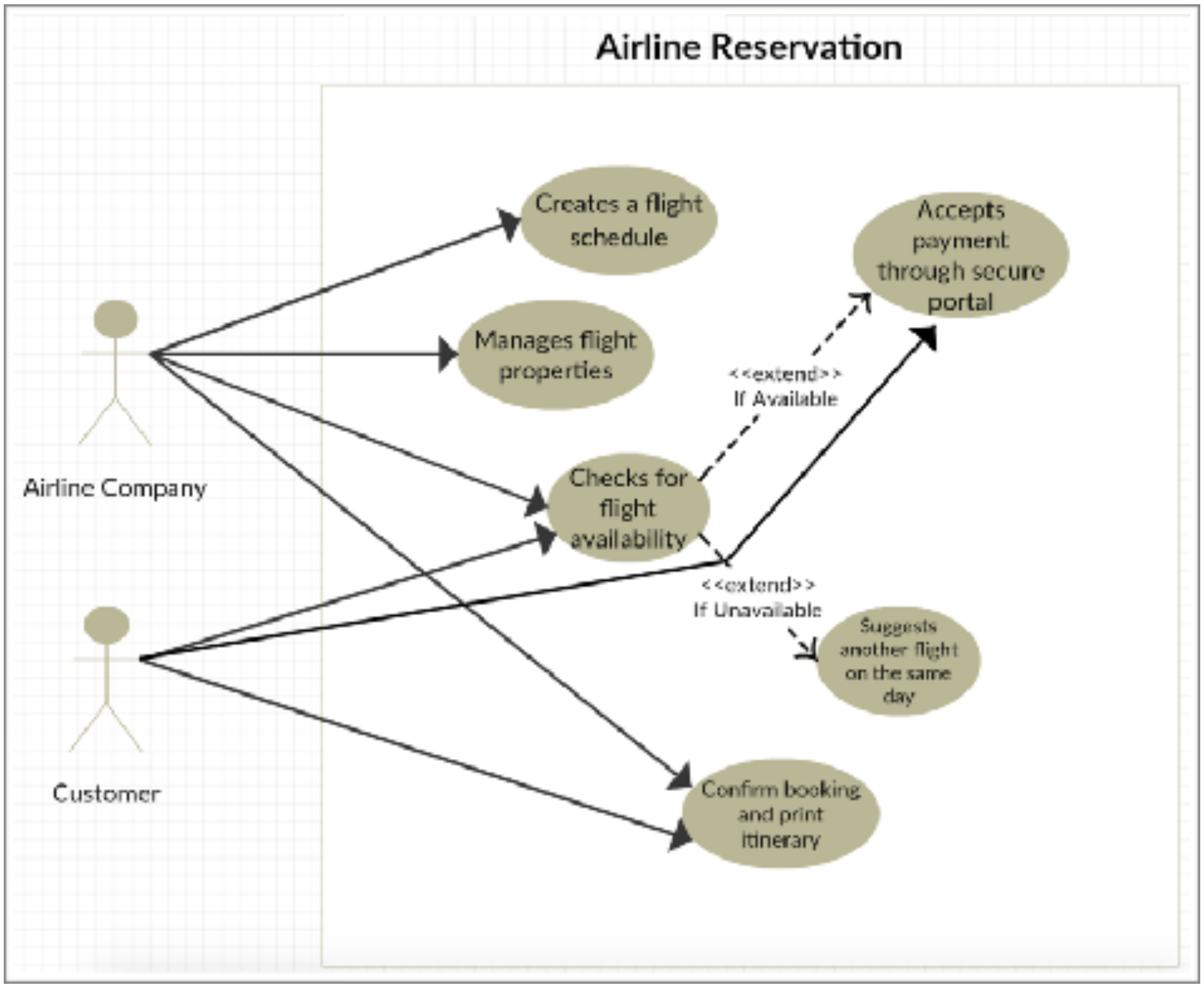


Component Diagram

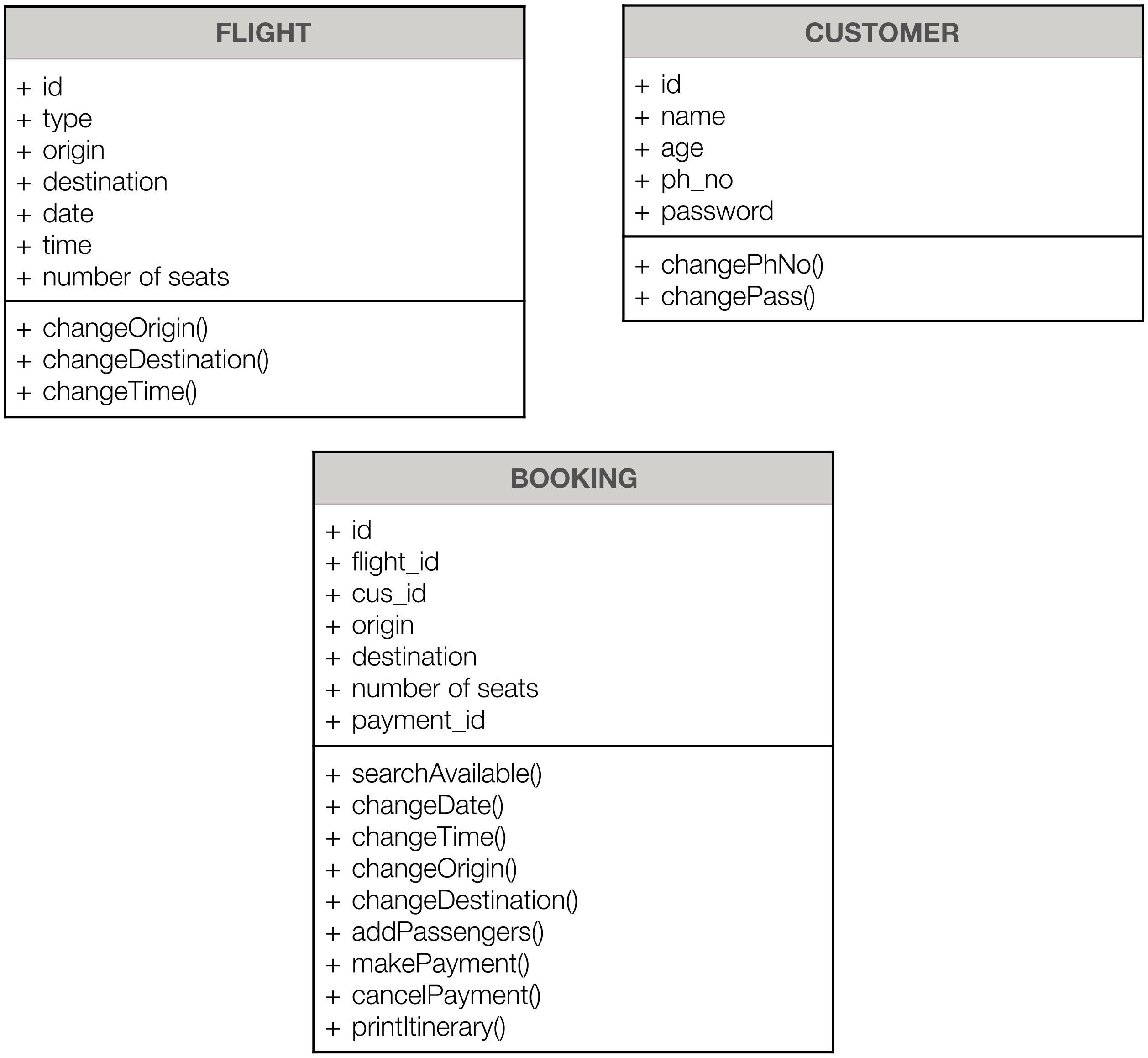
In the above example, a simple relationship structure is explained. The customer requires the order interface to order a product. The order class requires a product class to interact and order a particular item. However, the customer need not be registered (can be a guest user). Hence, the account component is not denoted using a solid line, rather a dotted line attached to payment. This means that a customer can pay without providing his/her account details, but if account details are given, the solid line shows the interface is needed.

Airline Reservation System

This is a very basic airline reservation system with limited components. It uses two diagrams from the ones listed above: Use Case Diagram and Class Diagram. It will serve as a starting point for future editing and references.


Airline Reservation
Airline Company
Customer
Creates a flight schedule
Manages flight properties
Checks for flight availability
Accepts payment through secure portal
<<extend>> If Available
<<extend>> If Unavailable
Suggests another flight on the same day
Confirm booking and print itinerary

Airline Reservation Use Case Diagram



Airline Reservation Class Diagrams

The above diagrams are a basic outline of the airline reservation system. In the Use case diagram, the airline manages bookings, flights, and manages the properties of different flights. They have access to check the availability of flights. The customer has access to flight details like date, time, origin, departure, etc. Customer can also make bookings, make payments, and both have access to the common printing of itinerary so that there is a record on both ends.

The Class diagrams have a detailed description of what functionalities are included for each user type. There is an additional class called booking which is accessible to both airline and customers. For security reasons, some of the attributes and functions have been labeled as private. This reservation system will allow for basic details and can be used to add attributes later. The Use Case diagram shows relations between entities while the Class diagram shows the components of each entity.

References

https://tallyfy.com/uml-diagram/

https://creately.com/blog/diagrams/uml-diagram-types-examples/

https://creately.com/app/?tempID=gc7qvpsj1&login_type=demo