What is SRS?

It is the complete description of the system that is to be developed. It is the official statement of the system that the developers is yet to developed.

Purpose of the SRS?

Describes what the software will do and how it is expected to perform.
Helps is finding out all the requirements and provides feedback to the customer.

Characteristics of SRS

Correct: Every requirement given in SRS is a requirement of the software.
Unambiguous: Every requirement has exactly one interpretation.
Complete: Includes all functional, performance, design, external interface requirements; definition of the response of the software to all inputs.
Consistent: Internal consistency.
Ranked importance: Essential vs. desirable.

Table of Contents

2.2.8 Logout

2.3 User Classes and Characteristics

The main actors in the system are (1) the user, (2) a flight and (3) a Flight Seat. The user will select a flight and book seats on the flight. They will then reserve specific seats on that flight. Brief descriptions of these classes follow: • User o Has properties like Name, Address, Age o Associated with Flight Miles accumulated and Credit Card information. • Flight o Has properties like Departing/Arriving City, Departure/Arrival dates and times, Miles, and an identifying Flight Number. • Flight Seat o Has properties of identifying seat number, reserved and flight number o Associated to Flight by flight number

2.4 Operating Encironment

2.5 Design and Implementing Constriants

2.5 User Documentation

User manual, Online help, tutorial

2.6 Assumptions and dependencies

Agile method used.

Depend upon the geographical and wether factors.

3 External Interface Requirements

4.1 User Interfaces

4.2 Communications Interfaces

API FTP Web services Middleware

Hardware

Software

4.. System Features

3.1 General Requirements

3.1.1 Login 6

3.1.2 Enrollment 7

3.1.3 Book Flights 9

3.1.4 Reserve Seats

3.1.5 Flight Status 12

3.1.6 Flight Schedule

3.1.7 My Account 13

3.1.8 Account Log out

USECASE DIAGRAM

5. Other Nonfunctional Requirements:

5.1 Performance Requirements

• The Airline Website shall have capabilities to accept 500 connections. For each session, system shall guarantee the connection time 5 minutes from last input, after which the connection will be deemed expired. A close operation will be performed when expired. This design is to satisfy each user's usability and connection quality. • The system shall send out verification request immediately (within 100ms) after the it receives a user submitted form. • The system shall update all flight status information every 5 minutes.

5.2 Security Requirements

• Passwords must be a minimum of eight characters and must contain one to seven digits. • Email addresses should be verified before the system grants user access. This verification shall be exercised by sending the prospective user a confirmation email after enrollment. This email must contain information specific to completing the enrollment process. • All exchanges from client to server involving private data shall occur using the highest available level of secure connection (e.g., https).

What are Data Flow Diagrams?

Pictorial representation of data that flows from one process to another process. Simply activities which transform data within a system.

Why they are useful? #

● A DFD shows an abstract or functional view of the system to be developed.
● The graphical representation easily overcomes any gap between 'user and system analyst' and 'analyst and system designer' in understanding a system.

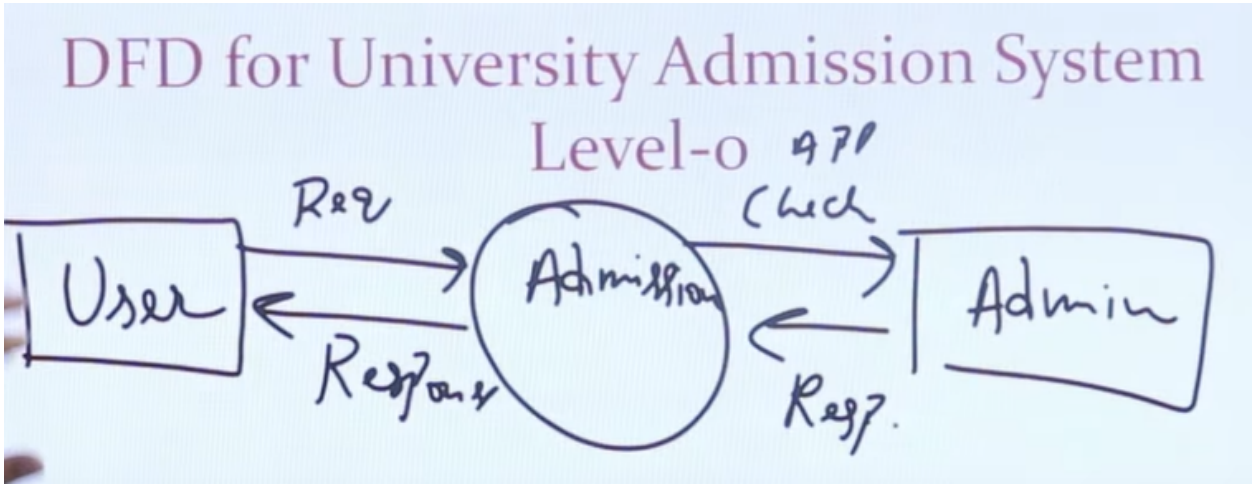
Data Flow Diagram Entites:

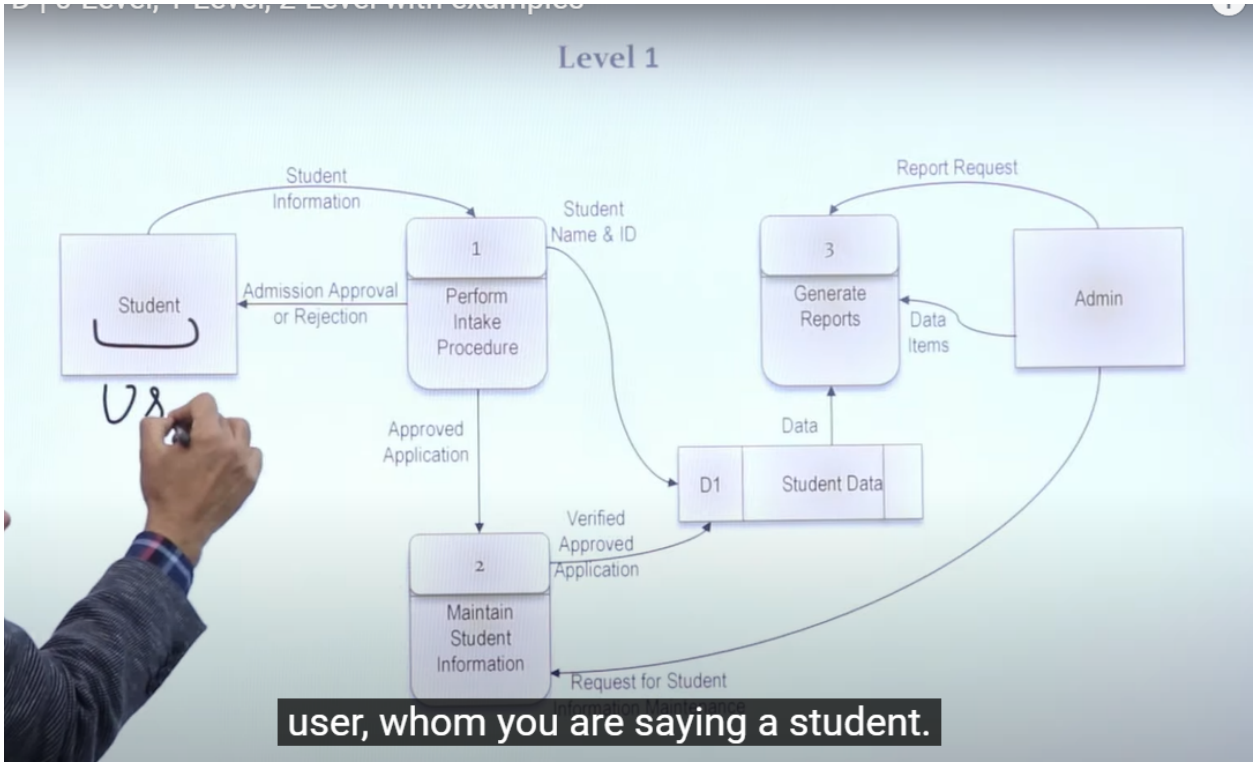
1. Source/ Sink : Represents the User.
2. Data Flow : Represented by the arrows.(Direction of flow)
3. Process: What is the work. Represented by the circle
4. Data store: Database used (Represented by the Parrel Lines.)

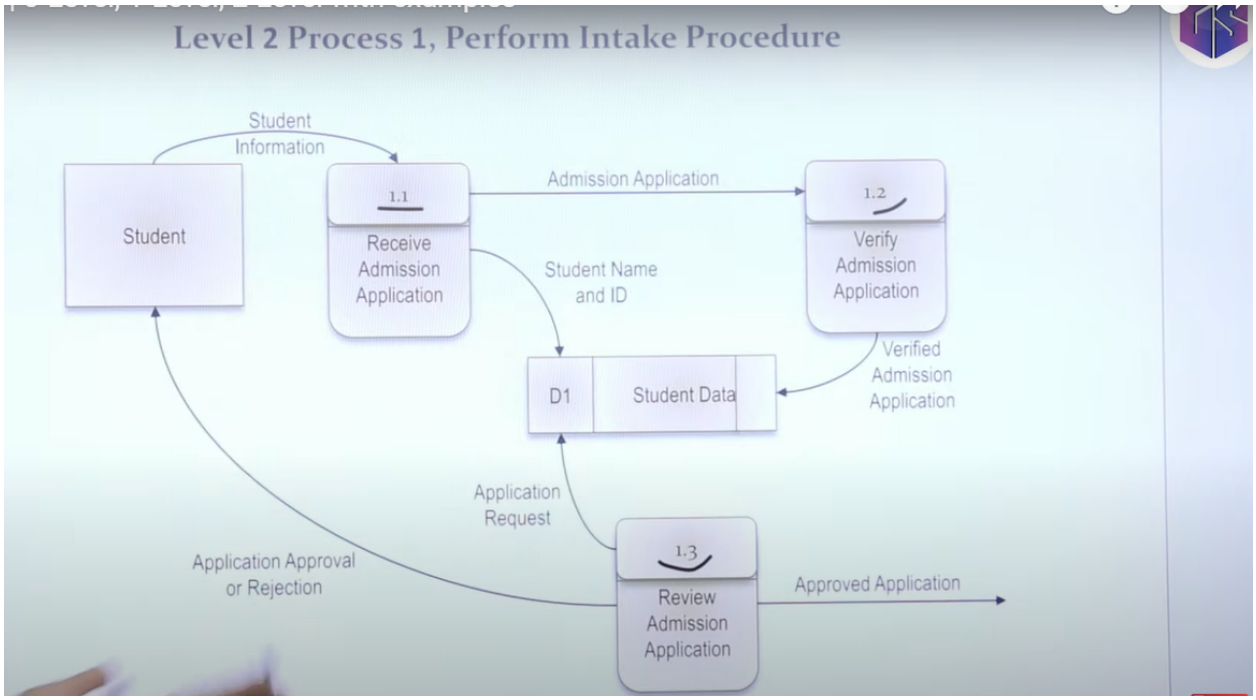
Rules associated:

Data can flow from
External entity to process
• Process to external entity
• Process to store and back
• Process to process

Data cannot flow from
External entity to external entity
External entity to store
• Store to external entity
• Store to store







Viva Questions:

1. What is a context diagram?
A context diagram is a high-level representation of a system that shows the system as a single process and its interaction with external entities. It helps to define the boundaries of the system and its scope.

2. Can the data flow directly between two data stores?

No, data flow cannot directly go between two data stores in a data flow diagram (DFD) because data stores represent data at rest, and they cannot directly communicate with each other. Data must flow through a process before it can be stored in a data store or retrieved from it.

3. Why data dictionaries are maintained?

Data dictionaries are maintained to provide a central repository for the definitions of data elements used in an organization's information systems. It helps to ensure consistency in the naming and use of data elements across the organization, which can help to improve data quality and facilitate system integration.

4. What are the rules to create DFDs?

There are several rules to create DFDs:
Each process should have at least one input data flow and one output data flow.
Data should flow in a single direction from sources to sinks.
Data should not flow back to the source process.
Each data flow should have a meaningful name that describes the data being transmitted.
Processes should be labeled with a verb-noun phrase that describes the transformation being performed.
Data stores should be labeled with a noun that describes the type of data being stored.

USE CASE:

Use Case diagrams are essential tools used by developers and analysts to describe the functionality of a system and how actors interact with it. A Use Case diagram is made up of four primary elements: Systems, Actors, Use Cases, and Relationships.

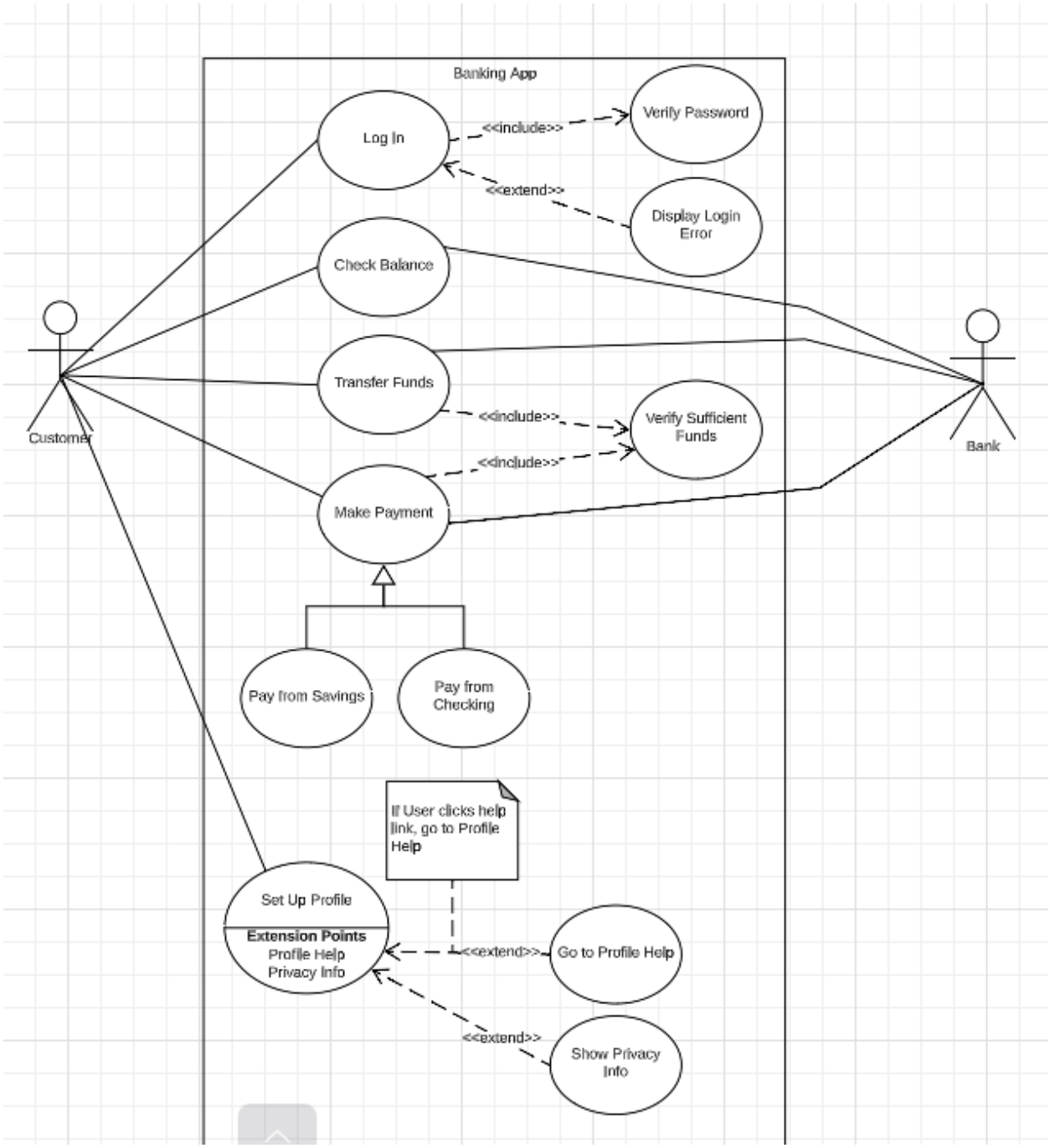
Systems:
A system is the thing that you're developing, which could be a website, software component, business process, or an app. In a Use Case diagram, a system is represented by a rectangle. You need to put the name of the system at the top, which defines the scope of the system.

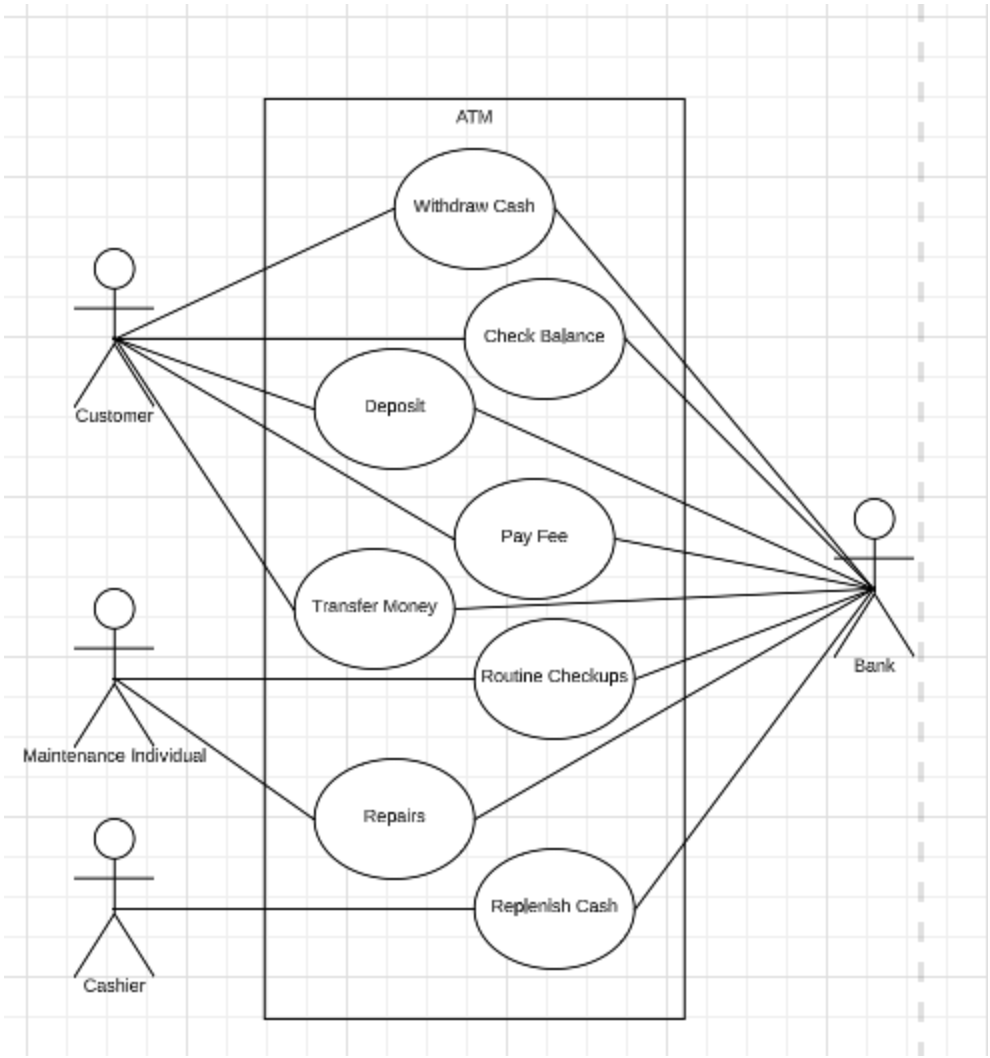
Everything within the rectangle happens within the system, and anything outside the rectangle doesn’t happen in the system.

Actors:
An actor is someone or something that uses the system to achieve a goal. It could be a person, an organization, another system, or an external device. In a Use Case diagram, actors are represented by stick figures. You need to place actors outside the system as external objects, and they should be thought of as types or categories. An actor initiates the use of the system, and they're either primary or secondary actors. Primary actors initiate the use of the system while secondary actors are more reactionary.

Use Cases:
A Use Case represents an action that accomplishes some sort of task within the system. Use Cases are depicted with an oval shape, and they're placed within the rectangle of the system because they're actions that occur within the system. Each Use Case starts with a verb, reinforces an action that takes place, and is sufficiently descriptive. It's good practice to put your Use Cases in a logical order when possible.

Relationships:
An actor, by definition, uses the system to achieve a goal. So each actor has to interact with at least one of the Use Cases within the system. In a Use Case diagram, the relationship between actors and Use Cases is shown by lines. The four types of relationships are an association, include, extend, and generalization. An association signifies a basic communication or interaction. An include relationship signifies that one Use Case includes another Use Case. An extended relationship signifies that one Use Case extends another Use Case. A generalization relationship signifies that one Use Case is a specialization of another Use Case.


Banking App
Log In
<<include>>
Verify Password
<<extend>>
Display Login Error
Check Balance
Transfer Funds
<<include>>
Verify Sufficient Funds
<<include>>
Make Payment
Customer
Bank
Pay from Savings
Pay from Checking
If User clicks help link, go to Profile Help
Set Up Profile
Extension Points
Profile Help
Privacy Info
<<extend>>
Go to Profile Help
<<extend>>
Show Privacy Info



Viva Questions:
1. What is use-case diagrams used for?

Use-case diagrams are used to illustrate the system's functionality by showing the relationships between actors and use cases. They provide a visual representation of the system's behavior by demonstrating how users interact with the system to achieve their goals.

2. What is extends and include? Differentiate between the two.

"Include" and "extend" are two relationships used in use-case diagrams. "Include" is used to indicate that one use case is a part of another use case. "Extend" is used to show optional functionality that can be added to a use case. The main difference between the two is that "include" is a mandatory relationship, while "extend" is optional.

3. What is a generalization relationship in a use-case?

In a use-case diagram, the generalization relationship is used to show that a child use case inherits the behavior and characteristics of a parent use case. This relationship can be used to simplify a complex use case by breaking it down into smaller, more manageable pieces.

4. “Use case diagrams belong to the category of behavioral diagram”. What does the statement mean?

The statement means that use-case diagrams are used to describe the behavior of a system. They focus on the system's functionality and how users interact with it to achieve their goals. Other types of behavioral diagrams in UML include sequence diagrams, activity diagrams, and state diagrams.

UML Sequence Diagram:

Particularly these diagrams show interactions in order. They take place. Communication is between the Objects and actors and also within the objects.

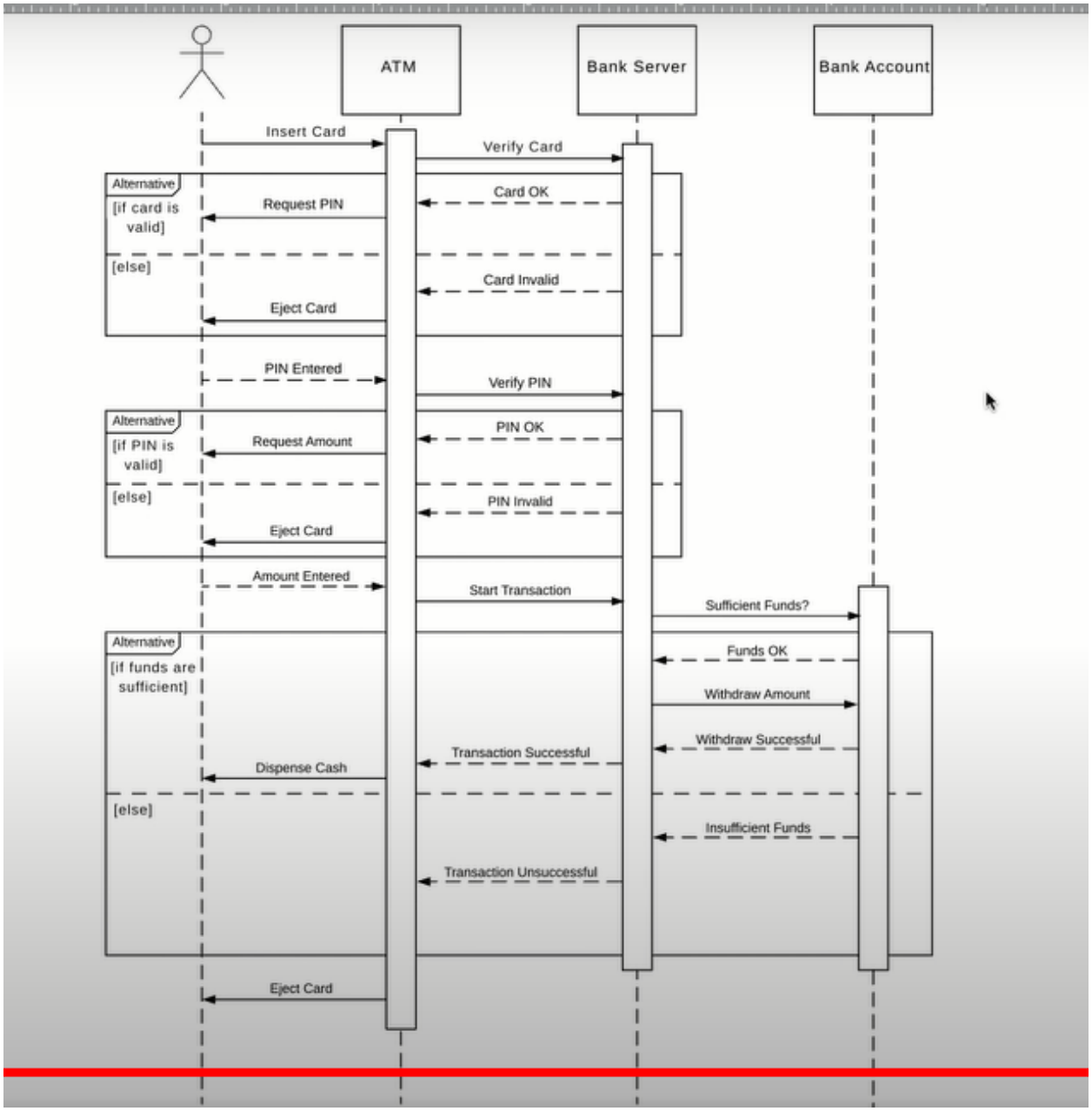
Lifelines are the vertical dashed lines that show the existence of an object or actor over time

Messages are of two types: Request and response. Request are shown bold arrows and response are shown using dashed arrow.

Label are written above message arrow.

An alternative frame symbolizes a choice between two or more message sequences.

Activation boxes show when and how long an object is performing a process



What is a sequence diagram?

1. **A sequence diagram is a UML behavioral diagram that shows how objects interact with each other in a particular sequence to complete a specific task or process. It is used to represent the flow of messages or interactions between objects or components in a system.**

What are advantages of sequence diagram?

2. **Some advantages of sequence diagrams are:**

- **They provide a clear and concise representation of object interactions in a system.**
- **They can be used to model both simple and complex systems.**
- **They help in understanding the behavior of a system at a specific point in time.**
- **They are easy to read and understand, making it easier for developers to communicate and collaborate with each other.**

What are entities in sequence diagram?

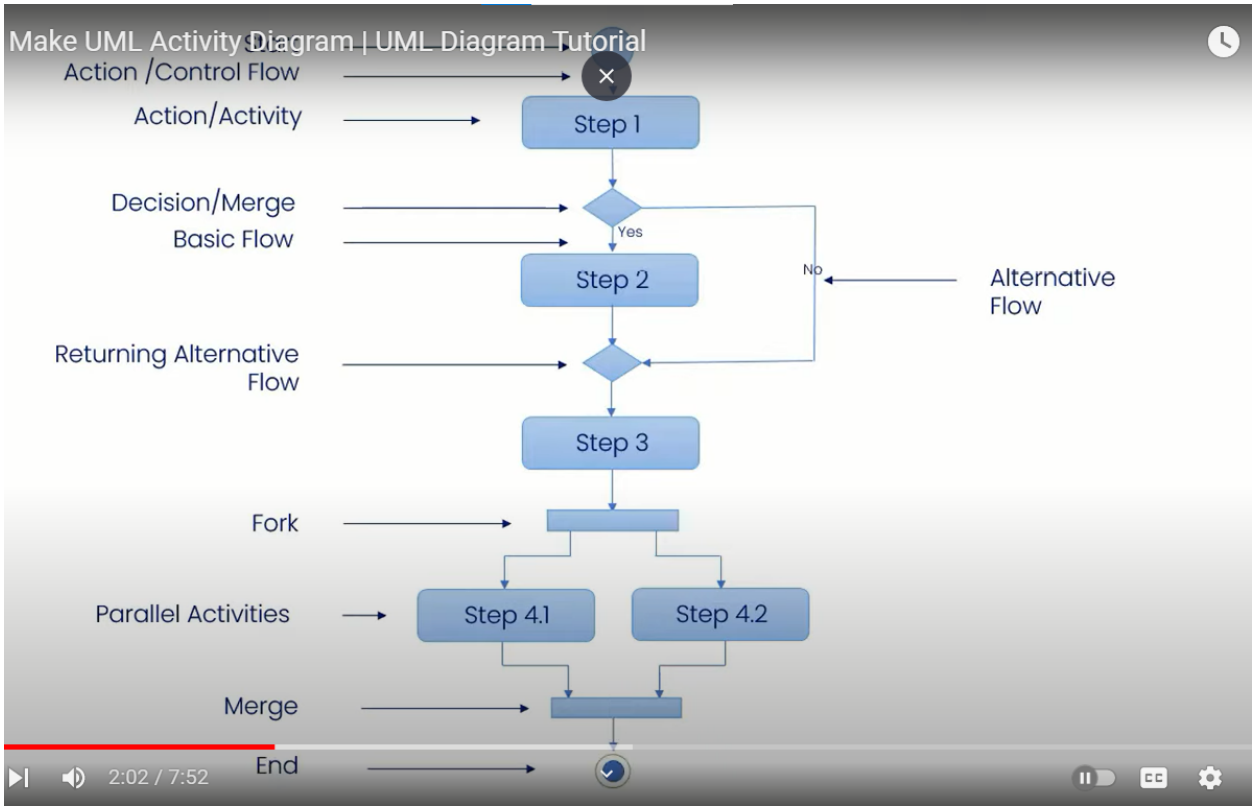
3. **In a sequence diagram, entities are represented as objects or components that participate in the interaction. Entities can be anything that has behavior, such as objects, components, classes, subsystems, or even actors.**

Explain its relation with the class diagram?

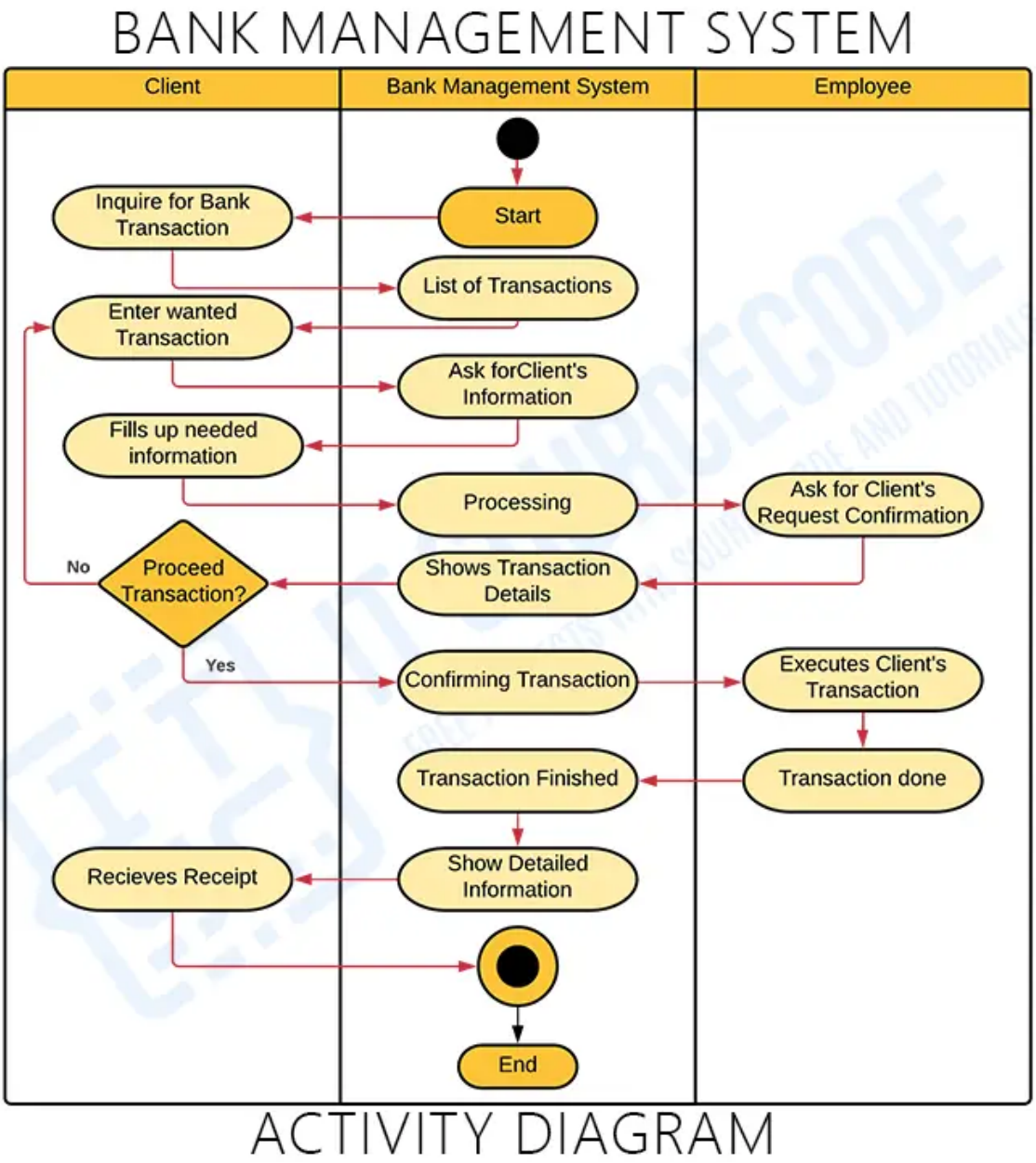
4. **Sequence diagrams and class diagrams are two different types of UML diagrams, but they are related to each other. A sequence diagram shows the interactions between objects in a system, while a class diagram shows the structure and relationships between classes and objects in a system. Both diagrams are used to model different aspects of a system, but they are complementary to each other. In a sequence diagram, the entities (objects, components, etc.) that are involved in the interaction are usually derived from the class diagram. Conversely, the class diagram can be used to create the objects or classes that are used in the sequence diagram.**

Activity Diagram:

activity diagrams in uml describe the dynamic aspects of the system it is basically a flowchart to represent the flow of one activity to another the activity



Segment of the activity that results into unusual stoppage or end of activity



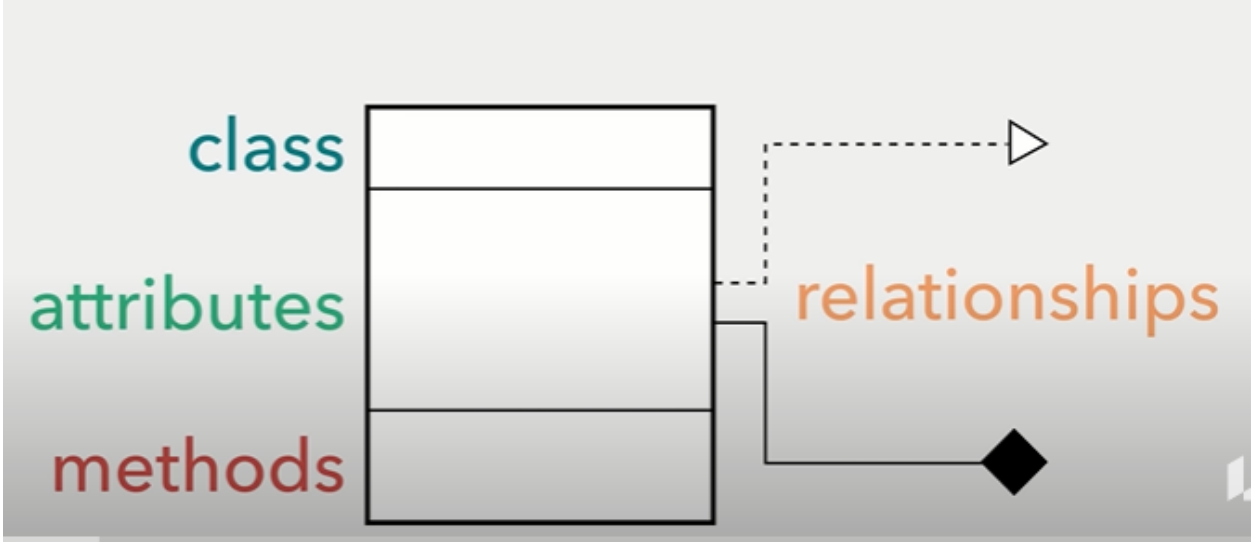
What is a state diagram used for?
A state diagram, also known as a state machine diagram, is a type of behavioral diagram in the Unified Modeling Language (UML) that represents the behavior of an object in response to external stimuli by describing the different states an object can be in and how it transitions from one state to another based on events and conditions.

Enumerate the steps to draw an activity diagram.

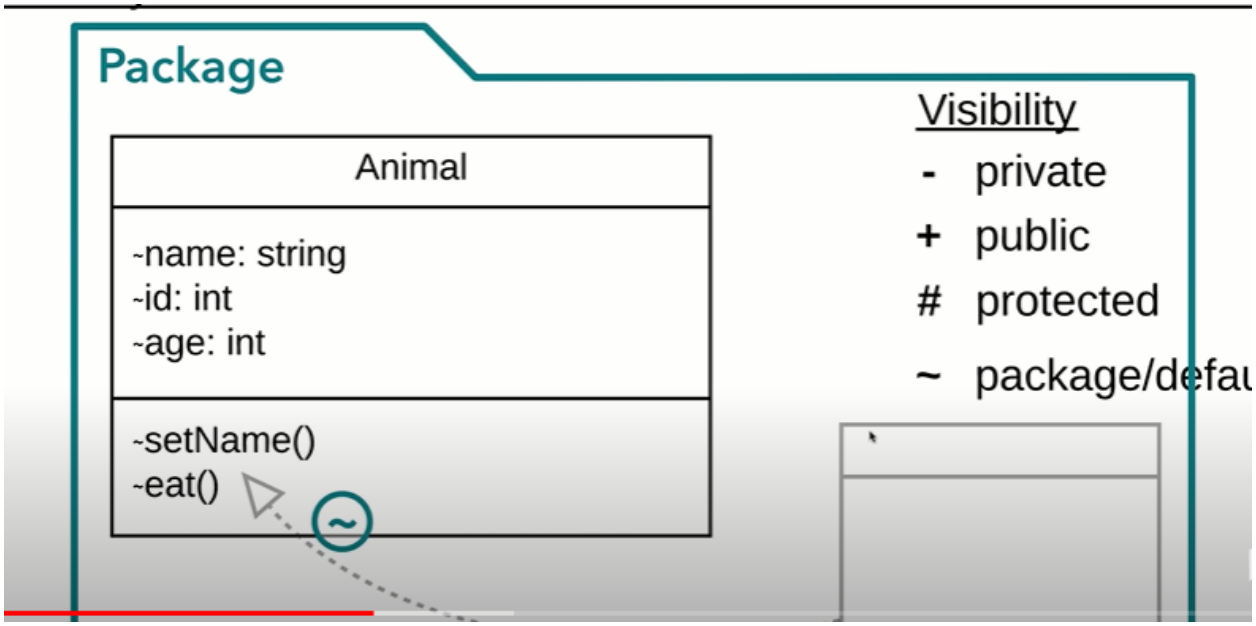
The steps to draw an activity diagram are as follows:

Identify the scope and objective of the activity.
Identify the actors involved in the activity.
Identify the initial and final states of the activity.
Identify the tasks or actions that need to be performed in the activity.
Identify the decision points or branches in the activity flow.
Identify the conditions or rules that apply to the activity flow.
Draw the activity diagram using UML notation, which includes activity nodes, activity edges, decisions, and merges.
Review and validate the activity diagram to ensure it accurately represents the activity and meets the requirements.

Class Diagram:

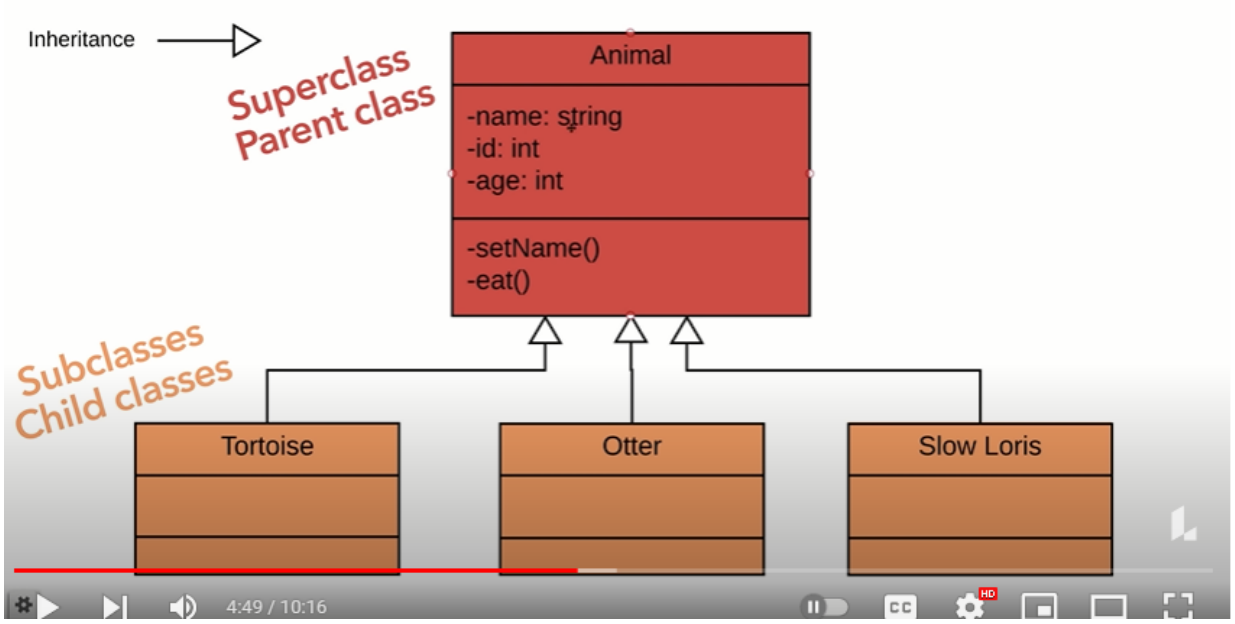


An attribute is a significant piece of data containing values that describe each instance.
Start with Lower casses.
Methods allow you to specify any behavioral features of a class.

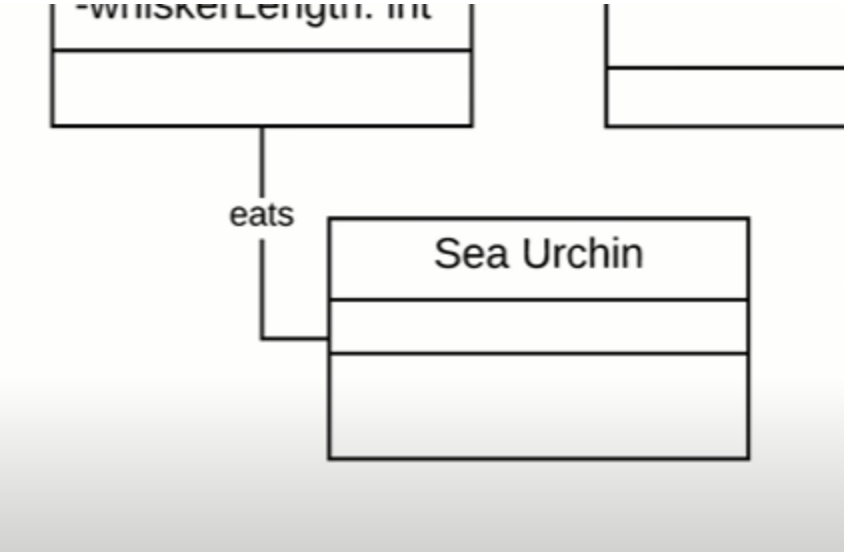


Relationships:

1. Inheritance:



2. Association:

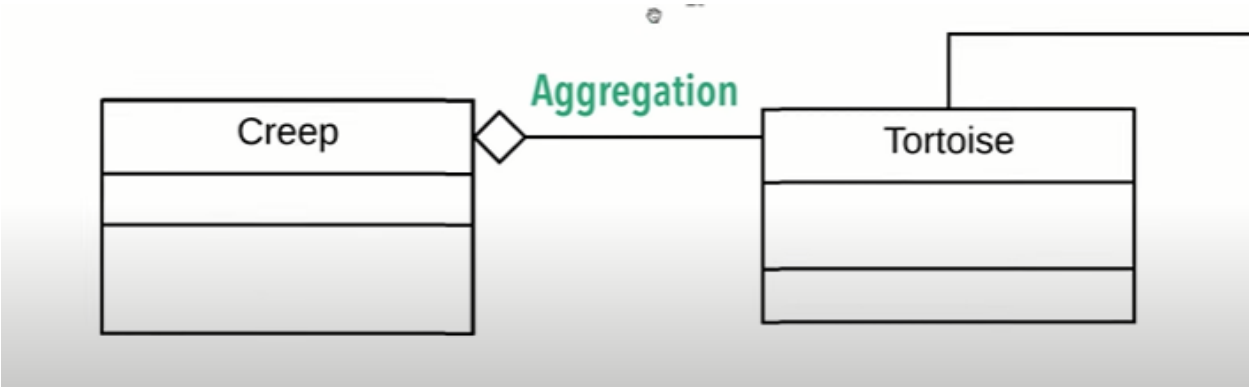


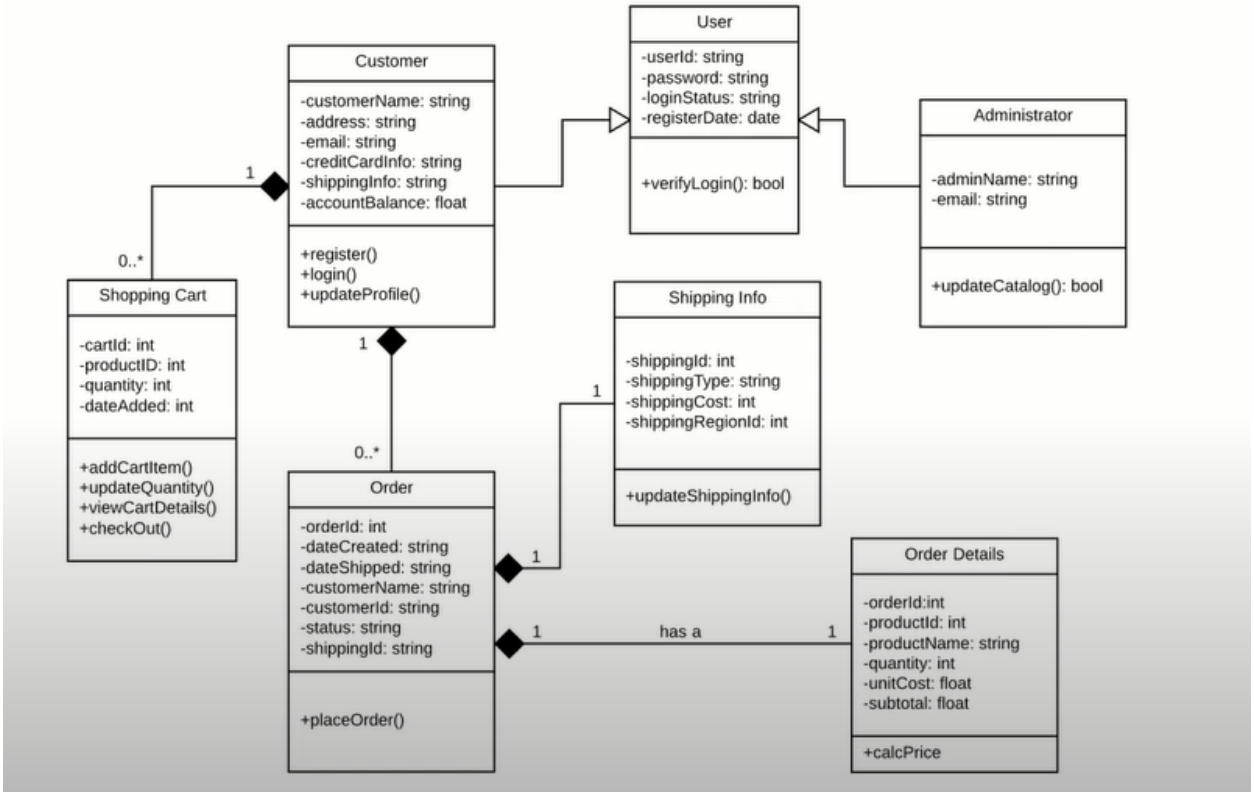
3. Aggrretion:

aggregation.

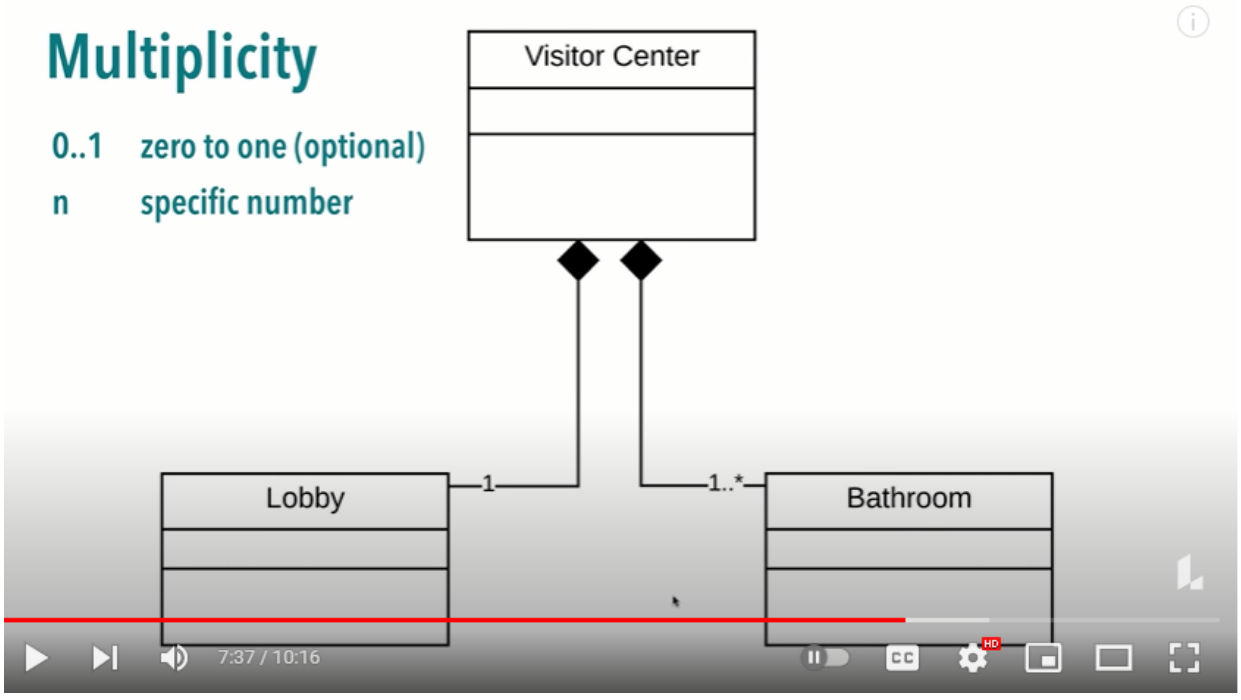
It’s a special type of association that specifies a whole and its parts.

Can exist outside an class.





4. Composition:
   Cannot exist without the base class.



What is class diagram used for?

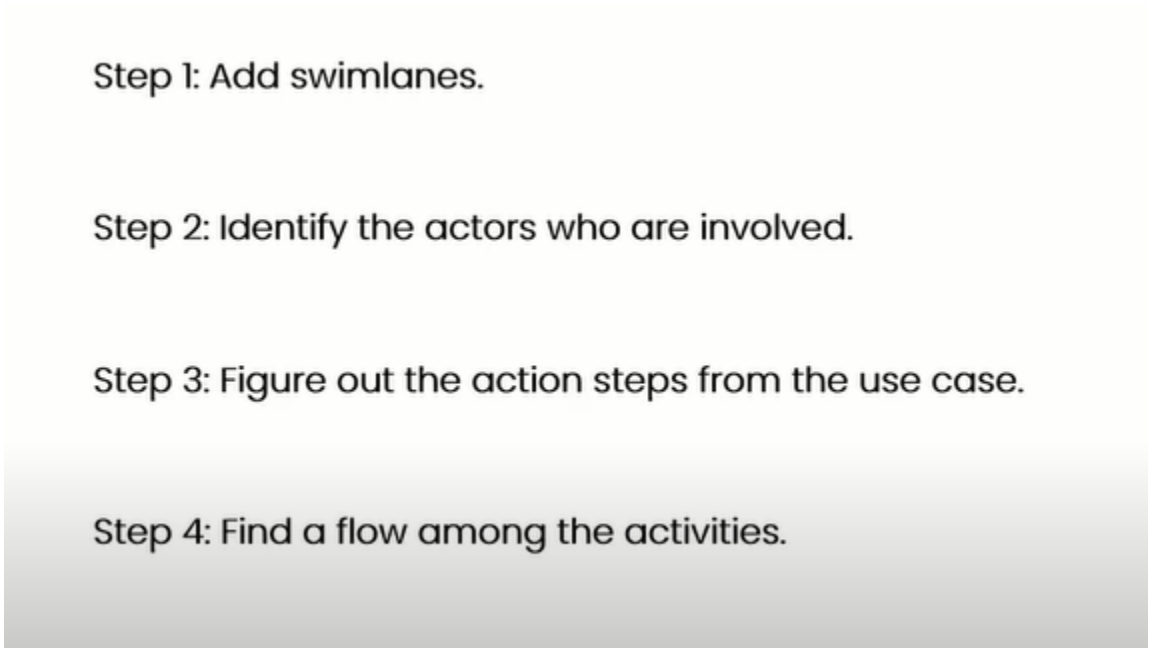
A class diagram is a type of static structure diagram in the Unified Modeling Language (UML) that shows the structure of a system by modeling its classes, attributes, operations (or methods), and the relationships among objects.

Enumerate on the type of relationships between classes.

The types of relationships between classes are:

Association
Aggregation
Composition
Generalization
Realization

What do you understand by multiplicity factor in class diagrams?
Multiplicity factor in class diagrams defines the number of instances of a class that can participate in a relationship with another class. It is represented by a range of values, such as 0..1, 1.., or 0... The first value indicates the minimum number of instances that can participate in the relationship, and the second value indicates the maximum number of instances that can participate. For example, 0..1 means that zero or one instance can participate in the relationship, while 1..* means that one or more instances can participate.



What is Selenium IDE?

Selenium IDE is an open source test automation tool that can record and playback your actions on the web. By using it, you can automate tests for web applications.

What is the KANBAN tool?

The Kanban system is a way to manage your work by visualising the workflow and limiting the amount of work-in-progress.

6 rules for an effective Kanban system

1. Prevent sending defective products to the next process

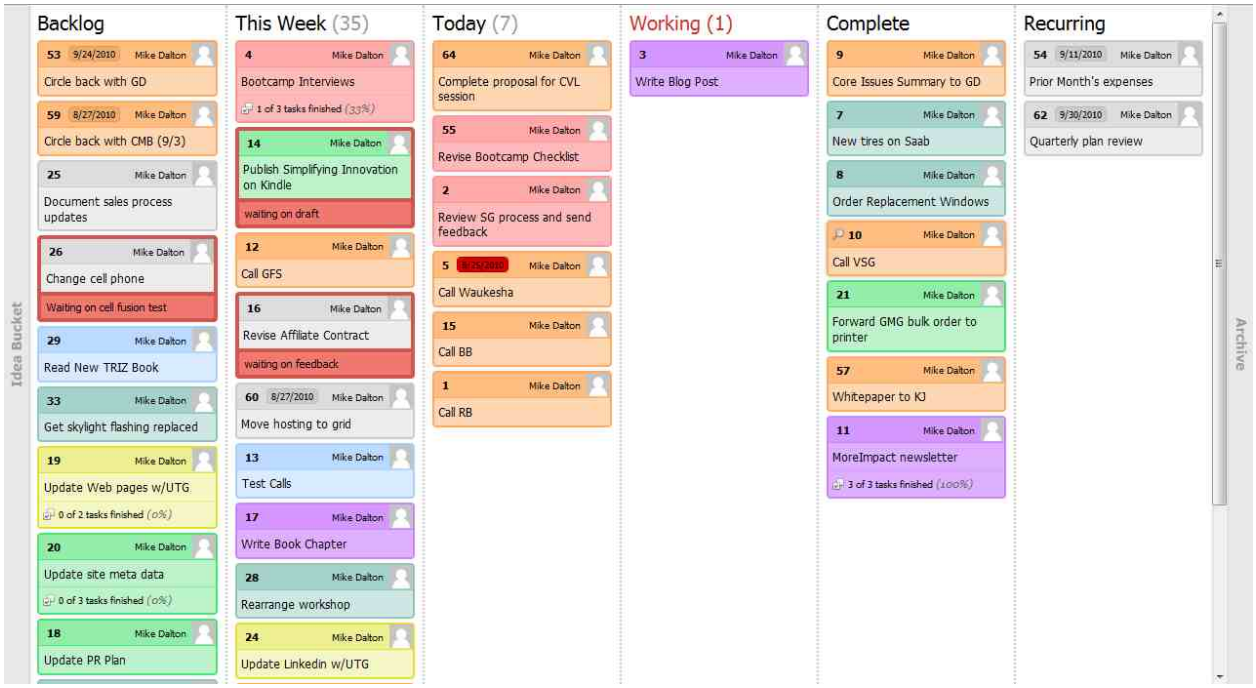
2. Take only what you need

3. Produce in exact quantities

4. Level the production

5. Optimise the production or process

6. Stabilise the process



Backlog(Number / Count) | This Week (Count) | Today Working (Count) | Complete(Count) | Recurring (Count)