1. Introduction to the Unified Modeling Language.

1) What is UML?
2) UML basic notations.
   a) Things.
      i) Structural things.
      ii) Behavioral things.
3) Benefits of using UML.

---

1) What is UML?

UML is a standard way of visualizing a system's design.

It is not a programming language but is used to visualize a system's behaviour and structure.

It is known for providing tools that allow software engineers and developers to analyze, design, develop, software systems and model processes.

UML is the perfect language to explain the inner workings of the software system to all the stakeholders involved.

2) UML basic notations.

3 main building blocks:

① Things
② Relationships.
③ Diagrams.

a) Things :-

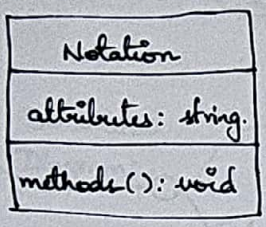
* Structural things
* Behavioral things
* Grouping things
* Annotation things

} functionalities are pretty much same as their names.

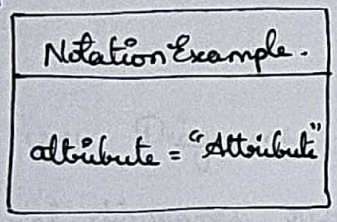
* **Structural Things :-**

represent the system's physical aspects such as a class, object, interface, use case, actor, component, and node.

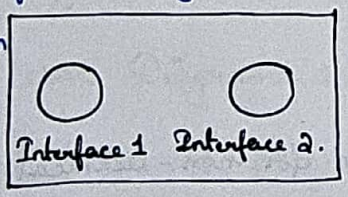
1. Class : The notation represents the attributes and methods of object.



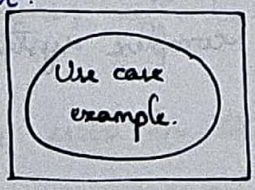
2. Object : This notation refers to instance of the class.



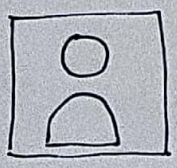
3. Interface : This notation represents functionality without its implementation



4. Use case : This notation describes the users' goals and possible interactions with the system.



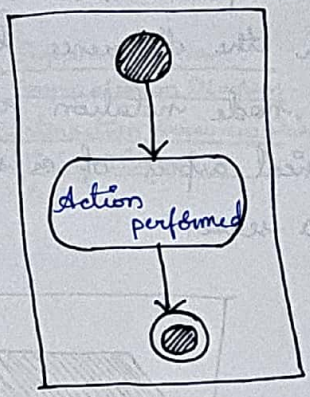
5. Actor This notation represents the entities interacting with system.



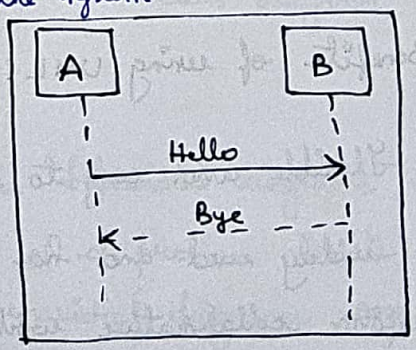
* **Behavioral Things :-**

represent the various system's interactions and functions such as state machines, activity, interaction diagrams.

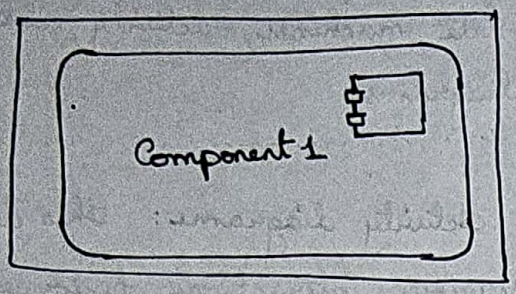
1. Activity diagrams : This describes various interactions performed by different components present in the system.



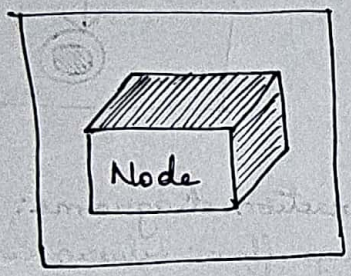
2. Interaction diagrams : These describe message flow between components present in the system.



6. Component: This notation represents a section of the system.



7. Node: This notation is similar to component notation, with the difference being that the node notation refers to the physical aspect of a system, such as a server.



3) Benefits of using UML:-

(1) Flexible and easy to understand even for non-technical.

(2) widely used and has large community, which makes it easier to perform collaborative work among teams.

(3) ambulance of tools that helps break complex systems into smaller pieces.

11. Types of UML Diagrams

1) Structural UML Diagrams
2) Behavioral UML Diagrams.
3) UML diagrams used in the OOD interview.

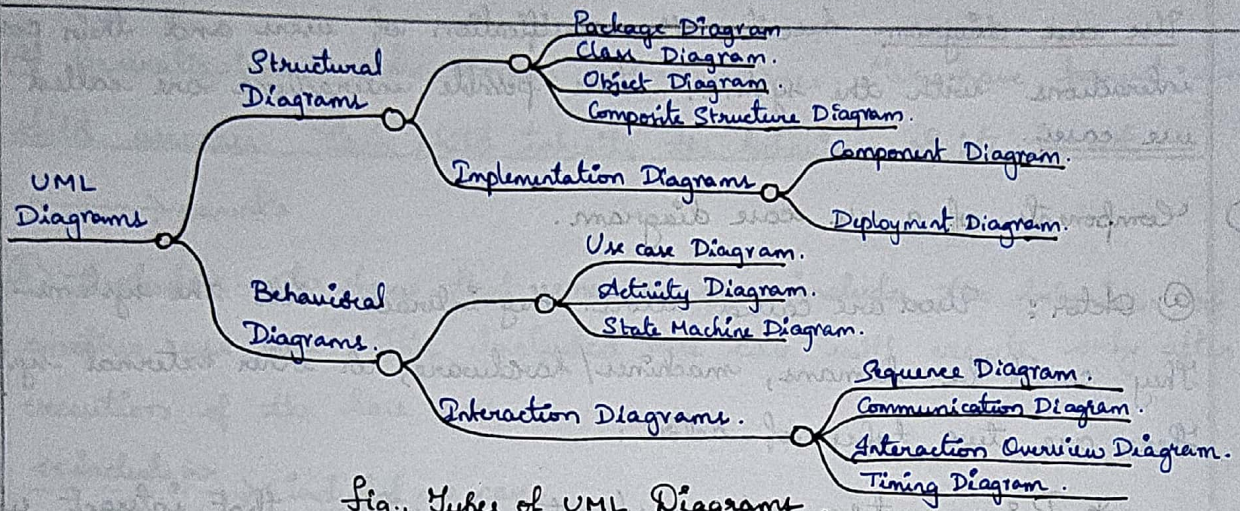


Fig., Types of UML Diagrams.

1) Structural UML Diagrams :-

Structural diagrams represent static structure of the system. They never depict the system's dynamic behavior. The most commonly used structural diagram in software development is the class diagram.

2) Behavioral UML Diagrams :-

In object-oriented programming, we use behavioral diagrams to illustrate the dynamic behavioral semantics of a problem or its implementation. The most commonly used behavioral diagrams are use case diagrams, activity diagrams and sequence diagrams.

3) UML diagrams used in OOD interview.

* Use case diagrams
* Class diagram.
* Sequence diagram.
* Activity diagram.

12. Use Case Diagram

1) Components of a use case diagram.
2) Relationships in use case diagrams
3) Benefits of use case diagrams.

Use case diagram describes the specification of users and their possible interactions with the system. These possible interactions are called use cases.

1) Components of a use case diagram.

(a) Actor: Users are called actors. They interact with the systems. They could be humans, machines/hardware, or other external systems. There are two types of actors:

* Primary actors: humans/external systems that interact with that system and are responsible for initiating the use case. They are placed on the left side in a use case diagram.

Also called as active actors.

* Secondary actors: are used by the system to assist the primary actors in a use case. They cannot interact with the system on their own. They need primary actors to initiate a use case.

Also called as passive actors and placed on the right side in a use case diagram.

(b) Use case: This is a single function performed on a system by an actor. It is represented by an oval shape.

(c) Package: This is a group of different elements. These groups are represented inside a folder icon.

* Note: This is used to add additional information about any component or relationship in a use case diagram.

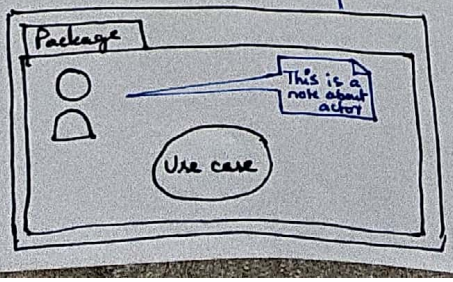


fig., Components of a use case diagram.

2) Relationships in use case diagrams:-

4 different types of relationships in a use case diagram:

1. Association:- relationship between/among the actors(s) and usecase(s). All actors in a use case diagram must have at least one association with any use case. ——

2) Generalization:- also known as inheritance. We have parent and child usecases. Each child inherits the behaviour of its parent.

——————>parent>

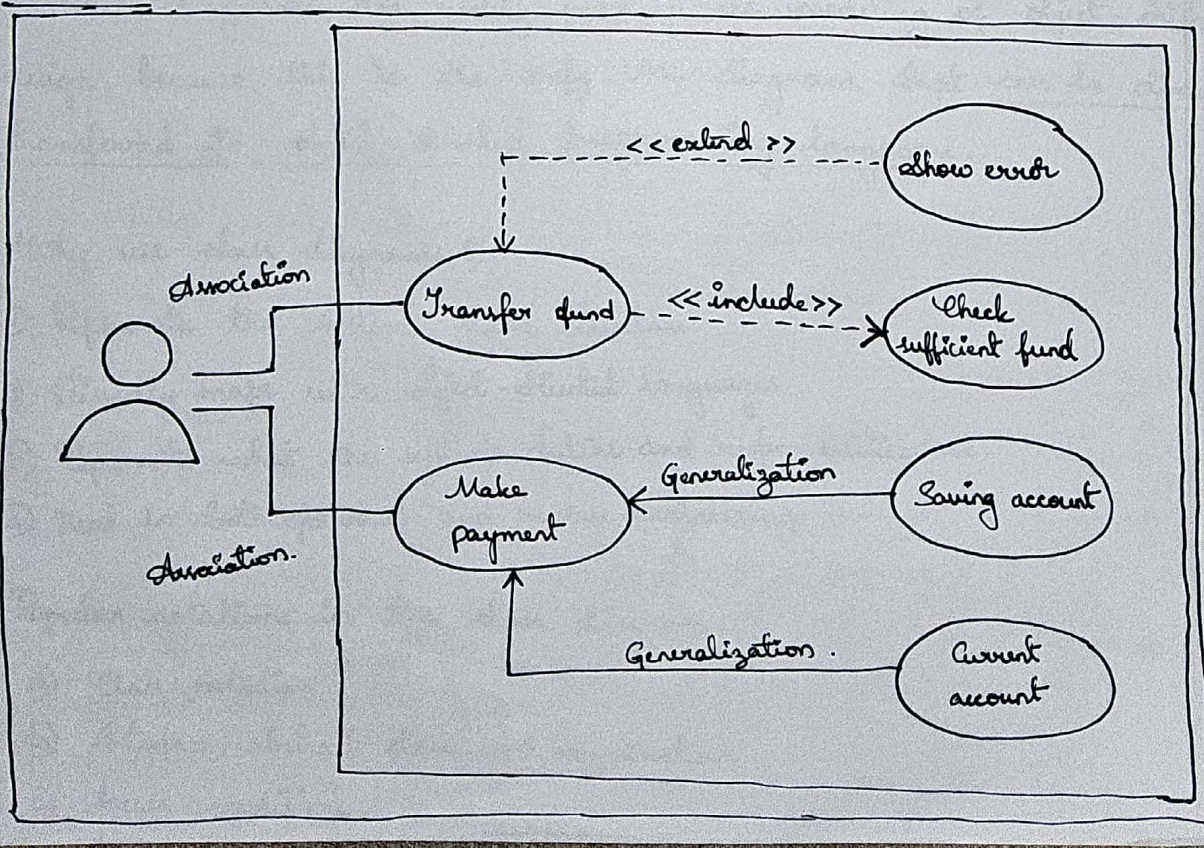
3) Include:- to show that one use case includes the behaviour of another use case. The included use case will execute only after the execution of the base case.

<<include>> - - - - - - - -><included use case>

4) Extend:- one use case extends the behaviours of another use case. The extended use case does not execute every time. It always depends on certain conditions.

<<extend>> - - - - - - - -> <base use case>

EXAMPLE:-

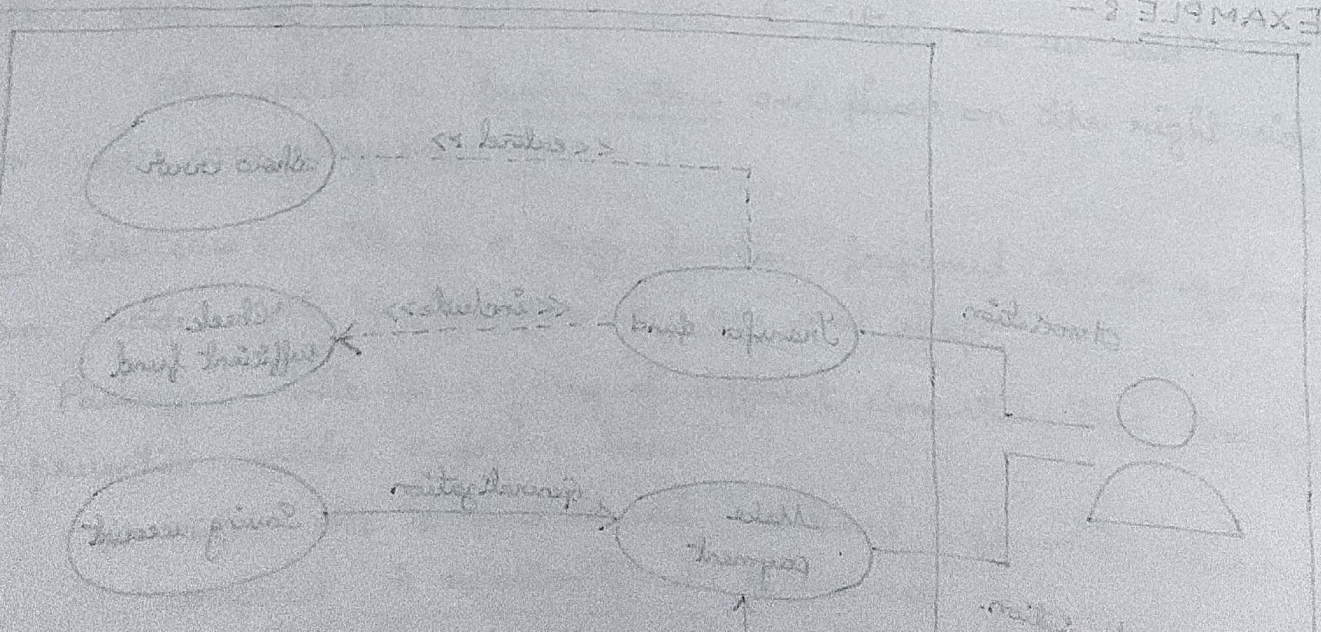


We have a small ATM (Automated Teller Machine) transaction system where customers can transfer funds and make payments. To validate the funds, the transfer system has to check if a sufficient amount of funds is available. Otherwise, an error message will be displayed. To make a payment, a customer has two choices. It can either pay via a current account & a savings account.

use case ① use case ② use case ③ use case ④

3) Benefits of use case diagrams :-

① It explains the flow and objective of all use cases.
② It helps in understanding the high-level functional requirements of the system.
③ It defines a system's context and needs.
④ It explains system behaviour from a user perspective.
⑤ It explains the scope of the system.



1. Class Diagram

1. Why use class diagrams?
2. Popular notations in the class diagram.
   - (a) Class notation.
   - (b) Interface, abstract class, and enumeration.
   - (c) Access modifiers.
3. Association.
   - (a) Class association.
   - (b) Object association.
     - (i) Simple association.
     - (ii) Aggregation.
     - (iii) Composition.
     - (iv) Some additional types of association.
4. Dependency.

Class diagrams are used to depict the system's static perspective. They are used in the design process to show the shared roles and responsibilities of the entities that produce the behavior of the system.

Class diagrams are widely used in the modeling of object-oriented design because this is the only UML diagram that can be directly transferred to object-oriented programming languages.

1. Why use class diagrams?
   1) Represents the system's static structure.
   2) Directly maps with object-oriented languages.
   3) Represents what the system's duties and responsibilities are.
   4) Used in both forward and reverse engineering.
2. Popular notations in the class diagram.
   a) Class notation.
   b) Interface, abstract class and enumeration.
   c) Access modifiers.

ⓐ Class notation :- class name, attributes, methods.

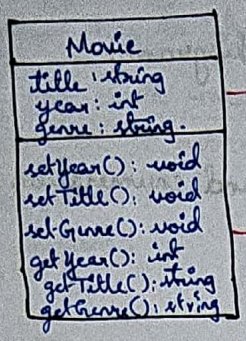


Fig., Notation of a class in a class diagram.

ⓑ Interface, abstract class and enumeration :-

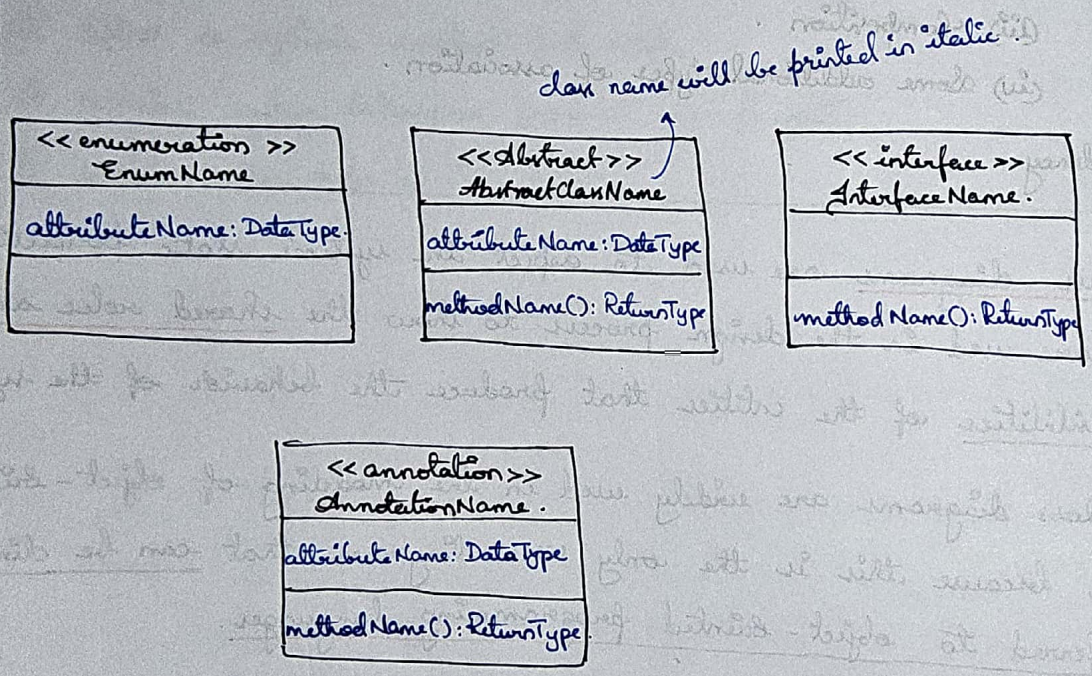


Fig., Class diagram notation for abstract classes, interfaces, enumerations, and annotations.

ⓒ Access modifiers :-

public (+) – can be seen anywhere.

private (-) – can only be accessible from within the class.

protected (#) – are only accessible within the class & derived classes.

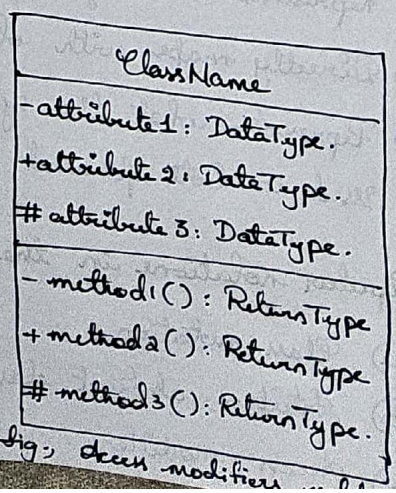
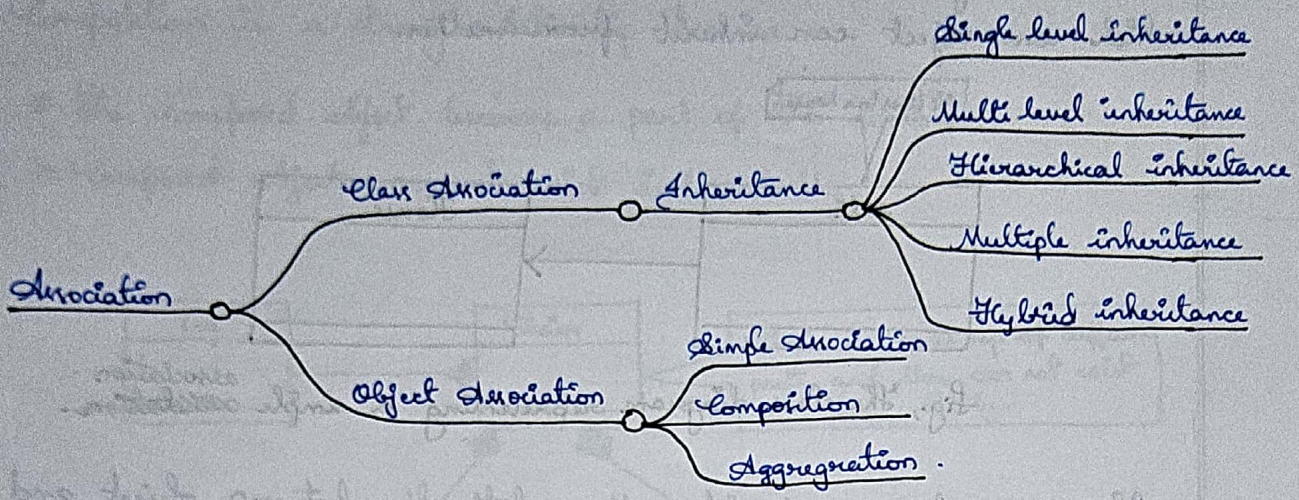


Fig., Access modifiers

③ Association :- Association provides a mechanism to communicate one object with another object, ie one object provides services to another object. It represents the relationship between classes.



a) Class association :-

Inheritance falls under the category of class association. Creating a new class from the existing class(es) is called inheritance.

Child ——▷ Parent.

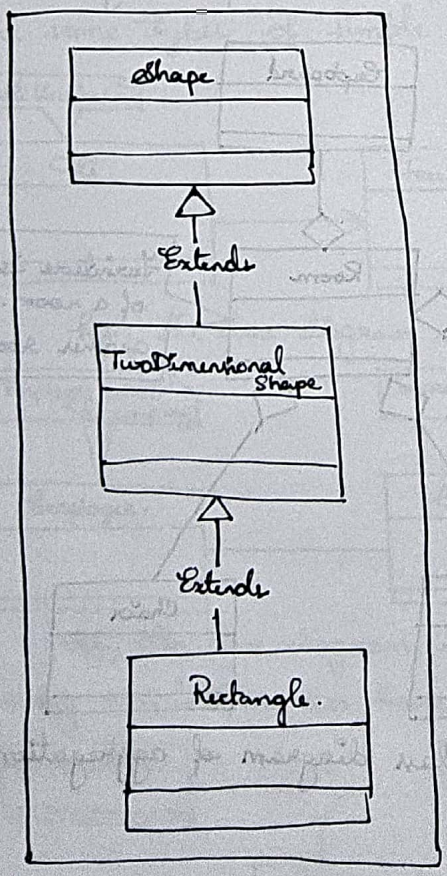


Fig., Class diagram to represent the multi-level inheritance.

b) Object association :-

(i) Simple association: The weakest connections between objects are made through simple association. It is achieved through reference, which one object can inherit from another.

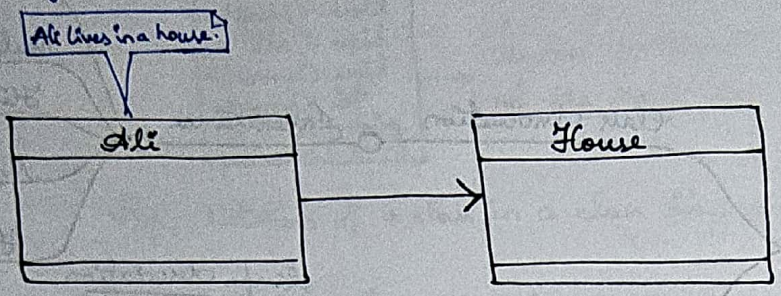


fig., The class diagram representing a simple association.

(ii) Aggregation: describes the relationship between object and its container.

Aggregation is a weaker relationship because:

* Aggregate objects are not a part of the container.
* Aggregate objects can exist independently.

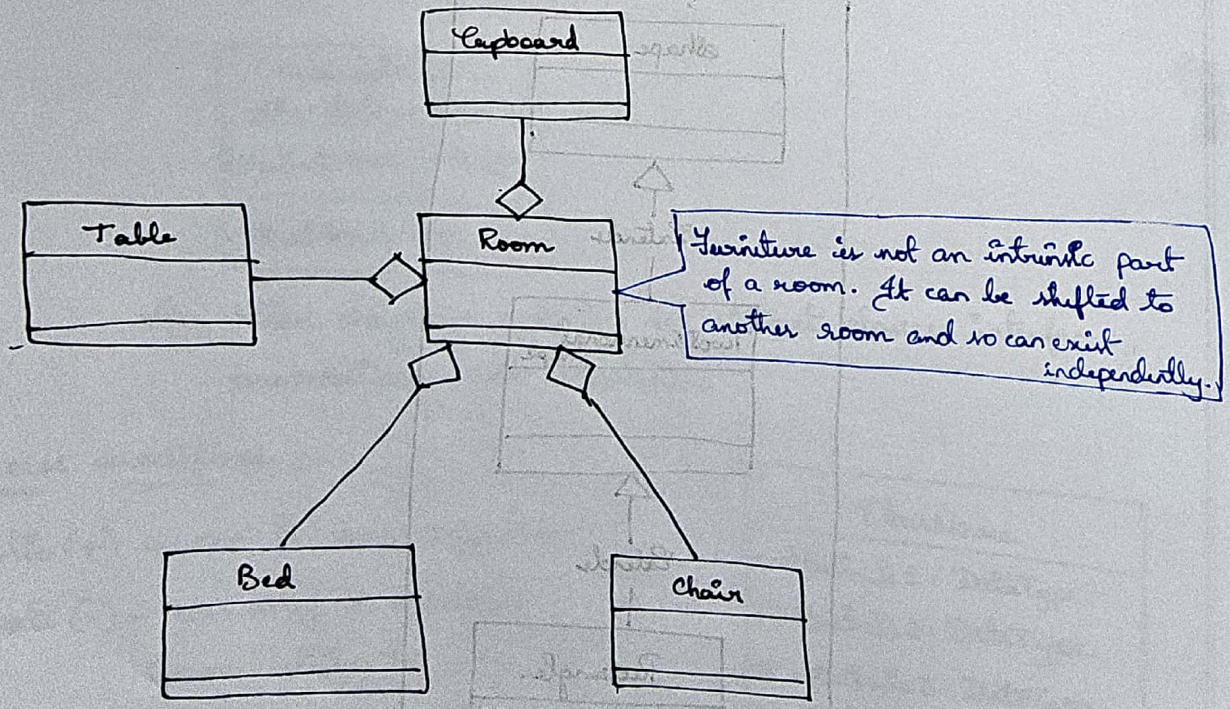


fig., The class diagram of aggregation.

c) Composition:- An object may be composed of smaller objects, and the relationship between the "part" objects and "whole" objects is known as composition.

Composition is a strong relationship because:

* The composed object becomes a part of the composer.
* Composed objects cannot exist independently.

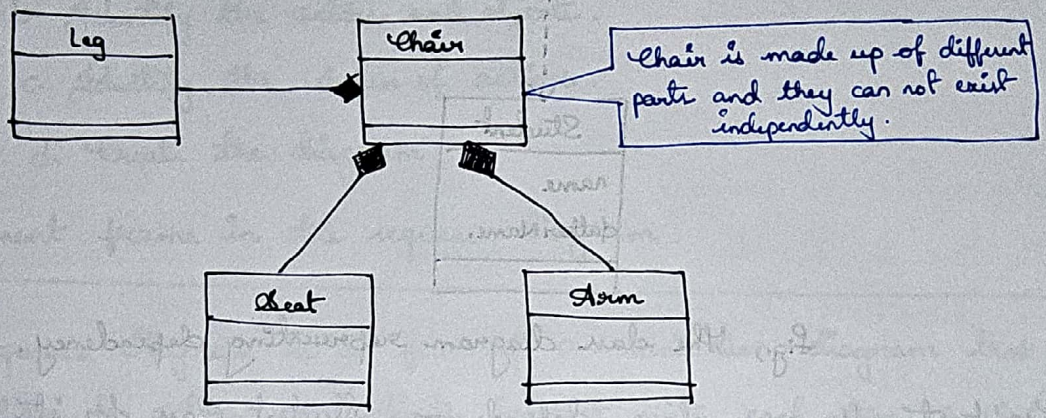


fig., The class diagram of composition.

d) Some additional types of association:-

The following are some types of simple association based on navigation:

(i) one-way association:
Single-direction navigation.

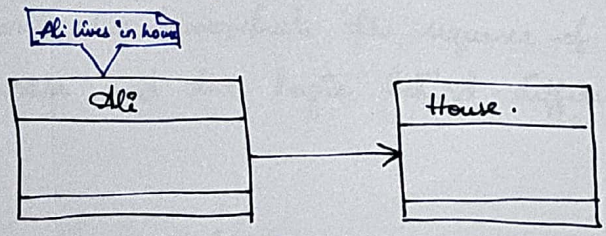


fig., The class diagram of one-way association.

(ii) two-way association:
If we navigate in both directions.

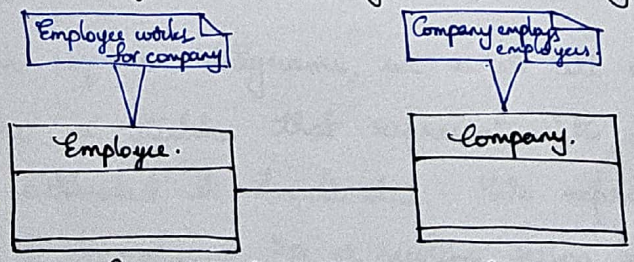
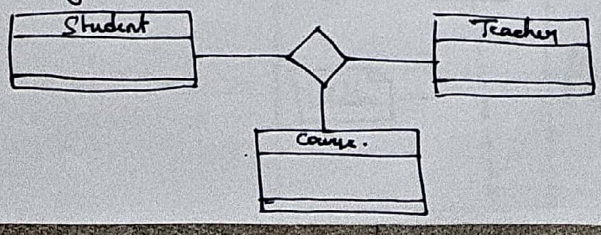


fig., The class diagram of two-way association.

(iii) binary, ternary, n-ary associations are based on the number of objects.



④ **Dependency:-**

Dependency indicates that one class is dependent on another class for its implementation.

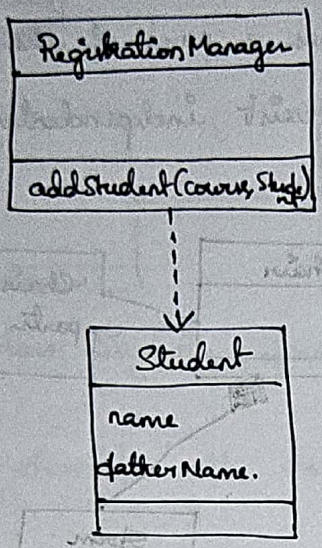


Fig.; The class diagram representing dependency.

RegistrationManager class depends on Student class for its behaviours because the object of the Student class is passed as parameter to one of the functions in the RegistrationManager class.

VI. Sequence Diagram

1. Elements of a sequence diagram.
   a. Lifeline
   b. Activation bars.
   c. Messages.
2. How to draw a sequence diagram.
   a. Identify the use case.
   b. Identify the actors and objects.
   c. Identify the order of actions.
   d. Create the diagram.
3. Fragment frame in the sequence diagram.

---

* A sequence diagram is a form of communication diagram that illustrates how different actors and objects interact with each other or between themselves.
* The diagram represents these interactions as an exchange of messages between various entities and the type of exchange.
* Sequence diagrams also demonstrate the sequence of events that occur in a specific use case and the logic behind different operations and functions.

1. Elements of a sequence diagram :-

a. **Lifeline :-** In sequence diagrams, we write all entities horizontally. Each entity has a lifeline that represents its existence, i.e., when the entity is activated or deactivated. We represent different objects, actors, entities, & boundaries in a system using lifelines, and they never overlap each other.

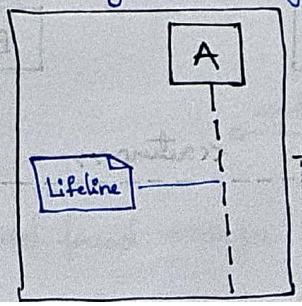


Fig., Lifeline in a sequence diagram.

b. Activation bars: Activation bars indicate the active period of an object, that is, the time when an object sends or receives messages.

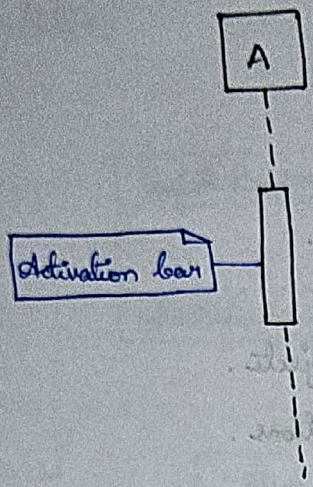


Fig., Activation bar in a sequence diagram.

c. Messages: A message is an interaction between two objects. It can be in the form of sending and receiving messages, invoking an operation, or creating a new object or entity. Messages are drawn horizontally in any direction: left to right, right to left, or back to themselves.

① Synchronous message: it a type of message where the sender has to wait for the receiver to return a response before it can perform another operation.

② Asynchronous message: it a type of message where the sender does not have to wait for a response from the receiver.

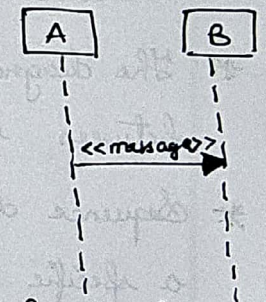


Fig., A synchronous message.

③ Synchronous return: it a type of message generated in response to a synchronous call. A synchronous message has to be paired with a synchronous return message. It means the receiver has processed the message and sent a response.

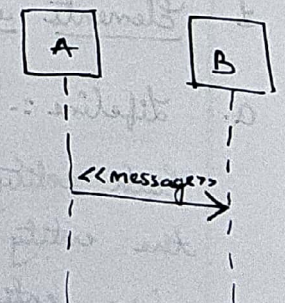


Fig., A asynchronous message.

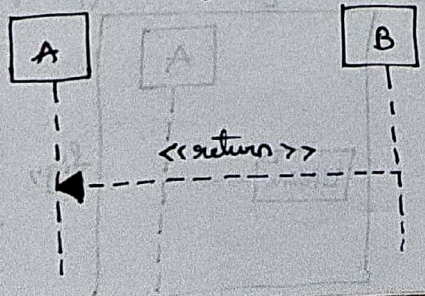
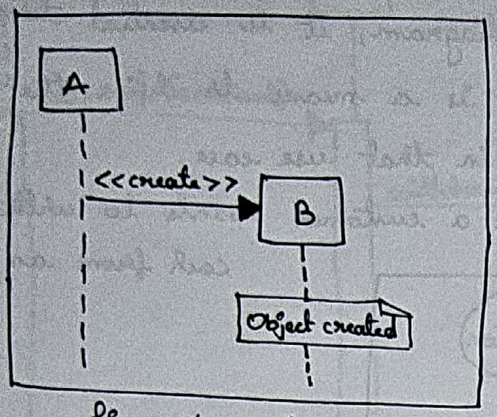


Fig., A synchronous return message

④ Create message: indicates that a new object is created during an interaction. New objects are created as a result of some message or operation.



Fig, A create message.

⑤ Destroy message: indicates that an object is destroyed during a sequence of events. Objects can be destroyed as a result of some message or operation, and their lifeline ends.

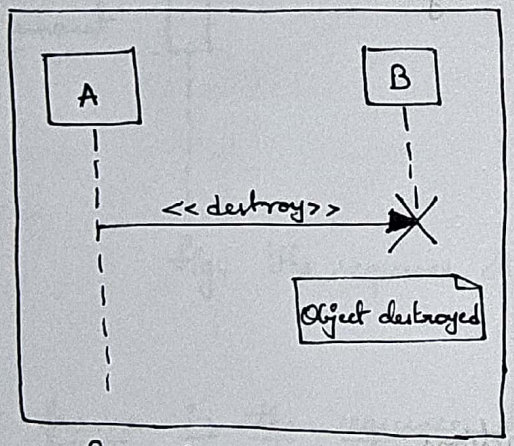


Fig., A destroy message.

⑥ Lost message: is a message that initiates from an object but does not reach its endpoint. It appears as a message that is terminated.

⑦ Found message: is a message that is received, but the sender is unknown. It appears as a message that reaches an endpoint but does not initiate from any object.

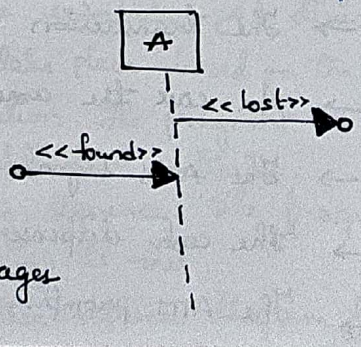


Fig., The lost and found messages

2. How to draw a sequence diagram :-

(i) Identify the use case :

To get started with the sequence diagram, it is essential to know our use case. The sequence diagram is a means to define the sequential order of events that occur in that use case.
Lets build on a simple scenario where a customer wants to withdraw cash from an ATM.

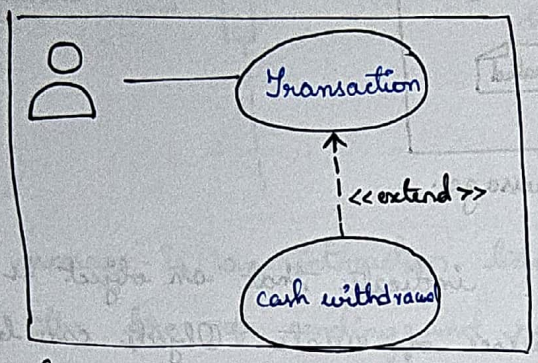


fig., Use case for cash withdrawal.

(ii) Identify the actors and objects :

* customer
* ATM
* Transaction
* Account.
* Cash Dispenser.

(iii) Identify the order of actions :

1 → The customer requests a cash withdrawal from the ATM with their account information and the amount.
2 → The ATM initiates a cash withdrawal transaction against the account and the given amount.
3 → The transaction amount is verified for the given account.
4 → In case the amount is valid, the account verifies the transaction.
5 → The ATM signals the cash dispenser to release the required cash amount.
6 → The cash dispenser confirms the release of cash.
7 → The ATM prompts the user to collect the cash.

(iv) Create diagram.

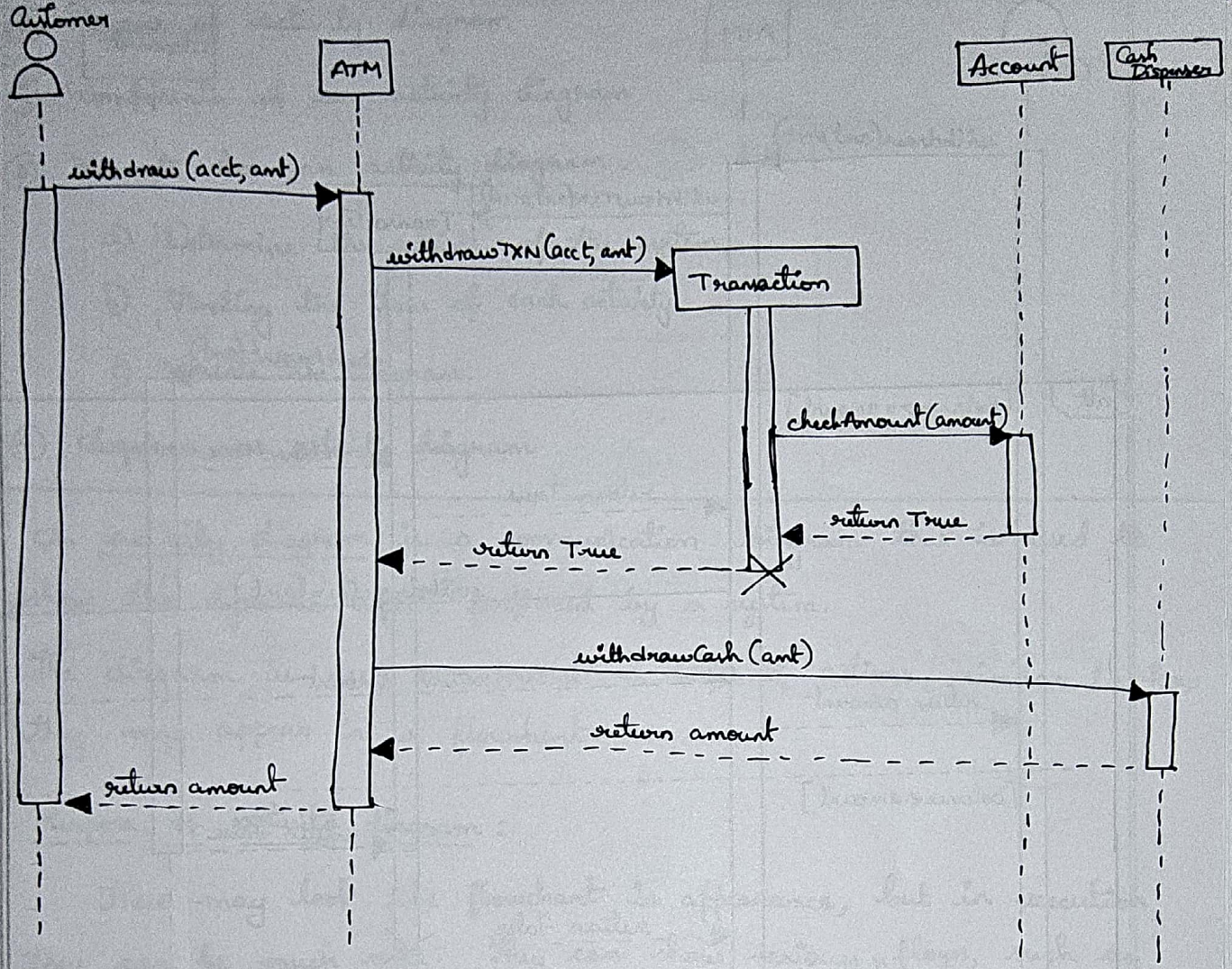


Fig., The sequence diagram for cash withdrawal.

3. Fragment frame in the sequence diagram:-

Sometimes we need to show more complex actions like conditional actions or loop sequences. Sequence diagrams allow us to represent this information using the sequence fragment component. We can identify the operator of a fragment in the top left corner of the fragment frame.

* Alternative (alt): This operator models the "if-else" condition. It divides the fragment into parts, and either of the parts can take place based on the guard condition.
* Option (opt): This operator models the "if" condition. The fragment is skipped if the guard condition is not met, and the interaction continues.
* Loop (loop): This operator represents a repetitive sequence. The fragment will keep repeating until the guard condition is met.
* Reference (ref): allows us to reuse parts of another sequence diagram by providing ref to it.

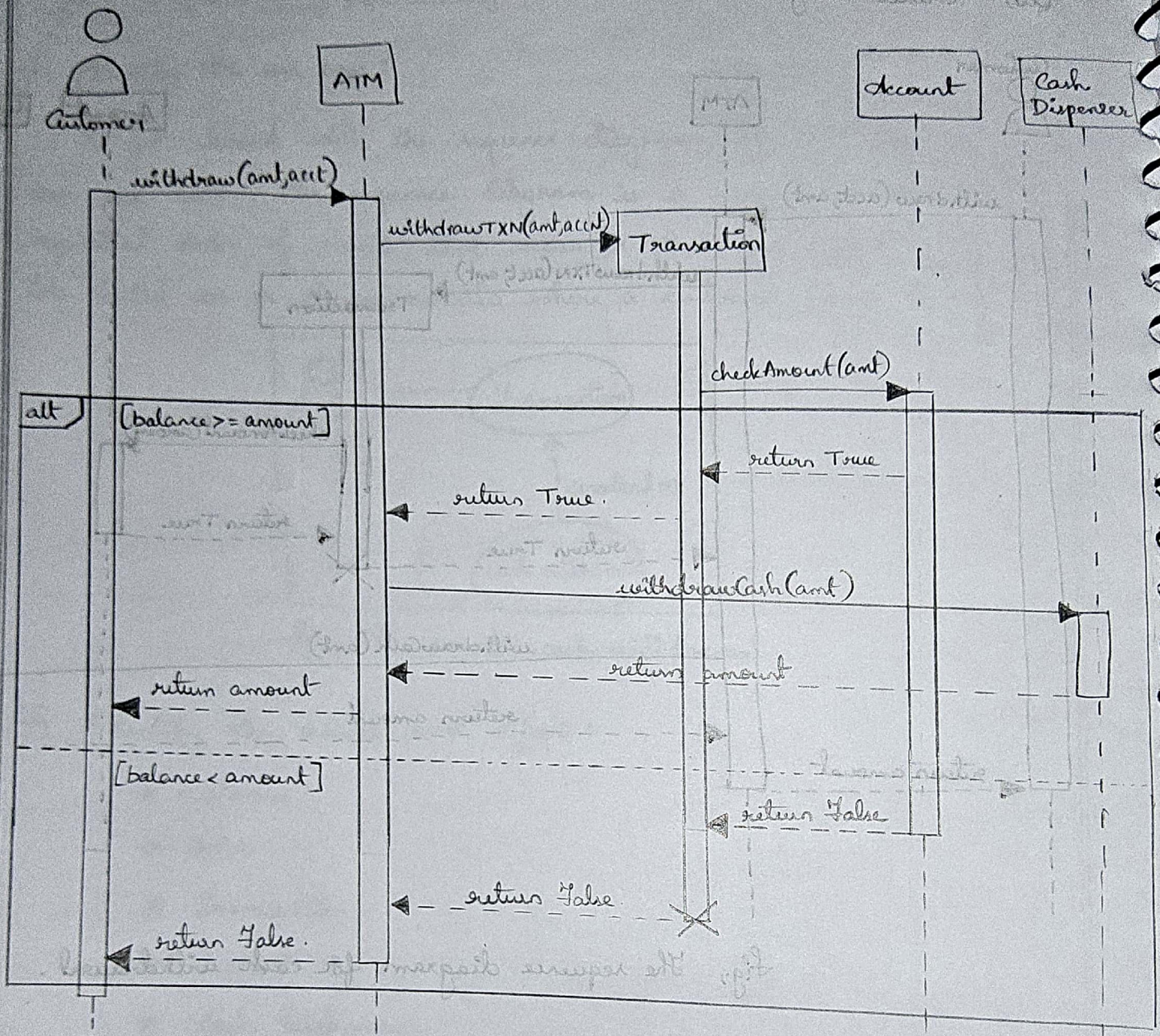


Fig., The sequence diagram for cash withdrawal with a sequence fragment

VII. Activity Diagram

1. Purpose of activity diagram
2. Components of an activity diagram
3. How to draw an activity diagram.
   a) Determine the actions of the system.
   b) Finding the flow of each activity.
   c) Create the diagram.
4. Sequence vs. activity diagram.

An activity diagram is a communication diagram that is used to show the dynamic aspects performed by a system.
The diagram is used to represent a series of actions, similar to how they may appear in a flowchart.

① Purpose of activity diagram :-

These may look like flowchart in appearance, but in execution, they can be much more. They can show various flows, such as single, parallel, concurrent and branched.

② Components of an activity diagram :-

* Initial: This represents start of the workflow of the activity diagram. They can be visualized as the node in a tree structure.

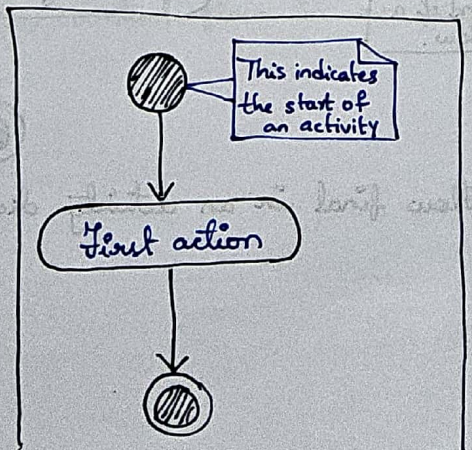


Fig., Initial node in an activity diagram.

* Action :- These are the main building blocks of an activity diagram and are used to show the activities that a modeled process is made of.

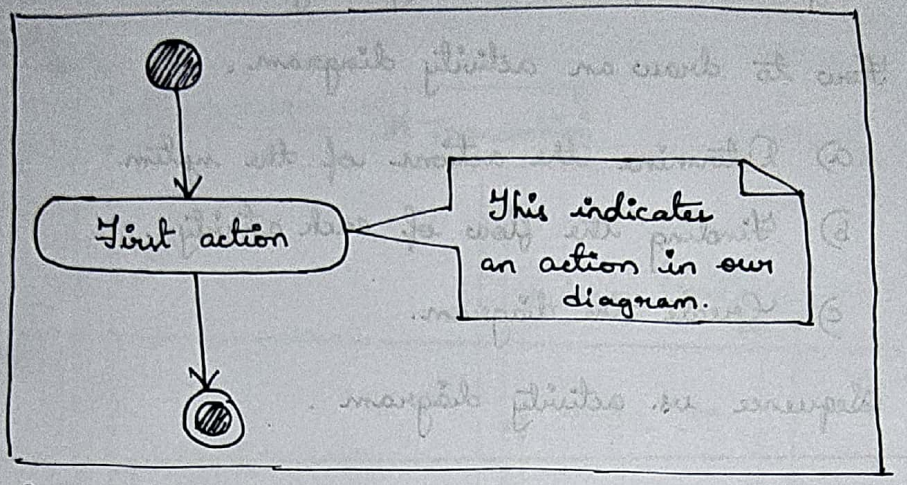


Fig., Action in an activity diagram.

* Flow final :- This represents the end of a single path in the activity diagram. They can be visualized as a leaf in a tree structure.

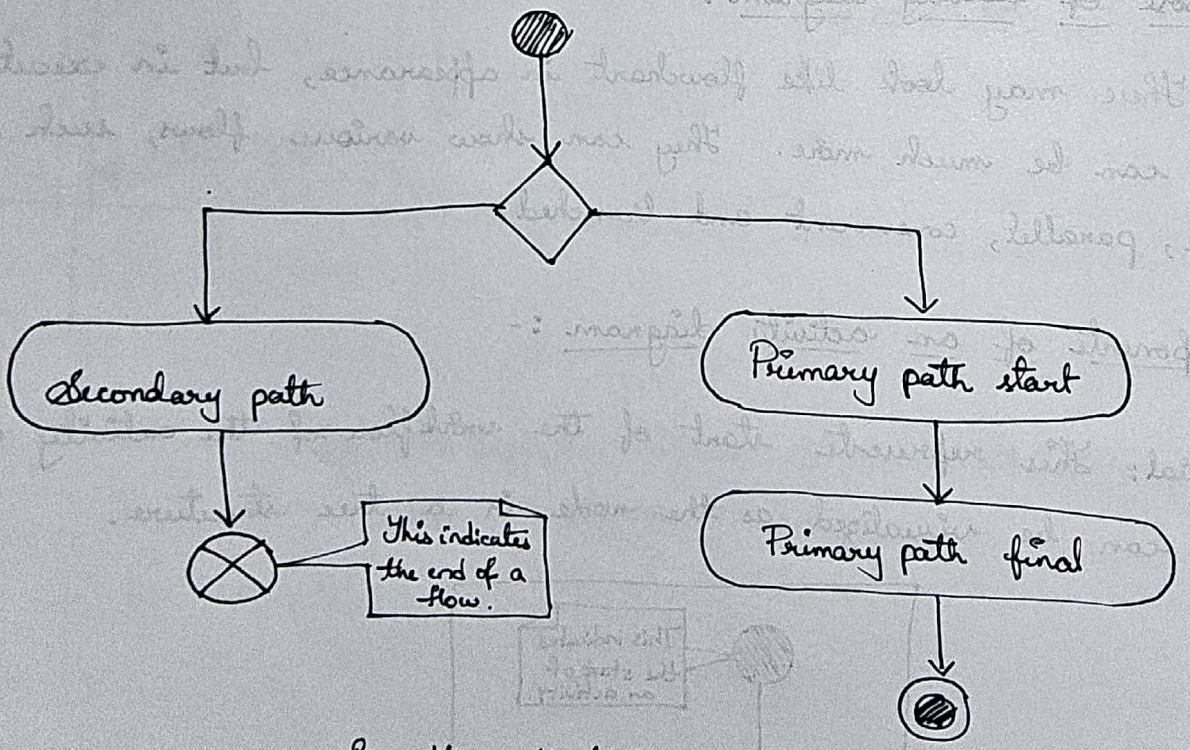


Fig., Flow final in an activity diagram.

* Activity final :- This represents the end of all activities in the activity diagram.

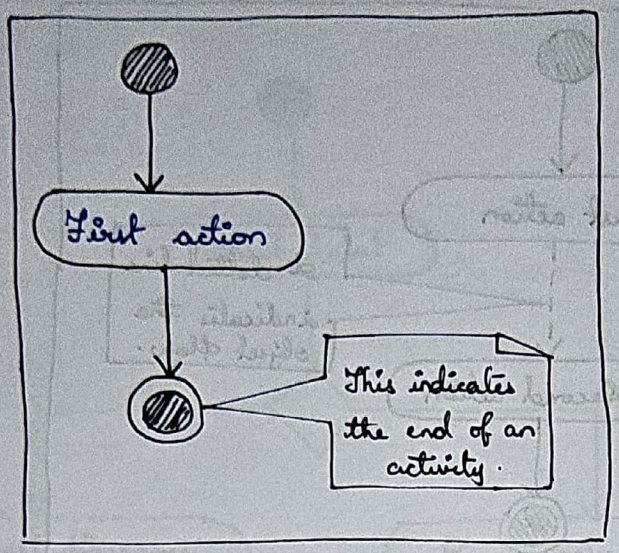


Fig., Activity final in an activity diagram.

* Control flow :- This shows the directional flow of the diagram. This exists as connector between one action and another.

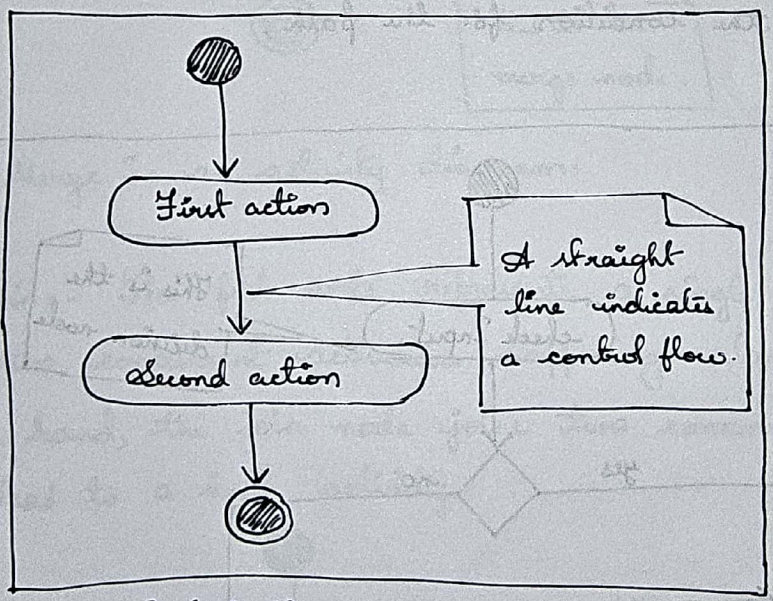


Fig., Control flow in an activity diagram.

* Object flow :- This shows the path of the objects as they move throughout the activity.

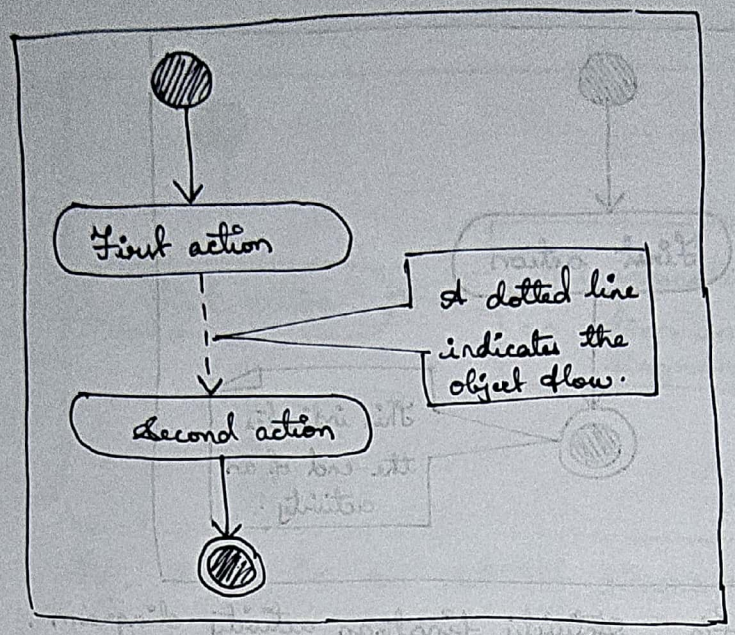


Fig., Object flow in an activity diagram.

* Decision :- This is used to represent multiple options that are possible in a system. They appear as a branch alongside the text describing the condition for the path.

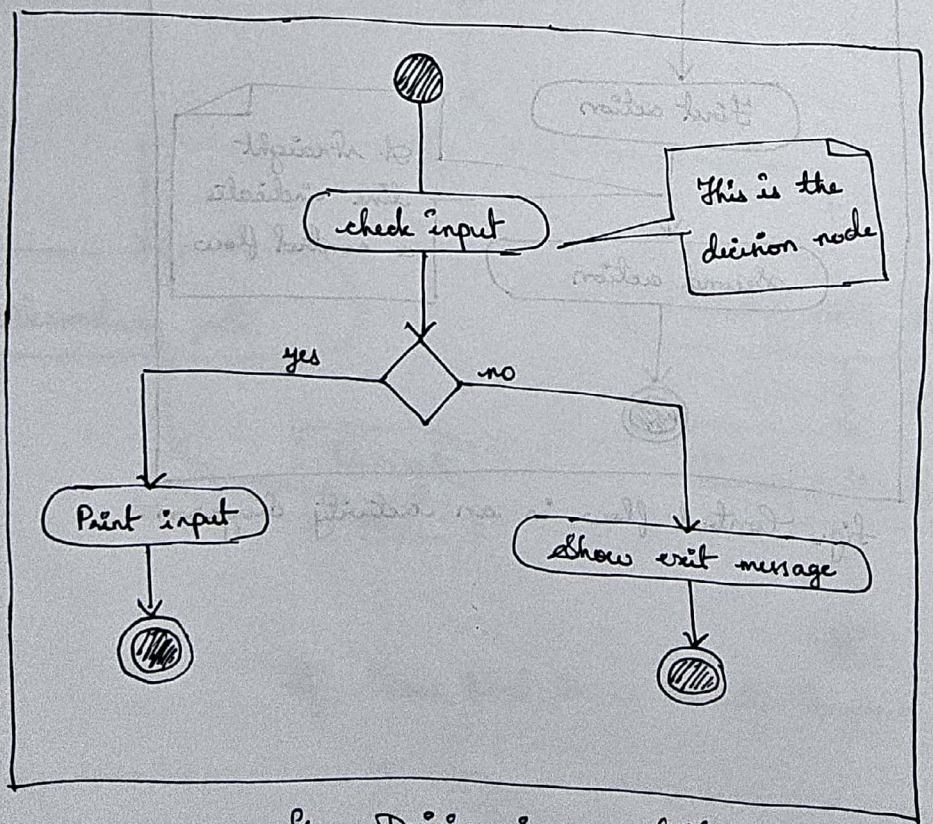


Fig., Decision in an activity diagram.

* Merge:- This uses the same symbol as a decision. However, this shows that multiple options join at this node but leads to a single output.

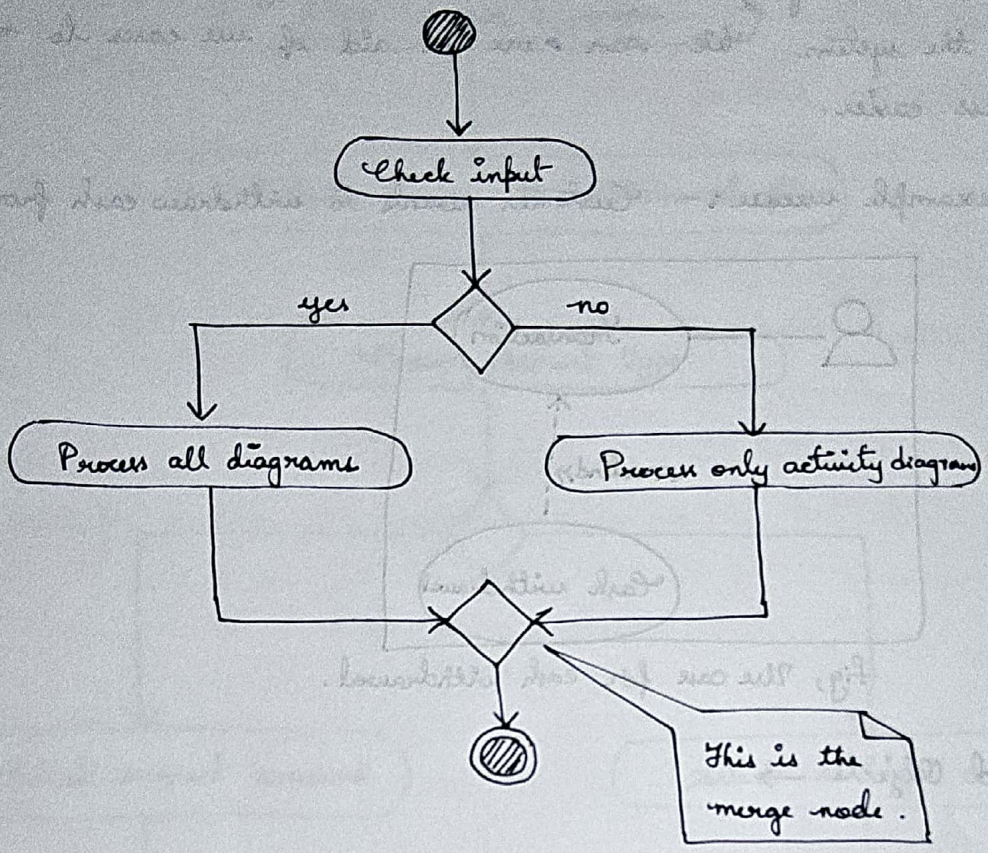
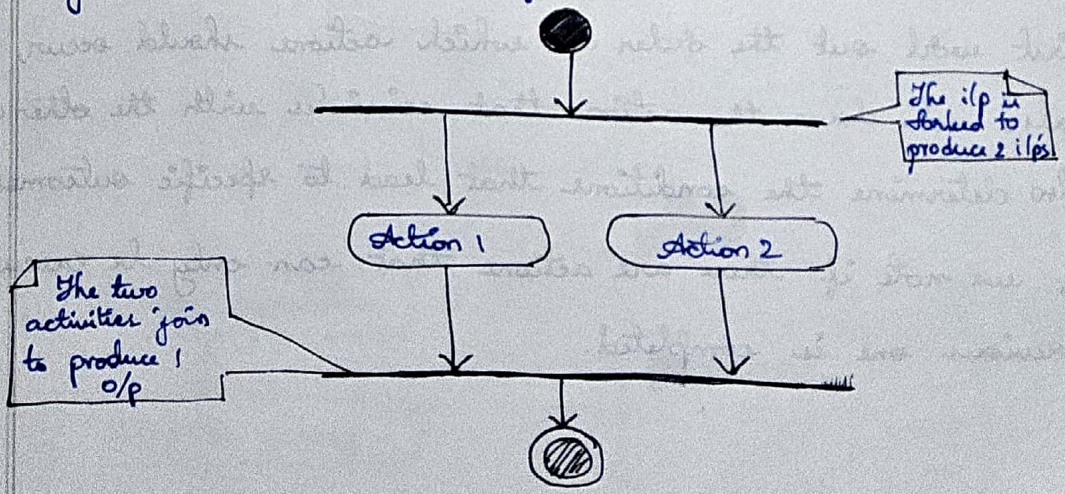


Fig., Merge in an activity diagram.

* Fork and join:- The fork node represents a single activity that is split into two concurrent activities happening alongside each other. On the other hand, the join node joins two concurrent activities together to lead to a single activity.



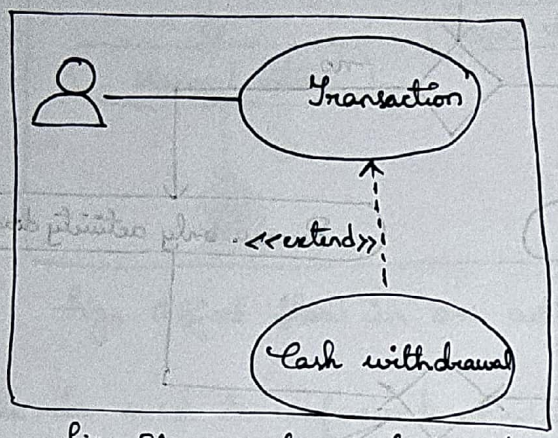
Fig, Fork and join in an activity diagram.

③ How to draw an activity diagram?

(i) Determine the actions of the system.

We need to recognize each action and how they interact with the others in the system. We can use the aid of use cases to make this process easier.

Previous example usecase :- Customer wants to withdraw cash from ATM.



fig, Use case for cash withdrawal.

Actors and Objects →

* Customer
* ATM
* Transaction
* Account
* Cash dispenser.

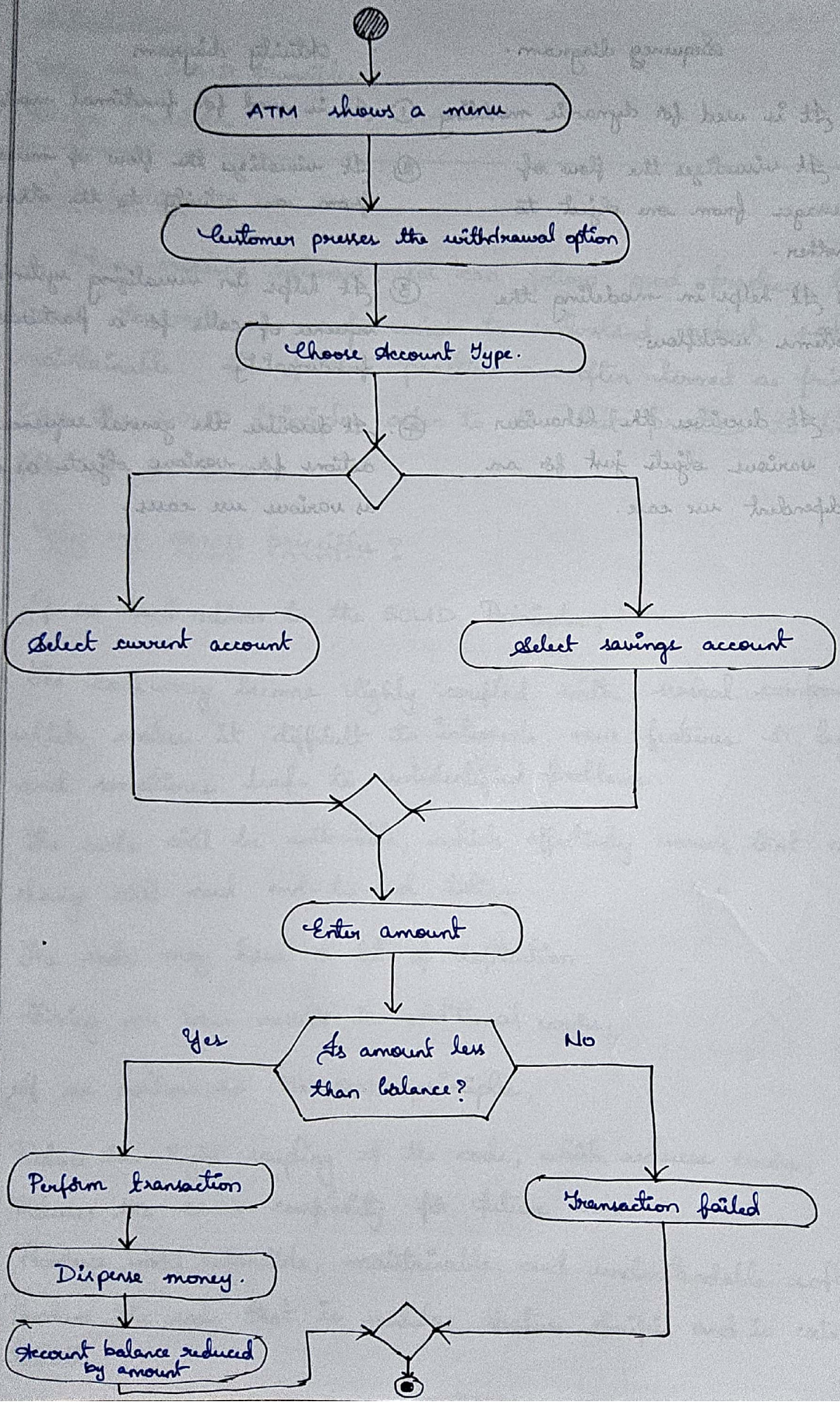
(ii) Finding the flow of each activity.

We first work out the order in which actions should occur, and we also note down the action that coincides with the other actions.

We also determine the conditions that lead to specific outcomes.

Lastly, we note if there are actions that can only be executed once a previous one is completed.

(iv) Create the diagram :-



④ Sequence vs Activity diagram:-

Sequence diagram.	Activity diagram
① It is used for dynamic modeling	① It is used for functional modeling
② It visualizes the flow of messages from one object to another.	② It visualizes the flow of messages from one activity to the other.
③ It helps in modeling the systems workflow.	③ It helps in visualizing system's sequence of calls for a particular functionality.
④. It describes the behaviour of various objects just for an independent use case.	④. It describes the general sequence of actions for various objects as well as various use cases.