Activity & Sequence Diagrams

Special class

Love Babbar • Mar 10, 2024

Object Oriented Design

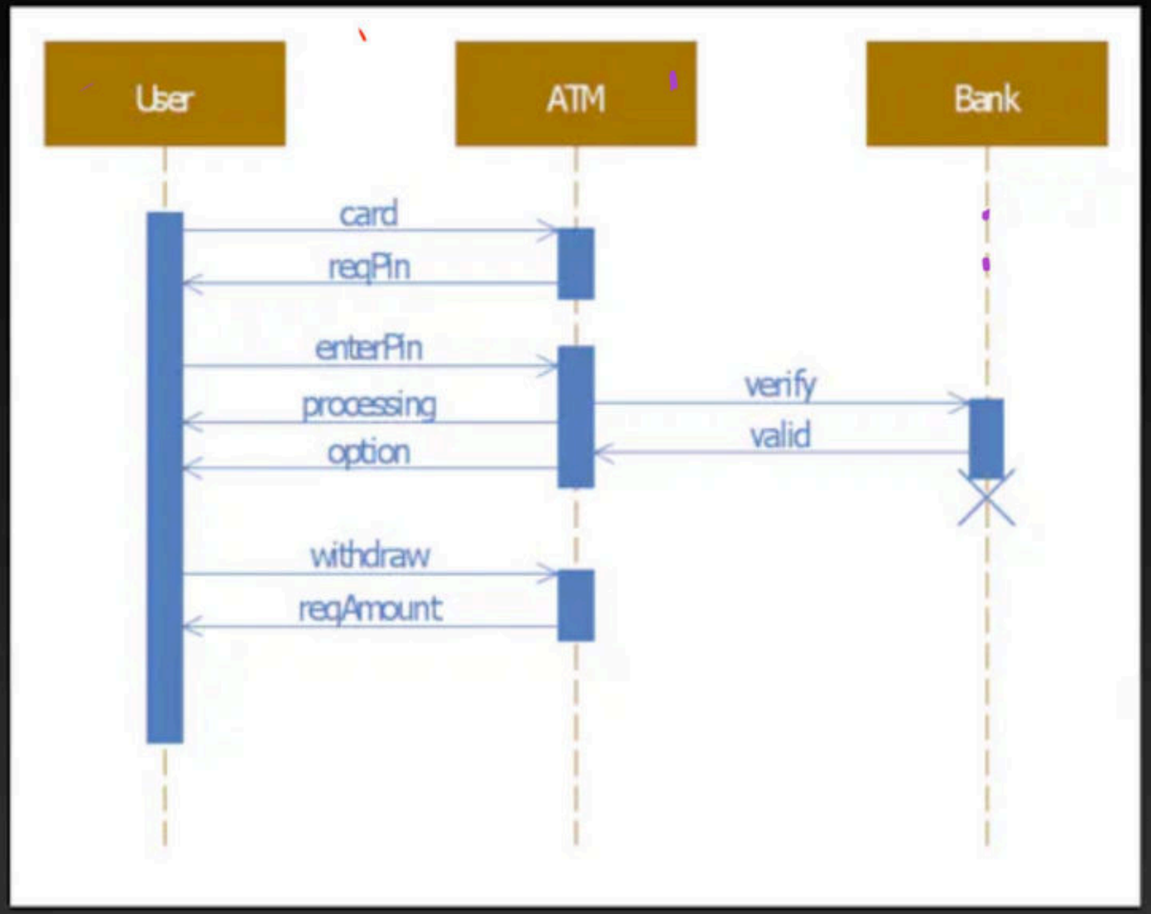
Module - II

- Love Babbar

Sequence Diagram

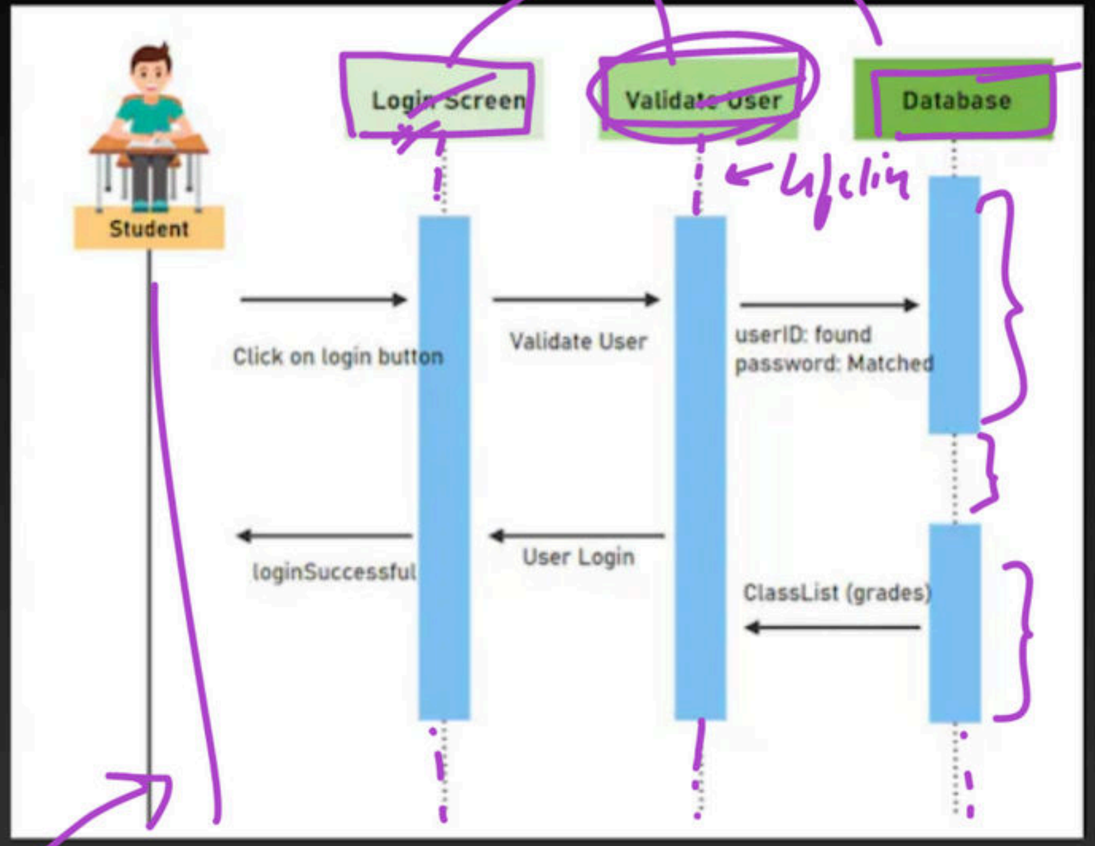
models / Actors / Objects communicate by msg.

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.

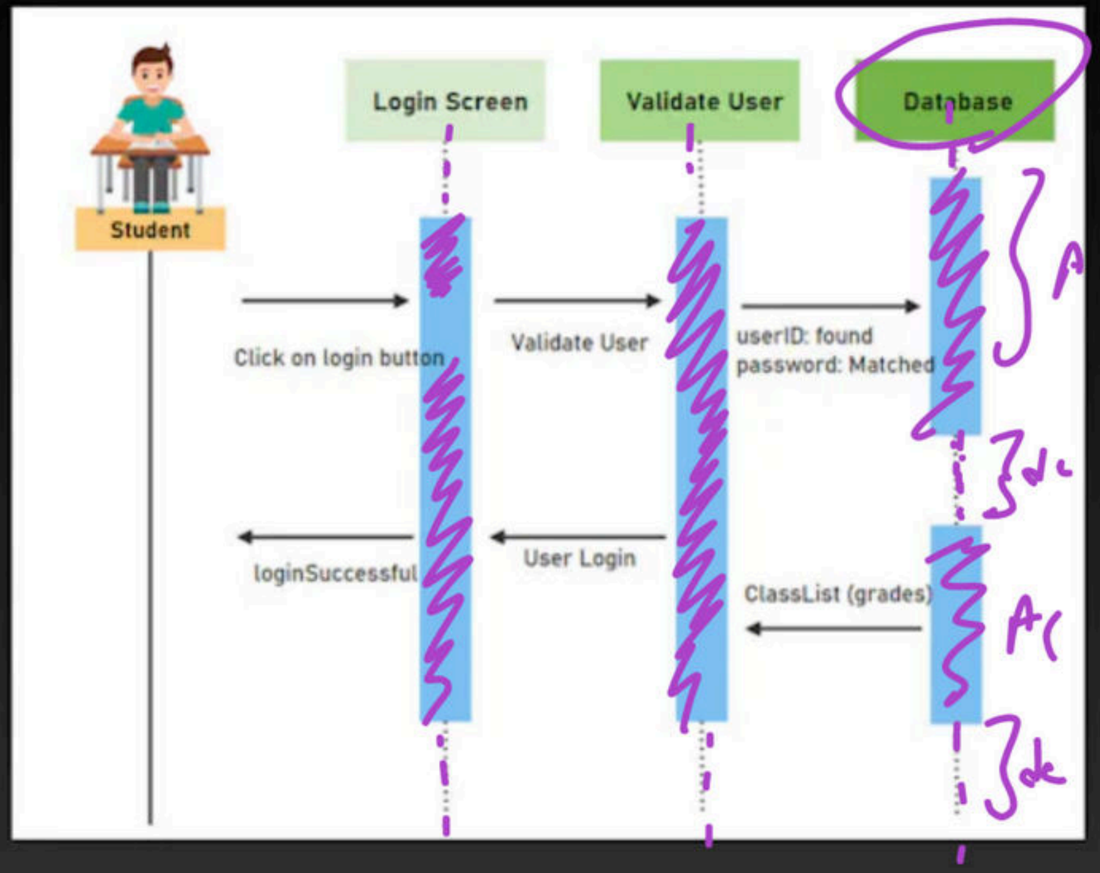


Elements:

- ***Lifeline:*** In sequence diagrams, we list everything involved across the top. Each thing has a lifeline showing when it's active or not, shown as a dotted line hanging down from it. These lifelines stand for different parts of a system, like objects or actors, and they don't cross each other. Below, there's a picture showing what these lifelines look like in UML. The things involved in what's happening are shown as horizontal boxes.



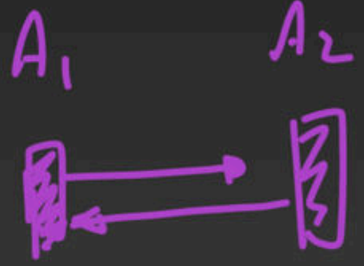
- ***Activation Bars***:Activation bars show when an object is busy sending or getting messages. We make these as straight up-and-down boxes on the object's lifeline. Here's what an activation bar looks like:

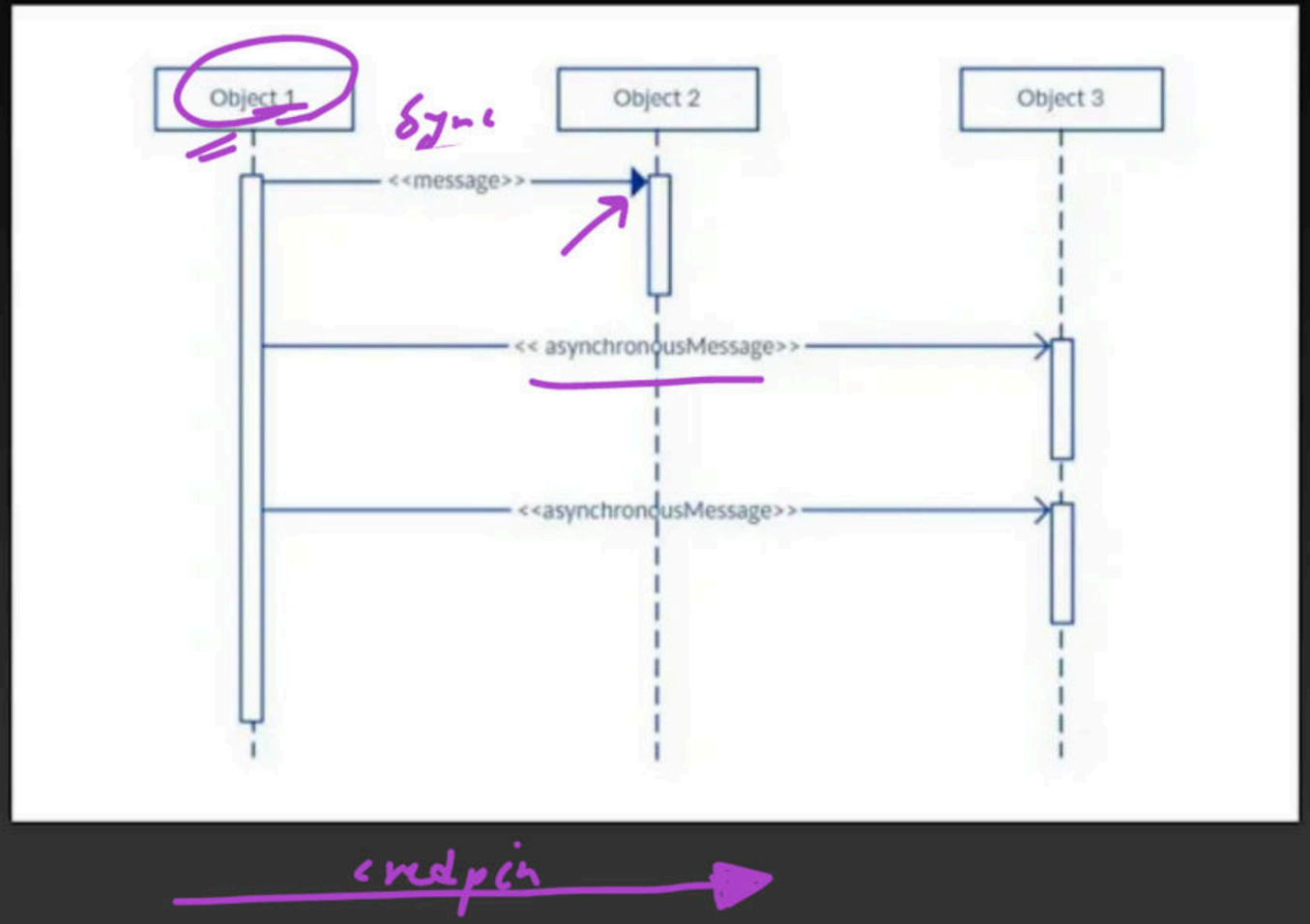


Messages in Sequence Diagrams:

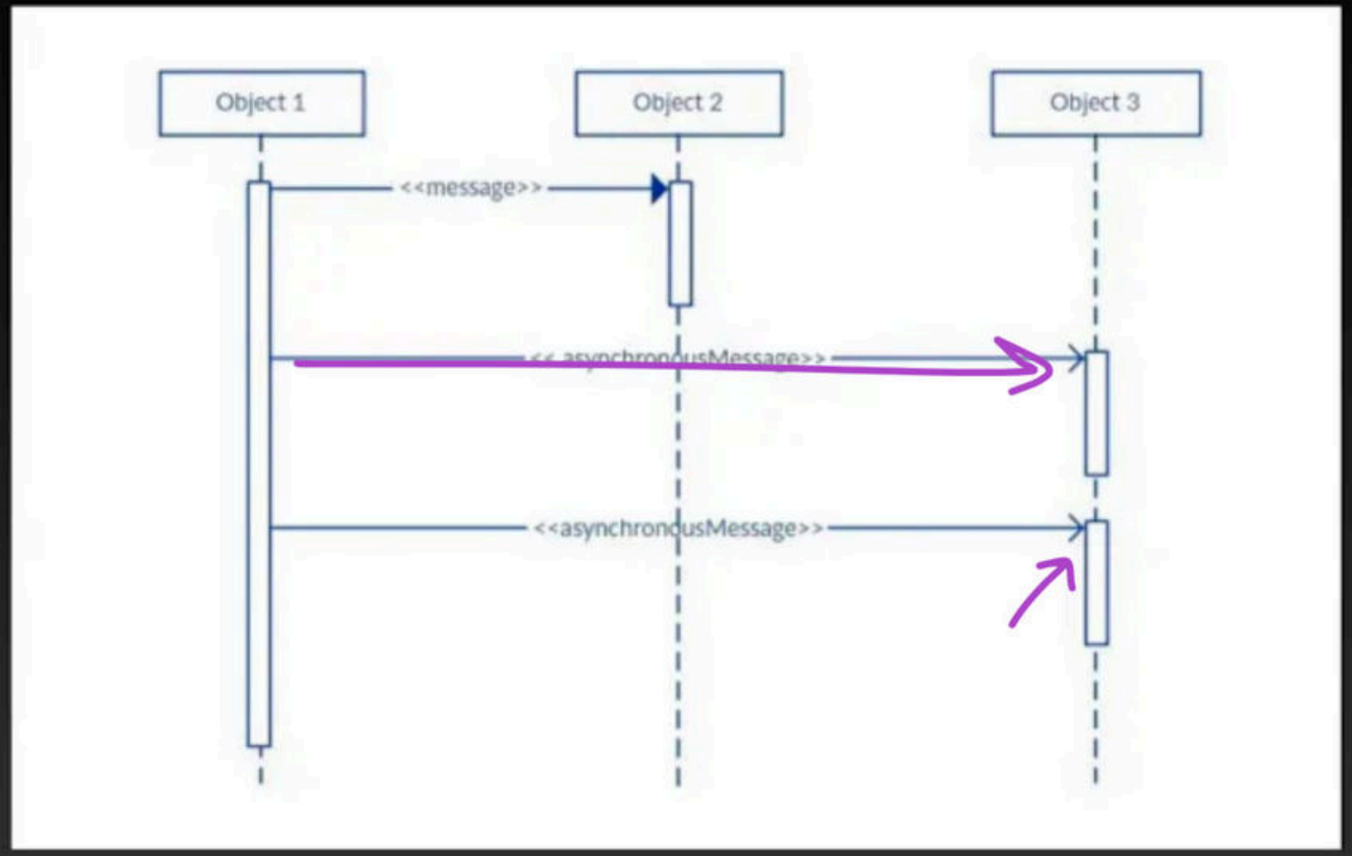
In sequence diagrams, a message is when two things talk to each other. This can happen when they send messages back and forth, start an action, or make a new thing or part of the system. Messages are shown as straight lines that can go from left to right, right to left, or even circle back to the same thing. We use different

- A synchronous message is when the sender sends a message and has to wait until it gets a reply before it can do anything else. We show these messages with a solid line and a solid arrow tip.

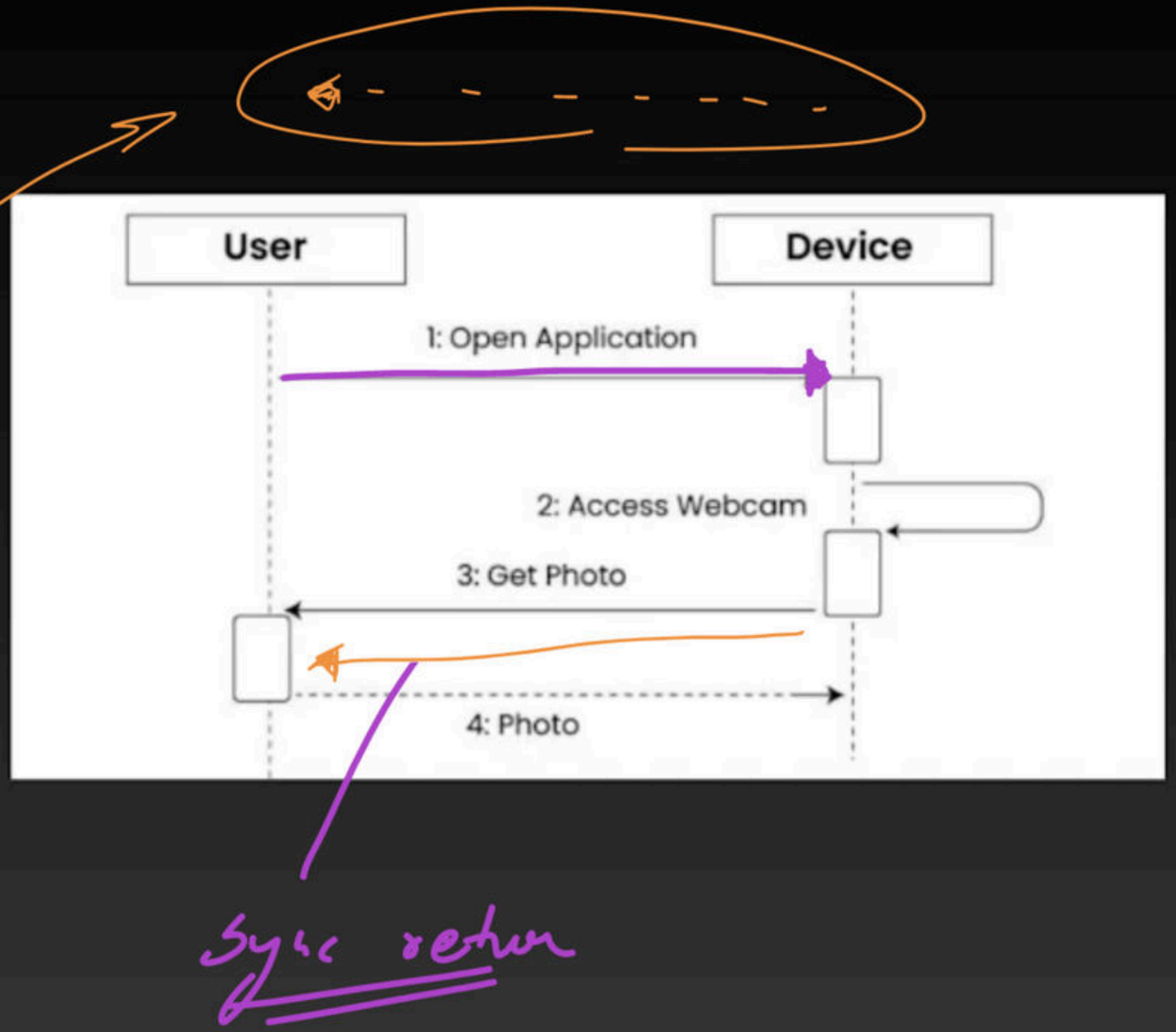




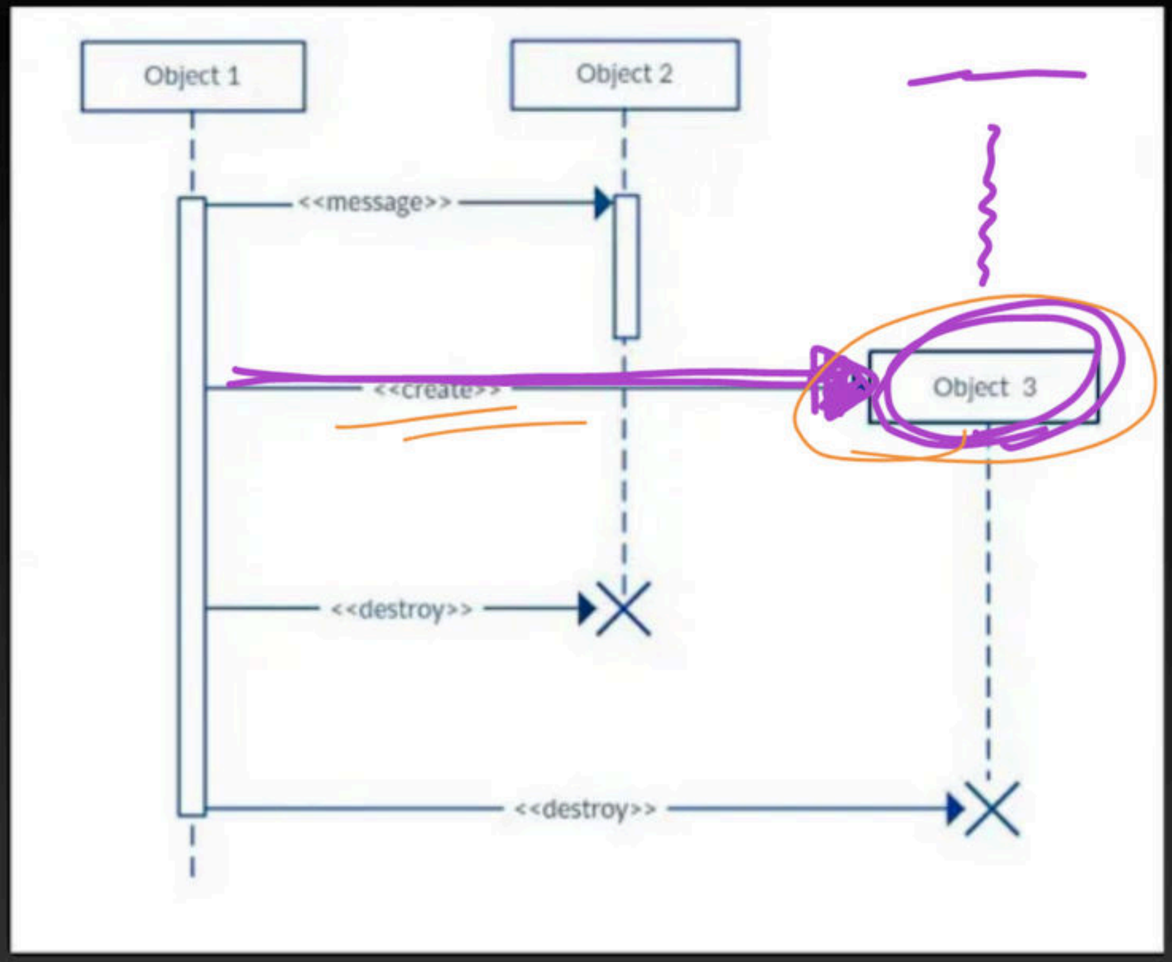
- An *asynchronous* message is when the sender sends a message but doesn't have to wait for an answer back. The sender can keep on sending messages to others. We draw these messages with a solid line that ends in an arrow that's not filled in.



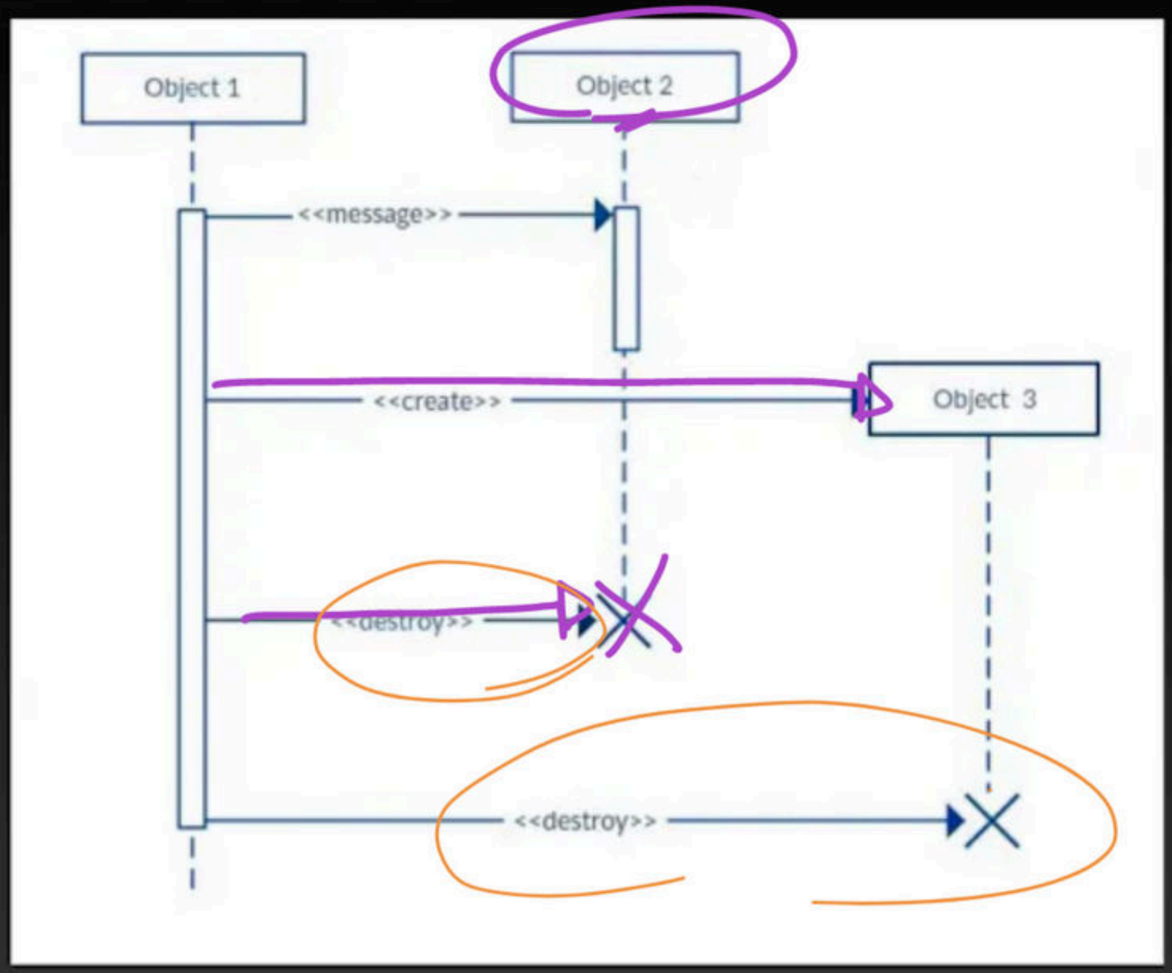
- A synchronous return is the reply you get back from a synchronous message. Whenever you send a synchronous message, you need to get one of these replies to show that the message was received and handled. We draw these replies with a dotted line that ends in a solid arrow tip.



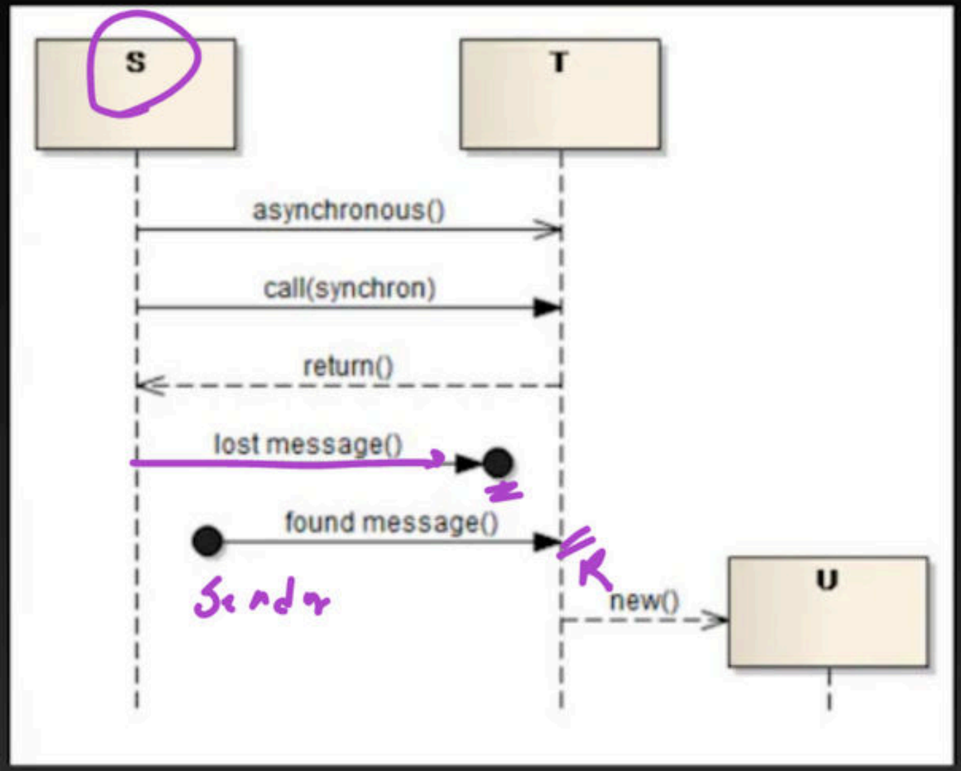
- A *create message* is used when a new thing is made during an interaction. This can happen because of a message or action. We show it like this:



- A *destroy message* is used when something is removed or ended during a series of events. This can happen because of a message or action, and it means the end of its timeline. We show it like this:



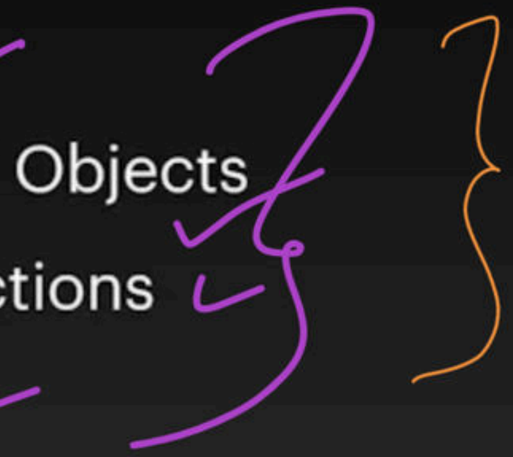
- A *lost message* is when a message starts from something but never makes it to where it was supposed to go. It looks like a message that just stops. A *found message* is when a message comes in, but we don't know who sent it. It looks like a message that arrives but doesn't seem to start from anywhere. We show lost messages with an arrow that ends in a circle and found messages with an arrow that begins with a circle.Some sequence diagrams use filled circles to represent the lost and found messages.



How to draw a Sequence Diagram ?

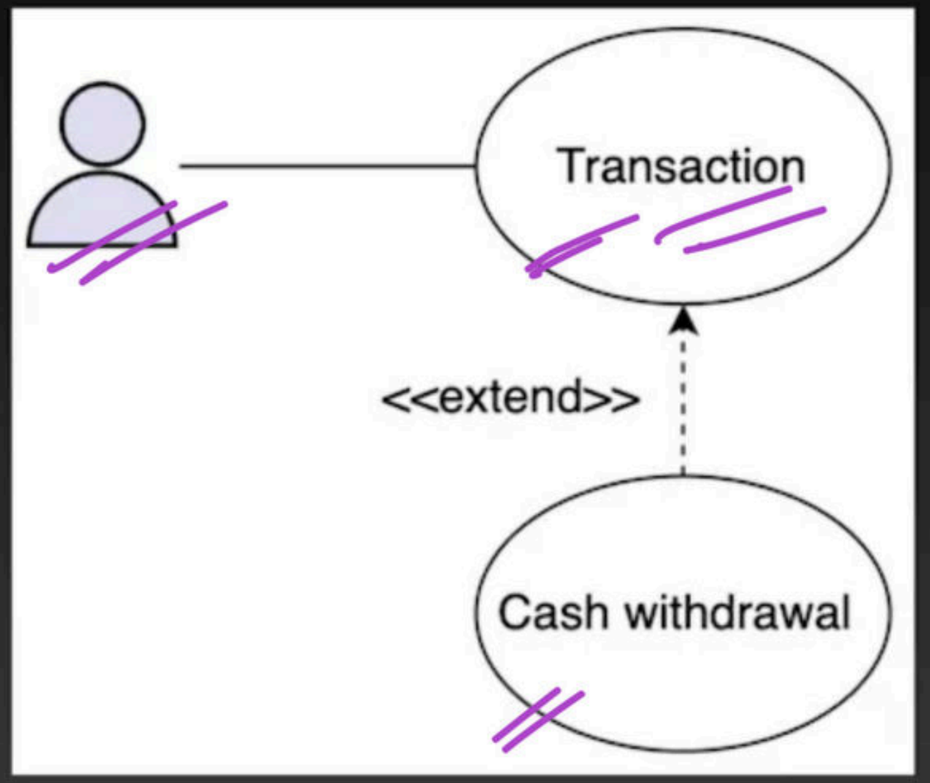
To make a clear sequence diagram, we have several steps to take. But first, remember that there's no single right way to make one. Different people might do it differently.

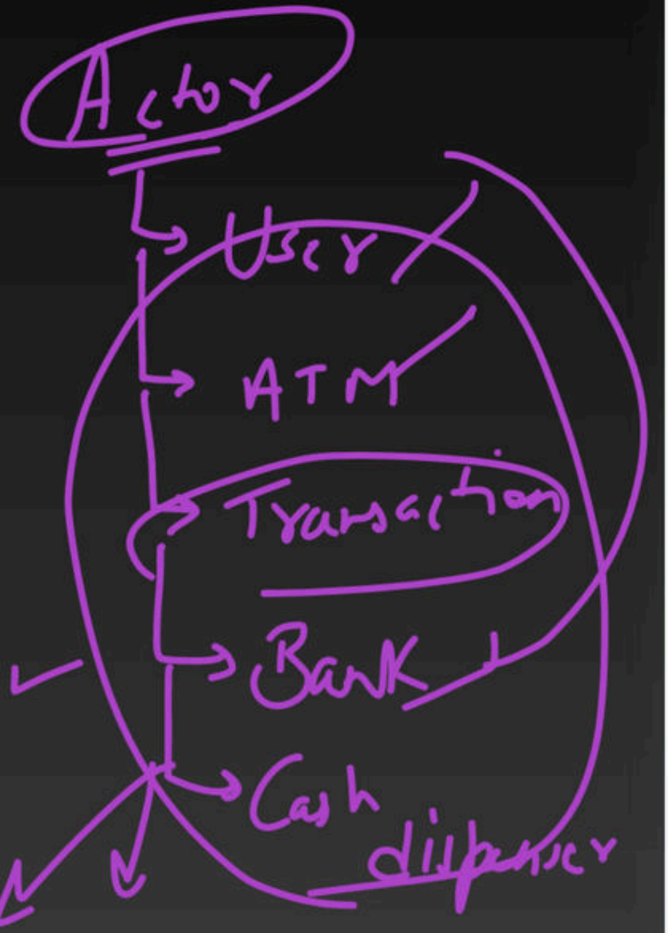
- Identify the Use-Case
- Identify the Actors and Objects
- Identify the order of Actions
- Create the diagram



Identify the Use Case

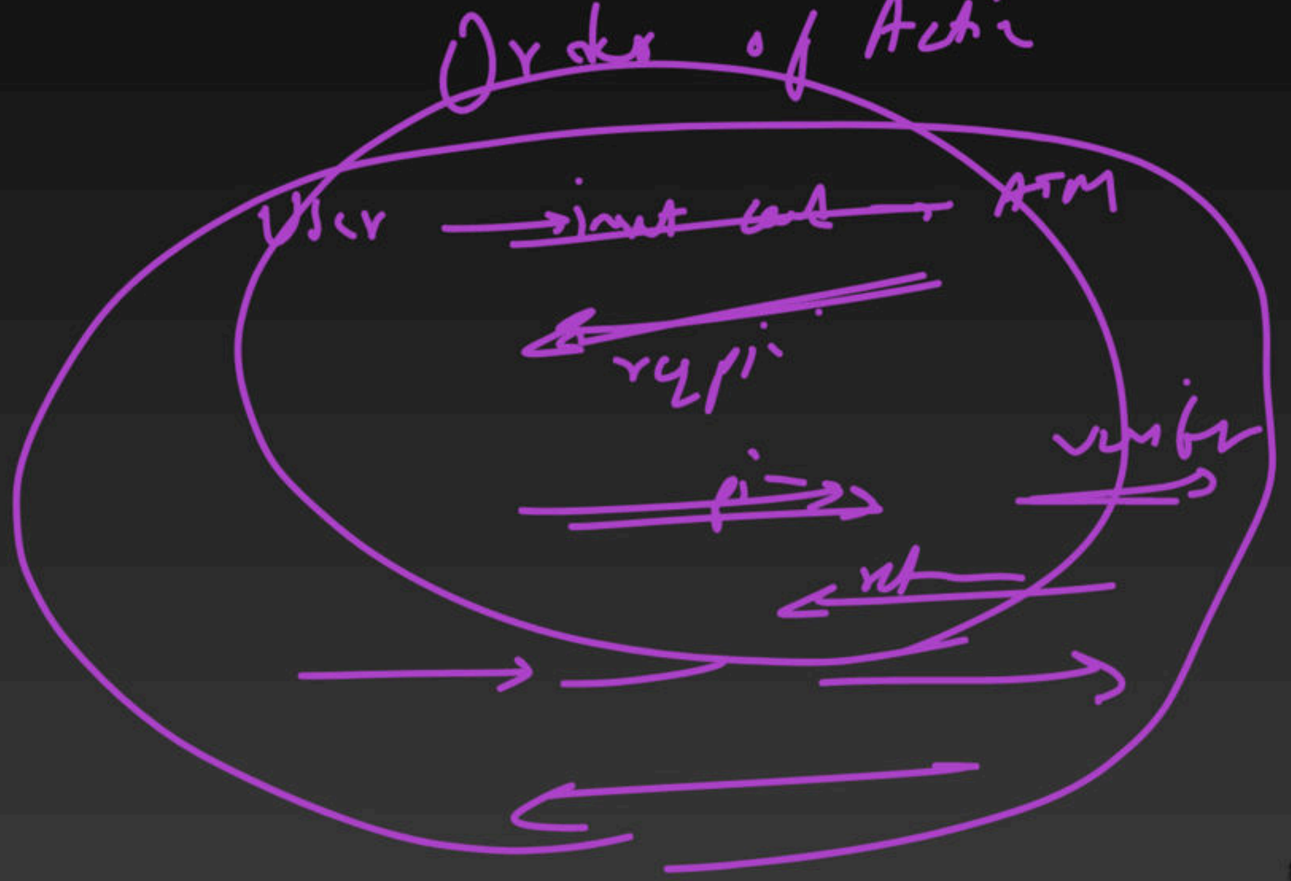
To begin with the sequence diagram, we need to understand our situation. The sequence diagram helps us show the steps in order, for that situation. Let's imagine a simple situation where a customer is taking out money from an ATM. Here's what that looks like:





Identify the Actors & Objects

- Now, we need to write down all the people and things that will take part in the steps. They are:
  - Customer
  - ATM
  - Transaction
  - Account
  - Cash dispenser



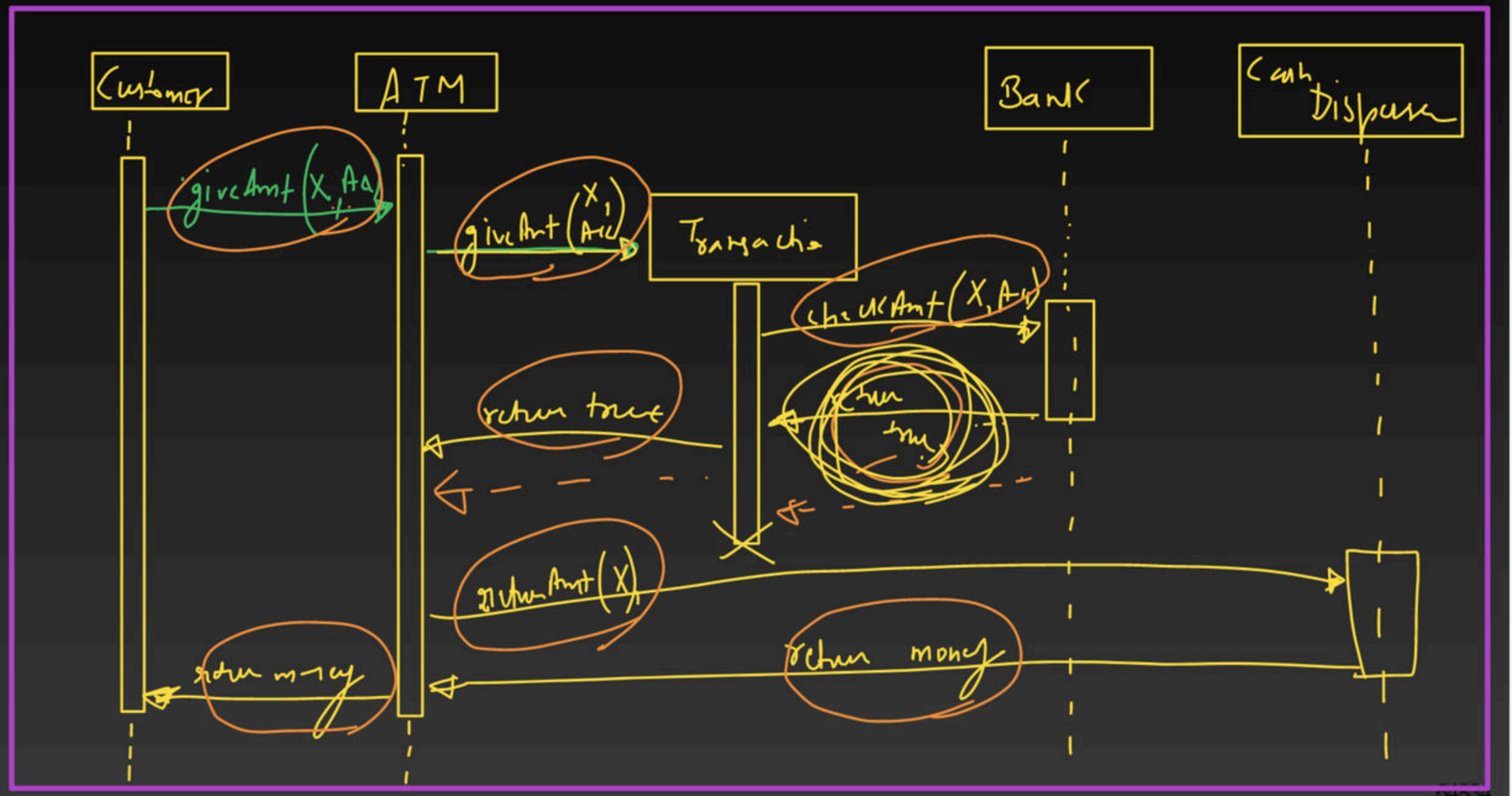
Identify the Order of Actions

- We know who's involved in getting cash from the ATM, but we need to figure out how they all work together. We should write down what each one does, step by step. Here's a simple way to understand their actions:
  - The customer asks the ATM for money, using their account info and how much they want.
  - The ATM starts the process to take out money from the account for the asked amount.
  - The transaction is checked to make sure the account can give that amount of money.
  - If the amount is okay, the account agrees to the transaction.
  - The ATM tells the cash dispenser to give out the needed money.
  - The cash dispenser confirms it's giving out the cash.
  - The ATM tells the customer to take the money.

Create a Diagram:

We've named all the things and people taking part in this activity and the steps they'll take. To make the sequence diagram, we need to connect these steps to the kinds of messages they'll send to each other. Here's how we could draw a diagram for taking out cash:

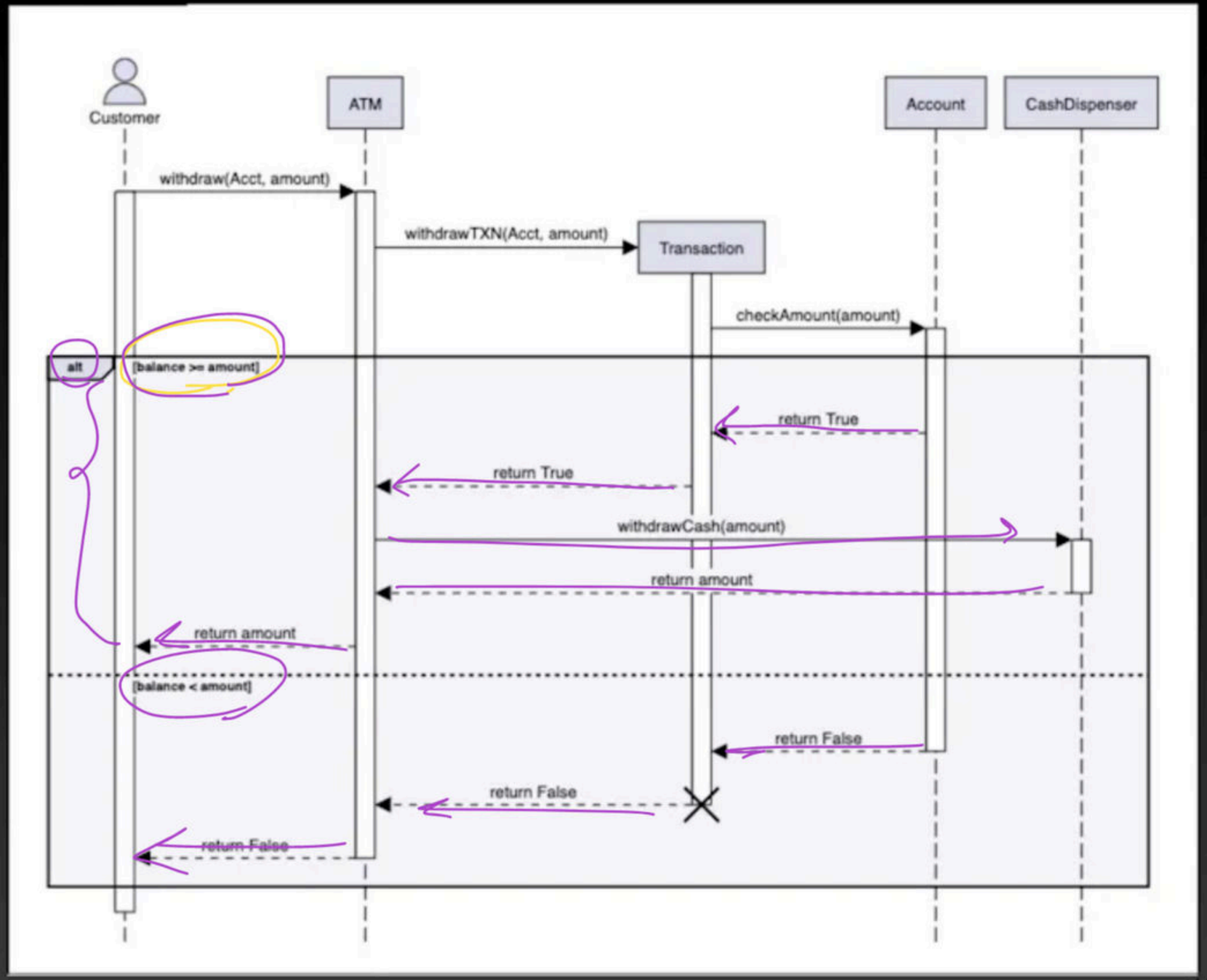
- Let's do it



Fragment frame:

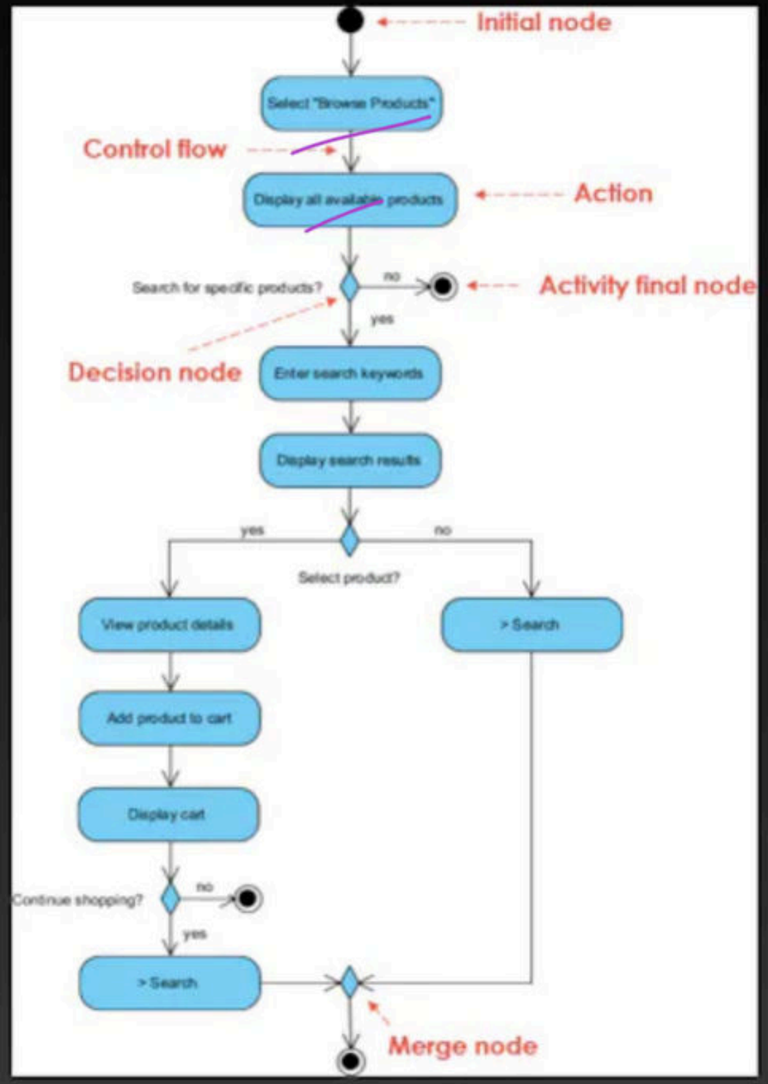
- We want our sequence diagram to be easy to understand, without too much detail all at once. But sometimes, we need to show more complicated things like "what if" scenarios or repeating actions. Sequence diagrams have a special part called "sequence fragment" for this. We can tell what a fragment does by a label in its top left corner. Here are some common types:
  - "Alternative" (alt): This is like an "if-else" in coding. It splits the fragment into parts, and depending on a certain condition, one part happens.
  - "Option" (opt): This is like an "if" condition. The fragment only happens if a certain condition is true. If not, it's skipped, and the diagram goes on.
  - "Loop" (loop): This is for things that repeat. The fragment keeps going over and over until a condition is met.
  - "Reference" (ref): This is for when diagrams get too big. It lets us refer to another part of the diagram, so we don't have to draw everything again.

- In the *diagram*, we use something called an "alternatives operator" that splits the sequence into two parts. The first part happens if a certain condition is true. If not, then the second part happens instead. In this case, the condition is checking if the account has enough money compared to what the customer wants to take out. If the account has enough money, the first part happens; if not, the second part does.



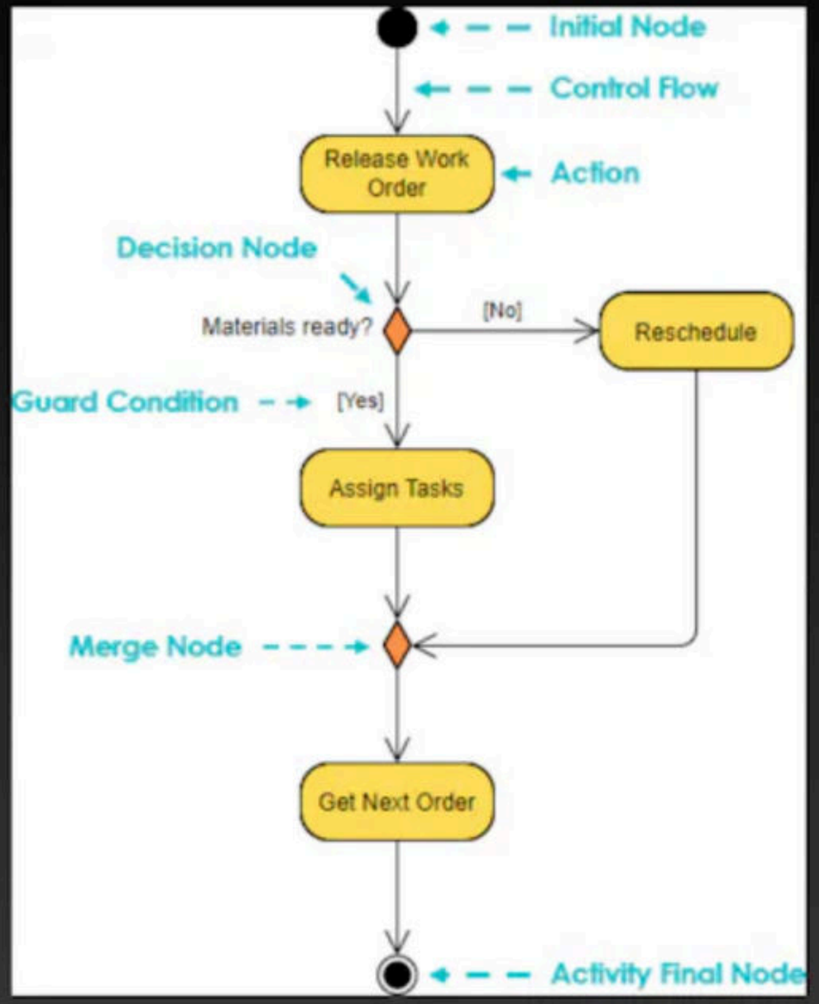
Activity Diagrams:

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



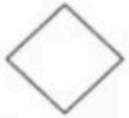
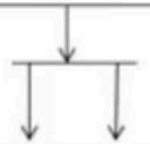
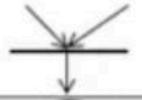
Why activity diagrams ?

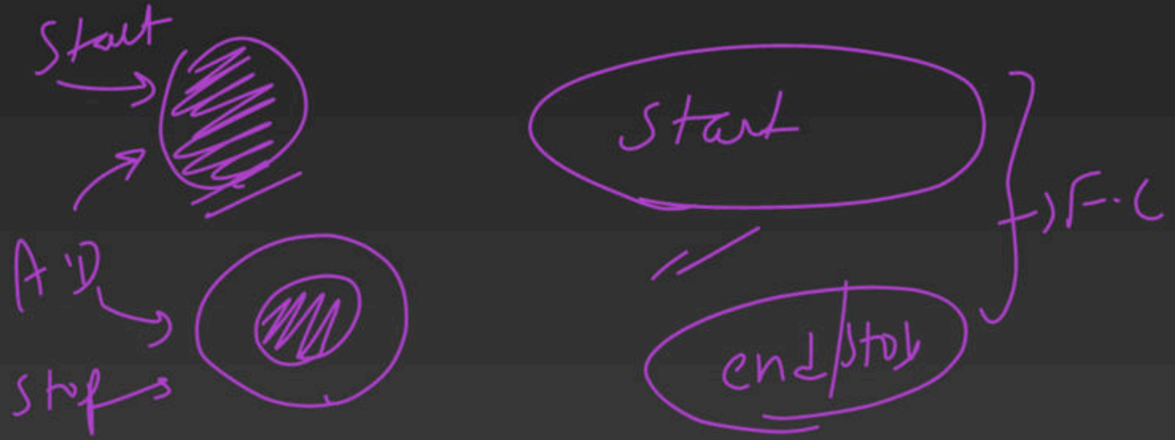
Activity diagrams are another type of diagram we use to show the order of events in a system, just like the other diagrams we talked about. But, activity diagrams are a bit different because they focus on how messages move from one activity to the next. They might look like flowcharts, but they can show a lot more, like things happening at the same time, one after the other, or even splitting into different paths.



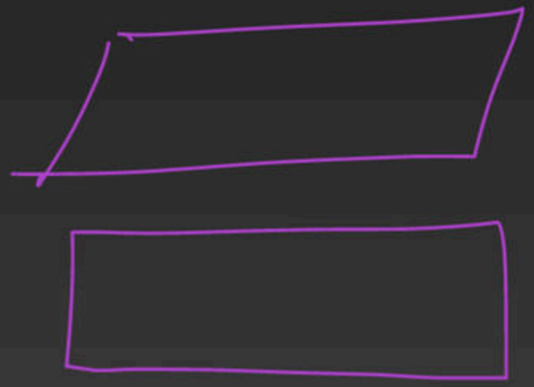
Elements of Activity diagrams:

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

Sr. No	Name	Symbol
1.	Start Node	
2.	Action State	
3.	Control Flow	
4.	Decision Node	
5.	Fork	
6.	Join	
7.	End State	

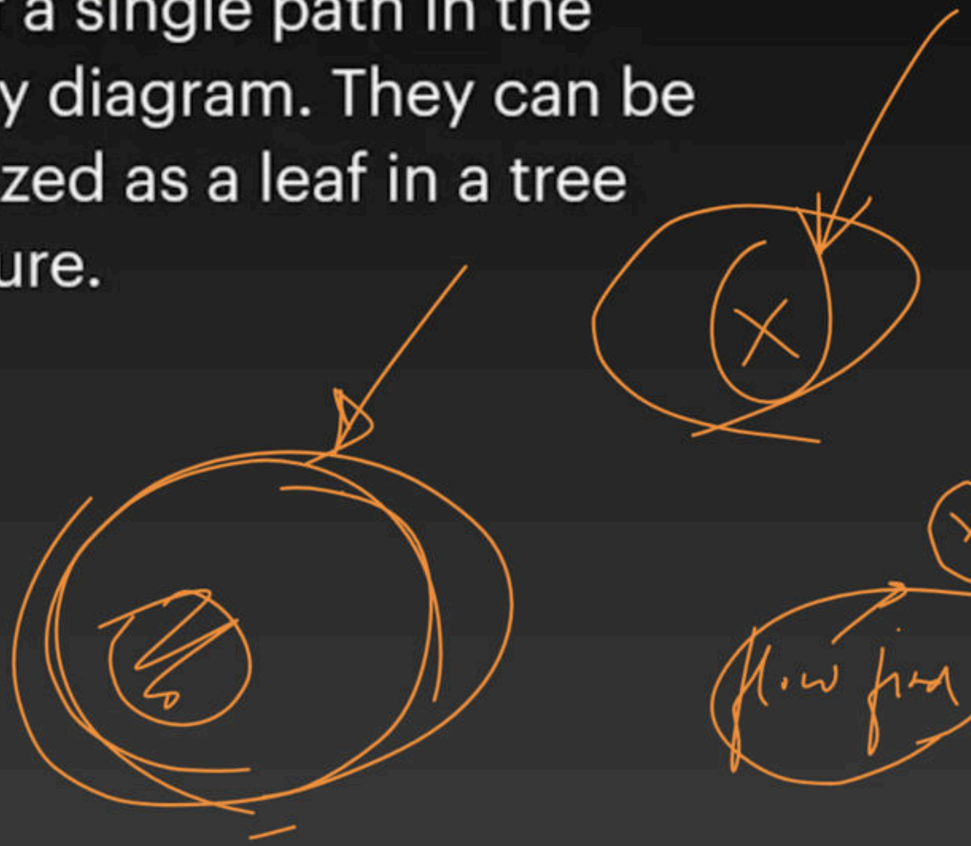


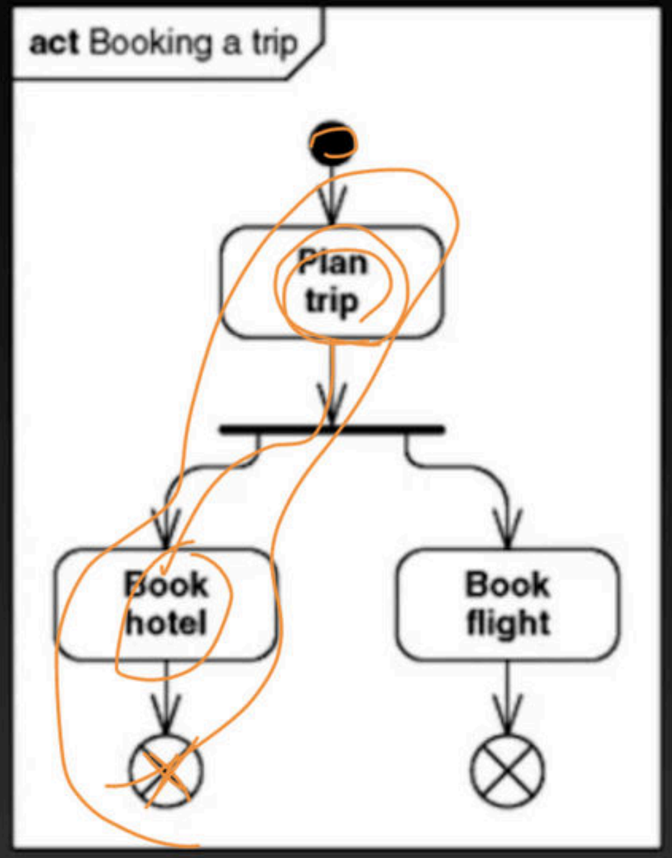
- **Action**: These are the main building blocks of an activity diagram and are used to show the activities that a modeled process is made of.



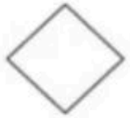
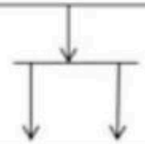
Sr. No	Name	Symbol
1.	Start Node	
2.	Action State	
3.	Control Flow	
4.	Decision Node	
5.	Fork	
6.	Join	
7.	End State	

- *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.

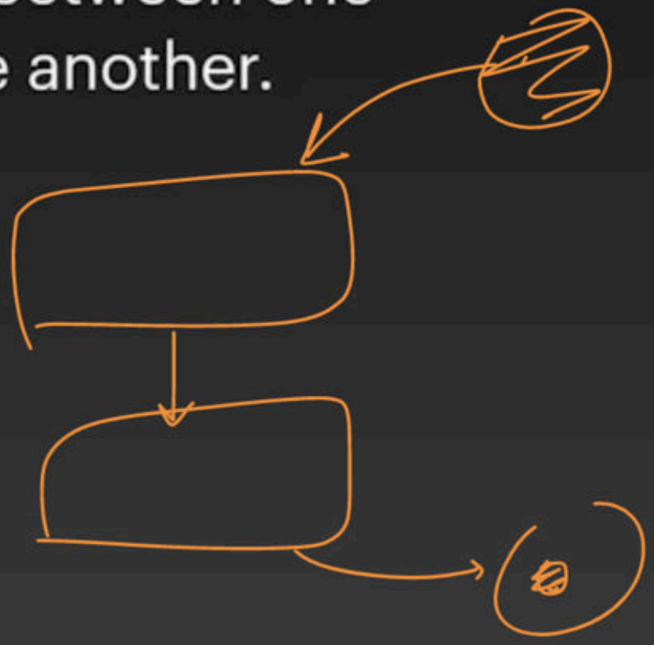




- *Activity final*: This represents the end of all the activities in the activity diagram.

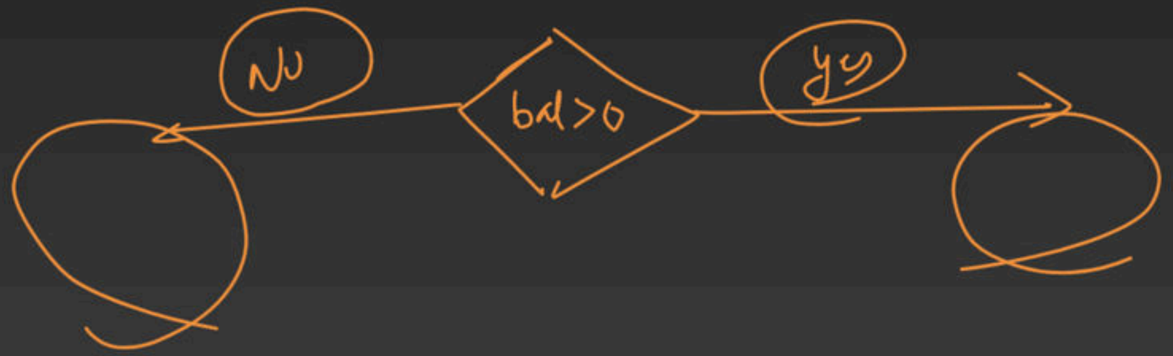
Sr. No	Name	Symbol
1.	Start Node	
2.	Action State	
3.	Control Flow	
4.	Decision Node	
5.	Fork	
6.	Join	
7.	End State	

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



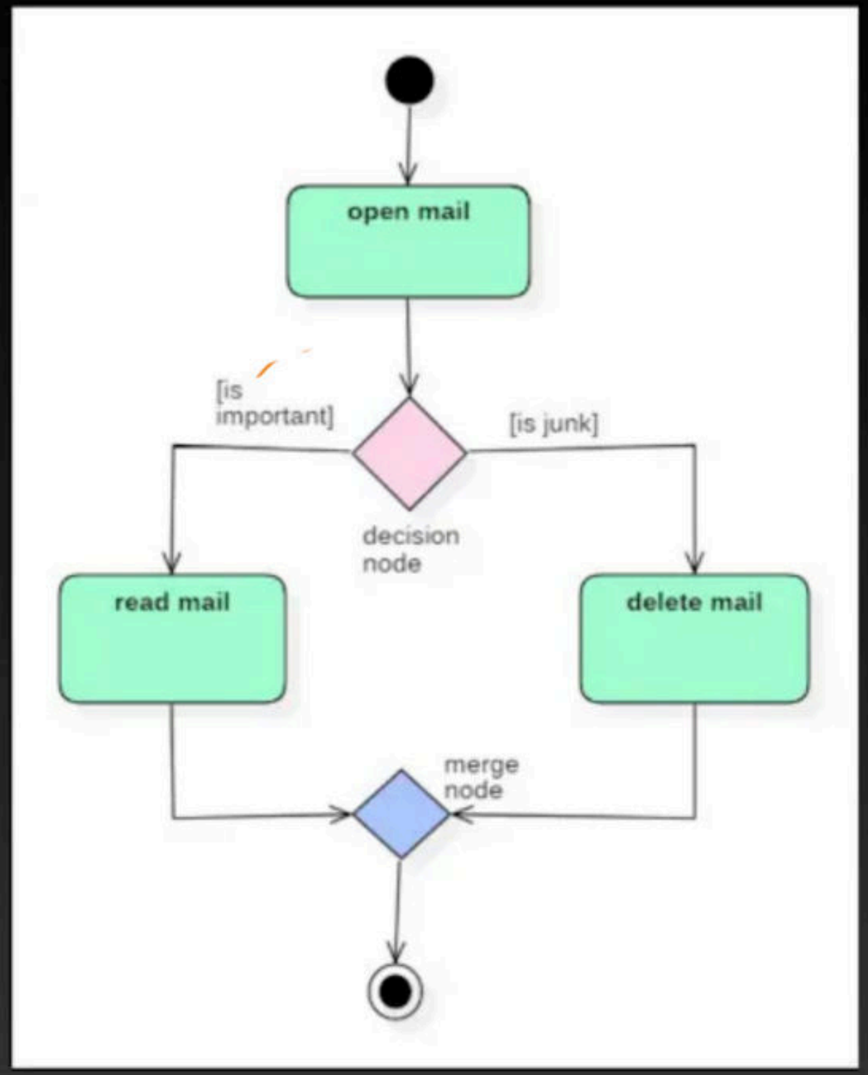
Sr. No	Name	Symbol
1.	Start Node	
2.	Action State	
3.	Control Flow	
4.	Decision Node	
5.	Fork	
6.	Join	
7.	End State	

- *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.

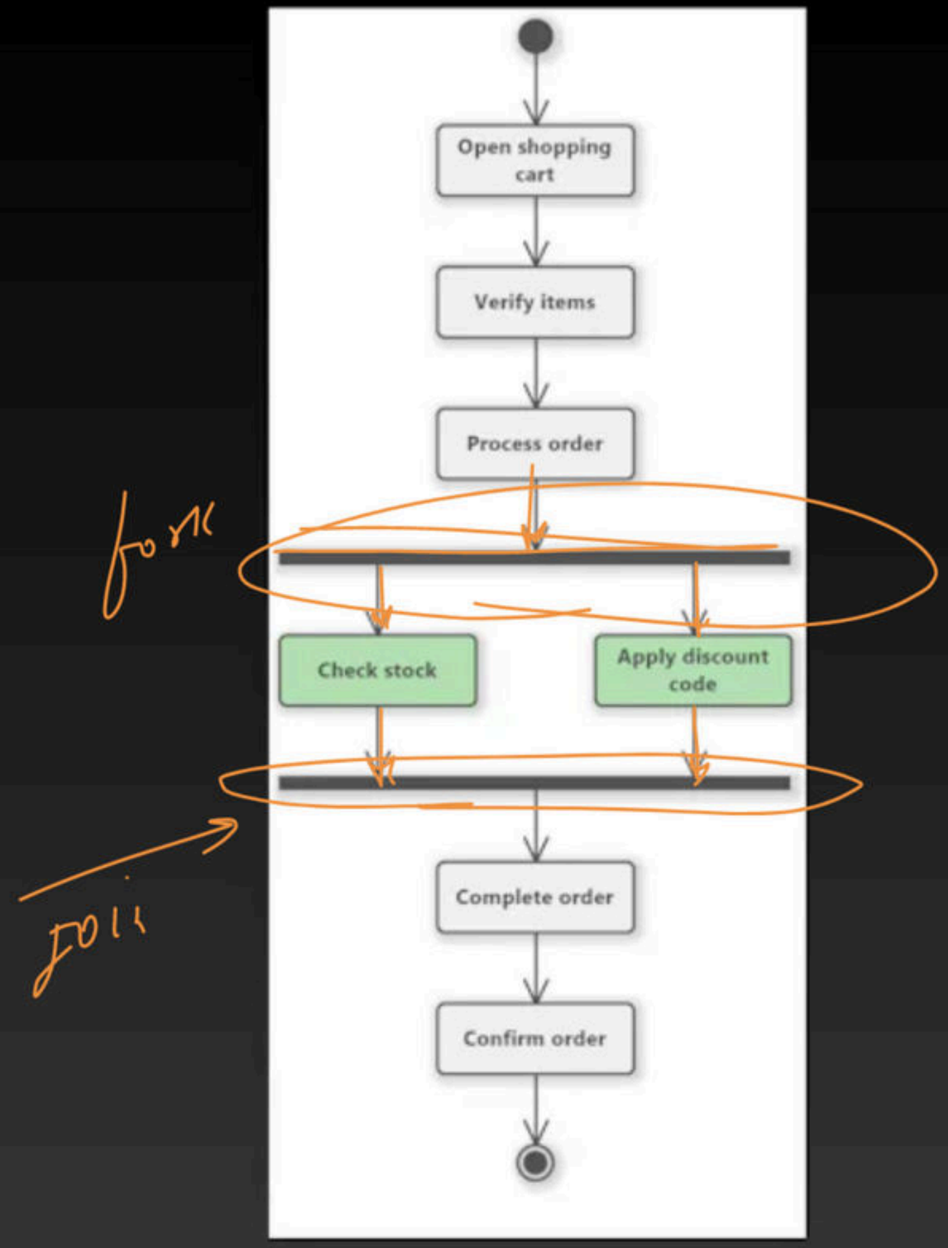


Sr. No	Name	Symbol
1.	Start Node	
2.	Action State	
3.	Control Flow	
4.	Decision Node	
5.	Fork	
6.	Join	
7.	End State	

- *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.



- *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.



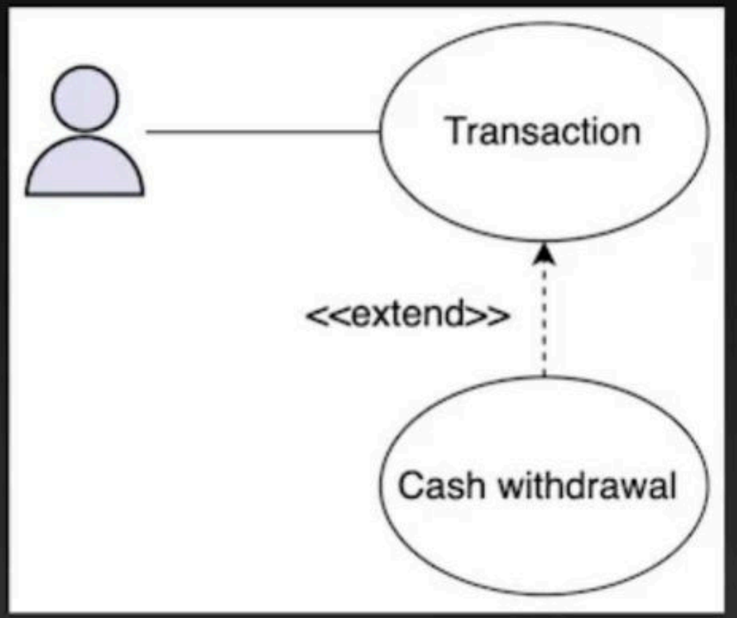
Steps to draw Activity Diagram ?

- Determine the actions of the system
- Finding the flow of each activity
- Create the diagram

Determine the actions of the system

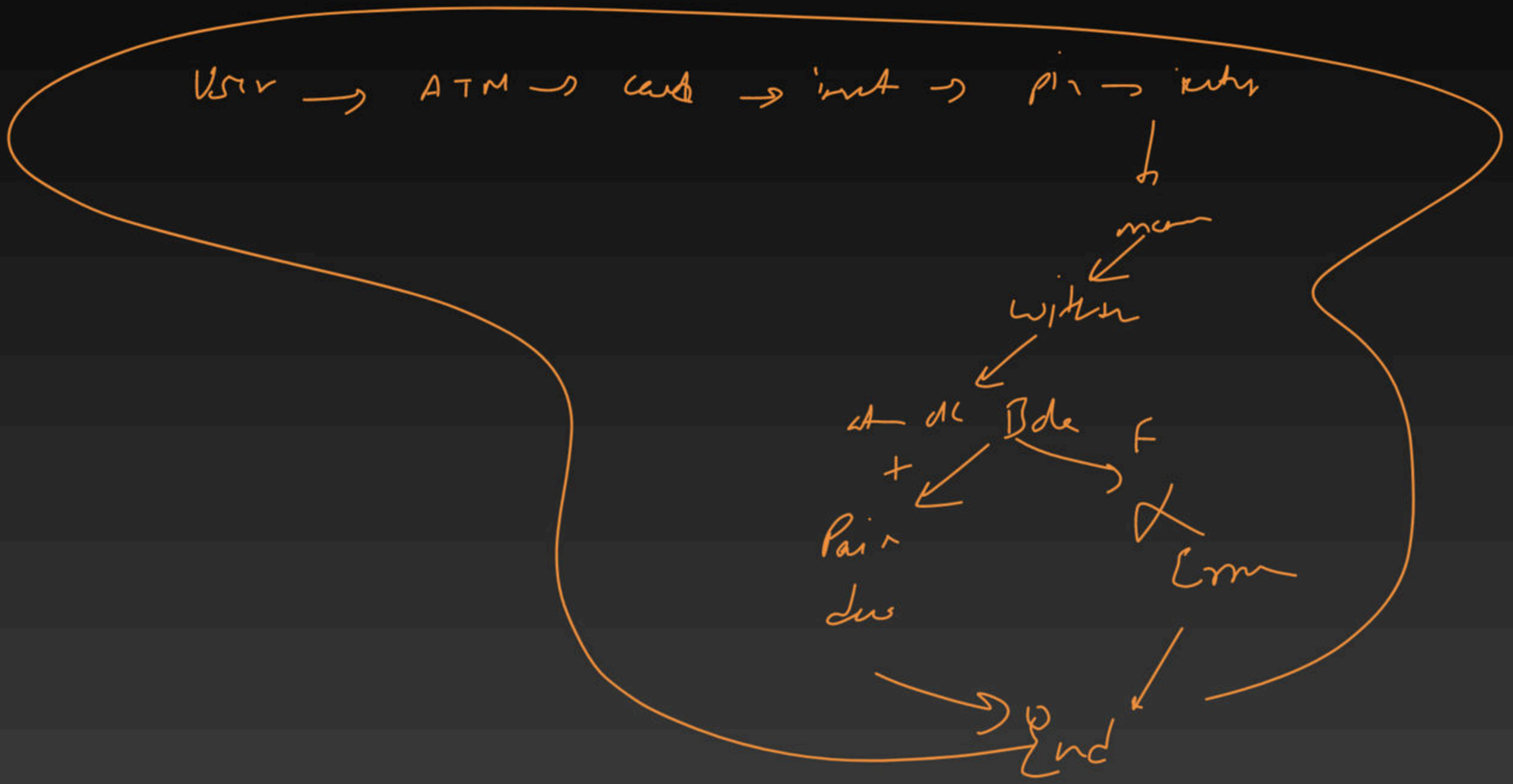
Here, we need to understand each step and how they connect with the rest in the system. Using examples, like stories of what people want to do, can help make this clearer. Let's use the same story as before, where a customer is taking out money from an ATM.

- Figure out who is involved and what they do, then write down all the people and things that take part in the whole process. They are:
  - Customer
  - ATM
  - Transaction
  - Account
  - Cash dispenser



Find flow of each activity

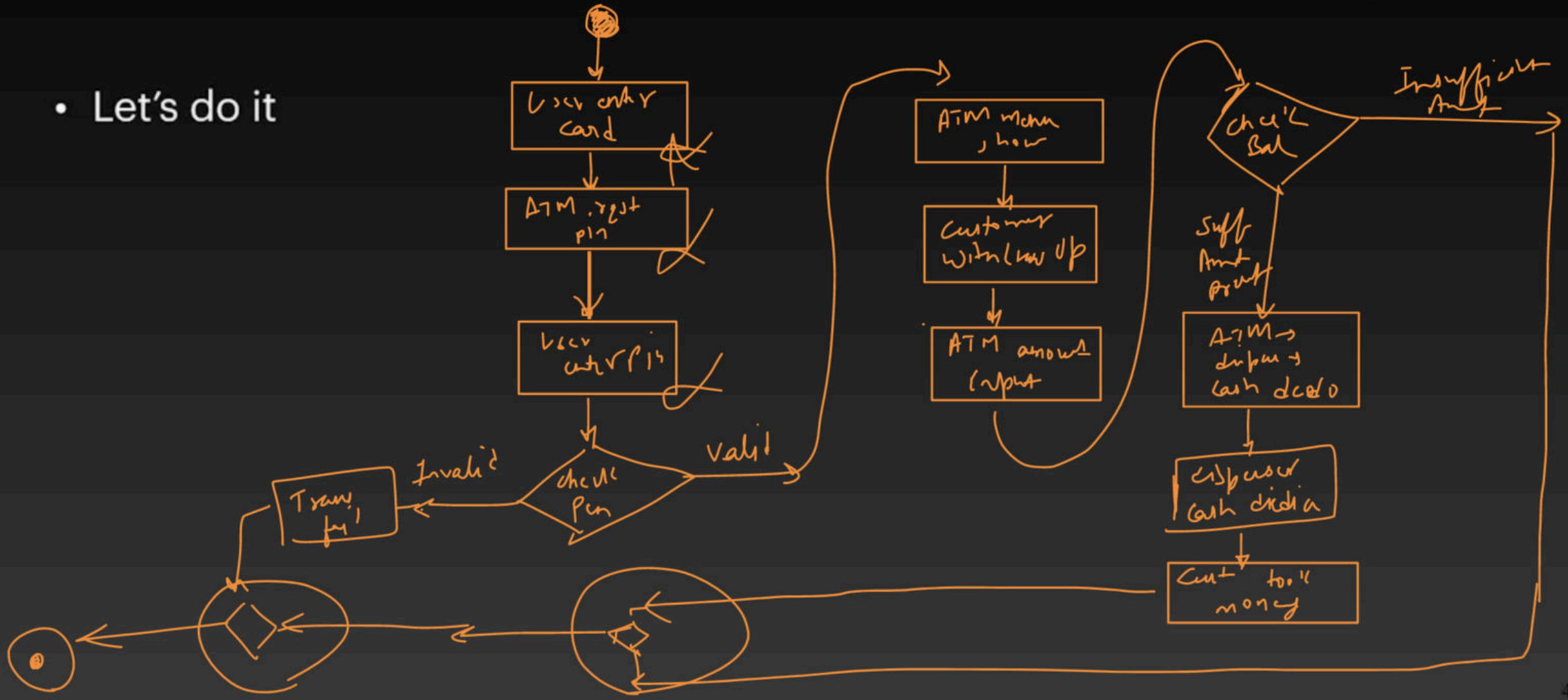
First, we figure out the steps and their order. We also check how some steps happen at the same time as others. We look at what conditions cause certain things to happen. And we note if some steps can only start after the one before is finished.



Create Diagram

Now, we will use these steps to build a sample activity diagram.

- Let's do it



HomeWork

Create activity and sequence diagrams for the same set of questions given in the previous class