SOFTWARE ENGINEERING LAB

EXERCISE – 3

TOPIC – 1

CONCEPTUAL MODEL OF UML

UML (Unified Modeling Language)

UML, or Unified Modeling Language, is a way to draw diagrams that show how software systems work. It's like drawing a map or blueprint to help people understand what's going on inside a program. Just like how an architect uses blueprints to plan a house, software designers use UML to plan their software before they start building it with code.

CASE Tools for UML

CASE tools (Computer-Aided Software Engineering tools) are special programs that help people draw these UML diagrams. They make it easy to show how the software will work. Some popular CASE tools are:

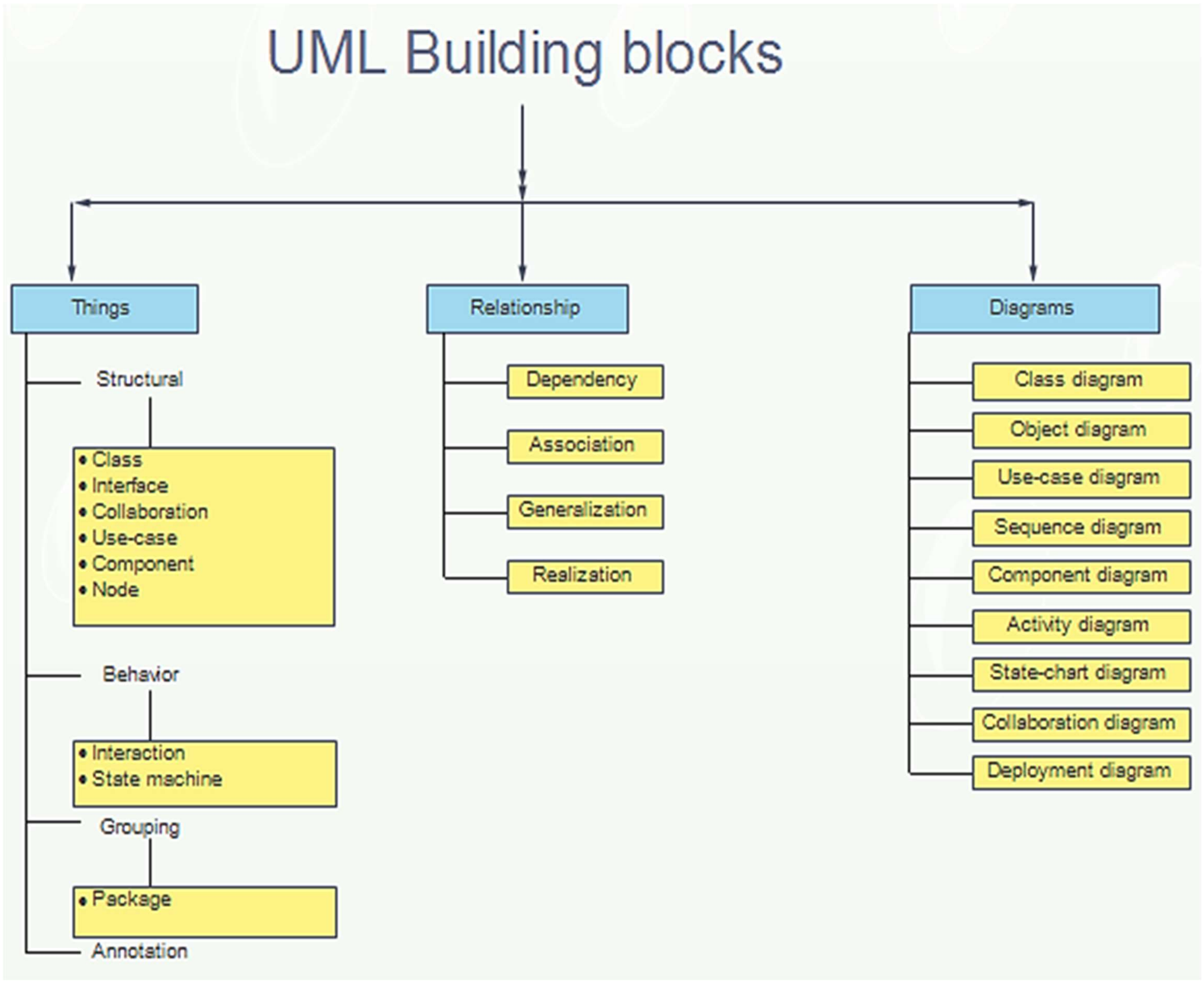
- **Rational Rose** – A well-known tool for drawing UML diagrams.
- **Enterprise Architect** – Helps create and organize diagrams.
- **Microsoft Visio** – A diagramming tool that can also make UML diagrams.
- **Lucidchart** – A web tool for drawing diagrams.
- **StarUML** and **ArgoUML** – Free tools for making UML diagrams.
- **Visual Paradigm** – Offers many features to help design software.
- **Balsamiq**, **Creately**, and **SmartDraw** – Simple tools to make diagrams quickly.
- **Umbrella** – Another tool that helps make UML diagrams.

UML Building Blocks

UML helps us think about how a software system works by using diagrams.

It focuses on three key points:

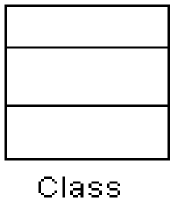
1. **Things/Objects**: The parts of the system, like people, data, or tasks.
2. **Relationships**: How these parts connect and work together.
3. **Diagrams**: Pictures that show us how everything fits together.

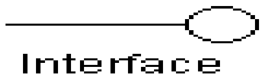


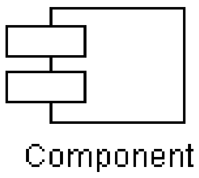
Parts of UML (Things)

In UML, the "things" are the parts that make up a system. These parts are divided into groups:

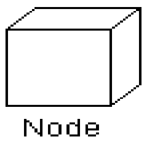
1. **Structural Things (Parts That Don't Change)** These describe the solid, unchanging parts of the system, like the structure of a building.

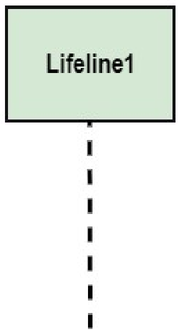
   - **Class**: A group of objects with similar features.
     - *Example: A "Car" class has properties like make, model, and color.*
     - *Symbol:* 

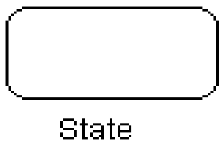
   - **Interface**: A list of rules or actions that certain objects must follow.
     - *Example: A "Drivable" interface makes sure that any class that uses it, like a Car, has a "drive" function.*
     - *Symbol:* 

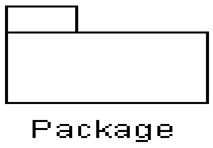
   - **Component**: A piece of the system that performs a specific job.
     - *Example: A "Payment" component handles all the payments in the system.*
     - *Symbol:* 

   - **Node**: A place where the software runs, like a server.
     - *Example: A "Database Server" is a node where customer information is stored.*

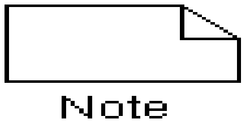
- *Symbol:* 

2. **Behavioral Things (How Things Act or Change)** These describe how the system behaves or changes over time, like how a door can open and close.
   - **Interaction**: Shows how different parts of the system talk to each other.
     - *Example: A customer asks for information from the system, and the system sends back the details.*
     - *Symbol:* 

   - **State Machine**: Shows the different states something can be in.
     - *Example: A door can be in either "Open" or "Closed" state.*
     - *Symbol:* 

3. **Grouping Things** These organize similar parts of the system into groups.
   - **Package**: Groups together related parts.
     - *Example: A "User Management" package might contain parts that deal with users, like login, registration, and permissions.*
     - *Symbol:* 

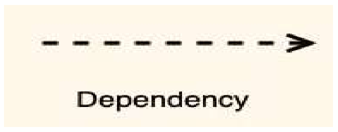
4. **<u>Annotational Things</u>** These are notes or comments that give extra information about the system.

- **Annotation**: A comment added to explain something in the diagram.
  - *Example: A note that explains why a particular feature is needed.*
  - *Symbol:* 

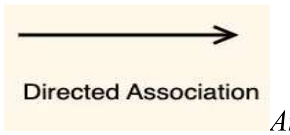
<u>Relationships in UML</u>

UML diagrams also show how the different parts of a system are connected. These connections are called **relationships**.

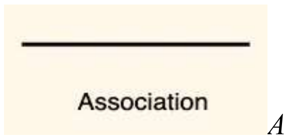
Some common types of relationships are:

- **<u>Dependency</u>**: When one thing needs another thing to work.
  - *Example: A "Car" class needs an "Engine" class to function properly.*
  - *Symbol:* 

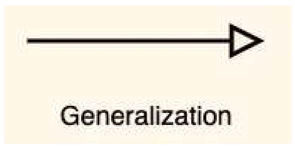
- **<u>Association</u>**: A simple connection between two things.
  - *Example: A teacher teaches a student, and the student learns from the teacher. They are associated with each other.*
  - *Symbol:*

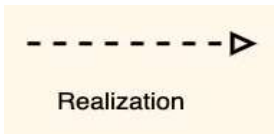


An "Order" depends on a "Product" to exist because an order is placed for products.

- 

A "Doctor" can treat many "Patients," and a "Patient" can see many different "Doctors." Both have a connection (doctor-patient relationship), but they can exist separately.

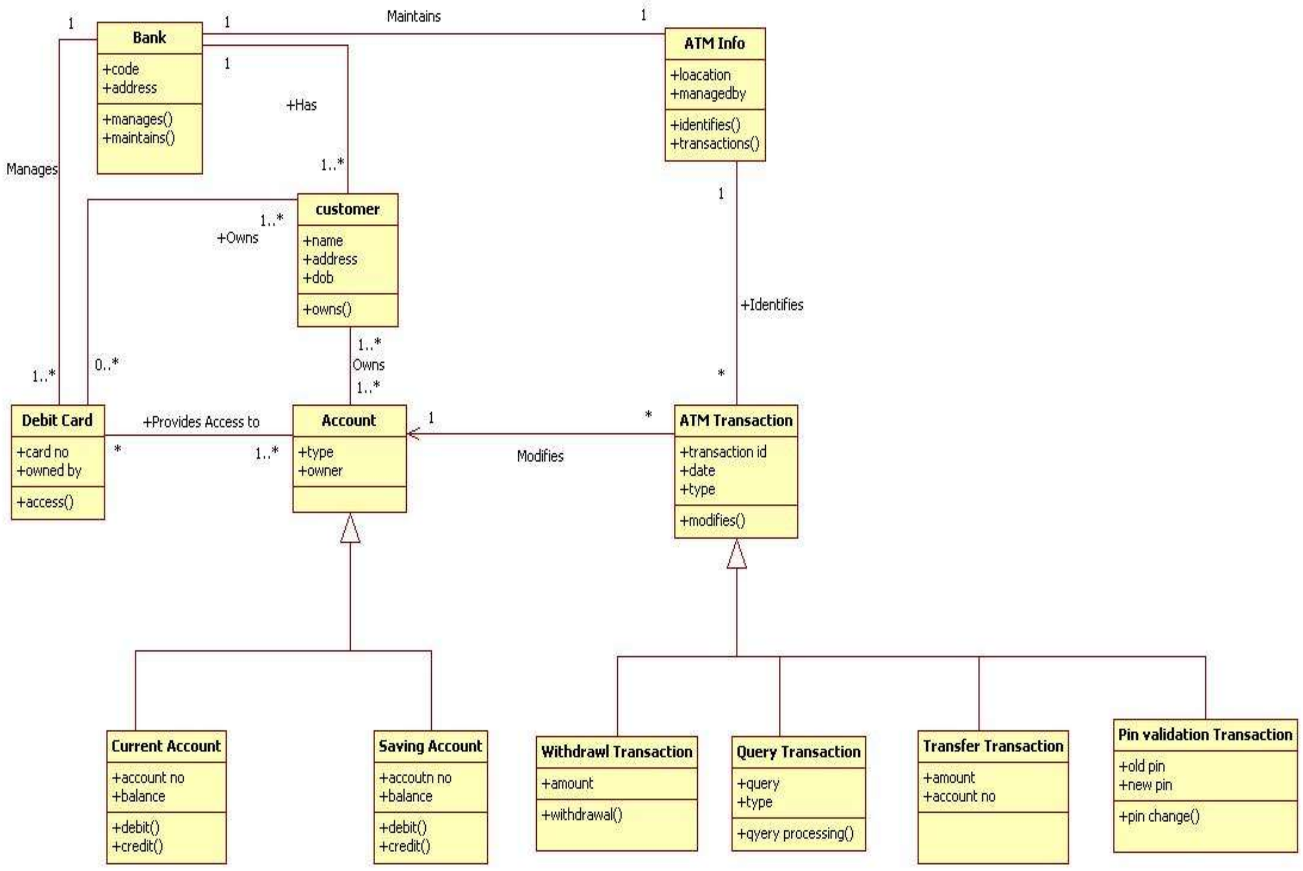
- **<u>Generalization</u>**: This shows inheritance, where one thing is a more general version of another.
  - *Example: A "Car" is a general category, and "SUV" and "Sedan" are more specific types of cars.*
  - *Symbol:* 

- **<u>Realization</u>**: When one thing follows the rules set by another (usually an interface).
  - *Example: A "Bird" class follows the "Flyable" interface by creating a "fly" function.*
  - *Symbol:* 

Types of UML Diagrams

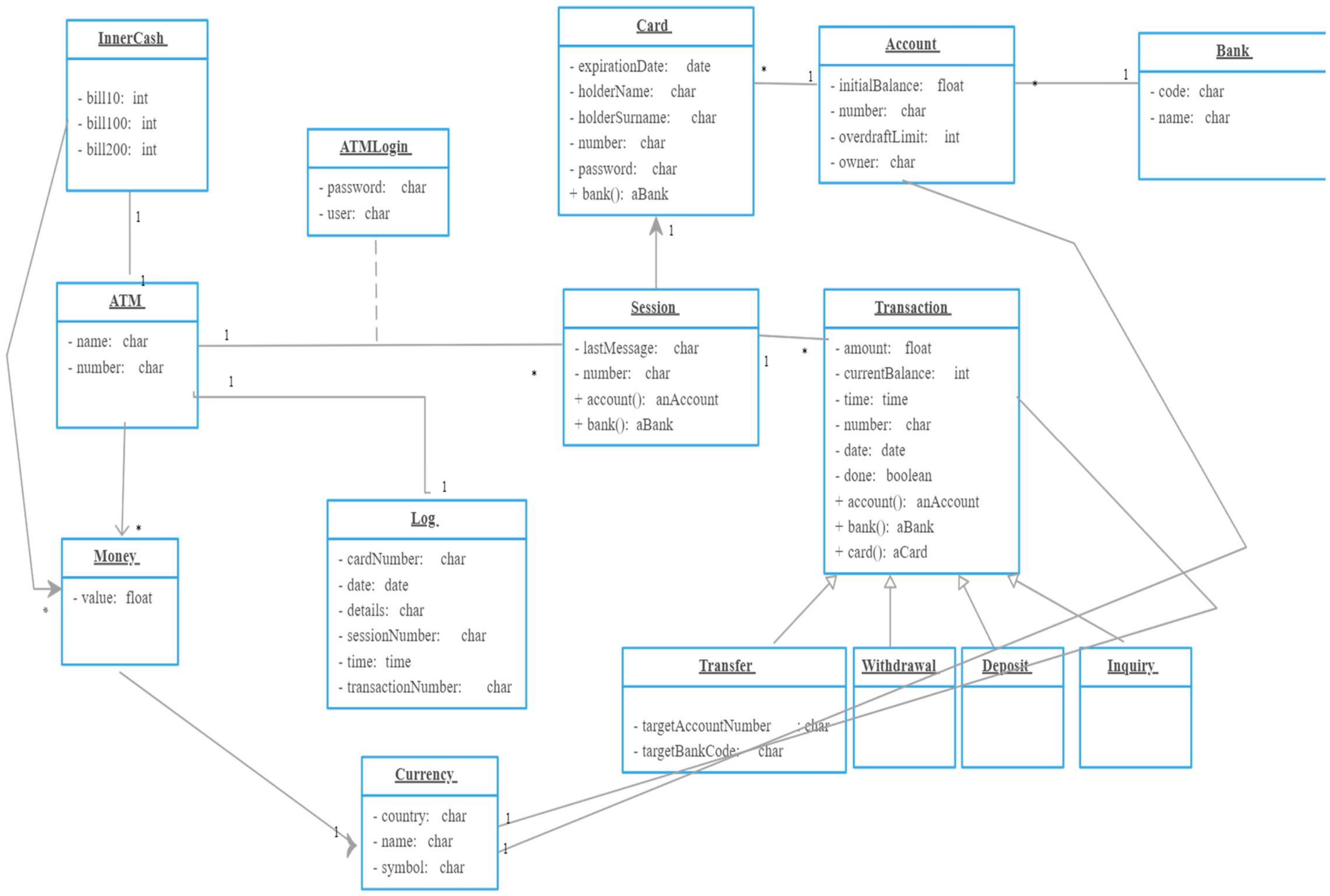
UML uses different types of diagrams to show various aspects of a system. Each diagram has its own way of explaining how the system works.

1. **Class Diagram**
   - Shows the different classes (types of objects) and how they relate to each other.
   - *Example: A "Bank Account" class connected to a "Card" class that handles money transactions.*
   - ***Example Class Diagram for an ATM system***



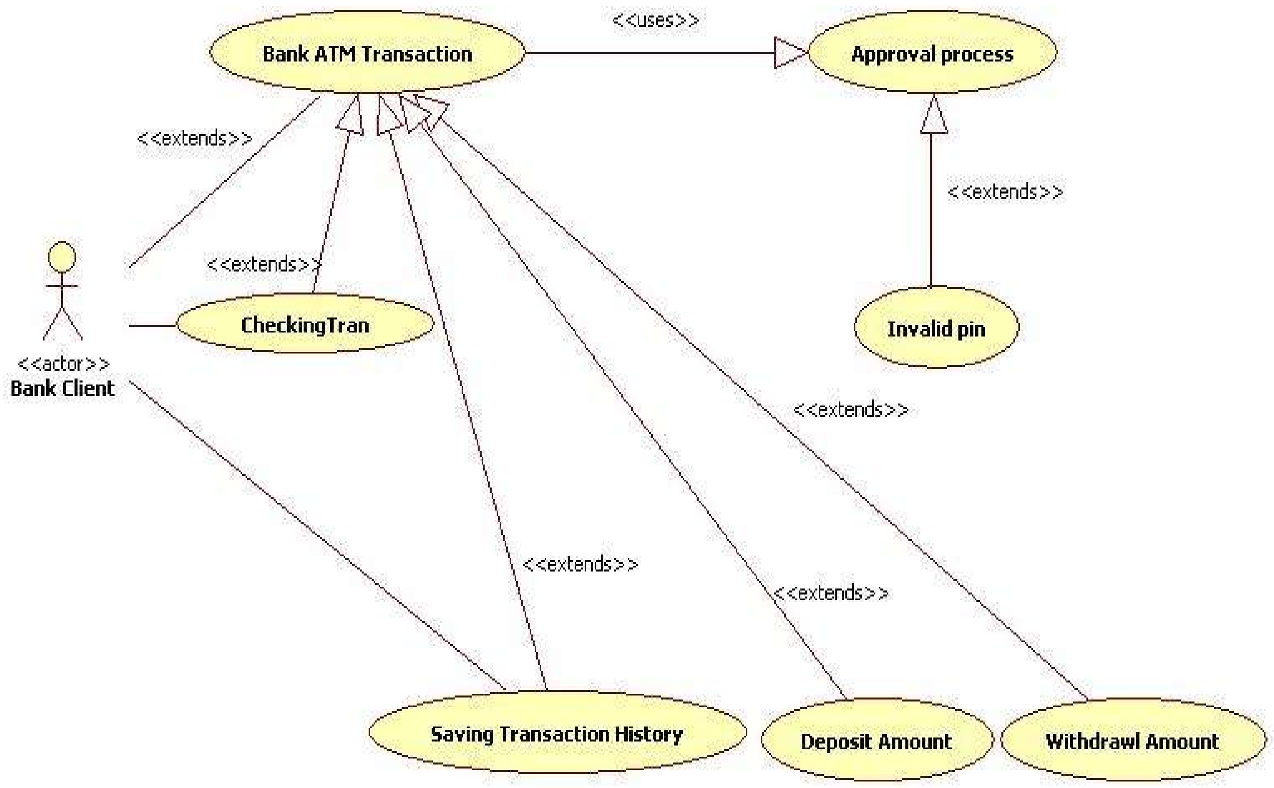
2. Object Diagram

- A snapshot of a specific instance in the system at a certain moment.
- *Example: A "Debit Card" showing details for one specific user, like card number and expiration date.*
- ***Example Object diagram for the ATM system***



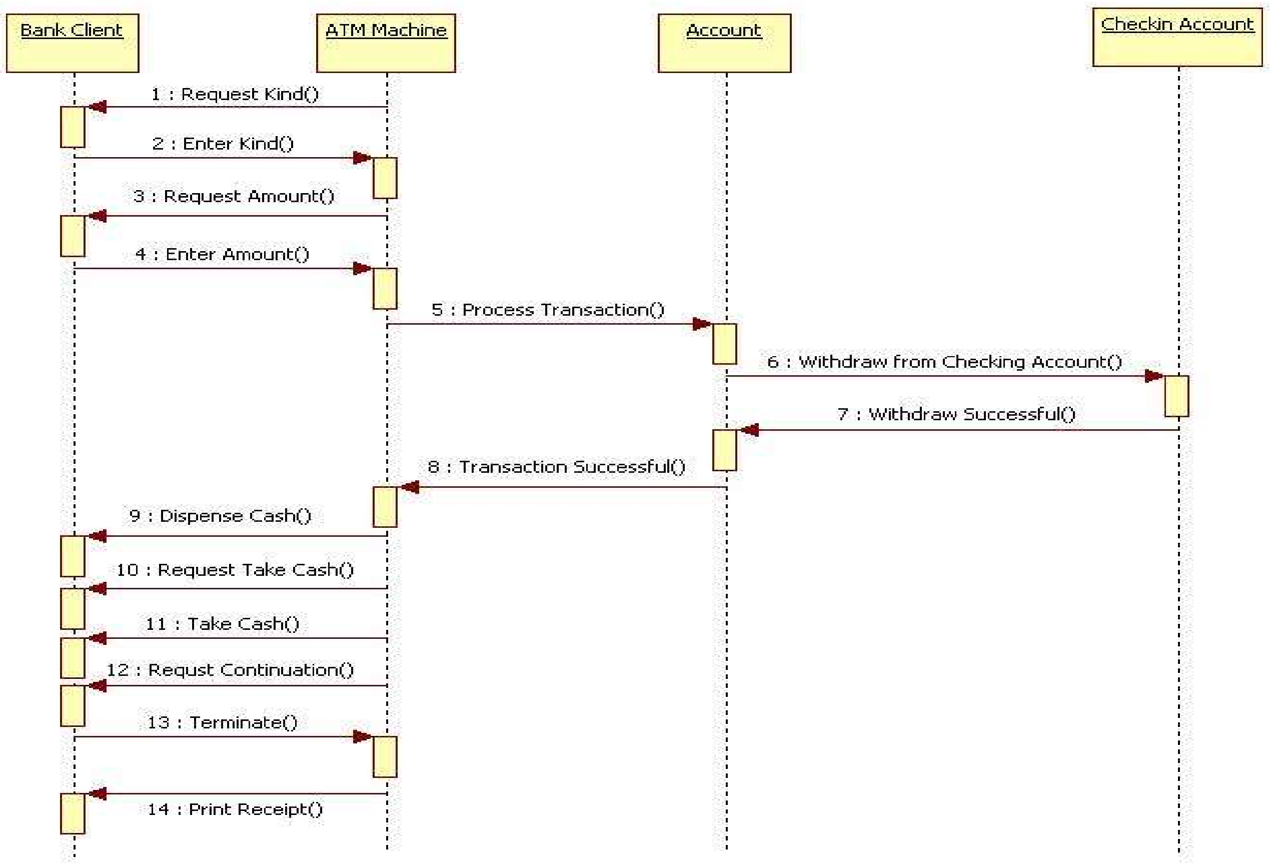
3. Use-Case Diagram

- Describes how users interact with the system.
- Example: A user withdrawing money from an ATM or checking their balance.
- ***Example Usecase diagram for the ATM system***



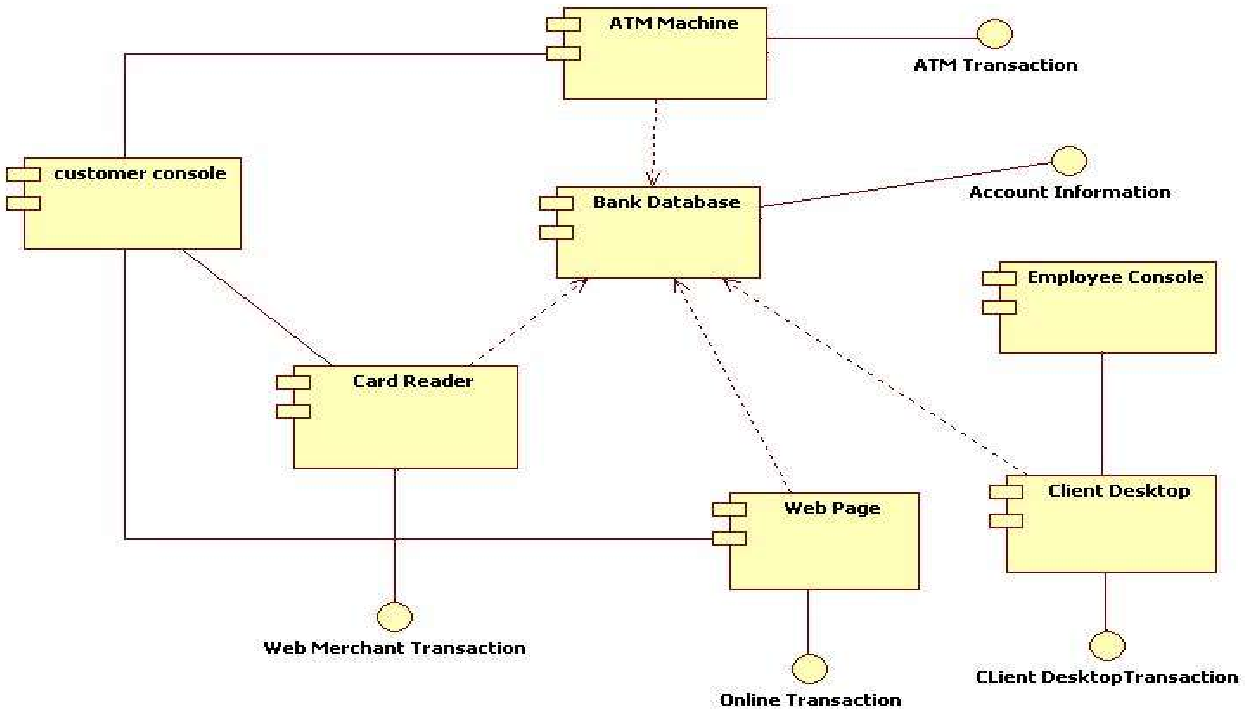
4. Sequence Diagram

- Shows the order in which things happen in a system.
- Example: A user requests cash, the ATM asks the bank, and money is dispensed.
- ***Example Sequence diagram for the ATM system***



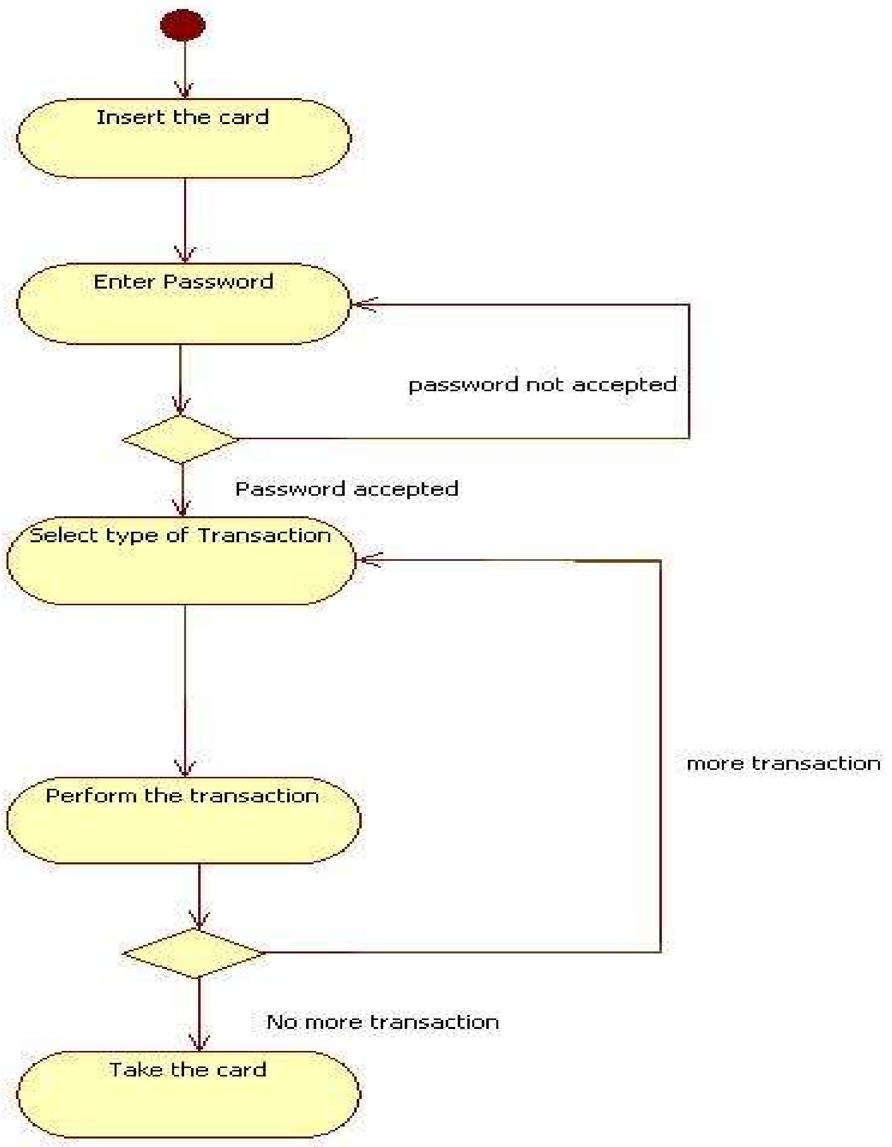
5. Component Diagram

- Shows how the different parts (components) of the system fit together.
- Example: An ATM machine's software connects to the bank's database to process a transaction.
- ***Example Component diagram for the ATM system***



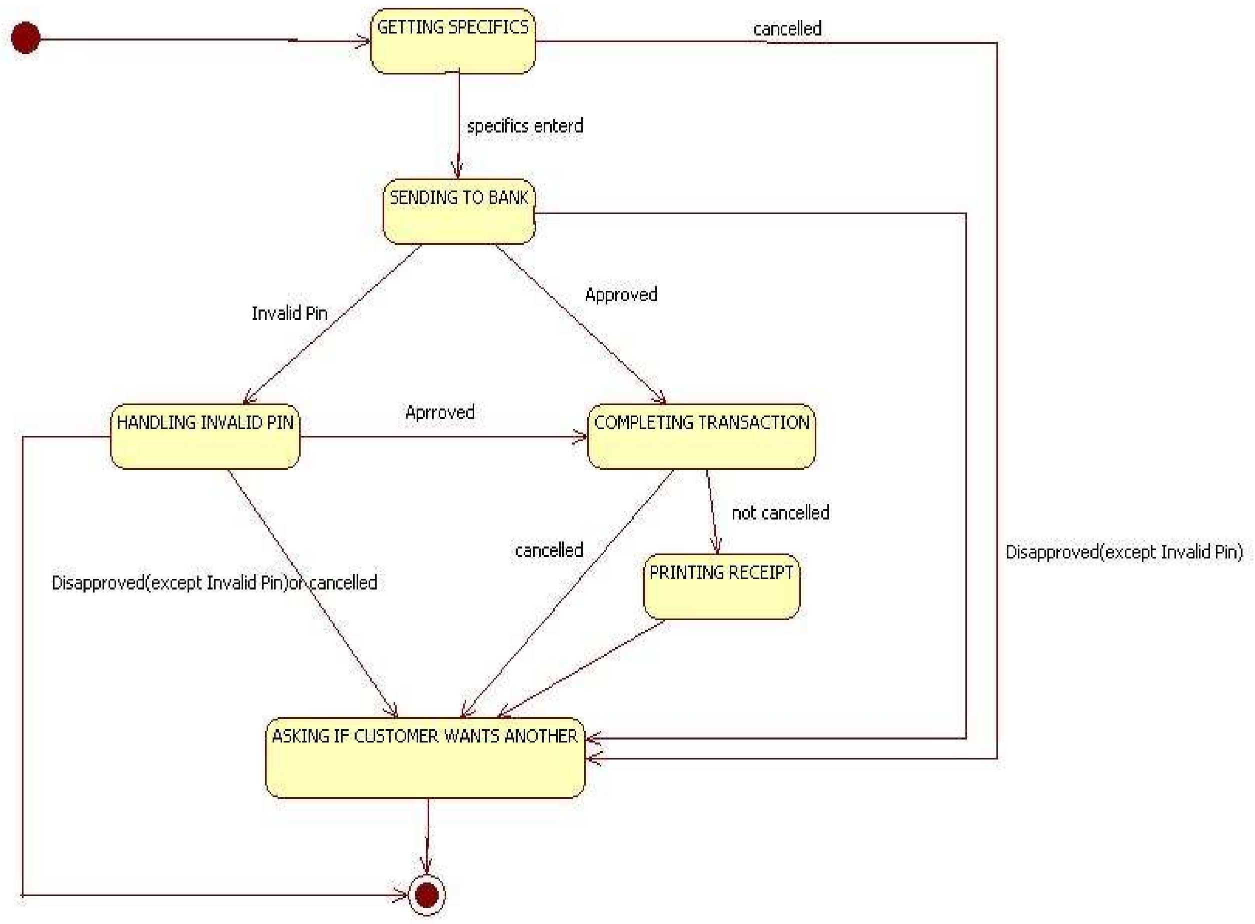
6. **Activity Diagram**

- Shows the steps in a process.
- Example: The process of withdrawing money from an ATM: insert card, enter PIN, select amount, and get cash.
- ***Example Activity diagram for the ATM system***



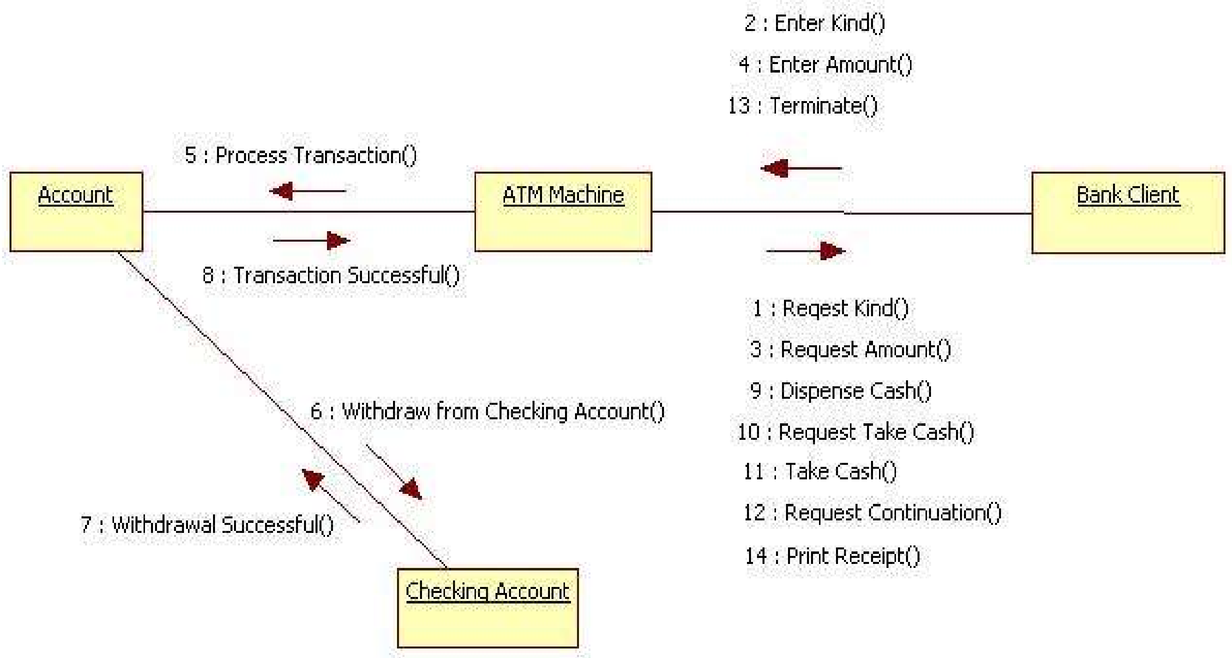
7. State-Chart Diagram

- Describes the different states an object can be in and how it changes between them.
- Example: An ATM machine can be "Idle" when not in use and "Dispensing Cash" when a user withdraws money.
- ***Example State-chart diagram for the ATM system***



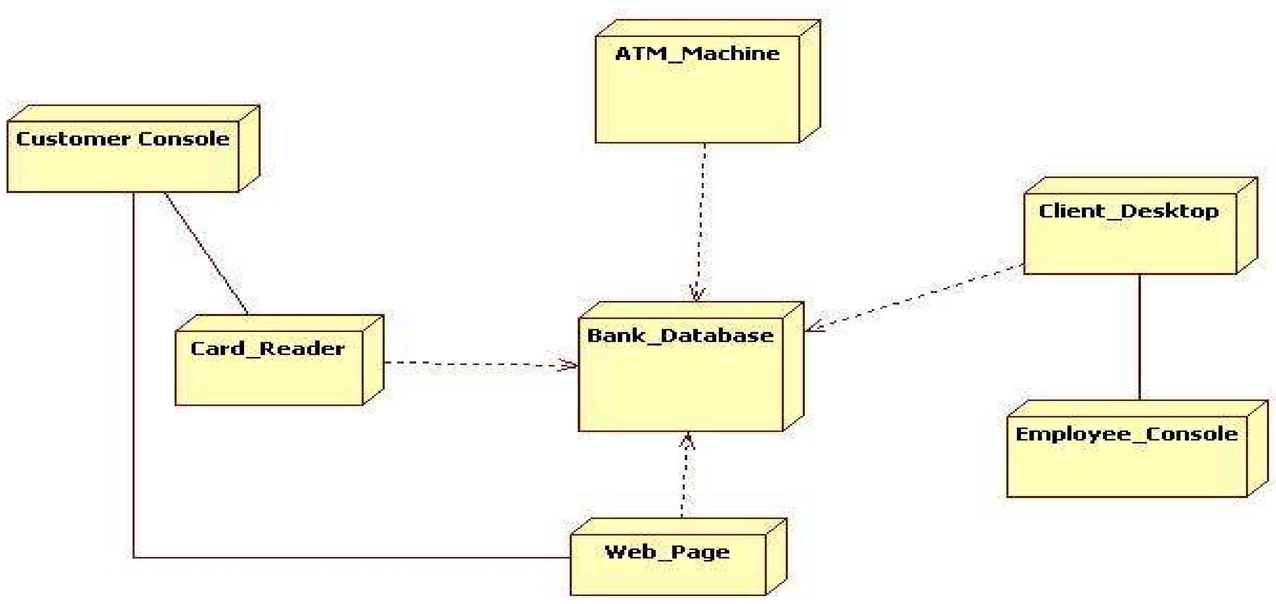
8. Collaboration Diagram

- Focuses on how different parts of the system work together to achieve something.
- Example: How the ATM, the user, and the bank's server work together during a cash withdrawal.
- ***Example Collaboration diagram for the ATM system***



9. Deployment Diagram

- Shows where the physical parts of the system are located and how they are connected.
- Example: An ATM machine at a bank connects to the bank's server, which might be in a different city.
- ***Example Deployment diagram for the ATM system***



UML makes it easier for people working on software to understand how everything fits together. By using diagrams to show how the system will work, teams can plan, communicate, and organize their ideas before they start building the actual software. UML is a simple but powerful way to visually map out a complex system, making it much clearer for everyone involved.