# SOFTWARE ENGINEERING LAB

## EXERCISE – 3

## TOPIC – 3

# UML DIAGRAMS – CLASS

## Class Diagram

A Class Diagram is a key building block in object-oriented modeling. It represents the static structure of a system by describing:

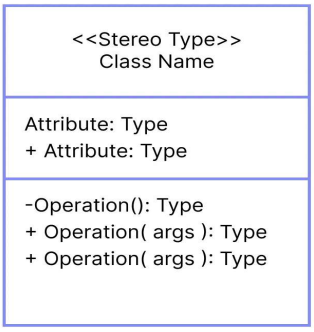
- **Classes:** The blueprint for objects.
- **Attributes:** Characteristics of the classes.
- **Methods:** The behaviors or actions associated with the class.
- **Relationships:** How the classes interact with each other.

Class diagrams are crucial in software development because they help engineers and stakeholders understand how the different components of a system fit together.

## Components of a Class Diagram:

1. **Classes:**

- A class in UML is a blueprint for objects. It is depicted as a rectangle divided into three sections:
  - **Top section:** Contains the name of the class.
  - **Middle section**: Lists the attributes (properties or data) of the class.
  - **Bottom section:** Shows the methods (functions or behaviors) that the class can perform.



### 2. Attributes:

- Attributes are variables that represent the state or properties of a class.
- They describe what an object of the class knows or possesses.
- Examples:
  - *A Car class might have attributes like color, model, and speed.*
  - *A Person class might have attributes like name, age, and height.*
- Attributes can be:
  - **Private (-):** Only accessible within the class.
  - **Public (+):** Accessible from outside the class.
  - **Protected (#):** Accessible by the class and its subclasses.

### 3. Methods:

- Methods represent the functions or behaviors that objects of the class can perform.
- They describe what actions an object of the class can take.
- Examples:
  - *A Car class might have methods like drive() and stop().*
  - *A Person class might have methods like walk(), talk(), and sleep().*

### 4. Relationships:

- The relationships between classes define how different classes are connected and interact with each other. UML supports various types of relationships, including:

**Association:**

- Represents a general connection between two classes. It implies that objects of one class interact with objects of another class.
- *Example: A Teacher and Student. The association shows that teachers teach students, and students are taught by teachers*.
- Shown by a solid line connecting the two classes.

**Inheritance (Generalization):**

- Also known as Generalization, this relationship indicates that one class inherits attributes and methods from another class.
- *Example: Dog is a subclass of Animal, meaning that all dogs inherit properties like legs and eyes from the Animal class, but dogs also have additional properties like barking.*
- Shown by a solid line with a hollow arrow pointing toward the parent class.


**5. Aggregation:**

- Represents a whole-part relationship where the part can exist independently of the whole.
- *Example: A classroom can exist without its students. Even if a student leaves, the classroom still exists.*
- Shown by a solid line with a hollow diamond near the class that represents the whole.

**6. Composition:**

- A stronger form of aggregation where the part cannot exist independently of the whole.
- *Example: A house and its rooms. If the house is destroyed, the rooms no longer exist.*
- Shown by a solid line with a filled diamond near the class that represents the whole.

**7. Dependency:**

- A weaker relationship where one class depends on another to perform its function.
- *Example: A Printer depends on a PrintJob. The printer class depends on the presence of a print job to function.*
- Shown by a dashed arrow from the dependent class to the class it depends on.

**8. Realization:**

- Represents that a class implements an interface or a contract.
- *Example: If a class Car implements an interface Vehicle, it must define all methods declared in the Vehicle interface, such as start() and stop().*
- Shown by a dashed line with a hollow arrow pointing towards the interface.

## Drawing a Class Diagram for an ATM System

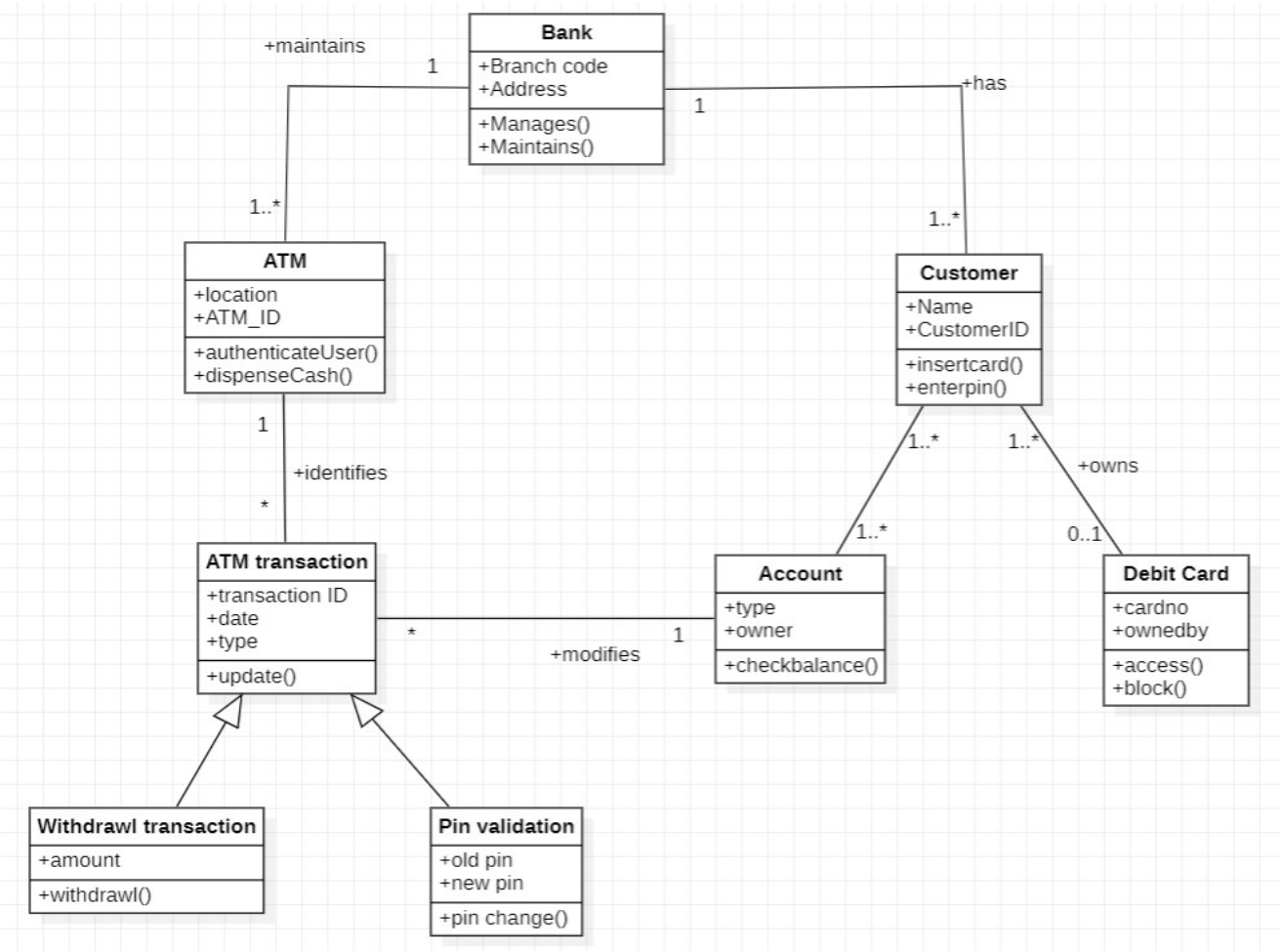
To illustrate how class diagrams are used, consider an ATM system. When modeling such a system, you break it down into classes, define their attributes and methods, and specify how they interact with each other.

**Steps to Create an ATM Class Diagram:**

1. Identify the Main Classes:
   - Think about the key components of the ATM system. These will form the classes.
   - *Examples of classes:*
     - **ATM:** The machine itself.
     - **Customer:** The person using the ATM.
     - **Bank Account:** The account that holds the customer's money.
     - **Transaction:** Actions like withdrawal, deposit, etc.
2. Define Attributes for Each Class:
   - List the important properties each class should have.
   - *Examples:*
     - **ATM:** location, ATM_ID.
     - **Customer:** name, customerID.
     - **Bank Account:** accountNumber, balance.
3. Define Methods for Each Class:
   - Methods are the actions that each class can perform.
   - *Examples:*
     - **ATM:**
     - authenticateUser(): Verify if the user is valid.
     - dispenseCash(): Give cash to the customer.
     - **Customer:**
     - insertCard(): The customer inserts their card into the ATM.
     - enterPIN(): The customer enters their PIN to authenticate.
     - **Bank Account:**
     - checkBalance(): Check the current balance in the account.
     - withdrawMoney(): Withdraw a specified amount of money.

4. Show the Relationships Between the Classes:

- **Association:** There is a relationship between the ATM and the customer, as well as between the ATM and the bank account.
- **Dependency:** The ATM depends on a Transaction to perform operations like withdrawal.
- **Composition:** An ATM contains components like a Card Reader, Cash Dispenser, and Display, and these components cannot exist without the ATM.



## Examples of Class Diagrams in the Real World

1. **E-Commerce System:**

- Classes:
  - Customer: Represents a user of the e-commerce platform.
  - Product: Represents an item that is available for purchase.
  - Order: Represents a customer's order, which includes multiple products.
- Relationships:
  - Association: A customer can place multiple orders.
  - Composition: An order consists of several products.

2. **Library System:**

- Classes:
  - Book: Represents a book in the library.
  - Member: Represents a library member.
  - Librarian: Represents the person managing the library.
- Relationships:
  - Association: A member borrows books.
  - Inheritance: A librarian and a member are both people, so they inherit from a Person class.