



DEPARTMENT OF  
COMPUTER SCIENCE AND ENGINEERING

---

Lab No. 11

Title: Develop UML Class Diagram (part 2: extended class diagram including inheritance, abstract class, interface etc.) for the given project

---

INTEGRATED DESIGN PROJECT I LAB  
CSE 324



GREEN UNIVERSITY OF BANGLADESH

---

# 1 Objective(s)

- To depict various aspects of the OOPs concept using Class Diagram and to show an inheritance hierarchy among classes.

## 1.1 Sub-Objective(s)

- To construct software applications using object oriented languages.
- To reduces the maintenance time by providing an overview of how an application is structured before coding.
- To model the collaboration among the elements of the static view.
- To incorporates forward and reverse engineering.

# 2 UML Class Diagram

## 2.1 What is Class Diagram?

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. A Class Diagram in Software engineering is a static structure that gives an overview of a software system by displaying classes, attributes, operations, and their relationships between each other. Class Diagram helps construct the code for the software application development. Class Diagram defines the types of objects in the system and the different types of relationships that exist among them. It gives a high-level view of an application. This modeling method can run with almost all Object-Oriented Methods. A class can refer to another class. A class can have its objects or may inherit from other classes.

## 2.2 Purpose of Class Diagrams

Most of the UML diagrams can't be mapped directly with any object-oriented programming languages except class diagrams. In other words, class diagram ideally can have one to one mapping to UML class diagrams. Class diagrams are the main building blocks of every object-oriented method. The class diagram can be used to show the classes, relationships, interface, association, and collaboration. UML is standardized in class diagrams. Since classes are the building block of an application that is based on OOPs, so as the class diagram has an appropriate structure to represent the classes in their context. It describes various kinds of objects and the static relationship between them.

## 2.3 Problem statement and motivation for the given project:

The project entitled Banking ATM system has a drastic change to that of the older version of banking system, customer feel inconvenient with the transaction method as it was in the hands of the bank employees. In our ATM system, the above problem is overcome here, the transactions are done in person by the customer thus makes the customers feel safe and secure. Thus the application of our system helps the customer in withdrawing money, checking the balance and transaction of the amount with mini-statement and transferring the balance by validating the pin number therefore ATM system is more user friendly.

# 3 Methodology

## 3.1 Essential elements of A UML class diagram

See the Essential component in lab manual 10.

## 3.2 Relationships between Classes

There are mainly three kinds of relationships in UML:

- Dependencies

- Generalizations
- Associations

## (ii) Generalization-

In UML modeling, a generalization relationship is a relationship that implements the concept of object orientation called inheritance. The generalization relationship occurs between two entities or objects, such that one entity is the parent, and the other one is the child. The child inherits the functionality of its parent and can access as well as update it. Generalization is sometimes called “is a kind of” relation, which is established through inheritance process.

In class diagrams, generalization relationship is represented by a solid line with a hollow arrowhead pointing towards the parent model element from the child model element shown in figure 1.

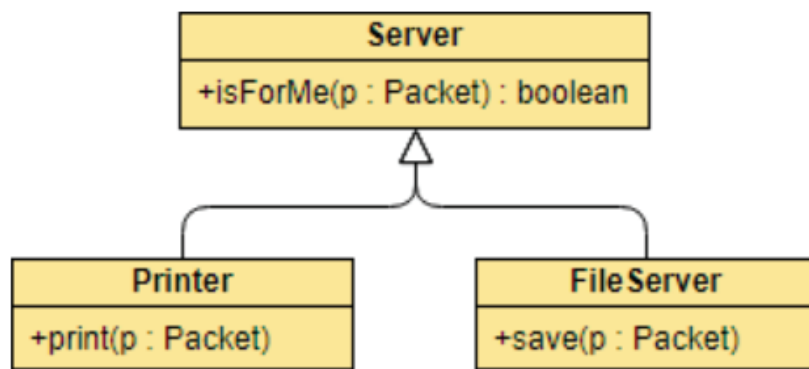


Figure 1: Symbols of Generalization

## Abstract classes and methods

In the abstract class, no objects can be a direct entity of the abstract class. The abstract class can neither be declared nor be instantiated. It is used to find the functionalities across the classes. The notation of the abstract class is similar to that of class; the only difference is that the name of the class is written in italics. Since it does not involve any implementation for a given function, it is best to use the abstract class with multiple objects shown in figure 2.

E.g. a class for Shape can be marked as abstract. It cannot be instantiated because we do not know what kind of shape it represents. It is a base (super) class for other shape classes (e.g. Ellipse, Rectangle).

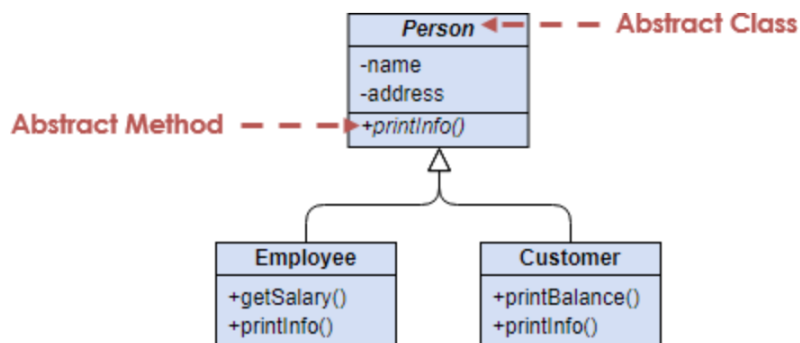


Figure 2: Symbols of Abstract classes and methods

In the above diagram there is an abstract class called person. In UML, the name of an abstract class is written in an italic font. This class contains one abstract method called printinfo, it is written in a italic font. An abstract method has no implementation. Typically an abstract class contains one or more abstract method.

### Realization / Implementation

Implementation is the relationship between two things, one thing (interface) specifies a contract, and the other thing (class) guarantees the execution of the contract by implementing the operations specified in the contract. It is used to show that the child is implemented by its parent, such that the child object inherits the structure and behavior of its parent object without disobeying the rules.

For example, the I shape interface might specify methods for showName. The two circle classes need to implement the methods, possibly in very different ways. In the class diagram, the implementation relationship appears as a dashed line with an open arrow pointing to the interface shown in figure 3.

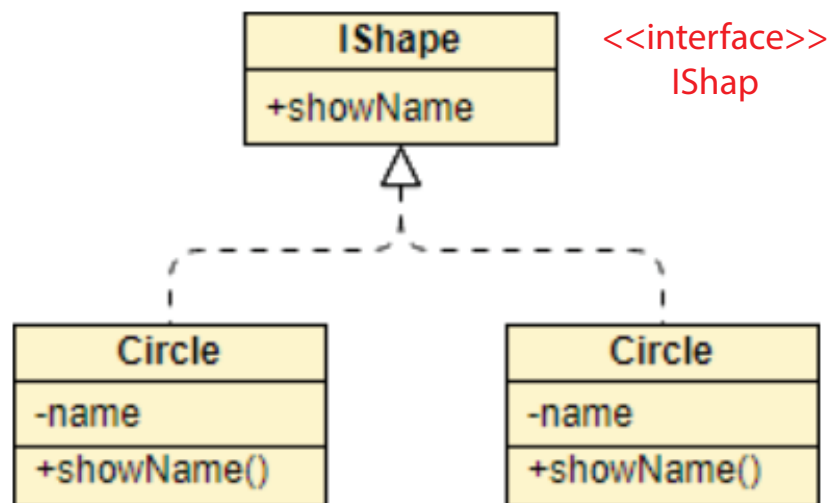


Figure 3: Symbols of Realization

An interface may be shown using a rectangle symbol with the keyword **<<interface>>** preceding the name. In UML modeling, interfaces are model elements that define sets of operations that other model elements, such as classes, or components must implement.

For example, the Owner interface might specify methods for acquiring property and disposing of property. The Person and Corporation classes need to implement these methods, possibly in very different ways shown in figure 4.

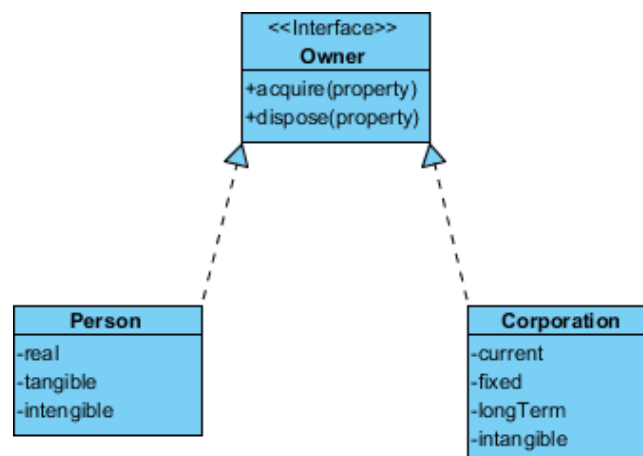








Figure 4: Symbols of Interface

## Table 1: Class Diagram Relationships Types

Class Diagram Relationship Type	Notation
Association	
Inheritance	
Realization/ Implementation	
Dependency	
Aggregation	
Composition	

To design your Class diagram, you may use platforms and editing tools online. These tools are helpful since they already have the needed symbols to illustrate your class diagram. You just have to plot the included classes, relationships, and visibility. Then you will put the appropriate attributes that the system requires.

platforms or online tools that you may use are:

1. Visual Paradigm for UML 8.2 (online link: <https://online.visual-paradigm.com/>)
2. StartUML
3. Lucidchart and other drawing tools

The diagram illustrates the MVC design pattern and its components, categorized into three main groups: Control, Boundary, and Entity.

- Control Class:**
  - DrawingContext** (Control Class): Contains methods `+setPoint()`, `+clearScreen()`, `+getVerticalSize()`, and `+getHorizontalSize()`. It has a **Dependency** on the **Event** entity.
- Boundary Class:**
  - Window** (Boundary Class): Contains methods `+open()`, `+close()`, `+move()`, `+display()`, and `+handleEvent()`. It is the **main window of the application**. It has a **Dependency** on the **Event** entity and an **Aggregation** relationship with the **Shape** entity (multiplicity 1 at Window, \* at Shape).
  - ConsoleWindow** (Boundary Class): Inherits from **Window**.
  - DialogBox** (Boundary Class): Inherits from **Window**.
- Entity Class:**
  - Event** (Entity Class): A simple entity class.
  - Shape** (Entity Class): An **Abstract Class** containing methods `+draw()`, `+erase()`, `+move()`, and `+resize()`. It is generalized by **Rectangle** and **Polygon**.
  - Circle** (Entity Class): Inherits from **Shape**. It contains attributes `-radius: float` and `-center: unsigned int`, and methods `+area(in radius: float): double`, `+circum()`, `+setCenter()`, and `+setRadius()`. It has a **Composition** relationship with the **Point** entity (multiplicity 1 at Circle, \* at Point).
  - Rectangle** (Entity Class): Inherits from **Shape**.
  - Polygon** (Entity Class): Inherits from **Shape**.
  - Point** (Entity Class): A simple entity class.
- Other Relationships:**
  - Association:** A dashed arrow points from **DialogBox** to **DrawingContext**.
  - Control:** A solid arrow points from **DrawingContext** to **Window**.
  - Operation:** A solid arrow points from **Circle** to **Point**.

© Dept. of Computer Science and Engineering, GUB

### 3.5 Design Procedure

Class diagrams go hand in hand with object-oriented design. So knowing its basics is a key part of being able to draw good class diagrams.

When required to describe the static view of a system or its functionalities, you'd be required to draw a class diagram. Here are the steps you need to follow to create a class diagram.

**Step 1:** Identify the class names

The first step is to identify the primary objects of the system.

**Step 2:** : Distinguish relationships

Next step is to determine how each of the classes or objects are related to one another. Look out for commonalities and abstractions among them; this will help you when grouping them when drawing the class diagram.

**Step 3:** : Create the Structure

First, add the class names and link them with the appropriate connectors. You can add attributes and functions/ methods/ operations later.

## 4 Implementation

You can draw class diagrams in VP-UML. A class diagram describes the structure of an object-oriented system by showing the classes in that system and the relationships between the classes. A class diagram also shows constraints, and attributes of classes. Go through the following link and follow procedure step by step to draw a class diagram. <https://www.visual-paradigm.com/support/documents/vpuserguide/94/2576/7190drawingclass.html>

The Class diagram of the Banking ATM system is shown in Fig.6.

## 5 Output

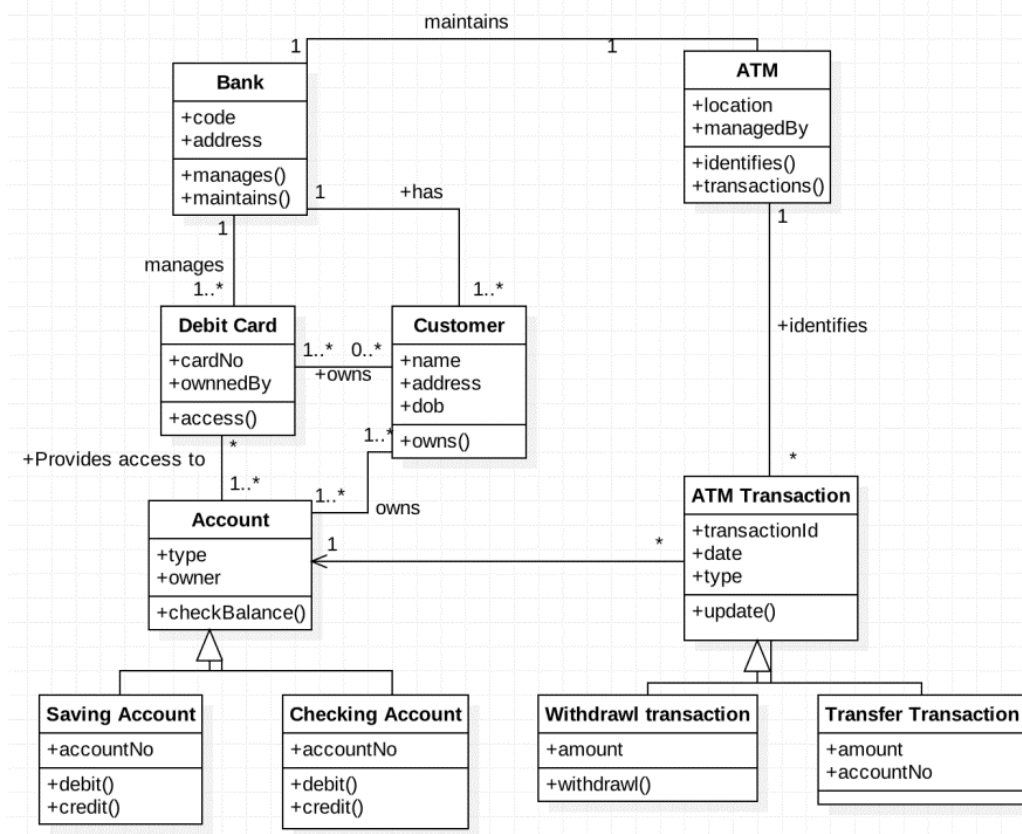


Figure 6: Class diagram of the Banking ATM system

---

The ATM Management System Management System is a modeled diagram that explain its classes and relationships. In the diagram classes are represented as boxes with three rectangles inside each box. The top rectangle has the class's name; the middle rectangle contains the class's properties; and the bottom rectangle contains the class's methods, commonly known as operations.

As you can see through the illustration, the classes were determined which is symbolized by boxes. They were designated with their corresponding attributes and shows the class' methods. Their relationships are also plotted to show the connections between classes and their multiplicity. You should also look into the visibility symbols displayed in the diagram. These are important because it declares the status of attributes in your Class Diagram. Some of the Class' attributes are for public (+) which means that they can be accessed by the classes connected to them. While the protected ( ) symbols, means that the attributes of the data can be accessed by the same classes or subclass and the (-) symbol means it cannot be accessed by other class.

## 6 Discussion & Conclusion

To design and visualize the software system artifacts, the standard language used is the UML. The relationship between the different objects is described by the class diagram, which ensures the design and analysis of an application and views it in its static form. Being the most important UML diagram, the class diagram consists of class, attributes, and relationships, which are its essential elements. To get an idea of the application structure, the class diagram is used, which helps in reducing the maintenance time.

## 7 Lab Task (Please implement yourself and show the output to the instructor)

1. Draw a class diagram for the given project '**Library Management System**'.

### 7.1 Problem analysis

A library database system is an infrastructure that allows users to search books and book content, add/remove, and download selected books. The problem faced is that library users require an efficient method to find a specific book or keyword(s) within a book given a continuously expanding library. Some scenarios of the 'Library Management System' are as follows:

1. User who registers himself as a new user initially is regarded as staff or student for the library system.
2. For the user to get registered as a new user, registration forms are available that is needed to be fulfilled by the user.
3. After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to cardholder or user.
4. After getting the library card, a new book is requested by the user as per there requirement.
5. After, requesting, the desired book or the requested book is reserved by the user that means no other user can request for that book.
6. Now, the user can renew a book that means the user can get a new due date for the desired book if the user has renewed them.
7. If the user somehow forgets to return the book before the due date, then the user pays fine. Or if the user forgets to renew the book till the due date, then the book will be overdue and the user pays fine.
8. User can fill the feedback form available if they want to.
9. Librarian has a key role in this system. Librarian adds the records in the library database about each student or user every time issuing the book or returning the book, or paying fine.
10. Librarian also deletes the record of a particular student if the student leaves the college or passed out from the college. If the book no longer exists in the library, then the record of the particular book is also deleted.

---

11. Updating database is the important role of Librarian.

## 8 Lab Exercise (Submit as a report)

- Draw an Class diagram for the given project for your team.

## 9 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.