**Unified Modeling Language (UML):**

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems.

The UML uses mostly graphical notations to express the design of software projects.

There are a lot of different diagrams (models) to get used to. The reason for this is that it is possible to look at a system from many different viewpoints. A software development will have many stakeholders playing a part.

For Example:

* Analysts
* Designers
* Coders
* Testers
* QA
* The Customer

All of these people are interested in different aspects of the system, and each of them requires a different level of detail. For example, a coder needs to understand the design of the system and be able to convert the design to a low level code.

Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts; there are seven types of structure diagram as follows:

* Class Diagram
* Component Diagram
* Deployment Diagram
* Object Diagram
* Package Diagram
* Composite Structure Diagram
* Profile Diagram

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time, there are seven types of behavior diagrams as follows:

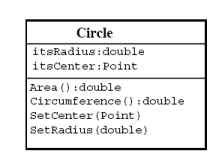
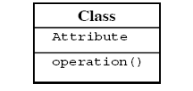
* Use Case Diagram
* Activity Diagram
* State Machine Diagram
* Sequence Diagram
* Communication Diagram
* Interaction Overview Diagram
* Timing Diagram

**Class Diagram:**

Class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or) methods and the relationships between the classes.

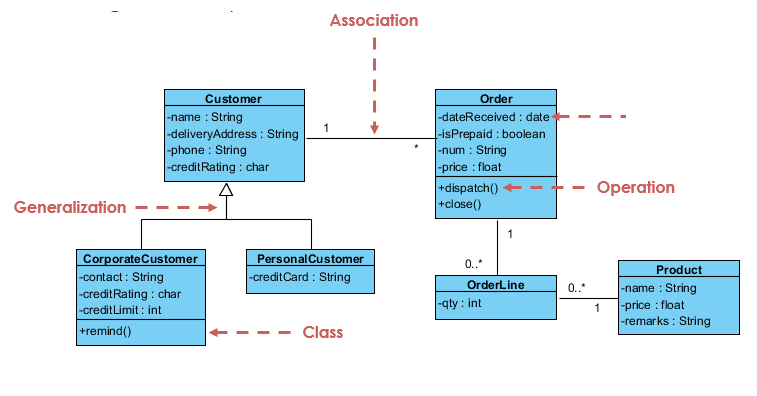
The class diagram is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts. A class with three sections:

* The upper part holds the name of the class.
* The middle part contains the attributes of the class.
* The bottom part gives the methods or operations the class can take or undertake.

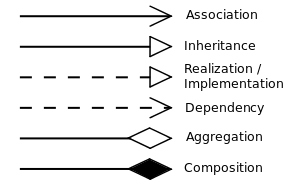


**Draw all the class diagrams given in this lab manual.**

**Example:**



**Relationship between Classes:**



**Multiplicity (Cardinality):**

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for just one company.

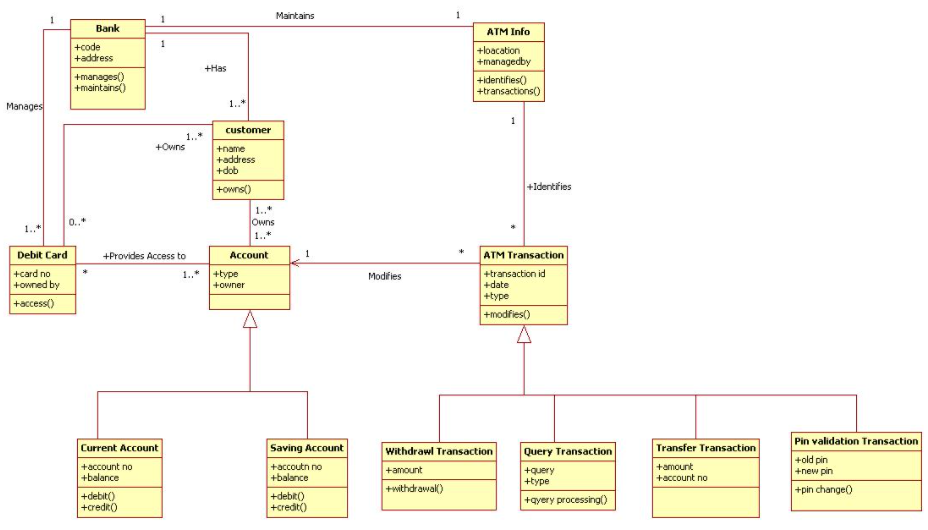
**Steps to make a class diagram:**

* Identify classes
* Identify attributes of each class
* Identify methods/functions of each class
* Identify relationship between them
* Identify Cardinality
* Model the class diagram

**Example:**

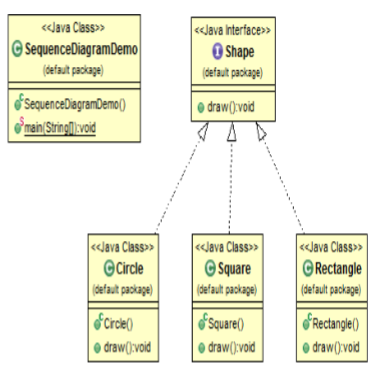
Below is class diagram of Bank ATM.

For the requirements of Bank ATM following class diagram is created .



**Code to class diagram:**

|  |
| --- |
| interface shape  {   Void draw( )  }  Class circle implements shape  {   Void draw( )  {   System.out.println(“circle”);   }  }    Class square implements shape  {   Void draw( )  {   System.out.println(“square”);   }  }    Class Rectangle implements shape  {   Void draw( )  {   System.out.println(“rectangle”);   }  }      public class SequenceDiagramDemo  {  public static void main(String[] args)  {  Shape circle = new Circle();  circle.draw();  Shape rectangle = new Rectangle();  rectangle.draw();  Shape square = new Square();  square.draw(); } } |

****

**Exercise:**

|  |
| --- |
| SEDCGS” is web based application, which provides facility to teachers to compose, design and generate unique exams. Any institution with IT background can use this system this application is capable of generation of multiple unique papers from composing to generation of paper all work done using this application. It is capable of pattern setting of papers. It is also capable of assigning different values to same questions by assigning dataset. It supports all categories of questions that is fill in the blanks , multiple choice question, short , detail questions. Customer support and help is also provided by providing visual and written aids. Our main aim was to conduct fair exam by eliminating cheating case opportunities. So our application is going to provide quality oriented unique question papers without making any overhead for teachers. |

Draw a class diagram of the above requirement specifications.