1.) What is inheritance in object-oriented technology? Give an example.

Inheritance is the relation between classes. It is a mechanism where you can derive a class from another class. one called base class or superclass that defines the base type and the other called derived class or subclass that defines the derived type. The members of the base class are shared among all the derived classes, but the derived class members are exclusive to that class.

So, by using inheritance, we need not create and define data members and functions recursively. We code once in a class, and they can inherit all properties of data members and functions in the subsequent subclass. This feature also helps in effective dynamic programming.

Ex: Take a case of a parent class called Shape (), this contains members like Area (), Volume (), Dimensions (). Now this class can be inherited to child classes like Rectangle (), Square (), Circle (). Each of these child classes will have custom implementation of the required methods from the Parent Class Shape ().

2.) What is the difference between an object and a class in OO technology?

In Object Oriented Programming, there is a huge difference between Class and Object. A class is a blueprint or template, which contains some values, known as member data or member, and some set of rules, known as behaviors or functions from which objects are created. It is basically a logical entity which can have group of similar objects. On the other hand, object is an instance of class. It is a physical entity which allocates memory on creation.

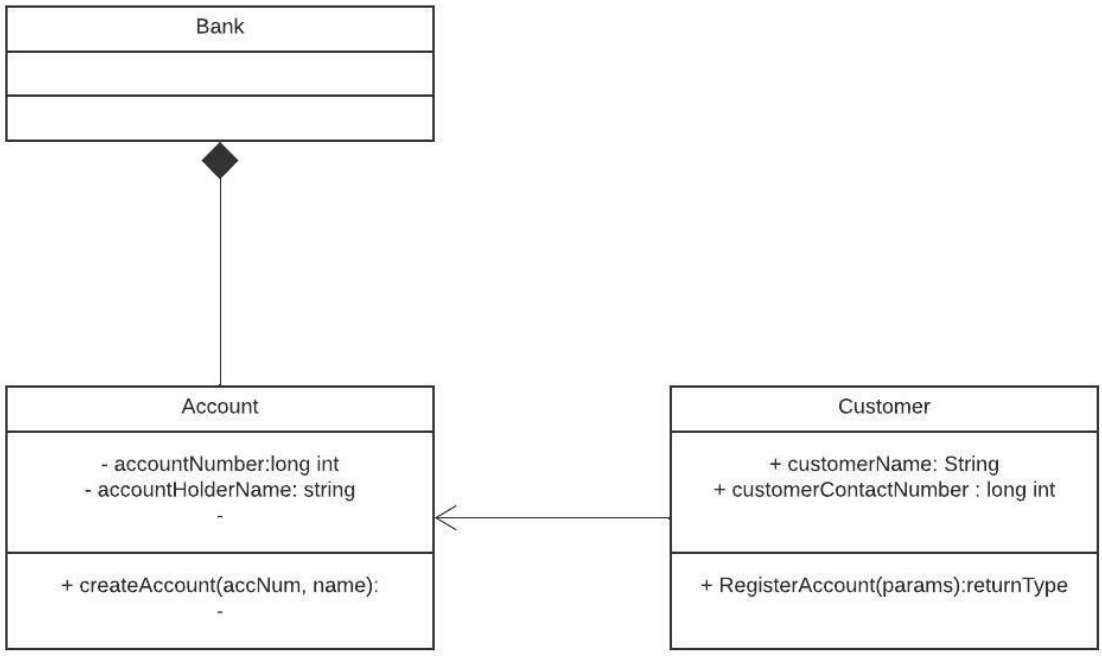
Ex: Best example for understanding the difference between class and object would considering class as Fruit and objects of this class as Apple, Mango, Banana.

3.) Describe the role of polymorphism in object-oriented technology. Give an example.

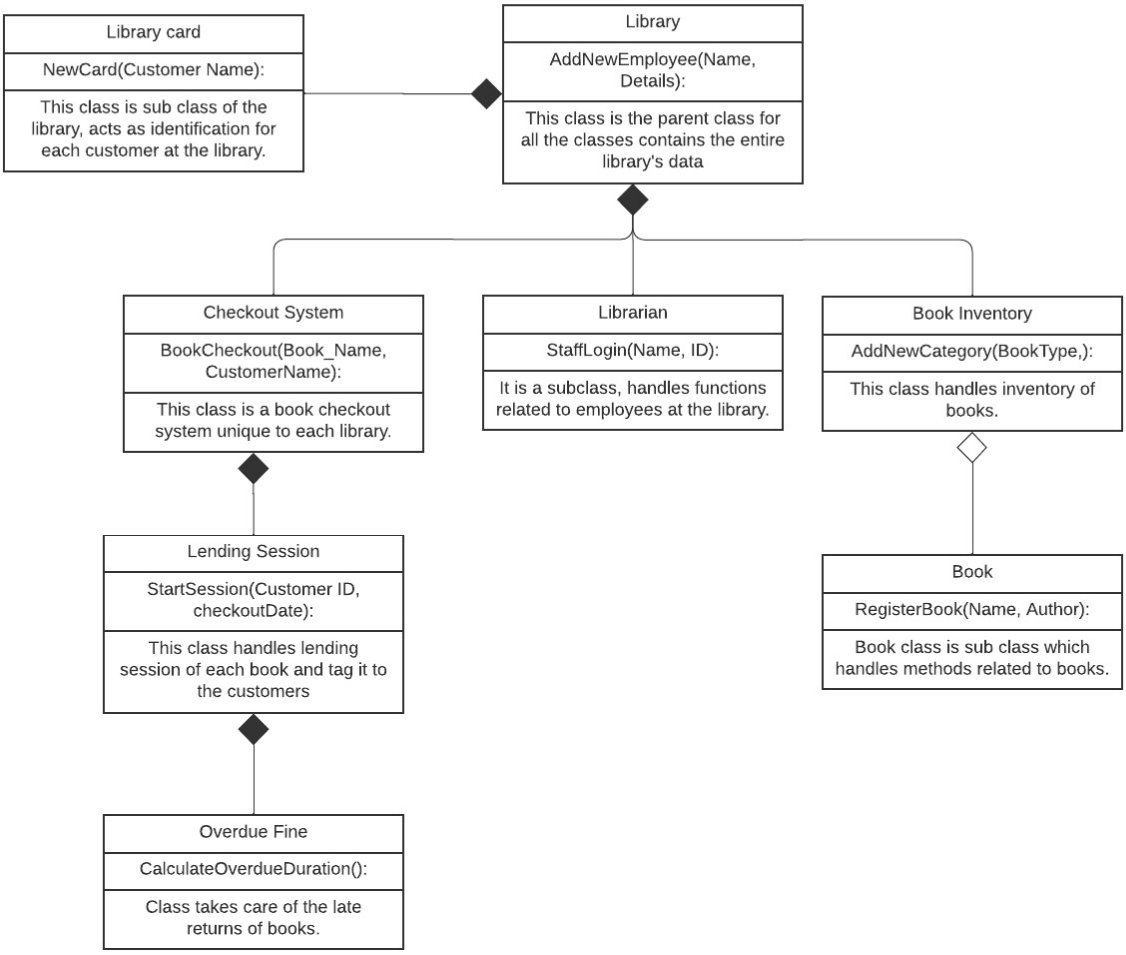
Polymorphism is one of the key aspects of Object-oriented technology. It is helpful when something occurs in several different forms. It allows you to use the derived type members in place of the base type. Based on which object being invoked either the base or derived it takes appropriate action belonging to that particular class or object. This implicit checking occurs during the run time and this mechanism is also called as late binding or dynamic binding. To be more specific in C++ polymorphism occurs in the following way, virtual methods are present in the base class, but when virtual methods are called and the object that calls it is of the derived type, then the derived type's method is called instead of the base type's method.

Ex: Consider a class Shape () which has a method CalculateArea () and a child class called Square () inherits this Shape class. Now this Square () class has a custom implementation of the method CalculateArea (). When an object of the square class is created, and it calls the method CalculateArea () it implements the custom implementation of that particular method in the square class but not the Shape class. This is because during run time the mechanism implicitly checks which object is being used based ion which it performs the action.

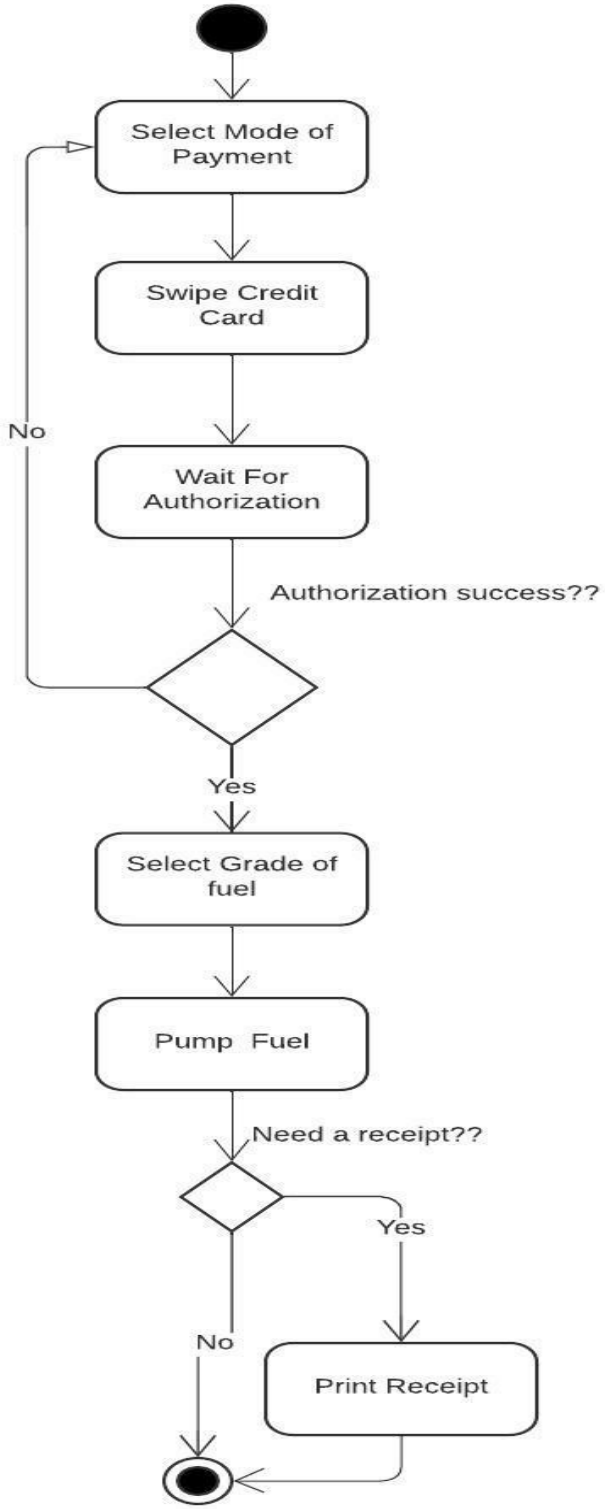
4.) Draw a class diagram of a small banking system showing the associations between three classes: the bank, customer, and the account.



5.) **Draw a class diagram of a library lending books using the following classes: Librarian, Lending Session, Overdue Fine, Book Inventory, Book, Library, Checkout System, and Library Card**.



6.) Draw an activity diagram of pumping gas and paying by credit card at the pump. Include at least five activities, such as "Select fuel grade" and at least two decisions, such as "Get receipt?"



7.) Explain how a class dependency graph differs from a UML class diagram.

A dependency graph is a data structure formed by a directed graph that describes the dependency of an entity in the system on the other entities of the same system. The underlying structure of a dependency graph is a directed graph where each node points to the node on which it depends.

Coming to UML class diagram, the UML Class diagram is a graphical notation used to construct and visualize object-oriented systems. A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's Classes, attributes, operations and relationship among objects.

The main difference in both these structures is that dependency graph showcases dependency among different entities in a system, whereas the class diagram signifies the relations between classes and their attributes. Hence though it appears to be similar, they differ because class diagram can only show relationship between classes.