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

   Inheritance is a relation between two classes: one is called parent class or base class and the other is called child class or derived class. The child class has all the properties of the parent class once it is defined. The Child class can also have extra properties or methods or the properties of the parent class can also be overridden to have different properties in the child class.

   If the variables or the methods are declared as private, then the child class doesn't inherit the properties of the parent class. To access these we need to have access specifiers in the base class and the child class can use these private variables using these access specifier methods.

   Example:

```
class Phone {
    public:
        string Type; //touch or keypad
        string model; //model number to keep track of number of phones
};

class Apple: public Phone {
    public:
        string Name; //iphone14, iphone14 pro etc
        string Price; //price
};
```

   The parent class has certain properties/attributues like type of phone (android or ios ex), Model etc. Now, the child class will have Name, price, type and model. you can always change the attributes of parent class in the child class. That is Once an object instance is created say for Apple, this object instances Type or model can also be modified.

   As we can see, this gives us a modular approach to coding but also reduces the repeatability of code further by using OOPS and also can improve run time if used properly.

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

   Let's say we are creating a shopping application for mobile phones. we know that every mobile phone has certain properties in common and certain properties which are different from each other. There will also be a number of mobile phones of the same model. let's say, we have 10 mobiles of iPhone 14 we want to sell, we don't want to create properties for each of these mobile phones seperatley. this is where objects come in handy.

   Now, I write a class for this mobile phone which has certain properties. Now, to create 10 mobile phones, I create 10 different objects from the same class. Now, I will have 10 different entities(objects) of the same phone.

   Object: Objects represent real-world entities. Class: It has the properties of the object.

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

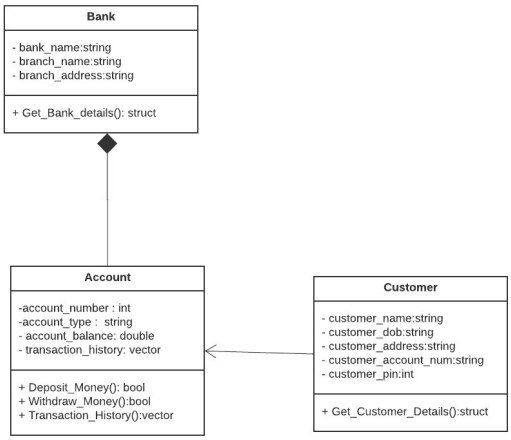
```
class Phone {
    public:
        string Type; //touch or keypad
        string model; //model number to keep track of the number of phones
    virtual void Phonecolor() {
        cout<<"virtual function called"<<endl;
    };
};

class Apple: public Phone {
    public:
        string Name; //iphone14, iphone14 pro etc
        string Price; //price
    void Phonecolor() {
        cout<< "phone is red in color"<<endl;
    }
};
```

```
int main()
{
    object1 = Phone();
    object2 = Apple();
    object1.Phonecolor();
    object2.Phonecolor();
}
output: virtual function called ; phone is red.
```

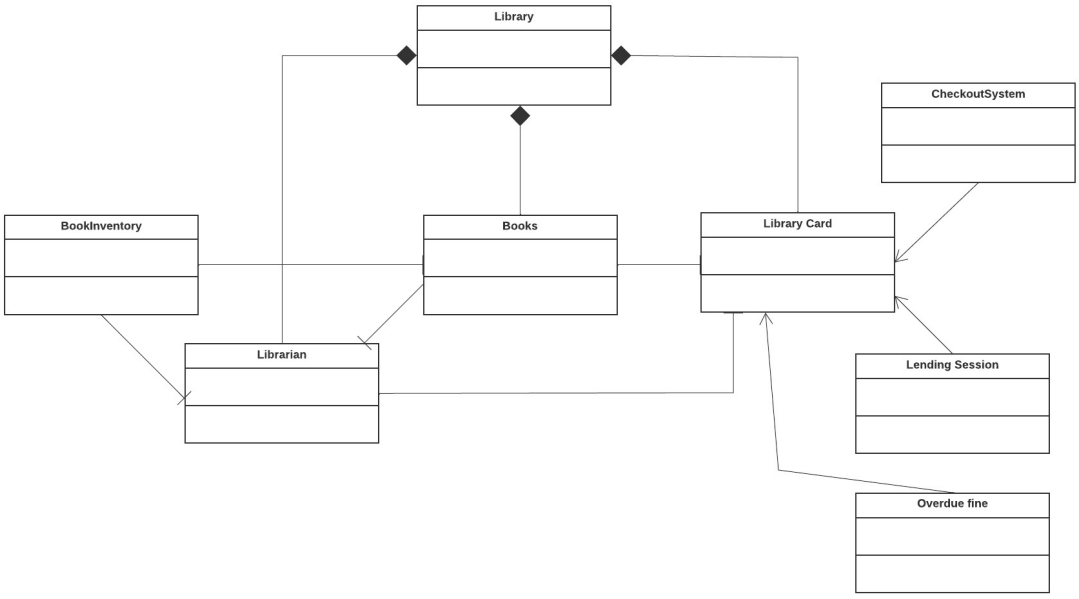
Virtual methods are present in the base class, but when virtual methods are called and the object that calls it is of the derived type, then the derived type's method is called instead of the base type's method. we know every phone has a color, so we defined a method to print the color. but in the base class, we just use it as a construct and add our properties in the child class. Now, when I call the apple object instance, it will print the phone red. Thus, polymorphism plays a huge role in the development of the code.

There are two types of polymorphism. One is Compile-time polymorphism and another is run-time polymorphism. Method overloading and operator overloading is the example of compile time polymorphism and method overriding is an example of run-time polymorphism.

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

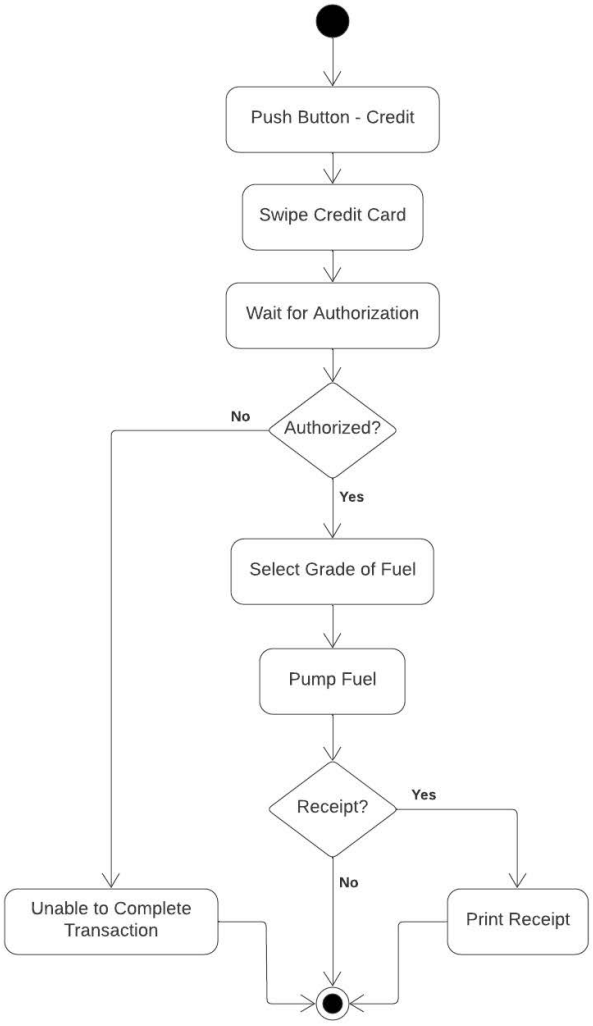


5. **Draw a class diagram of a library lending books using the following classes: Librarian, Lending Session, Overdue Fine, Book Inventory, Book, Library, Checkout System, and Library Card.**



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?"**

Activity Diagram of Pumpng Gas and paying by Credit Card



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

UML class diagrams model the classes of the system and their relations(part-of, is-a). A class dependency graph is a directed graph where nodes are all classes of the program and edges are all dependencies. Let's say I want to change some things in a class, and that class takes attributes from some other different classes. so, the class is dependent on other classes. this is called dependency between classes. this dependency is modelled in the CDG. If there is a part-of relation between classes A and B, there is also a dependency A, B. If there is a nonpolymorphic inheritance of X from Y, then X, Y is a dependency. If there is a polymorphic inheritance between X and Y, then both X, Y, and Y, X are dependencies. A CDG is used to model the dependencies.