Rohith Jayarajan
StudentID: 115458437

**Software Engineering**

**3.13**
**What is inheritance in object-oriented technology? Give an example.**

The relation between the two classes: the base class and and the derived class is called as inheritance. The base class or superclass is what defines the base type and the derived class or subclass is what defines the derived type. The class members that are shared among all derived classes is defined by the base class. The members that are specified to the particular class is contained by the derived class.

An example of inheritance is as follows, where `class Fruit` inherits from public `class Food`

```
class Food
   {
   private:
         String name;
         String color;
   };

class Fruit: public Food
   {
   public:
         quantities_to_buy(int buy_quantities);
         quantities_to_sell(int sell_quantities);
   private:
         int calories;
   };
```

**3.14**
**What is the difference between an object and a class in OO technology?**

A class can be defined as a template for objects. A class defines the properties for an object which includes but is not restricted to a valid range of values, and a default value. The behavior of the object is defined by the class. The “instance” of a class is defined as an object.

| S.No. | Object | Class |
|---|---|---|
| **1.** | An instance of a class is called as an object. | Class if defined as a template. It is from this template that objects are created. |
| **2.** | Object is a physical entity. | Class is a logical entity. |
| **3.** | Objects can be created multiple times. | A class declaration happens only once. |
| **4.** | This is an example of how objects are created: | This is an example of how classes are created: |

| | | |
|---|---|---|
| | Fruit orange();<br>Here orange is an instance(object) of class Fruit | class Fruit()<br>Here Fruit is the class |

**3.15**
**Describe the role of polymorphism in object-oriented technology. Give an example.**

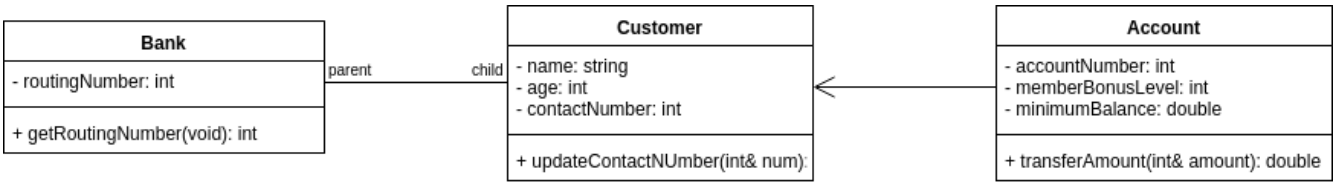
Polymorphism is derived from the Greek roots Poly=many and Morph=change of form. Polymorphism is a construct of object-oriented technology that allows the use of a derived type in the place of the base type, for example, in a variable declaration, function parameter, and so forth. Polymorphism is the ability in programming to present a same interface for different underlying data types.

An example of polymorphism is as follows where the method `getHorsePower` has different meaning in every derived type:

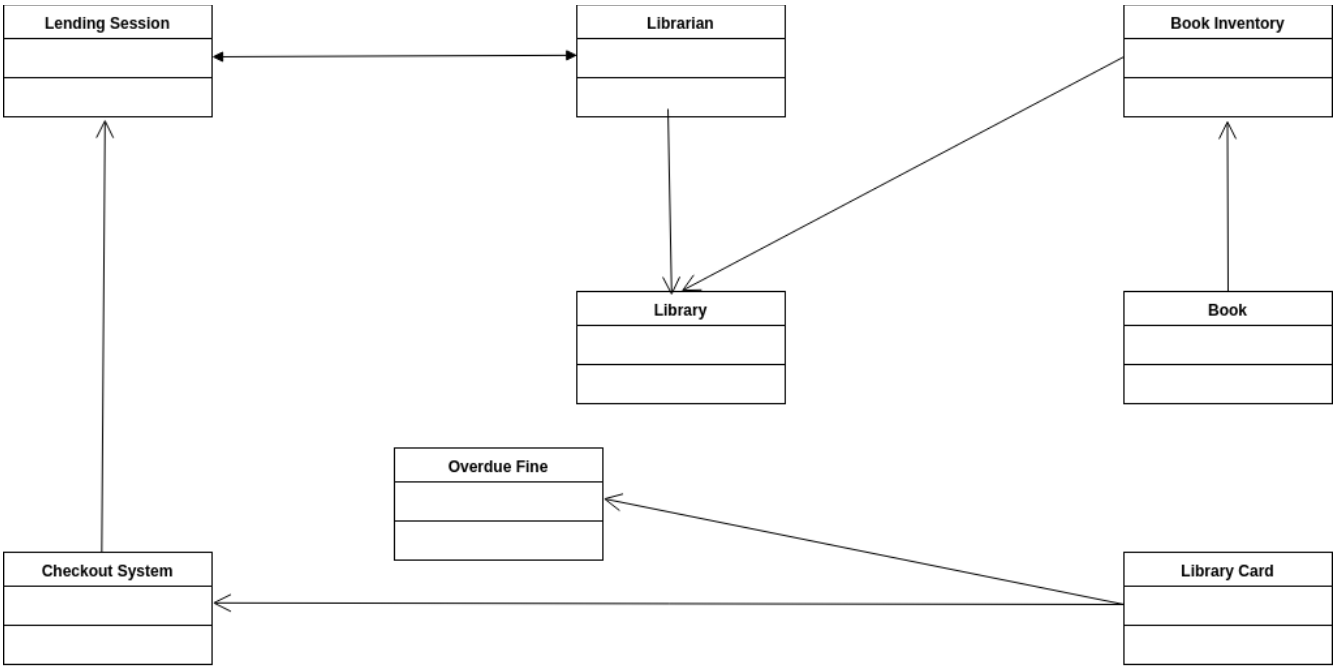
```
class Car
   {
   public:
         virtual void getHorsePower() {}
   };

class Prius : public Car
   {
   public:
         void getHorsePower() {cout<<" 200hp";}
   };

class TeslaModelS : public Car
   {
   public:
         void getHorsePower() {cout<<" 259hp";}
   };
```

**4.1**
**Draw a class diagram of a small banking system showing the associations between three classes: the bank, customer, and the account.**

Rohith Jayarajan
StudentID: 115458437




**4.2 (provide one-sentence definition of each class)**
**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.**




Librarian: Librarian has access to members of public class Library and has the methods to start the lending session. It also will have access to the methods of public class Lending Session.

Lending Session: This class has access to members of public class Librarian.

Overdue Fine: This keeps a track of the overdue fine for a given Library card

Book Inventory: The book inventory tracks the books available in the library and updates that database in the Library class.

Book: The status of available/checked-out books is stored in the Book class and updates the Book Inventory accordingly.

Rohith Jayarajan
StudentID: 115458437

Library: The library maintains a database of the book inventory that is updated and accessed by the Book Inventory and accessed by the Librarian.

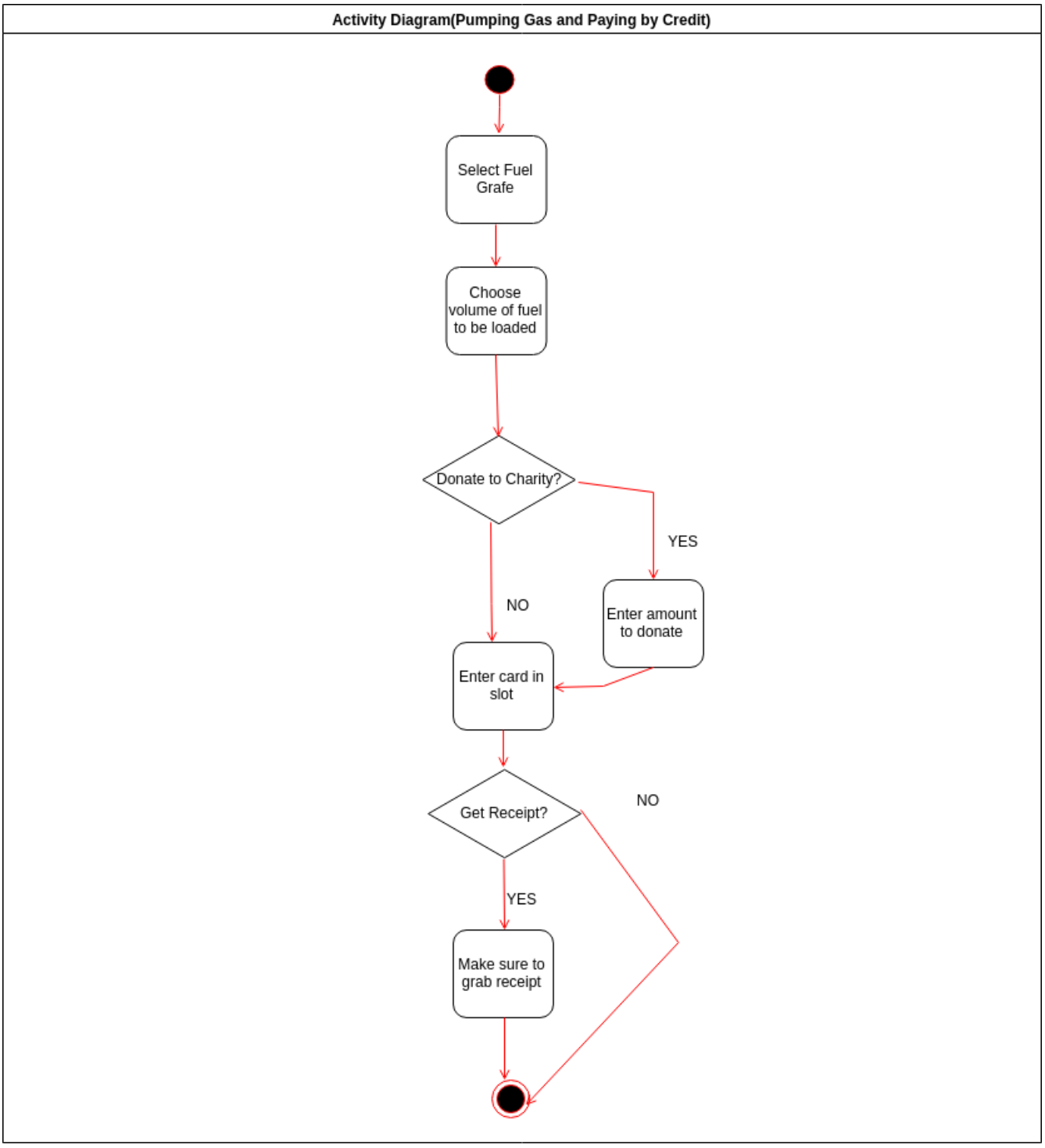
Checkout System: The checkout system has access to the public members of the Lending Session Class and is used to checkout/lend a book from the lending session to library card holder

Library Card: The library card class keeps track on the overdue fine and also used to update the inventory by checking out a book in the lending session.

**4.3**

**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?”**

Rohith Jayarajan
StudentID: 115458437




**4.5**

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

Week2 Written Responses

Rohith Jayarajan
StudentID: 115458437

A class dependency graph is a directed graph where nodes are all classes of the program and edges are all dependencies. UML class diagram is more extensively used than class dependency graphs. Also if we are more concerned about the logic of the algorithm, then we prefer class dependency graphs and on the other hand, UML class diagrams are used to facilitate better coding practices by modeling the static structure of the model. Class dependency graphs introduce a notion of responsibility whereas UML class diagrams don’t. The relationship between classes as supplier and client is the emphasis in Class dependency graphs whereas in UML class diagram, the relationship between classes is part-of and is-a.