Week 2 Written responses

Name: Rohitkrishna Nambiar
Student ID: 115507944

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

Inheritance is one of the core concepts in object oriented programming where we can create a new class with characteristics of another class. This way, the new class is called the derived class and the class from which it has inherited members is called the parent class. This will be better explained from the example below:

```
#include <iostream>
using namespace std;

class Fish{
   protected:
      double weight_;
   public:
      void setWeight(double weight){
         weight_ = weight;
      }
};

class Tuna : public Fish{
   public:
      float cost(){
         return weight_ * 50;
      }
};

class Salmon : public Fish{
   public:
      float cost(){
         return weight_ * 45;
      }
};

int main()
{
   Salmon s;
   Tuna t;
   s.setWeight(100);
   t.setWeight(95);
   cout << "Salmon cost for 100 pound is $" << s.cost() << endl;
   cout << "Tuna cost for 95 pound is $" << t.cost();

   return 0;
}
```

Here in this example we see that Fish is the base class and Salmon and Tuna are the derived class. Both of the fishes have a common attribute called weight which is set through a function written in base class. The function to calculate cost is dependent on the fish and is written in the derived class. The protected variable weight_ is only accessible to the base and inherited class.

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

A class is a template for an object. A class defines the names and types of methods and variables that can exist for an object. We write a class only once using class keyword, but we can create multiple objects for a class using the new keyword. No memory is allocated for a class, but only when an object of that class is created, memory is allocated. We can think of class as a type and object as a variable of that type.

ex. If we have a class Car, we can have one of object as BMW and another object as Audi.

In object oriented programming, our entire program deals with manipulation of objects which have methods and attributes.

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

Polymorphism in object oriented technology is used when we have classes related by inheritance. It means that we can use a derived class member function in place of the base class. We do this by declaring the base class method as virtual. This is called dynamic binding and the implementation is decided based on the object calling the function. If we have no specific implementation in the base class, we declare the function as a pure virtual function.

Ex: Using the same example we used in question 3.13, we add the cost method in the base class with the following implementation.

```
// Changes to class Fish
float cost(){
        return 0;
    }

// Changes to main
int main()
{
   Salmon s;
   Tuna t;

   s.setWeight(100);
   t.setWeight(95);

   Fish *f;
   f = &s;
```

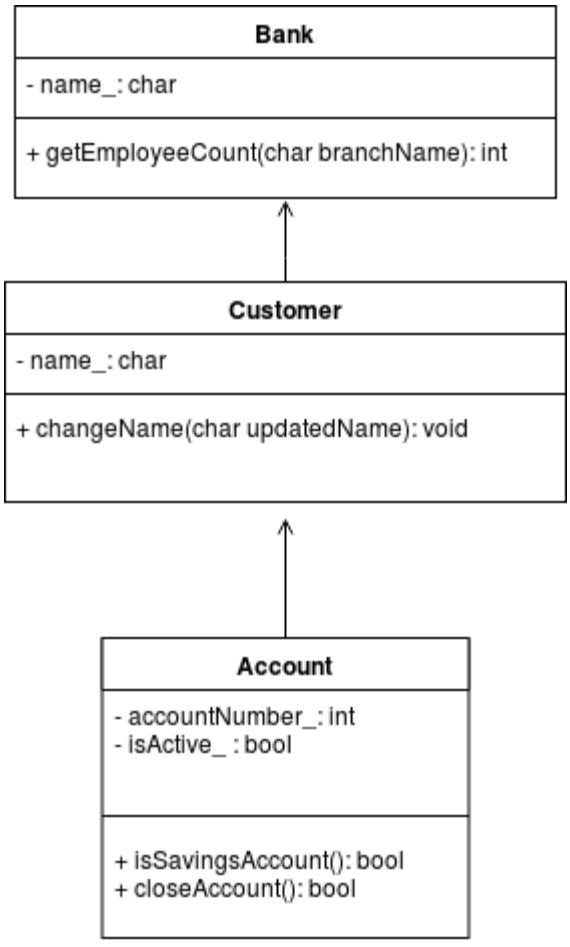
```
    cout << "Cost for fish is $" << f->cost() << endl;

    return 0;
}
```

In this case, when we use a base class pointer and call the method cost, we get the output as 0. To use the implementation of the derived class we declare the cost function as a virtual function.

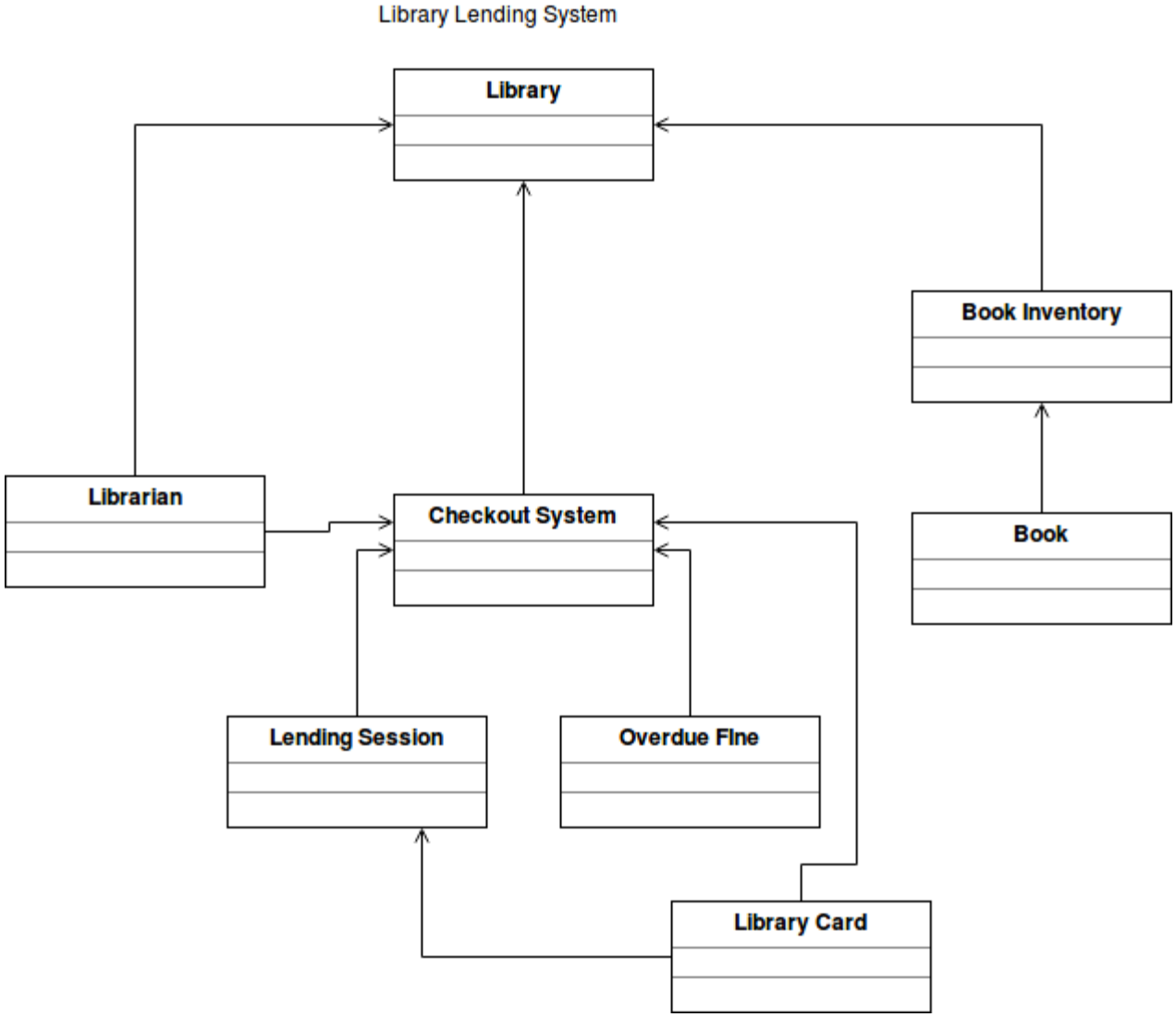
```
// Changes to class Fish
virtual float cost(){
        return 0;
    }
```

This, during execution, based on the object type, the appropriate base class method will be called. This is called polymorphism.

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



4.2 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.



Library: This class contains the information about the library.

Librarian: This class has the attributes and member functions for librarian. It is a part of the Library class.

Checkout System: This class handles the checkout of the book. It is a part of the Library Class.

Lending Session: This is a part of the Checkout System class. It uses the Library Card Class.

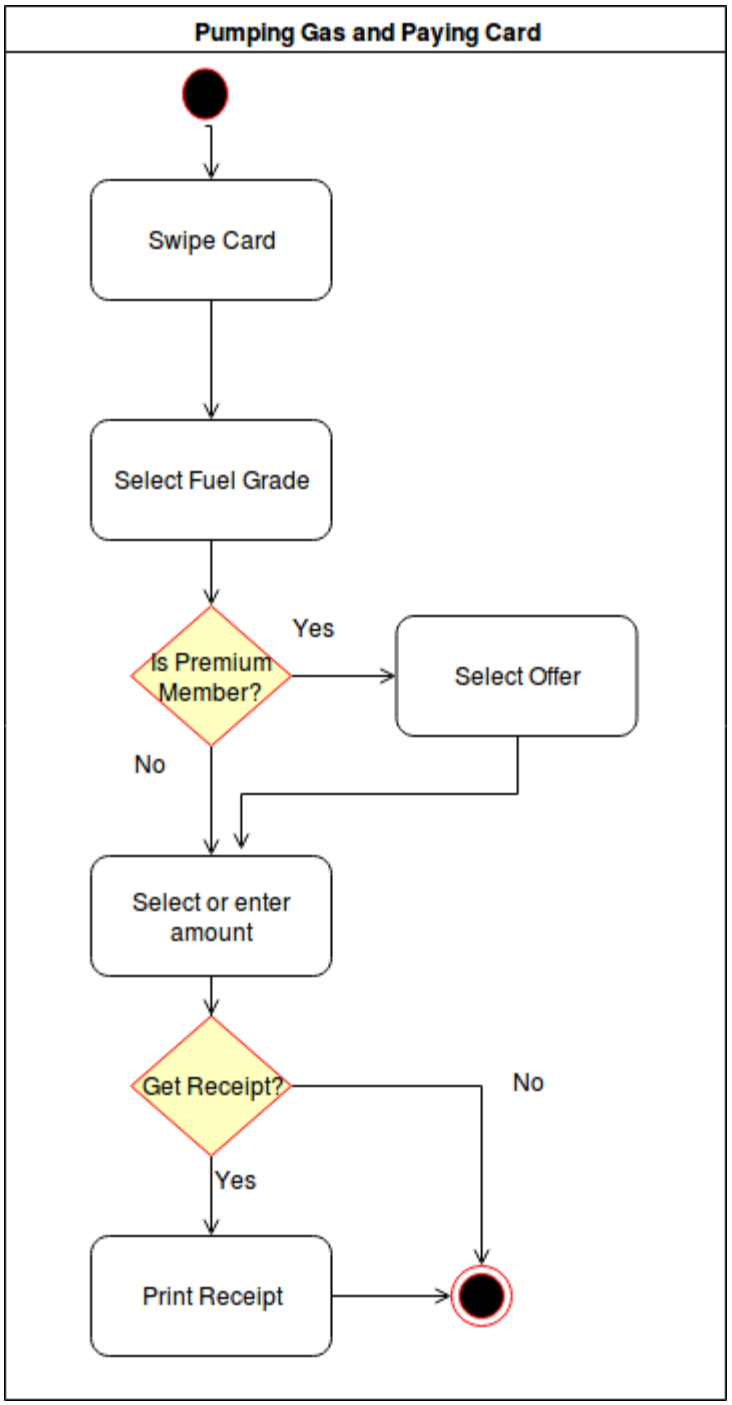
Overdue Fine: This class handles the methods and attributes for overdue fines. It is a part of the Checkout System class.

Book Inventory: This class has information about all the books in the library. The Book Inventory class is a part of the Library class.

Book: This class has information about a particular book. It has methods and attributes for individual book. Part of the Book Inventory class.

Library Card: This class has methods and attributes about the library cards.

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?"



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

A class dependency graph is a directed graph where nodes are all classes of the program and edges are the dependencies. A class diagram is used for general conceptual modeling of the application.

The class dependency graph has a client supplier relationship between the classes whereas the class diagram has a is-a and a part-of relationship between the classes.

The class diagram is more popular than the dependency graph.