Week 2 Written Exercises

Software Engineering 3.13 - ***What is inheritance in object-oriented technology? Give an example.***
Ans. Inheritance is defined as a "IS-A" relationship between two classes. If class X "IS-A" type of class Y, then class Y is called a base class and X is called a derived class. Inheritance allows the use of all the member functions of base class in derived class.
e.g.

```
// Base Class
class Animal {
        public:
        void soundAnimalMakes();
        }

// Derived class
class Cow :: public Animal {
        public:
        void  eat();
        void sleep();
        // can also use Animal member function
        }
```

Software Engineering 3.14 - ***What is the difference between an object and a class in OO technology?***
Ans. Class is like a template which contains functionality. Objects are instances of the classes created to use that functionality. A class contains nothing but the properties or characteristics of the object and once an object is instantiated, these properties can be used by using a dot(.) operator.

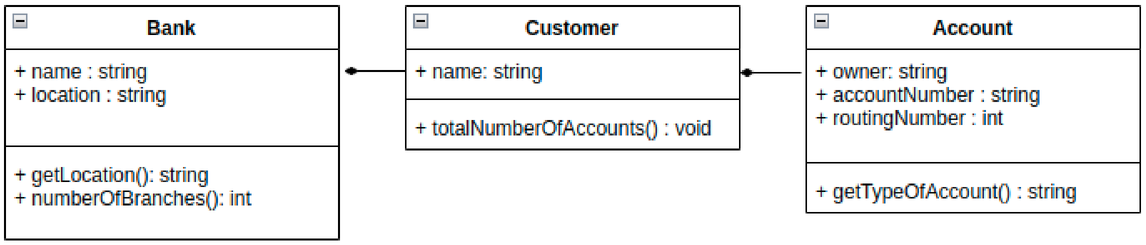
Software Engineering 3.15 - ***Describe the role of polymorphism in object-oriented technology. Give an example.***
Ans. Polymorphism literally means "many-forms". It is an essential part of OO technology that allows the same function to be reused in different ways without the need to write extra code.
e.g.

```
// Base Class
class Animal {
        public:
        void soundAnimalMakes(){std::cout << " animals makes some noise"};;
        }
// Derived class
class Cow :: public Animal {
        public:
        void  eat();
        void sleep();
        void soundAnimalMakes(){std::cout << " cow says Moo!"};
        }
// Derived class
class Dog :: public Animal {
        public:
        void  eat();
        void sleep();
        void soundAnimalMakes(){std::cout << " Dogs say woof!"};
        }
```

Here, the same function in all classes called "soundAnimalMakes()" outputs different things based on their respective classes when called in the main method, hence polymorphism is seen here.

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

Ans. Customer is a part-of bank. BankAccount is a part-of customer.



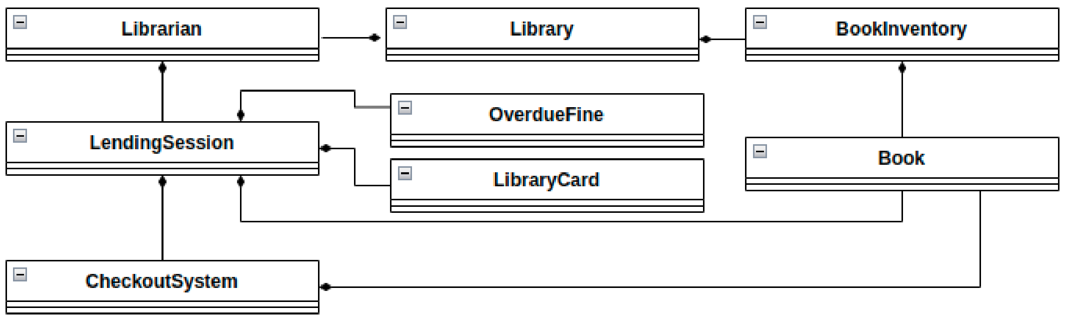
Software Engineering 4.2 - Draw a class diagram of a library lending books using the following classes: Librarian, Lending Session, Overdue Fine, Book Inventory, Book, Library, Checkout System, and Library Card.

Ans: The description of each class is as follows:

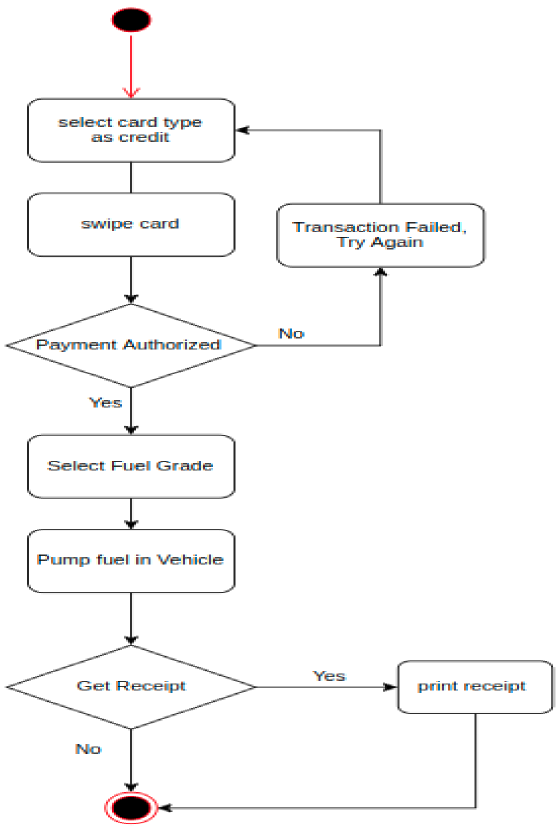
1. **Library**: Contains information about the Library including books, people who work there and the inventory..
2. **Librarian(***Part-of library*): Contains information about Librarian like name and member functions include starting new sessions to lend books and manage book checkout systems.
3. **Book Inventory(***Part-of library*): Contains information about all books like ID, title, status(checked-out, in library or due).
4. **Book(***Part-of book inventory*): Contains information about all books like genre, year of publication, ID, title, status(checked-out, in library or due).

5. **Lending session(***Part-of librarian*): Contains information about lending sessions eg. patron's name, library card number, book to issue, fines and due date.
6. **Overdue Fine(***Part-of lending session*): Class that calculates fines due for the patron, and due date.

7. **Checkout System(***Part-of librarian*): Contains information about book checked out, due date, issued to which patron, receipt info.
8. **Library Card(***Part-of lending session and checkout system*): Class that contains information about patron id, name.



Software Engineering 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?"



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

Ans. There is not much difference between CDG and UML as both are used to provide more structure and information for the code but here are few points:

CDG	**UML**
CDG are usually made after code has already been written.	UML is drawn before the coding begins. It is used for structuring the code.
CDG can easily determine *responsibility* of a class.	UML cannot easily determine and track things like responsibility and contracts.
Connections between classes are determined as supplier and client.	Connections between classes are determined as IS-A and Part-of.