Written Responses: Software Engineering

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

Inheritance is based on the observation that some classes share common members. It is a relation between two classes: 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 base class defines class members that are shared among all derived classes, while derived classes contain members that are specific to that particular class. (*Chapter 3.2.4)*. For example, both the banking customers, class Customer, and banking tellers, class Teller, share a name and an address from class Person. They are subclasses of class Person.

```
class Person
{
   private:
   stirng name;
   string address;
   };
class Customer: public Person
{
   public:
   open_account(int acount_number);
   //.....
   private:
   int account_number;
};
class Teller: public Person
{
   public:
   check_in();
   check_out();
   //......
   private:
   int teller_ID;
};
```

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

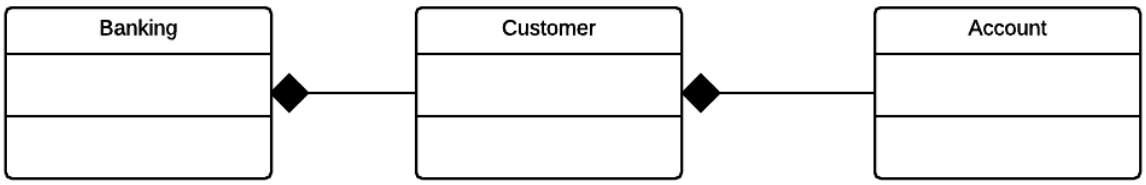
Objects in software represent the objects of the real world. (*Chapter 3.2.1*). Classes are modules of the source code that define the properties of several objects. (*Chapter 3.2.2*)

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

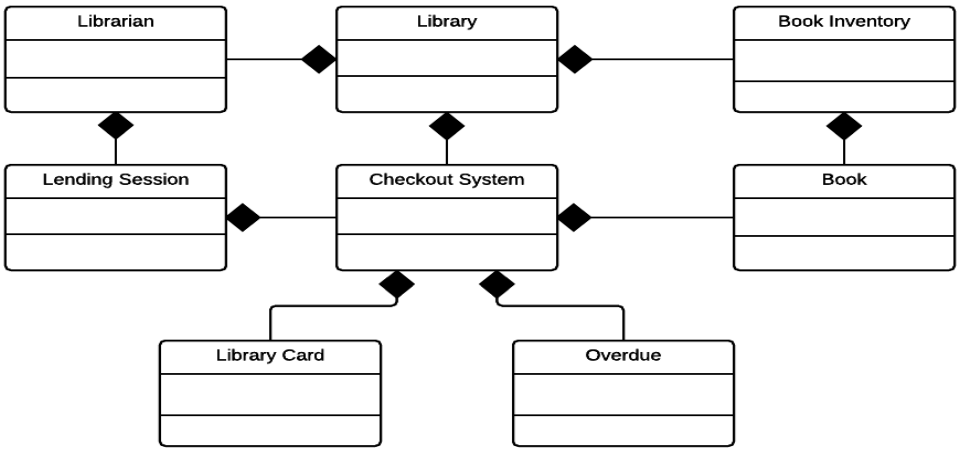
Polymorphism is a construct of object-oriented technology that allows the use of a derived type in the place of the base type (*Chapter 3.2.5*). In the following codes, a virtual method is present in the base type, and then, subclasses can call the virtual method and derive it to different methods.

```
class FarmAnimal
{
   public:
   virtual void makeSound() {}
};
class Cow : public FarmAnimal
{
   public: void makeSound () {cout << " Moo-oo-oo";}
};
class Sheep : public FarmAnimal
{
   public:
   void makeSound() {cout<<"Be-e-e";}
};
```

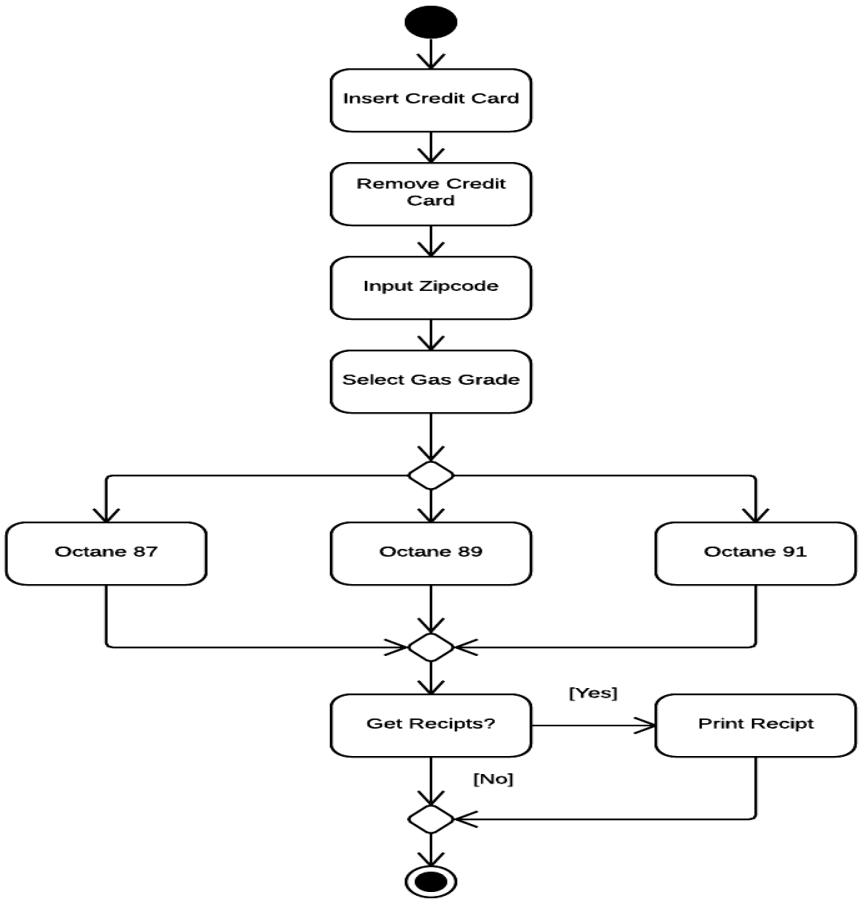
Exercise 4.1: Draw a class diagram of a small banking system showing the associations between three classes: the bank, customer, and the account.



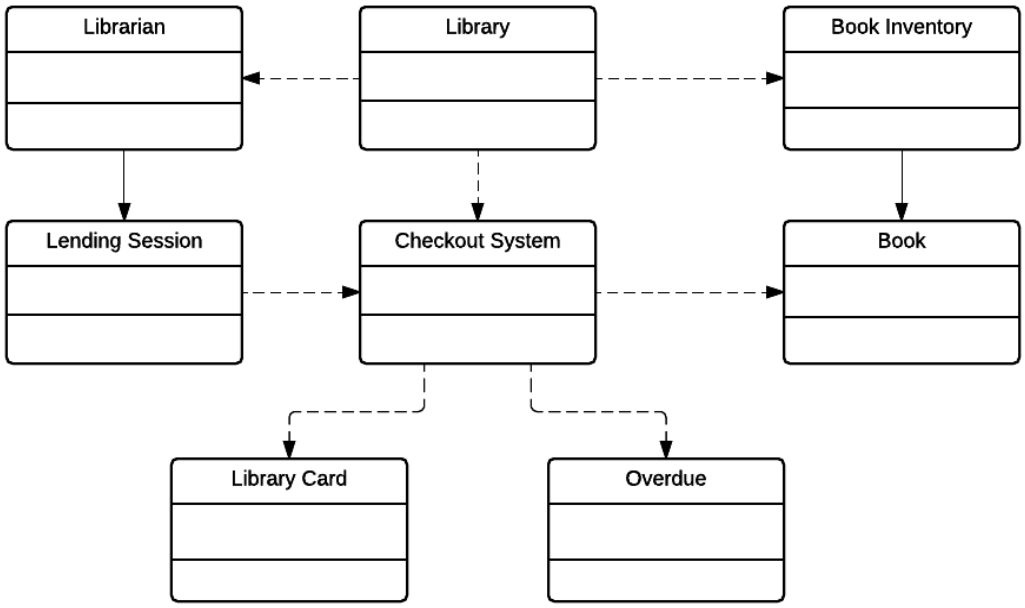
Exercise 4.2: Draw a class diagram of a library lending books using the following classes: Librarian, Lending Session, Overdue Fine, Book Inventory, Book, Library, Checkout System, and Library Card



Exercise 4.3: Draw an activity diagram of pumping gas and paying by credit card at pump. Include at least five activities, such as "Select fuel grade" and at least two decisions, such as "Get receipt?"



Exercise 4.4: Draw a class dependency graph of a software for a library lending books.



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

UML class diagram: represent *Associations* between two classes. UML models either static structure (for example, with class diagrams) or dynamic properties (for example, with activity diagram) (Chapter 4 Summary).

Class dependency graph: represent *dependency* between two classes. Software evolution uses dependency diagrams that reflect delegation of responsibilities among the classes.