# Final Term Exam

**Note: Attempt all question**

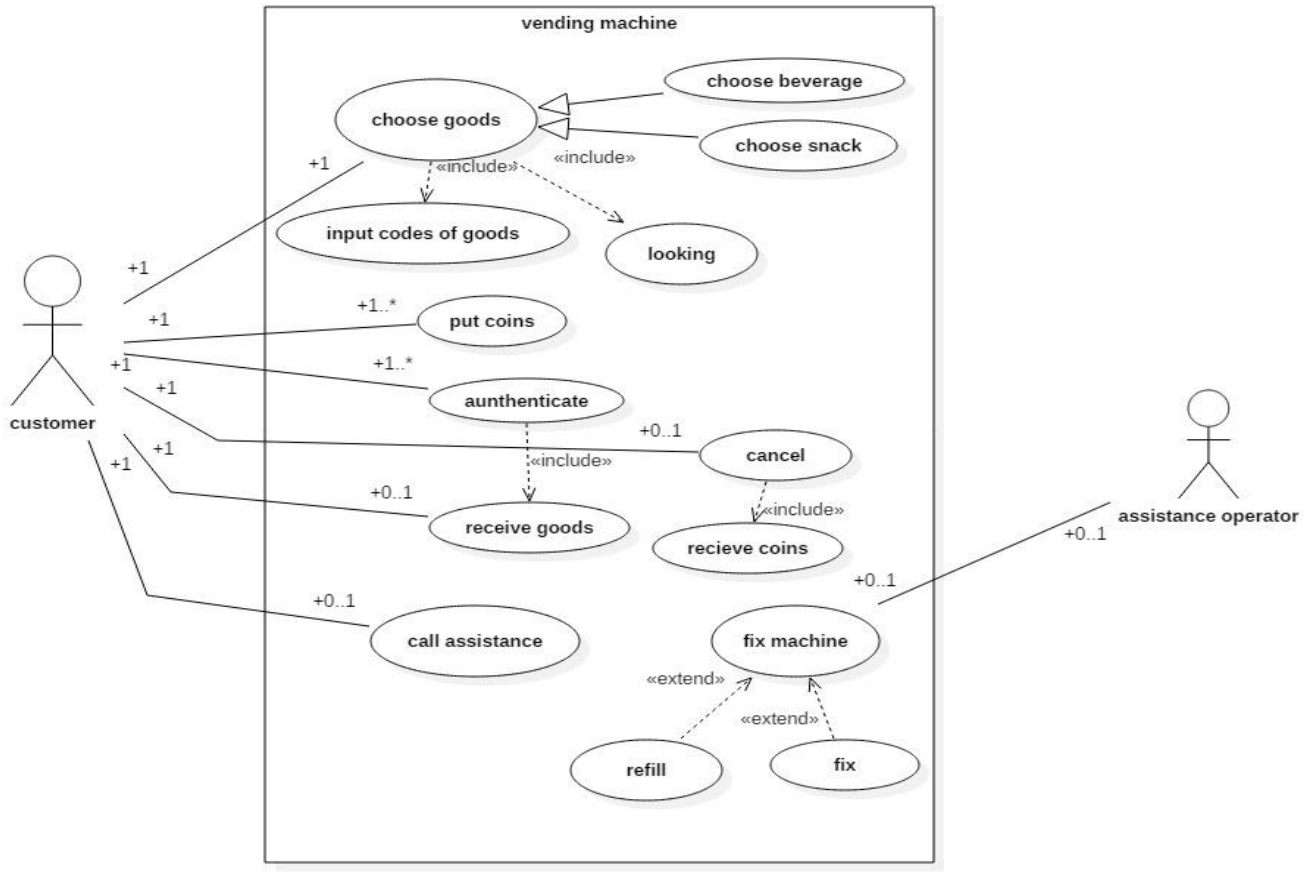
**June 23, 2020**

| Name:Abdul Musawir | Roll NO: 12991 | Class/Section:BSSE VIII A |
|---|---|---|

## Q1: Draw Use Case diagram 10 Marks

Propose a use case diagram for a vending machine that sells beverages and snacks. Make use of inclusion and extension associations and remember that a vending machine may need technical assistance from time to time.

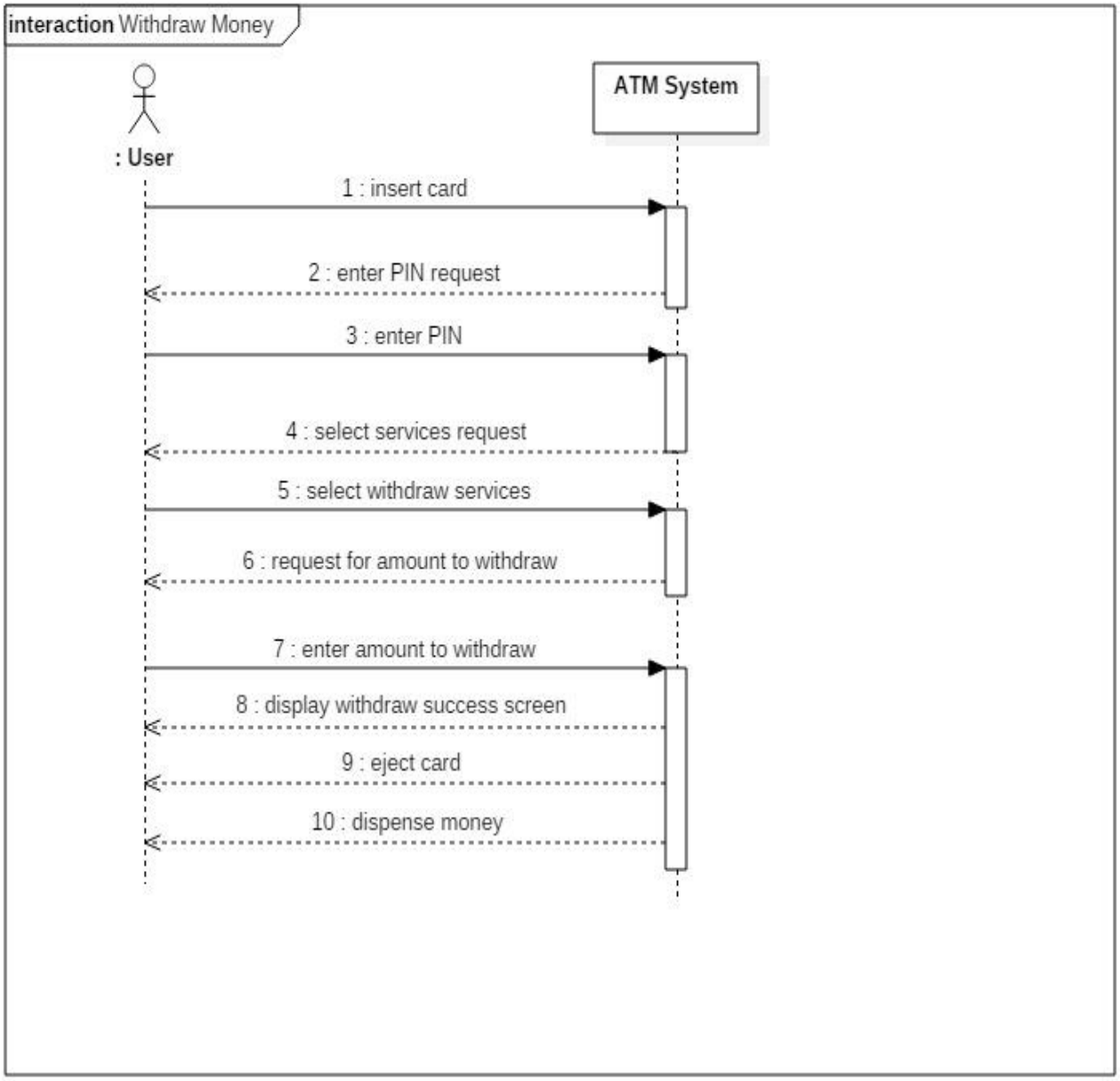
**Answer:**




## Q2: Draw Sequence Diagram 10 Marks

Model a scenario of the Withdraw Money use case of a Bank ATM system. The user is able to make withdrawal of money. The system employs a standard procedure of validating the card and account holder's password.

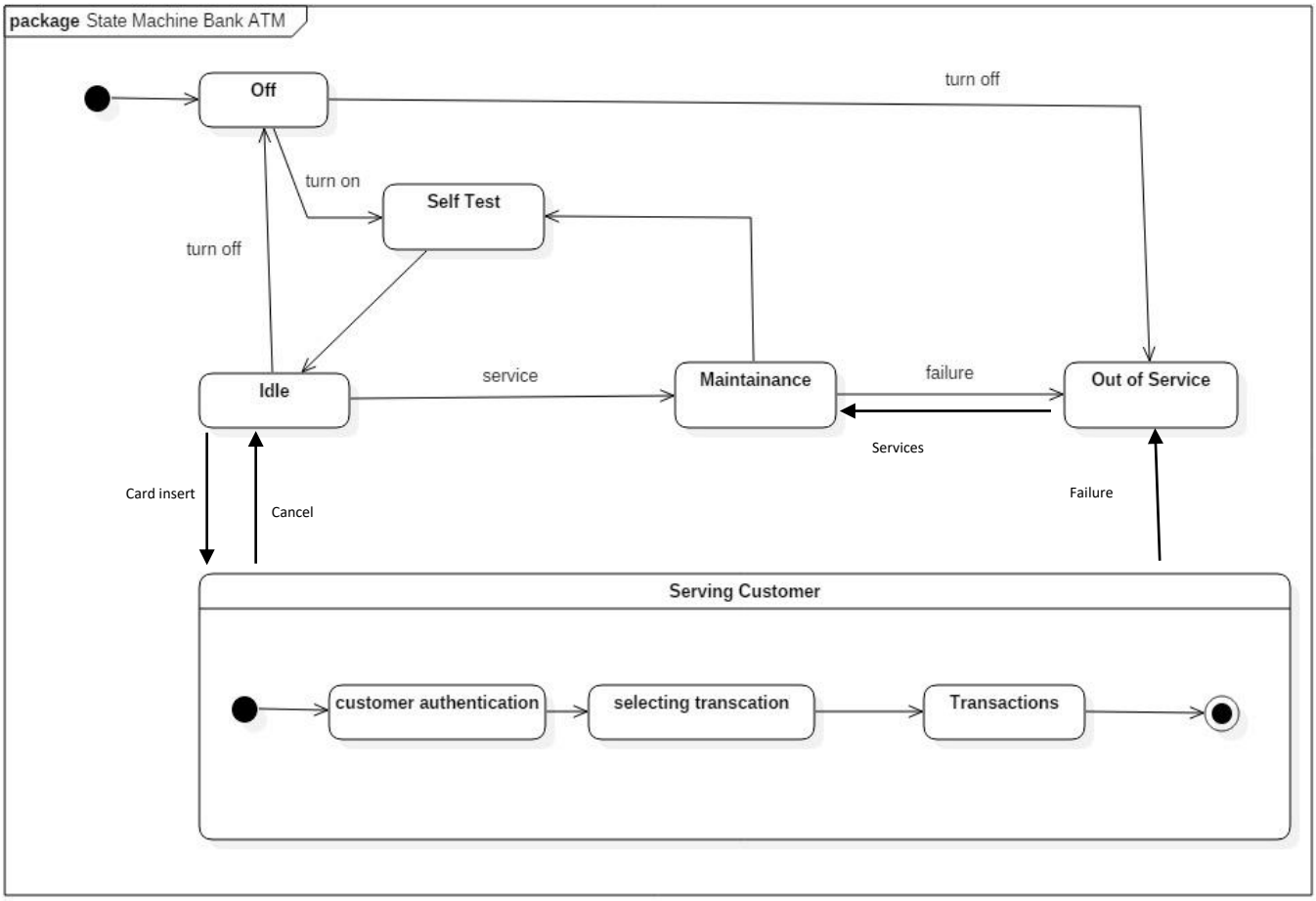
**Answer:**




## Q3: Draw State chart diagram **10 Marks**

ATM is initially turned off. After the power is turned on, ATM performs startup action and enters Self Test state. If the test fails, ATM goes into Out of Service state, otherwise transition to the Idle state. In this state ATM waits for customer interaction. The ATM state changes from Idle to Serving Customer when the customer inserts banking or credit card in the ATM's card reader. On entering the Serving Customer state that is composed of basic ATM functions i.e authentication, money withdrawal etc

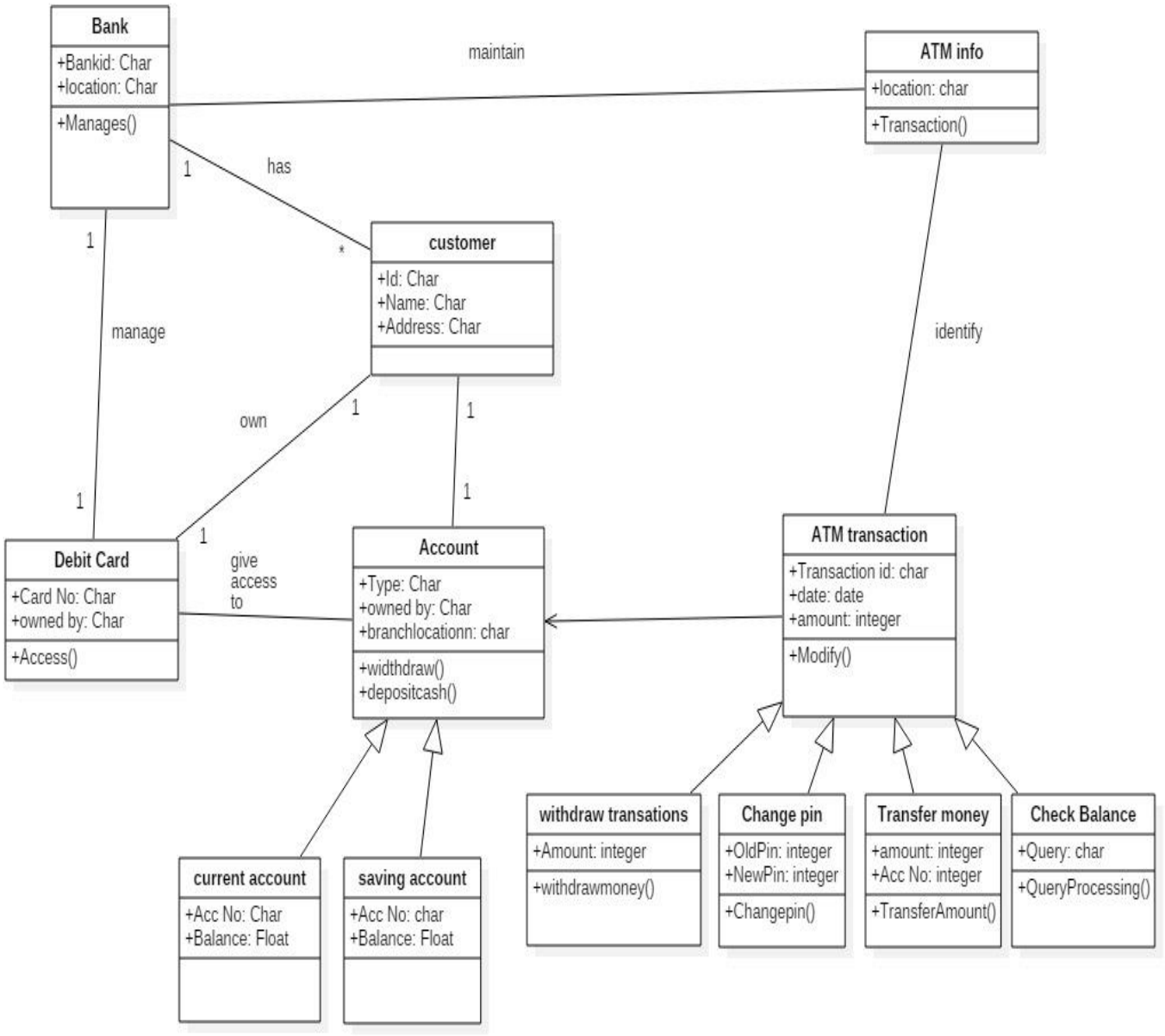
**Answer:**




## Q4: Draw Class Diagram 10 Marks

Illustrate Class diagram for ATM Machine. The various Classes involved in the system are: Bank, Account, Customer Info, Debit Card, Current Account, Saving Account, ATM Info, ATM Transaction, Withdraw Transaction, Change Pin, Transfer Money, Check Balance. The Bank maintains personal and ATM information of each customer. The customer can access their account using Debit Card issued by the Bank. In this system there could be two types of Account: Current Account and Saving Account. Both use to share many of the properties and methods. The ATM Machine can perform multiple transactions such as Withdrawing cash, change pin, check balance and Transfer Money to each account.

**Answer:**




## Q5: Design Pattern 10 Marks

Suppose we have the following java files. Identify the pattern also Considering the java files draw class diagram.

(answer is mentioned at the end of question)

*Shape.java*

```
public interface Shape {
   void draw();
}
```

*Rectangle.java*

```
public class Rectangle implements Shape {

   @Override
   public void draw() {
      System.out.println("Shape: Rectangle");
   }
}
```

*Circle.java*

```
public class Circle implements Shape {

   @Override
   public void draw() {
      System.out.println("Shape: Circle");
   }
}
```

*ShapeDecorator.java*

```
public abstract class ShapeDecorator implements Shape {
   protected Shape decoratedShape;

   public ShapeDecorator(Shape decoratedShape){
      this.decoratedShape = decoratedShape;
   }

   public void draw(){
      decoratedShape.draw();
   }
}
```

*RedShapeDecorator.java*

```
public class RedShapeDecorator extends ShapeDecorator {

   public RedShapeDecorator(Shape decoratedShape) {
      super(decoratedShape);
   }

   @Override
   public void draw() {
      decoratedShape.draw();
      setRedBorder(decoratedShape);
   }

   private void setRedBorder(Shape decoratedShape){
      System.out.println("Border Color: Red");
   }
}
```

*DecoratorPatternDemo.java*

```
public class DecoratorPatternDemo {
   public static void main(String[] args) {

      Shape circle = new Circle();

      Shape redCircle = new RedShapeDecorator(new Circle());

      Shape redRectangle = new RedShapeDecorator(new Rectangle());
      System.out.println("Circle with normal border");
      circle.draw();

      System.out.println("\nCircle of red border");
      redCircle.draw();

      System.out.println("\nRectangle of red border");
      redRectangle.draw();
   }
}
```

---

**Answer:**

Decorator pattern allows a user to feature new functionality to present object without modifying its structure. This sort of design pattern proceed under structural pattern as this pattern proceed as a cover to existing class. This pattern makes a decorator class which covers the first class and deliver extra functionality keeping class methods signature intact. We are demonstrating the utilization of decorator pattern via following code during which we'll decorate a shape with some color without modify shape class.

**Class diagram**