Object Oriented Design and Modeling through UML

Chapter-4

Object Oriented Implementation

Facilitator: Er. Rudra Nepal

Mapping Design to Code

- While mapping UML class diagram to particular object-oriented programming language, we need to implement classes and interfaces definitions and methods and attributes definitions.

Mapping Design to Code

- It includes :
  1. **Defining class with simple methods and attributes :**
     - Here, we follow a particular syntax in order to represent the methods and attributes of a given class.
     - **For example,**

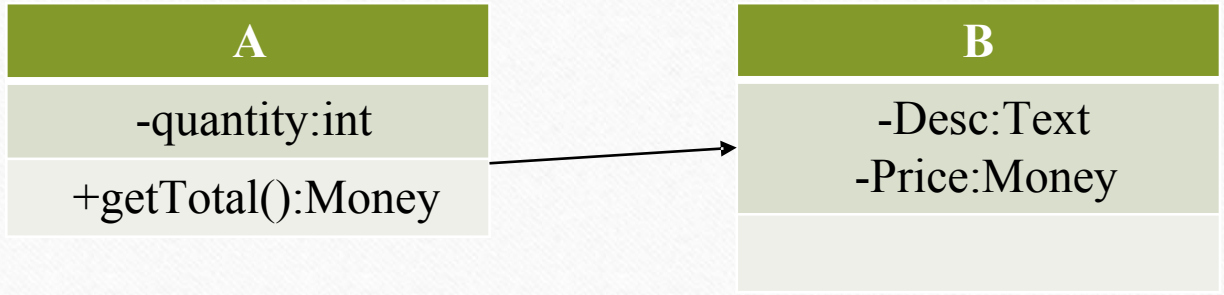
A
-quantity:int
+getTotal():Money

Defining class with simple methods and attributes

```
public class A
    {
    private int quantity;
    public A(int qty)
    {
        this.quantity=qty;
      }
    public Money getTotal()
    {
          ...
      }
}
```

Mapping Design to Code

2. **Adding reference attribute:**
   - A reference attribute is an attribute that refers to a particular class type object that is defined inside another class to maintain the visibility between two classes.
   - It is different from simple attributes of type such as in integer, float, string and so on.
   - **For example,**



Adding reference attribute

```
public class A
    {
    private int quantity;
    private B b;
   public A(int qty)
    {
       this.quantity=qty;
       b=new B();
     }
   public Money getTotal()
    {
          ...
     }
}
```

Mapping Design to Code

3. **Role names and reference attributes:**
   - A role can be defined as the responsibility of one class in another.
   - It is recommended that if the role of a particular class is given along with the association, we should name the reference attribute according to the role name.
   - **For example,**

Role names and referenc attributes

```
public class A
    {
    private int quantity;
    private B description;
   public A(int qty)
    {
       this.quantity=qty;
       description=new B();
     }
   public Money getTotal()
    {
          ...
     }
}
```

Mapping Design to Code

4. **Mapping some attributes from design to code:**
   - In some programming language, it is allowed to represent two attributes with a single attribute name while implementing class diagram to code.
   - **For example,**

Sale
-date:Date -time:Time -isComplete:Boolean

Mapping some attributes from design t code

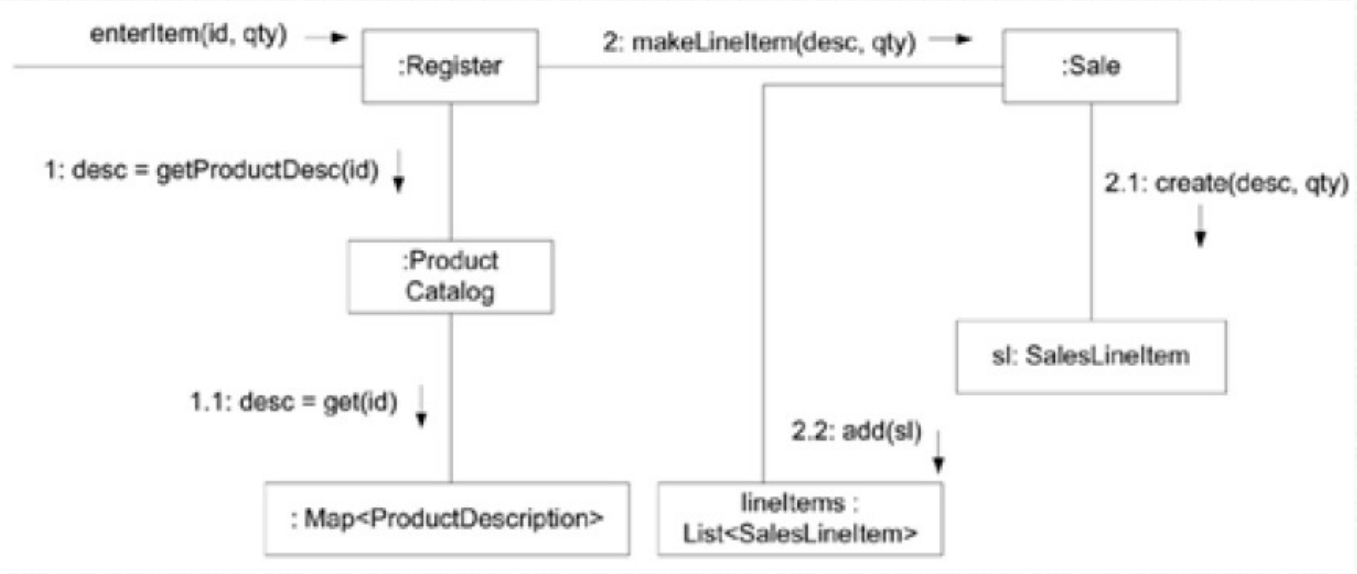
```
import java.util.Date;
public class Sale
    {
    Date dateTime=new Date();
    …
}
```

Mapping Design to Code

5. **Creating methods from collaboration diagram:**
   - All the incoming messages or methods in a class in collaboration diagram must be represented in that particular class in class diagram.

Mapping Design to Code

5. **Creating methods from collaboration diagram:**
   - **For example:**



Mapping Design to Code

5. **Creating methods from collaboration diagram:**
   - **For example:**

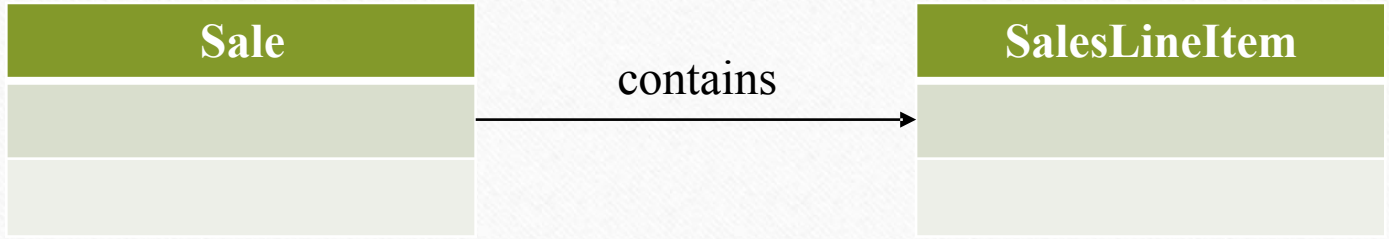
Register
+enterItem(id,qty)

Mapping Design to Code

6. **Updating class definitions:**
   - If the desired software is not built during the implementation phase, the developer then should perform **reverse engineering** i.e. the generation of UML diagrams from the particular code in order to build the software as per the users requirement.

Mapping Design to Code

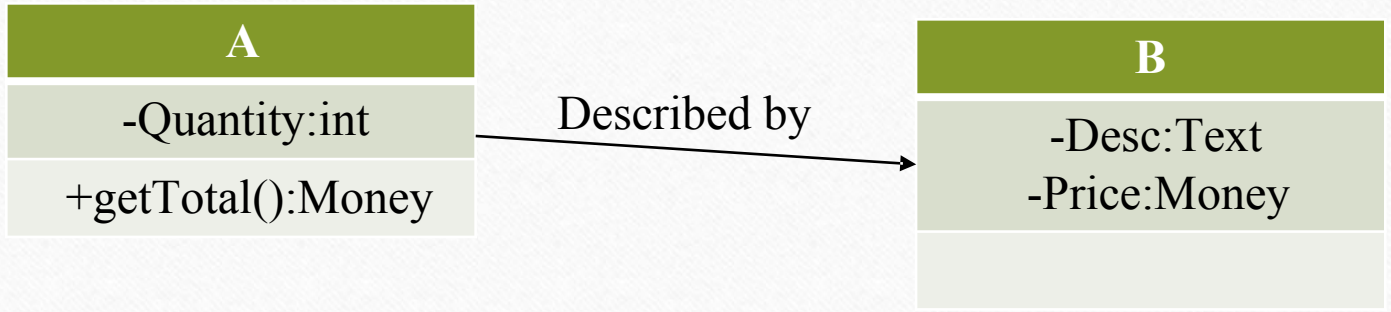
7. **Container or collection classes in code:**
   - If a particular class contains multiple objects of another class, we can use already defined container classes in that case.
   - **For example, in Java, we have already defined container class called ArrayList that can be used in order to show the composition or collection of multiple objects in a class.**



Container or collection classes in code

```
import java.util.ArrayList;
public class Sale
    {
    ArrayList <SalesLineItem> sales= new
ArrayList <SaleslineItem>();
    …
  }
```

Question number 1



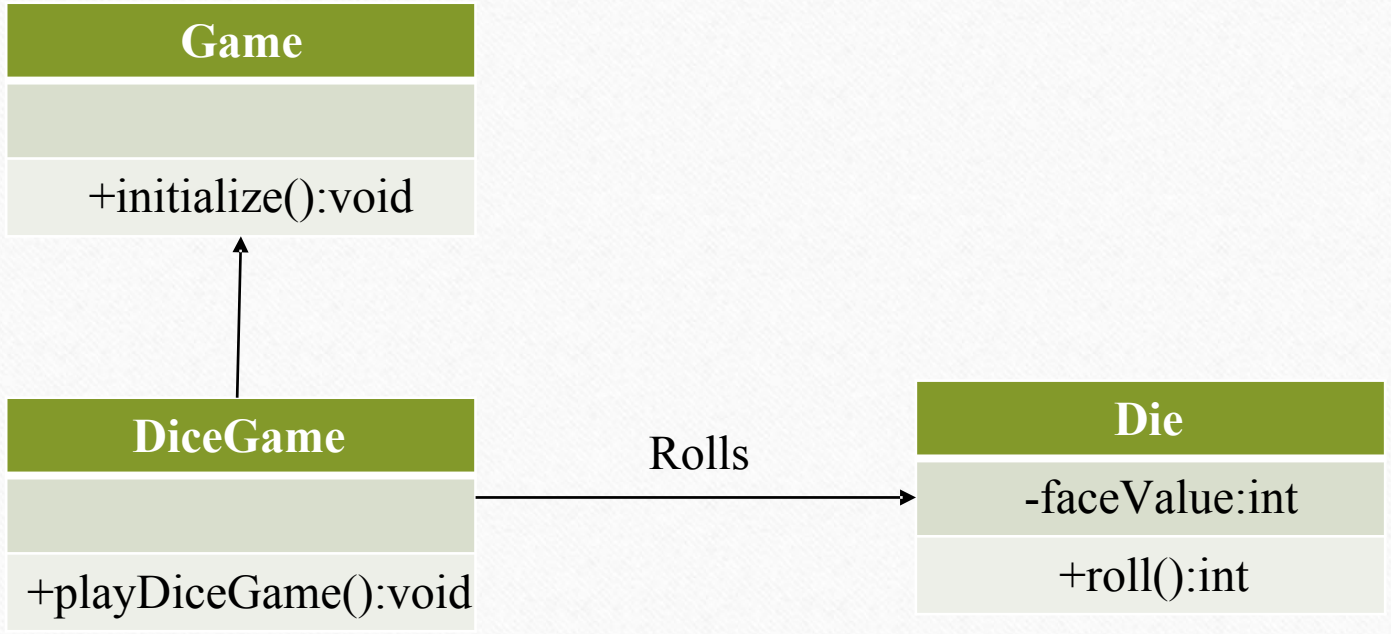
Solution of Question number 1

```
public class B
    {
     private Text Desc;
    private Money Price;
    public B(Text d,Money P)
     {
          this.Desc=d;
        this.Price=P;
      }
    public Money getPrice()
     {
          return Price;
      }
  }
```

Solution of Question number 1

```
public class A
    {
     private int Quantity;
    private B description;
    public A(int qty)
     {
          this.Quantity=qty;
        description=new B(“tshirt”,2000);
      }
    public Money getTotal()
     {
          return (description.getPrice()*Quantity);
      }
  }
```

Question number 2



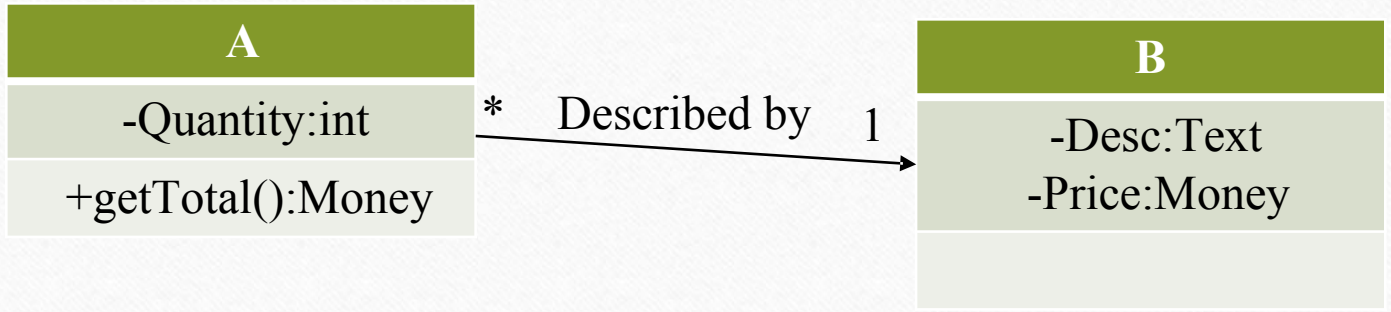
Solution of Question number 2

```
public class Game
        {
        public void initialize()
        {
                System.out.println("Game has begun");
        }
  }
public class Die
        {
        private int faceValue;
      public Die(int fv)
        {
                this.faceValue=fv;
        }
      public int roll()
        {
                return faceValue;
        }
  }
```

Solution of Question number 2

```
public class DiceGame
    {
     private Game g;
    private Die rolls;
    public DiceGame()
     {
          g=new Game();
        rolls=new Die(5);
      }
    public void playDiceGame()
     {
          g.initialize();
          System.out.println(“and the facevalue is
"+rolls.roll());
      }
  }
```

Question number 3



Solution of Question number 3

```
public class B
    {
     private Text Desc;
    private Money Price;
    public B(Text d,Money P)
     {
          this.Desc=d;
        this.Price=P;
      }
    public Money getPrice()
     {
          return Price;
      }
  }
```

Solution of Question number 3

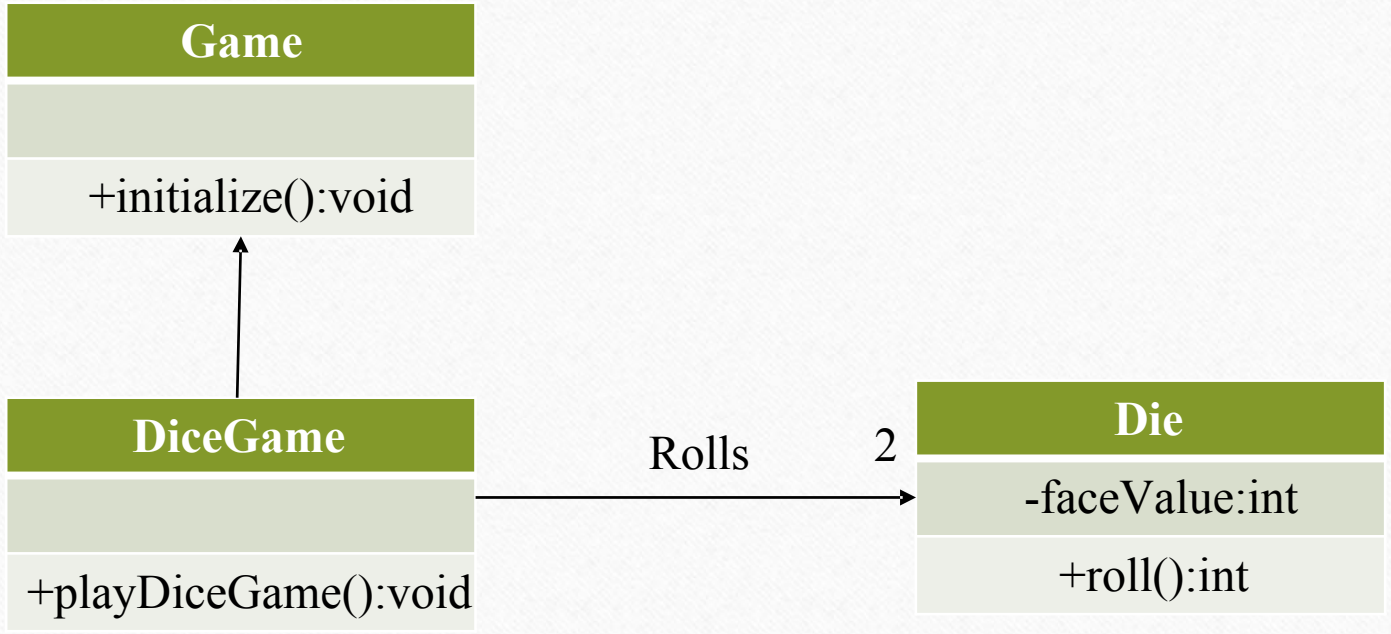
```
public class A
    {
    private int Quantity;
    private B description;
    public A(int qty)
    {
        this.Quantity=qty;
        description=new B("shirt",2500);
    }
    public Money getTotal()
    {
        return (description.getPrice()*Quantity);
    }
```

Solution of Question number 3

```
public static void main(String[] args)
        {
            Arraylist<A> list=new ArrayList<A>();
          A a1=new A(5);
          A a2=new A(10);
          A a3=new A(15);
          list.add(a1);
          list.add(a2);
          list.add(a3);
          for(A a:list)
              {
              System.out.println("Total price for item
is"+a.getTotal());
              }

        }
   }
```

Question number 4



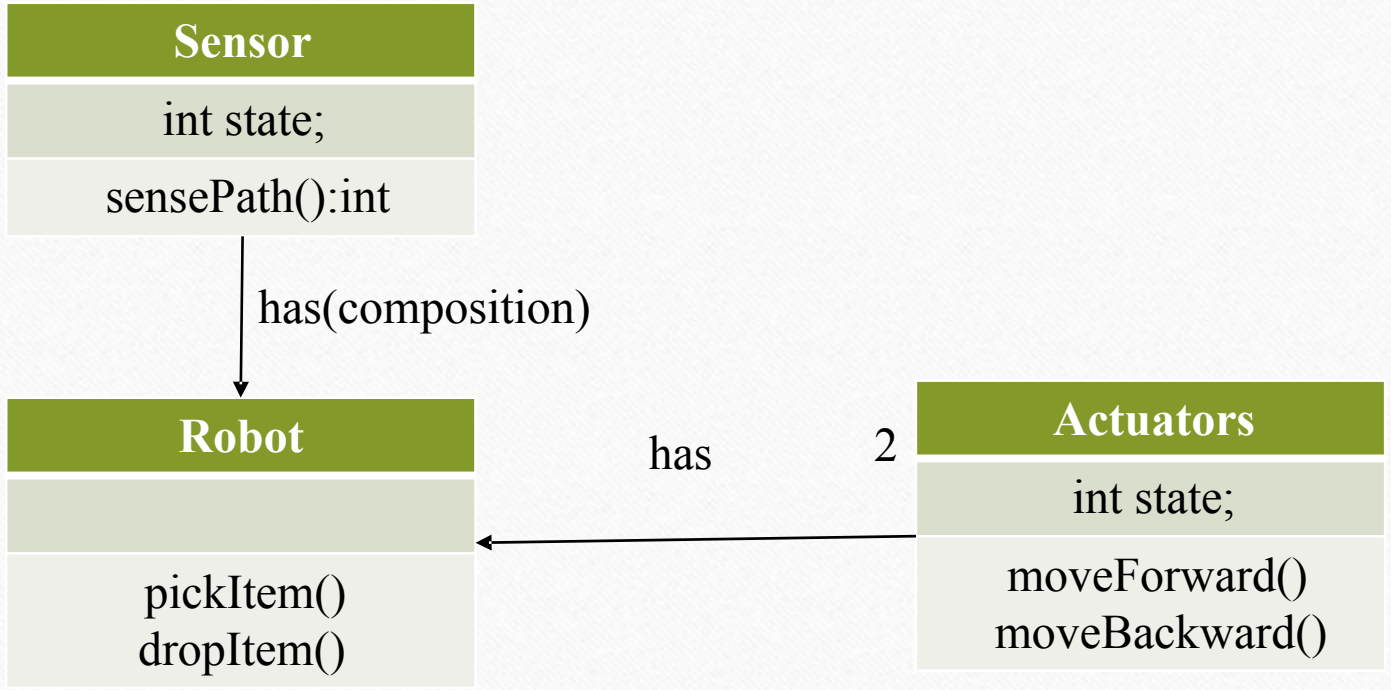
Solution of Question number 4

```
public class Game
        {
        public void initialize()
        {
                System.out.println("Game has begun");
        }
  }
public class Die
        {
        private int faceValue;
      public Die(int fv)
        {
                this.faceValue=fv;
        }
      public int roll()
        {
                return faceValue;
        }
  }
```

Solution of Question number 4

```
public class DiceGame
    {
    private Game g=new Game();
   public static void main(String[] args)
   {
    ArrayList<Die> rolls=new ArrayList<Die>(2);
    Die d1=new Die(4);
    Die d2=new Die(6);
    rolls.add(d1);
    rolls.add(d2);
    g.initialize();
    for(Die d:rolls)
        {
        System.out.println("Face value is "+d.roll());
        }
   }
 }
```

Question number 5(2012 5b)



Solution of Question number 5

```
public class Sensor
    {
    private int state;
    public Sensor(int s)
    {
      this.state=s;
      }
    public int sensePath()
    {
          return state;
      }
  }
```

Solution of Question number 5

```
public class Actuators
    {
     private int state;
    public Actuators(int s)
     {
          this.state=s;
      }
    public int moveForward()
     {
          state++;
        return state;
      }
    public int moveBackward()
     {
          state--;
        return state;
      }
  }
```

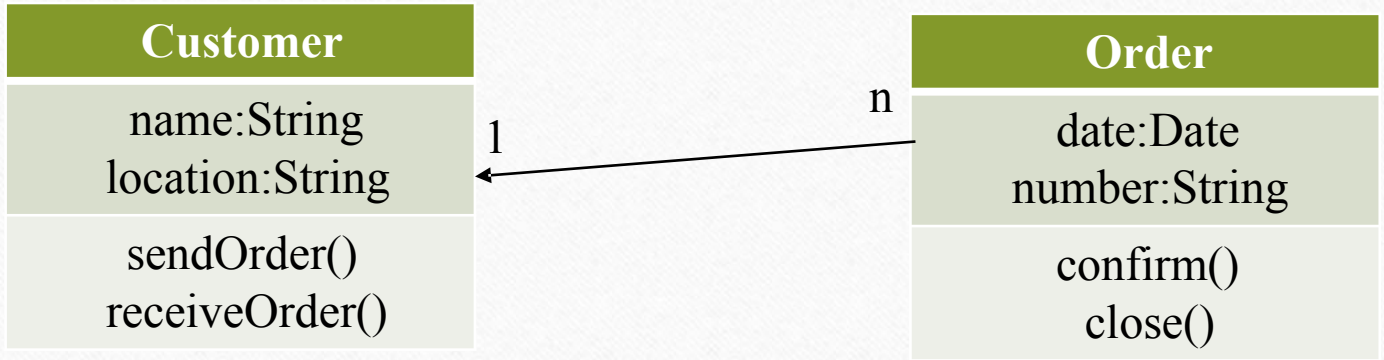
Solution of Question number 5

```
public class Robot
        {
        Sensor s=new Sensor(1);
    public static void main(String[] args)
    {
        ArrayList<Actuators> list=new ArrayList<Actuators>(2);
        Actuators a1=new Actuators(1);
        Actuators a2=new Actuators(-1);
        list.add(a1);
        list.add(a2);
        public void pickItem()
        {
                if(s.sensePath()>0)
                {
                for(Actuators a:list)
                {
                        System.out.println("Robot is moving forward and its new state is
"+a.moveForward());
                }
            }
          else
                {
                System.out.println("State of  Robot is less than zero");
            }
        }
```

Solution of Question number 5

```
public void dropItem()
        {
        if(s.sensePath()>0)
        {
            for(Actuators a:list)
            {
                System.out.println("Robot is moving
backward and its new state is "+a.moveBackward());
             }
          }
        else
        {
            System.out.println("State of  Robot is less
than zero");
            }
       }
    }
}
```

Question number 6



Solution of Question number 6

```
public class Customer
    {
     private String name;
    private String location;
    public Customer(String name,String location)
     {
          this.name=name;
       this.location=location;
     }
    public void sendOrder()
     {
          System.out.println("Order has been made for
"+name+" at location "+location);
     }
    public void receiveOrder()
     {
          System.out.println("Order has been received");
     }
    }
```

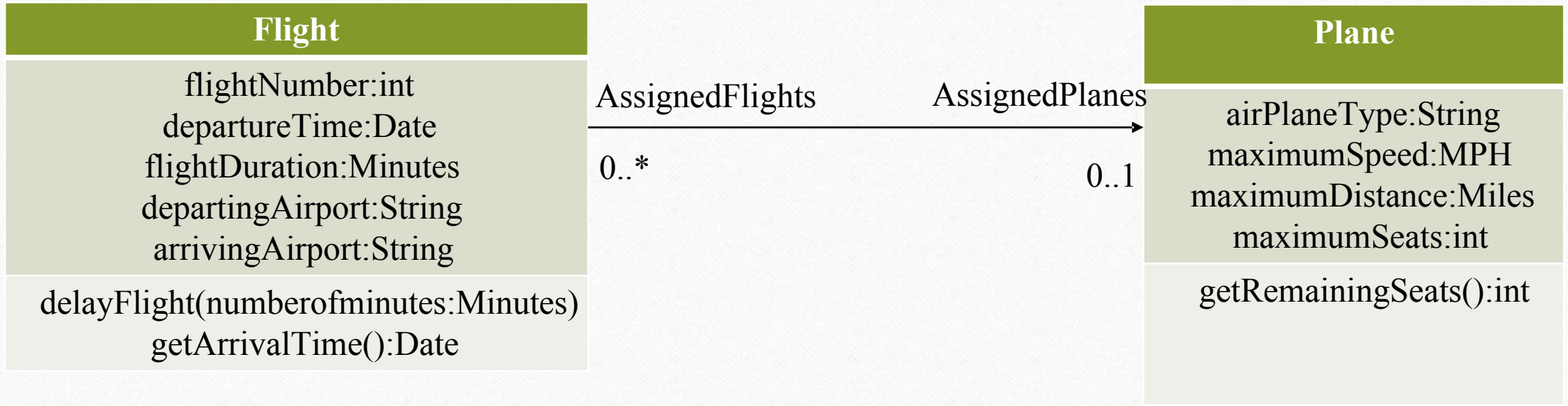
Solution of Question number 6

```
public class Order
      {
       private Date date;
      private String number;
      private Customer C;
      public Order(Date date,String number)
       {
            this.date=date;
          this.number=number;
          C=new Customer("Book","Balkumari");
        }
      public void confirm()
       {
            C.sendorder();
          System.out.println("Item "+C.name+" is
ordered at "+date);
        }
```

Solution of Question number 6

```
public void close()
        {
                C.receiveOrder();
           System.out.println("Order has been closed");
        }
     public static void main(String args[])
        {
         ArrayList<Order> list=new ArrayList<Order>();
         Order o1=new Order("2018/02/15","1917");
         Order o2=new Order("2018/12/30","1918");
         list.add(o1);
         list.add(o2);
         for(Order o:list)
                {
                        o.confirm();
              o.close();
                }
        }
   }
```

Question number 7



Solution of Question number 7

```
public class Plane
    {
     private String airPlaneType;
   private MPH maximumSpeed;
   private Miles maximumDistance;
   private Integer maximumSeats;
   public Plane (String a1,MPH mS,Miles mD,Integer maS)
    {
          this.airPlaneType=a1;
      this.maximumSpeed=mS;
      this.maximumDistance=mD;
      this.maximumSeats=maS;
    }
   public Integer getRemaining Seats()
    {
          return maximumSeats;
    }
 }
```

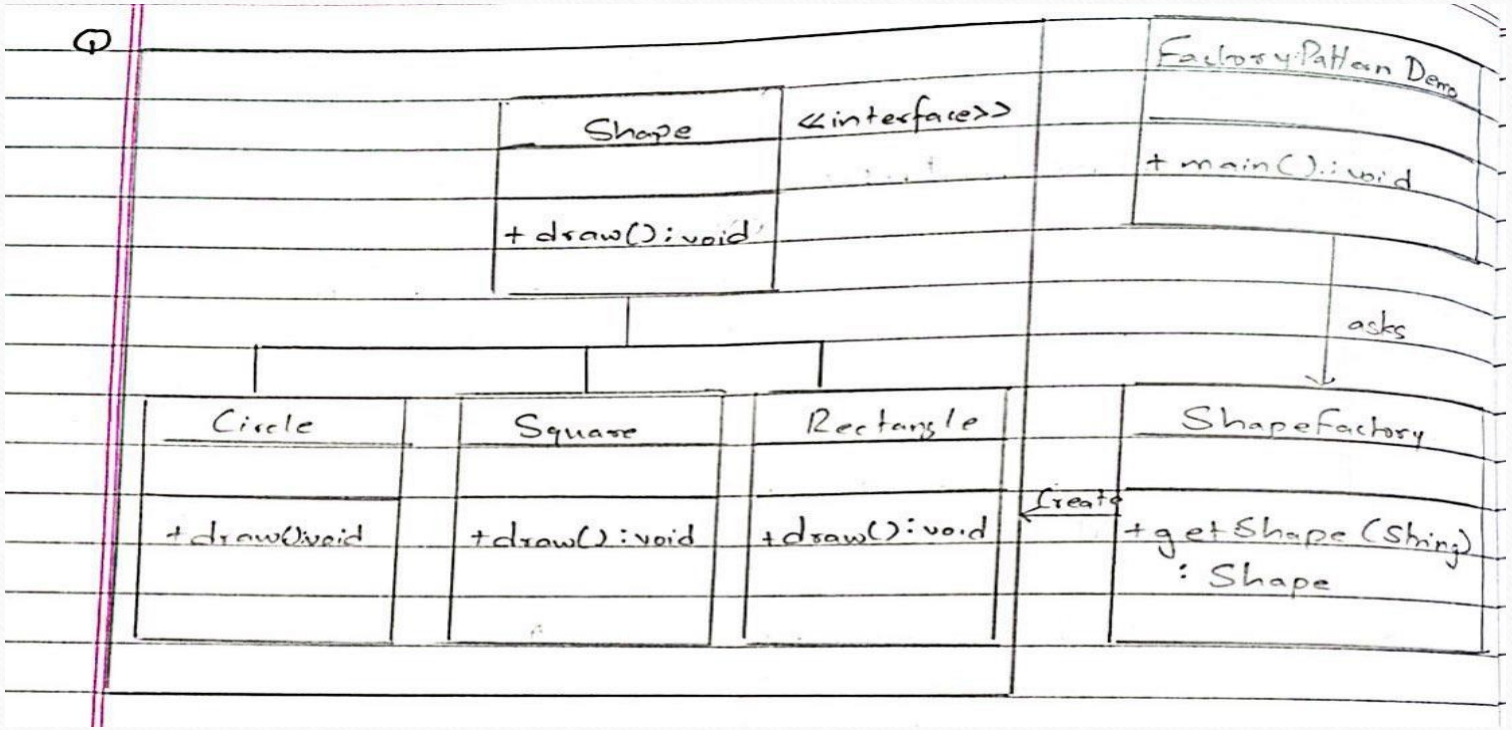
Solution of Question number 7

```
public class Flight
      {
       private Integer flightNumber;
      private Date departureTime;
      private Minutes flightDuration;
      private String departingAirport;
      private String ArrivingAirport;
      private Plane AssignedPlanes;
      public Flight(Integer fN,Date dT,Minutes fD,String
dA, String aA)
      {
            this.flightNumber=fN;
          this.departureTime=dT;
          this.flightDuration=fD;
          this.departingAirport=dA;
          this.arrivingAirport=aA;
          AssignedPlanes=new Plane("Yeti
Airlines",140,500,100);
       }
```

Solution of Question number 7

```
public void delayFlight(Minutes numberofminutes)
        {
       System.out.println("Flight has been delayed for "+numberofminutes);
     }
    public Date getArrivalTime()
        {
        return (departureTime+flightDuration);
     }
    public static void main(String args[])
        {
     ArrayList<Flight> flights=new ArrayList<Flight>(2);
        Flight f1=new Flight(1234,"18:00",20,"Kathmandu","Pokhara");
        Flight f2=new Flight(1235,"8:00",35,"Kathmandu","Biratnagar");
        flights.add(f1);
        flights.add(f2);
        for(Flight f:flights)
                {
        f.delayFlight();
        System.out.println("The flight departed at "+f.departureTime+" and will arrive
after ");
        f.getArrivalTime();
        }
     }
  }
```

Question number 8



Solution of Question number 8

```
public interface Shape
        {
        public void draw();
  }
public class Circle implements Shape
        {
        public void draw()
        {
        System.out.println("Draw function inside circle");
      }
  }
public class Square implements Shape
        {
        public void draw()
        {
        System.out.println("Draw function inside square");
      }
  }
public class Rectangle implements Shape
        {
        public void draw()
        {
        System.out.println("Draw function inside Rectangle");
      }
  }
```

Solution of Question number 8

```
public class ShapeFactory
        {
        public Shape getShape(string shapeType)
        {
        if(shapeType==null)
                {
                return null;
          }
        else if(shapeType.equalsIgnoreCase("Circle"))
                {
                return new Circle();
          }
        else if(shapeType.equalsIgnoreCase("Square"))
                {
                return new Square();
          }
        else if(shapeType.equalsIgnoreCase("Rectangle"))
                {
                return new Rectangle();
          }
        else
                {
                return null;
          }
      }
  }
```

Solution of Question number 8

```
public class FactoryPatternDemo
    {
    public static void main(String[] args)
    {
        ShapeFactory shapeFactory=new
ShapeFactory();
        Shape
shape1=shapeFactory.getShape("Circle");
        shape1.draw();
        Shape
shape2=shapeFactory.getShape("Square");
        shape2.draw();
        Shape
shape3=shapeFactory.getShape("Rectangle");
        shape3.draw();


    }
  }
```

Exception Handling in UML diagrams

- Exception is an undesirable event that occurs during runtime. There are basically two kinds of errors in a program :
  - ➢ **Syntactical error :**
    - ✓ This kind of error occurs when the developer writes the syntax of a program in an incorrect way. This kind of error can be easily detected by compiler during compile time.
  - ➢ **Semantic error :**
    - ✓ This kind of error occurs when the logic used in a program is somehow incorrect and it occurs during the runtime.

Exception Handling in UML diagrams

- Exception generally occurs due to semantic errors in a program.
- Exception might occur in following conditions :
  - **divide by zero**
  - **trying to open a file that does not exist**
  - **running out of memory**
  - **trying to initialize an object to an impossible value**
  - **and so on**

In UML diagrams, there are several ways for showing the exception occurrence in a program.

For example Division by zero problem

```
public class Division
    {
   int num1=3;
   int num2=0;
    public void calculate() throws ArithmeticException
    {
          try
          {
          int result=num1/num2;
            System.out.println(result);
          }
        catch(ArithmeticException e)
          {
          e.log();
            System.out.println("Division by zero
exception");
          }
     }
  }
```

Exception Handling in UML Diagrams

- In class diagram, there are 4 ways to represent exception:
- **1st way:**

Division
-num1:int -num2:int
+calculate():void throws Arithmetic Exception

Exception Handling in UML Diagrams

- In class diagram, there are 4 ways to represent exception:
- **2nd way:**

Division
-num1:int -num2:int
+calculate():void {throws= (Arithmetic Exception)}

Exception Handling in UML Diagrams

- In class diagram, there are 4 ways to represent exception:
- **3rd way:**

Division
-num1:int -num2:int
+calculate():void <<exception>> Arithmetic Exception

Exception Handling in UML Diagrams

- In class diagram, there are 4 ways to represent exception:
- **4th way:**

Exception Handling in UML Diagrams

- In SSD, exception can be represented as: