**Lab 4 Source Code:**

**Task 1**

package linesandshapes;

import java.awt.\*;

import java.awt.geom.\*;

import javax.swing.\*;

import java.io.\*;

abstract class Shape{

         abstract void  draw();

}//end of Shape class.

class  Circle extends Shape{

int x, y, width, height;

public Circle(int x, int y) {

this.x = x;

this.y = y;

}

public void draw(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

Ellipse2D.Double circle = new Ellipse2D.Double(x, y, 10, 10);

g2d.setColor(Color.GRAY);

g2d.fill(circle);

}}

class Square extends Shape{

    Public void Draw(){

g.drawRect(100,100,100,100);

g.drawLine(100,100,100,100);

g.setBackground(Color.BLACK);

System.out.println(&quot;Trasut&quot;);

credits.setText(&quot;Kitebbiv&quot;);

credits.setBackground(null);

credits.setEditable(false);

credits.setFocusable(false);

credits.setBounds(0,0,100,100);

credits.setForeground(Color.BLACK);

panel.add(credits);

g.getPaint();        }

        }

} /end of Square class.

class  Rectangle extends Shape{

   //@override

Public void Draw(){

g.drawRect(100,100,300,300);

g.drawLine(100,100,100,100);

g.setBackground(Color.BLACK);

System.out.println(&quot;Trasut&quot;);

credits.setText(&quot;Kitebbiv&quot;);

credits.setBackground(null);

credits.setEditable(false);

credits.setFocusable(false);

credits.setBounds(0,0,100,100);

credits.setForeground(Color.BLACK);

panel.add(credits);

g.getPaint();        }

} /end of Rectangle class.

class FactoryPatternDemo{

    public static void main(String args[])throws IOException{

       FactoryPatternDemo  shapeFactory = new  FactoryPatternDemo();

      System.out.print(&quot;Enter the name of plan for which the bill will be generated:);

      BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

      String planName=br.readLine();

    System.out.print(&quot;Enter shape you want to create: (CIRCLE/SQUARE/RECTANGLE) &quot;);

      int units=Integer.parseInt(br.readLine());

      Shape s = ShapeFactory.getShape(shapeName);

       System.out.print(&quot;Bill amount for &quot;+shapeName);

           s.getDraw();

            s.draw();

           }   }

----------------------------------------------------------------------

**Task 2**

public interface Shape {

   void draw();

}

public class Rectangle implements Shape {

   @Override

   public void draw() {

      System.out.println("Inside Rectangle::draw() method.");

   }

}

public class Square implements Shape {

   @Override

   public void draw() {

      System.out.println("Inside Square::draw() method.");

   }

}

public class Circle implements Shape {

 Override

   public void draw() {

      System.out.println("Inside Circle::draw() method.");

   }

}

public interface Color {

   void fill();

}

 public class Red implements Color {

   @Override

   public void fill() {

      System.out.println("Inside Red::fill() method.");

   }

}

public class Green implements Color {

   @Override

   public void fill() {

      System.out.println("Inside Green::fill() method.");

   }

}

public class Blue implements Color {

   @Override

   public void fill() {

      System.out.println("Inside Blue::fill() method.");

   }

}

public abstract class AbstractFactory {

   public abstract Color getColor(String color);

   public abstract Shape getShape(String shape) ;

}

public class ShapeFactory extends AbstractFactory {

   @Override

   public Shape getShape(String shapeType){

      if(shapeType == null){

         return null;

      }

      if(shapeType.equalsIgnoreCase("CIRCLE"))

{

         return new Circle();

      }

else if(shapeType.equalsIgnoreCase("RECTANGLE"))

{

         return new Rectangle();

      } else if(shapeType.equalsIgnoreCase("SQUARE"))

{

         return new Square();

      }

      return null;

   }

   @Override

   public Color getColor(String color) {

      return null;

   }

}

public class ColorFactory extends AbstractFactory {

   @Override

   public Shape getShape(String shapeType){

      return null;

   }

   @Override

   Color getColor(String color) {

      if(color == null){

         return null;

      }

      if(color.equalsIgnoreCase("RED")){

         return new Red();

      } else if(color.equalsIgnoreCase("GREEN")){

         return new Green();

      } else if(color.equalsIgnoreCase("BLUE")){

         return new Blue();

      }

      return null;

   }

}

public class FactoryProducer {

   public static AbstractFactory getFactory(String choice){

      if(choice.equalsIgnoreCase("SHAPE")){

         return new ShapeFactory();

      } else if(choice.equalsIgnoreCase("COLOR")){

         return new ColorFactory();

      }

      return null;

   }

}

public class AbstractFactoryPatternDemo {

   public static void main(String[] args) {

             / / Get the shape factory

      AbstractFactory shapeFactory = FactoryProducer.getFactory("SHAPE");

             / / Get the object of the shape of Circle

      Shape shape1 = shapeFactory.getShape("CIRCLE");

             / / call Circle's draw method

      shape1.draw();

             / / Get the object of the shape of Rectangle

      Shape shape2 = shapeFactory.getShape("RECTANGLE");

             / / call the Rectangle draw method

      shape2.draw();

             / / Get the object of the shape of Square

      Shape shape3 = shapeFactory.getShape("SQUARE");

             / / Call Square's draw method

      shape3.draw();

             / / Get the color factory

      AbstractFactory colorFactory = FactoryProducer.getFactory("COLOR");

             / / Get the object of color Red

      Color color1 = colorFactory.getColor("RED");

             / / call Red's fill method

      color1.fill();

             / / Get the object of color Green

      Color color2 = colorFactory.getColor("Green");

             / / call Green's fill method

      color2.fill();

             / / Get the object of color Blue

      Color color3 = colorFactory.getColor("BLUE");

             / / call Blue's fill method

      color3.fill();

   }}

------------------------------------------------------------------------------------