Lab 4 : Abstract Factory Design

1. Shape interface

```
package abstractfactorydesign;

public interface shape {
   void drawshape();
}

public class Rectangle implements shape{

   @Override
   public void drawshape() {
      System.out.println("shape Rectangle..");
   }
}

public class circle implements shape{

   @Override
   public void drawshape() {
      System.out.println("shape Circle..");
   }
}

public class Square implements shape{

   @Override
   public void drawshape() {
      System.out.println("shape square..");
   }
}
```

2. Shape Factory

```
public class shapefactory extends Abstractfactory
{


   @Override
   color getcolor(String color) {
      return null;
   }

   @Override
   shape getshape(String shape) {
      {if(shape==null)
      {return null;
      }
```

```
            if(shape.equalsIgnoreCase("Circle"))
            {return new circle();}
            else if(shape.equalsIgnoreCase("Rectangle"))
            {return new Rectangle();}
            else if(shape.equalsIgnoreCase("Square"))
            {return new Square();}
            return null;}
    }
}
```

3. Color interface

```
public interface color {

    void fillColor();
}

public class Red implements color{

    @Override
    public void fillColor() {
        System.out.println("Red color");
    }

}

public class green implements color{

    @Override
    public void fillColor() {
        System.out.println("Green color");
    }
}

public class Blue implements color{
    @Override
    public void fillColor() {
        System.out.println("Blue color");
    }

}
```

4. Color Factory

```
public class colorfactory extends Abstractfactory{
    @Override
    color getcolor(String color) {
        if(color.equalsIgnoreCase("BLUE"))
        {
            return new Blue();
        }
        else if(color.equalsIgnoreCase("RED"))
        {
            return new Red();
        }
        else if(color.equalsIgnoreCase("GREEN"))
        {
            return new green();
        }
        else {
            return null;
        }
    }

    @Override
    shape getshape(String shape) {
        return null;
    }
}
```

5. Abstract Factory

```
public abstract class Abstractfactory {
    abstract color getcolor(String color);
    abstract shape getshape(String shape);
}
```

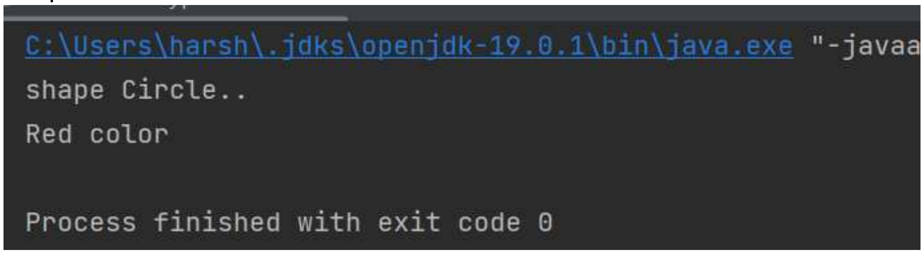
6. Factory producer

```
public class factoryproducer {
    public static Abstractfactory getFactory(String Choice)
    {
        if (Choice.equalsIgnoreCase("shape"))
        {
            return new shapefactory();
        }
        else if(Choice.equalsIgnoreCase("color"))
        {
            return new colorfactory();
        }
        else
        {
            return null;
        }
    }
}
```

7. Abstract Factory Pattern Test (Main part)

```
public class abstractfactorypatterntest {
    public static void main(String[] args) {
        Abstractfactory shapeFactory = factoryproducer.getFactory("Shape");
        shape shape1 =shapeFactory.getshape("Circle");
        shape1.drawshape();
        Abstractfactory colorfactory = factoryproducer.getFactory("Color");
        color color1 = colorfactory.getcolor("Red");
        color1.fillColor();
    }
}
```

Output:



```
C:\Users\harsh\.jdks\openjdk-19.0.1\bin\java.exe "-javaa
shape Circle..
Red color

Process finished with exit code 0
```