Lab Assignment - 26

Name: M Bhagyalaxmi **Student Id: AF0339449**

1)Create a simple Factory Pattern for creating shapes (e.g., Circle, Square, Triangle). Define an interface called Shape with a draw method, and create concrete classes Circle, Square, and Triangle that implement the Shape interface. Implement a ShapeFactory that has a method createShape which takes a string (e.g., "circle", "square", "triangle") as input and returns the corresponding shape object. Write a program to demonstrate the usage of the factory to create different shapes and call their draw methods.

Program:

```
//Factory method pattern Example

 package com.DesignPattern;

 public interface Shape  {

 public void drawshape();

 }

package com.DesignPattern;

public class Circle implements Shape  {
```

```java
public void drawshape()
    {
        System.out.println("circle is drawn");
    }
}
```

```java
package com.DesignPattern;

public class Square implements Shape {

    public void drawshape()
    {
        System.out.println("square is drawn");
    }
}
```

```java
package com.DesignPattern;

public class Triangle implements Shape {

    public void drawshape()
    {
        System.out.println("Triangle is drawn");
    }
}
```

```java
package com.DesignPattern;

public class ShapeFactory {
```

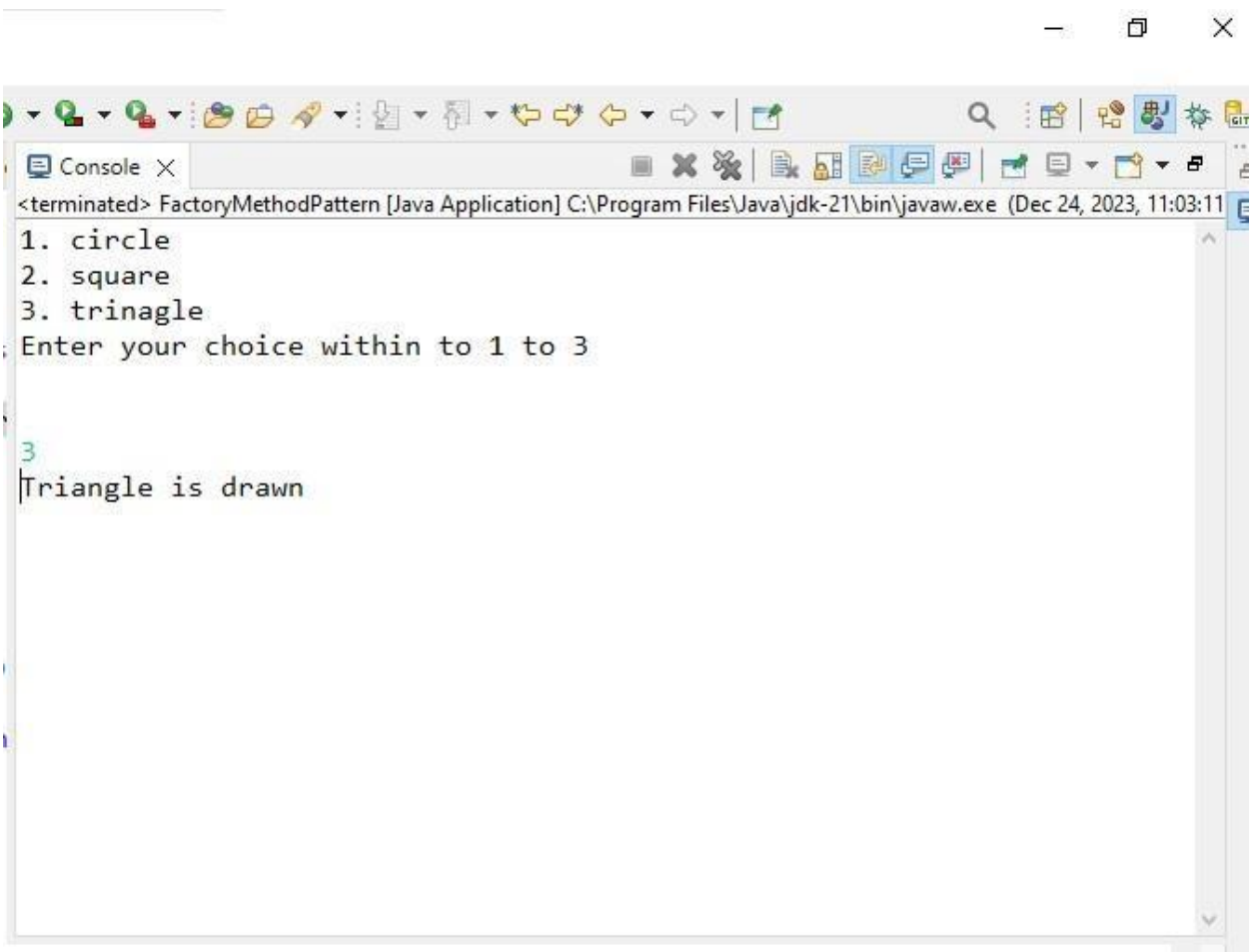
```
    public Shape createShape(String s)
    {
        Shape shape;

     if(s.equalsIgnoreCase("circle"))

        shape = (Shape) new Circle();

     else if(s.equalsIgnoreCase("square"))

         shape = (Shape) new Square();

     else if(s.equalsIgnoreCase("triangle"))

         shape = (Shape) new Triangle();

      else
            shape=null;

        return shape;
    }
}
```

```
package com.DesignPattern;

import java.util.*;

public class FactoryMethodPattern {

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

```
        ShapeFactory sf = new ShapeFactory();

        int choice;
        Scanner obj = new Scanner(System.in);

        System.out.println("1. circle");
        System.out.println("2. square");
        System.out.println("3. trinagle");

System.out.println("Enter your choice within to 1 to
3\n");
        choice=obj.nextInt();


        switch(choice)
        {
            case 1:
            {
                Shape s = sf.createShape("circle");
                s.drawshape();
                break;
            }

            case 2:
            {
                Shape s1 = sf.createShape("square");
                s1.drawshape();
                break;
            }
            case 3:
            {

             Shape s2 = sf.createShape("triangle");
            s2.drawshape();
```

```
                break;
            }
            default:
            {
System.out.println("Invalid choice!!");
    }
      }
        }

          }
```

Output:



2)Create a simple Singleton Pattern for a logging class. Implement a Logger class that logs messages. Ensure that only one instance of the Logger class canbe created, and all log messages are written to a single log file. Write a program to demonstrate the usage of the Logger class to log messages from multiple parts of the application.

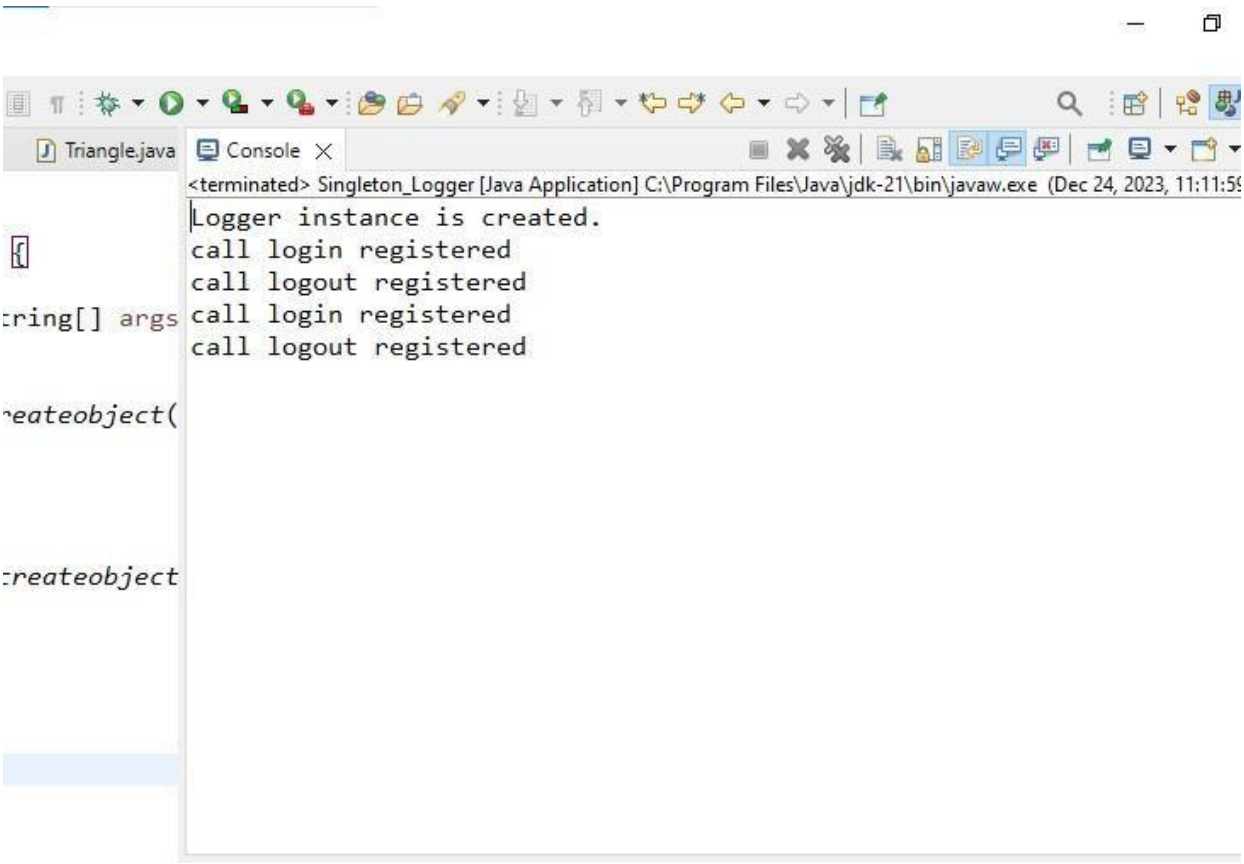
Program:

```
package com.Designpattern2;

public class Logger {

public  static Logger log = new Logger();

    private Logger()
{

System.out.println("Logger instance is created.");
    }

    public static Logger createobject()
    {
        return log;
    }

    public void loggerInmsg()
    {
        System.out.println("call login registered");
    }

    public void loggerOutmsg()
```

```
    {
        System.out.println("call logout registered");
    }

}
```

```
package com.Designpattern2;

public class UseSingletonclass {

public static void main(String[] args) {

        Singleton s ;
        s = Singleton.getinstance();
        s.mybusinesslogic();
       System.out.println("Hashcode of the object " +
s.hashCode());

        Singleton s1 ;
        s1 = Singleton.getinstance();
        s1.mybusinesslogic();
    System.out.println("Hashcode of the object " +
s1.hashCode());

        Singleton s2;
        s2 = Singleton.getinstance();
        s2.mybusinesslogic();
    System.out.println("Hashcode of the object " +
s2.hashCode());

    }
}
```

Output:


Triangle.java
Console
<terminated> Singleton_Logger [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (Dec 24, 2023, 11:11:59
Logger instance is created.
call login registered
call logout registered
call login registered
call logout registered