|  |  |
| --- | --- |
| **WEEK 1 – Design Patterns and Principles** | **Superset Id : 6429486**  **Name : Akshaya V** |

**Exercise 1: Implementing the Singleton Pattern**

**Scenario:**

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **SingletonPatternExample**.
2. **Define a Singleton Class:**
   * Create a class named Logger that has a private static instance of itself.
   * Ensure the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.
3. **Implement the Singleton Pattern:**
   * Write code to ensure that the Logger class follows the Singleton design pattern.
4. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

**Code:**

**Logger.java**

package SingletonPatternExample;

public class Logger{

    private static Logger instance;

    private Logger(){

        System.out.println("Logger Initialized.");

    }

    public static Logger getInstance(){

        if(instance==null){

            instance=new Logger();

        }

        return instance;

    }

    public void log(String message){

        System.out.println("LOG: "+message);

    }

}

**LoggerTest.java**

package SingletonPatternExample;

public class LoggerTest{

    public static void main(String[]args){

        Logger logger1=Logger.getInstance();

        logger1.log("First msg-6429486");

        Logger logger2=Logger.getInstance();

        logger2.log("Second msg-Akshaya V");

        if(logger1==logger2){

            System.out.println("Singleton verified(Both logger instances are the same).");

        }else{

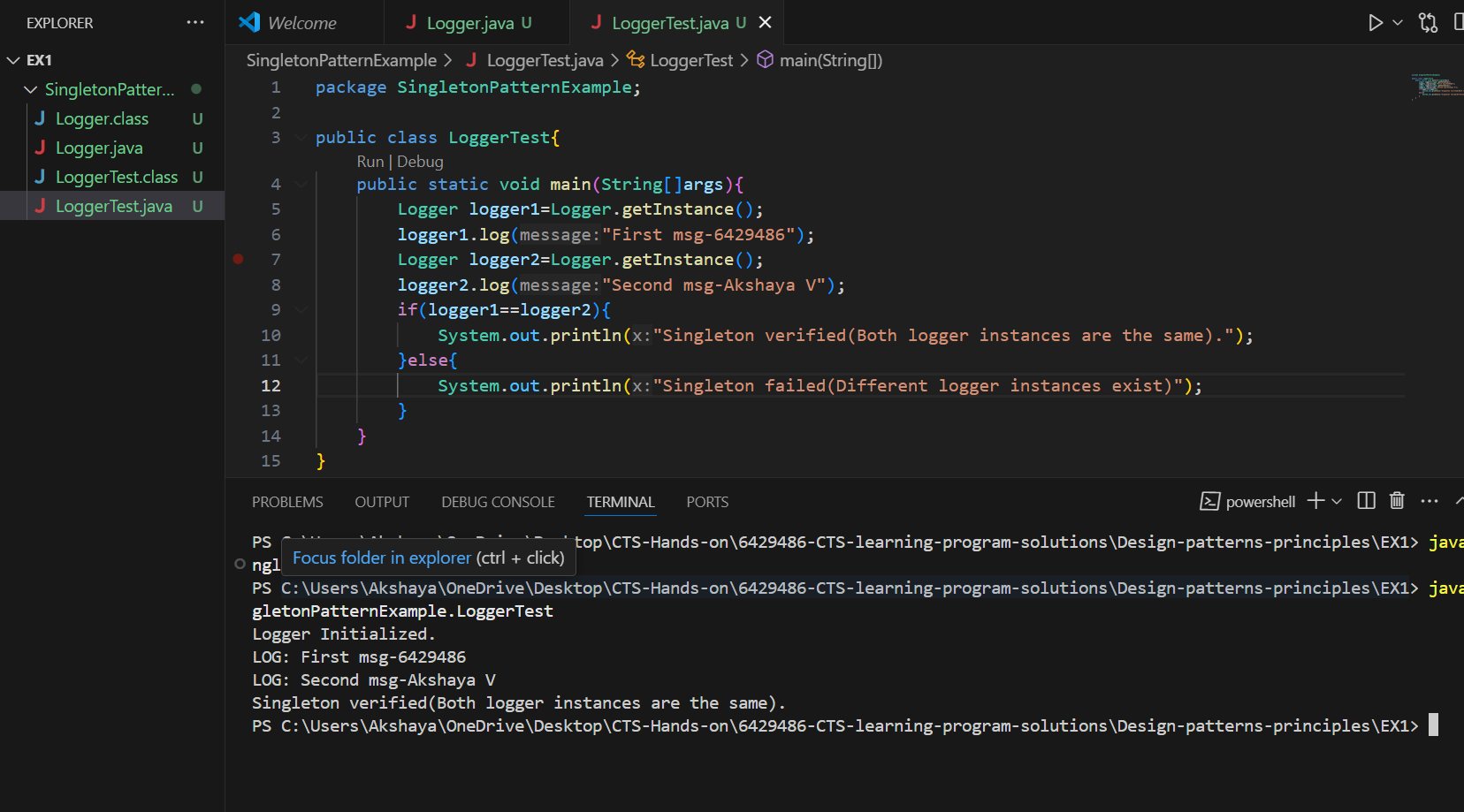
            System.out.println("Singleton failed(Different logger instances exist)");

        }

    }

}

**Output:**



**Exercise 2: Implementing the Factory Method Pattern**

**Scenario:**

You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **FactoryMethodPatternExample**.
2. **Define Document Classes:**
   * Create interfaces or abstract classes for different document types such as **WordDocument**, **PdfDocument**, and **ExcelDocument**.
3. **Create Concrete Document Classes:**
   * Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.
4. **Implement the Factory Method:**
   * Create an abstract class **DocumentFactory** with a method **createDocument()**.
   * Create concrete factory classes for each document type that extends DocumentFactory and implements the **createDocument()** method.
5. **Test the Factory Method Implementation:**
   * Create a test class to demonstrate the creation of different document types using the factory method.

**Code:**

**Document.java**

package FactoryMethodPatternExample;

public interface Document{

    void open();

}

**DocumentFactory.java**

package FactoryMethodPatternExample;

public abstract class DocumentFactory{

    public abstract Document createDocument();

}

**ExcelDocument.java**

package FactoryMethodPatternExample;

public class ExcelDocument implements Document{

    public void open(){

        System.out.println("Opening the Excel document");

    }

}

**ExcelDocumentFactory.java**

package FactoryMethodPatternExample;

public class ExcelDocumentFactory extends DocumentFactory{

    public Document createDocument(){

        return new ExcelDocument();

    }

}

**PdfDocument.java**

package FactoryMethodPatternExample;

public class PdfDocument implements Document{

    public void open(){

        System.out.println("Opening the PDF Document");

    }

}

**PdfDocumentFactory.java**

package FactoryMethodPatternExample;

public class PdfDocumentFactory extends DocumentFactory{

    public Document createDocument(){

        return new PdfDocument();

    }

}

**WordDocument.java**

package FactoryMethodPatternExample;

public class WordDocument implements Document{

    public void open(){

        System.out.println("6429486 - Akshaya V");

        System.out.println("Opening the Word Document");

    }

}

**WordDocumentFactory.java**

package FactoryMethodPatternExample;

public class WordDocumentFactory extends DocumentFactory{

    public Document createDocument(){

        return new WordDocument();

    }

}

**FactoryPatternTest.java**

package FactoryMethodPatternExample;

public class FactoryPatternTest{

    public static void main(String[] args){

        DocumentFactory wordFactory=new WordDocumentFactory();

        Document wordDoc=wordFactory.createDocument();

        wordDoc.open();

        DocumentFactory pdfFactory=new PdfDocumentFactory();

        Document pdfDoc=pdfFactory.createDocument();

        pdfDoc.open();

        DocumentFactory excelFactory=new ExcelDocumentFactory();

        Document excelDoc=excelFactory.createDocument();

        excelDoc.open();

    }

}

**Output:**

