# Week 1 - Design Principles & Patterns - Hands-On

## 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.

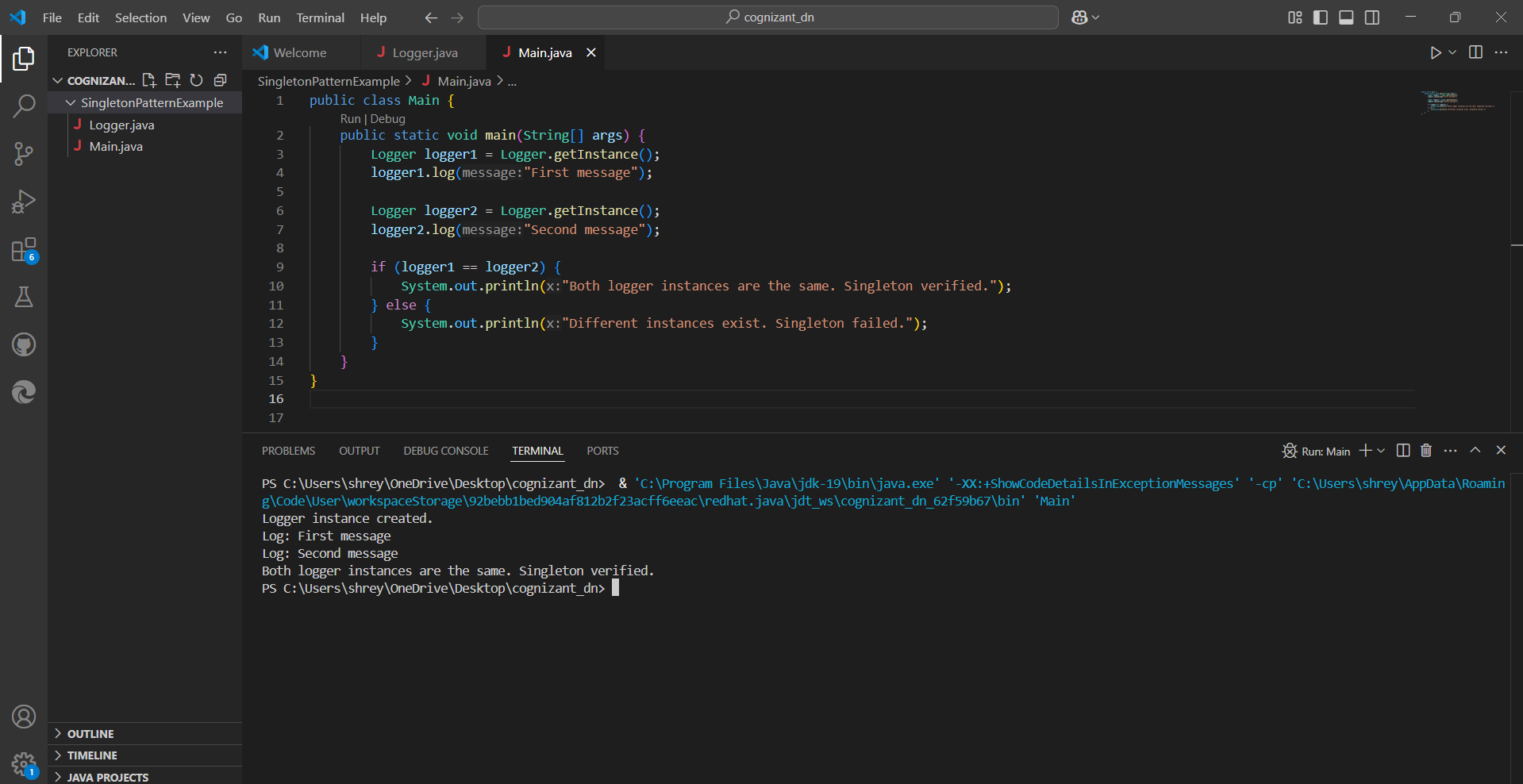
Logger.java:

public class Logger {  
 private static Logger instance;  
  
 private Logger() {  
 System.out.println("Logger instance created.");  
 }  
  
 public static Logger getInstance() {  
 if (instance == null) {  
 instance = new Logger();  
 }  
 return instance;  
 }  
  
 public void log(String message) {  
 System.out.println("Log: " + message);  
 }  
}

Main.java:

public class Main {  
 public static void main(String[] args) {  
 Logger logger1 = Logger.getInstance();  
 logger1.log("First message");  
  
 Logger logger2 = Logger.getInstance();  
 logger2.log("Second message");  
  
 if (logger1 == logger2) {  
 System.out.println("Both logger instances are the same. Singleton verified.");  
 } else {  
 System.out.println("Different instances exist. Singleton failed.");  
 }  
 }  
}

Output Screenshot:



# 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.

CODE

package FactoryDemo;

interface Document {

void open();

}

class WordDocument implements Document {

public void open() {

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

}

}

class PdfDocument implements Document {

public void open() {

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

}

}

class ExcelDocument implements Document {

public void open() {

System.out.println("Opening Excel Document.");

}

}

abstract class DocumentFactory {

public abstract Document createDocument();

}

class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

public class FactoryDemo {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document word = wordFactory.createDocument();

word.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdf = pdfFactory.createDocument();

pdf.open();

DocumentFactory excelFactory = new ExcelFactory();

Document excel = excelFactory.createDocument();

excel.open();

}

}

OUTPUT

