**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.

**Solution:**

**Logger.java**

public class Logger {

private static volatile Logger instance;

private Logger() {

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

}

public static Logger getInstance() {

if (instance == null) {

synchronized (Logger.class) {

if (instance == null) {

instance = new Logger();

}

}

}

return instance;

}

public void log(String message) {

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

}

}

**LoggerTest.java**

public class LoggerTest {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

logger1.log("Message from logger1");

Logger logger2 = Logger.getInstance();

logger2.log("Message from logger2");

if (logger1 == logger2) {

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

} else {

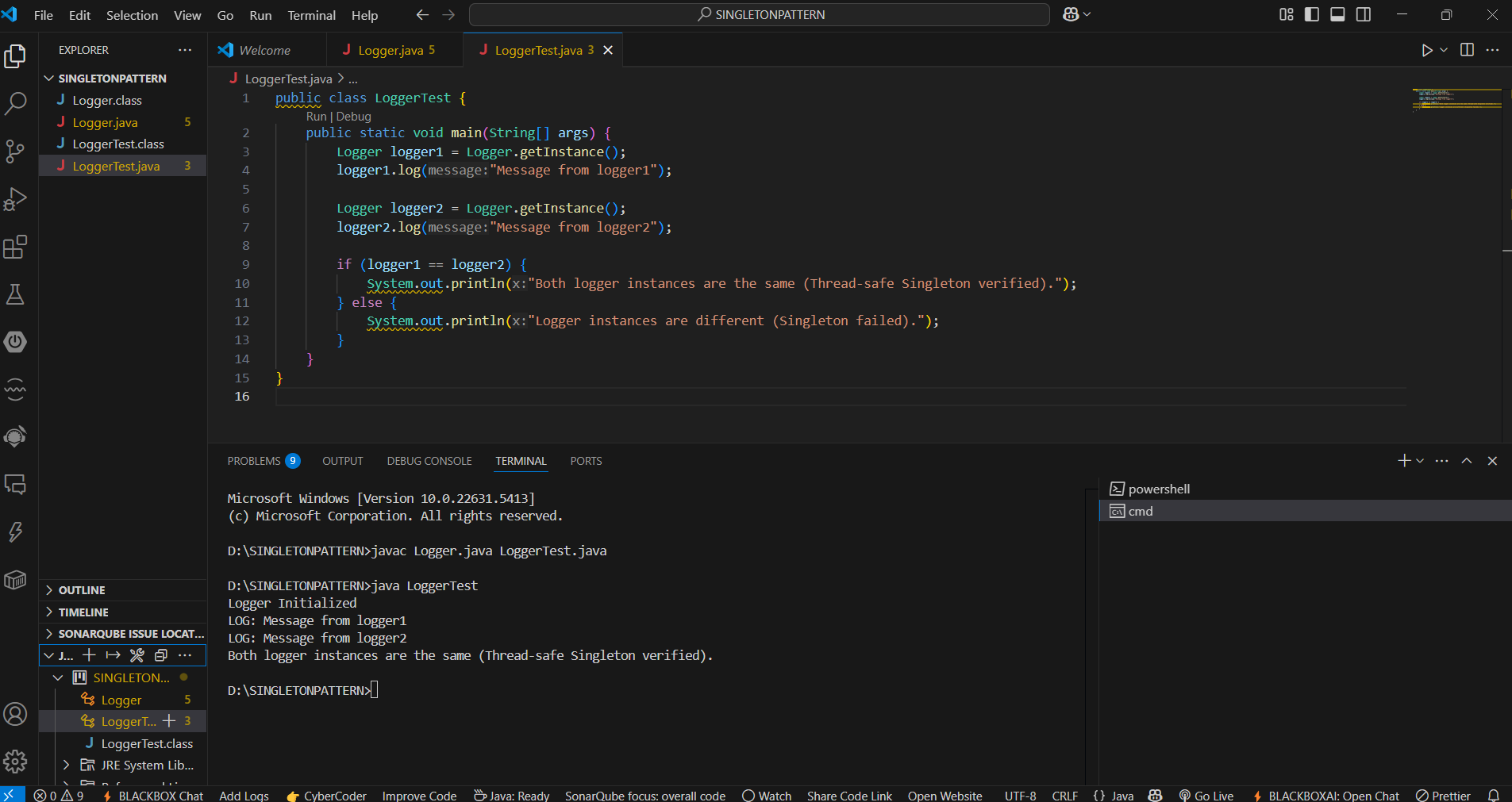
System.out.println("Logger instances are different (Singleton failed).");

}

}

}

**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.

**Solution:**

**Document.java**

public interface Document {

void open();

}

**WordDocument.java**

public class WordDocument implements Document {

@Override

public void open() {

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

}

}

**PdfDocument.java**

public class PdfDocument implements Document {

@Override

public void open() {

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

}

}

**ExcelDocument.java**

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

}

**DocumentFactory.java**

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**WordFactory.java**

public class WordFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

**PdfFactory.java**

public class PdfFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

**ExcelFactory.java**

public class ExcelFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

**DocumentTest.java**

public class DocumentTest {

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 screenshot:**

