Design Patterns and Principles Mandatory 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.

Code:

public class LoggerTest {

// Singleton Logger class (inner static class)

static class Logger {

// private static instance of Logger

private static Logger instance;

// private constructor

private Logger() {

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

}

// public static method to get the instance

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

// Logging method

public void log(String message) {

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

}

}

// Main method to test Singleton implementation

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

logger1.log("First log from logger1");

Logger logger2 = Logger.getInstance();

logger2.log("Second log from logger2");

if (logger1 == logger2) {

System.out.println("Singleton verified: Only one instance exists.");

} else {

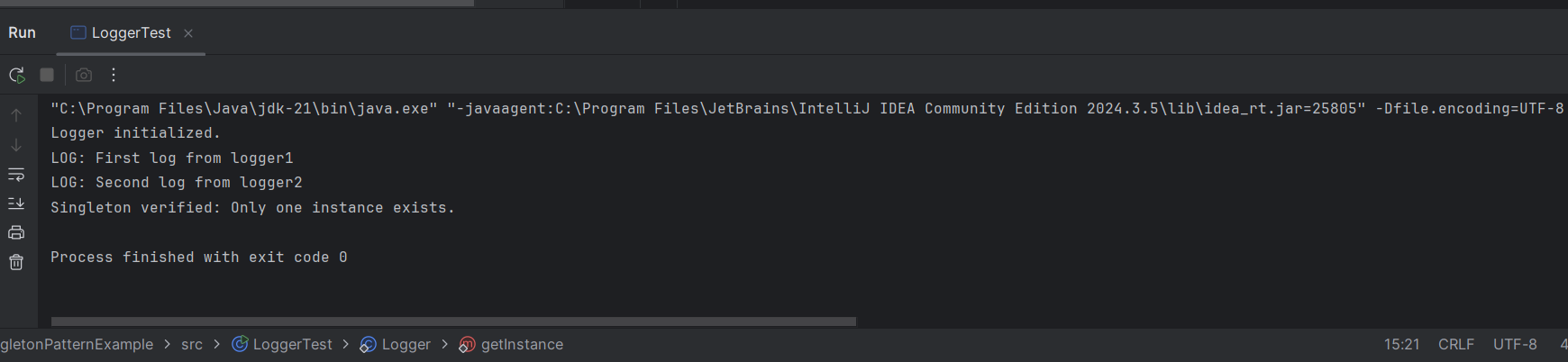
System.out.println("Singleton FAILED: Multiple instances exist.");

}

}

}

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:

// Step 1: Document interface

interface Document {

void create();

}

//Step 2: Concrete document classes

class WordDocument implements Document {

public void create() {

System.out.println("Word document created!");

}

}

class PdfDocument implements Document {

public void create() {

System.out.println("PDF document created!");

}

}

class ExcelDocument implements Document {

public void create() {

System.out.println("Excel document created!");

}

}

// Step 3: Abstract factory

abstract class DocumentFactory {

public abstract Document createDocument();

}

// Step 4: Concrete factories

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();

}

}

// Step 5: Test class

public class FactoryMethodPatternExample {

public static void main(String[] args) {

// Create factories

DocumentFactory wordFactory = new WordFactory();

DocumentFactory pdfFactory = new PdfFactory();

DocumentFactory excelFactory = new ExcelFactory();

// Create documents

Document word = wordFactory.createDocument();

Document pdf = pdfFactory.createDocument();

Document excel = excelFactory.createDocument();

// Use documents

word.create();

pdf.create();

excel.create();

}

}

Output:

