**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:**

**1.Create a New Java Project**

**2.Defining Singleton Class**

package SingletonPatternExample;

public class Logger {

private static Logger logger;

private Logger(){

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

}

public static Logger getInstance(){

if(logger==null){

logger = new Logger();

}

return logger;

}

public void displayMsg(String message){

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

}

}

**3.Implementation and Testing**

package SingletonPatternExample;

public class TestLogger {

public static void main(String[] args) {

Logger logger1=Logger.getInstance();

Logger logger2=Logger.getInstance();

logger1.displayMsg("This is the first log message.");

logger2.displayMsg("This is the second log message.");

if(logger1==logger2){

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

}else{

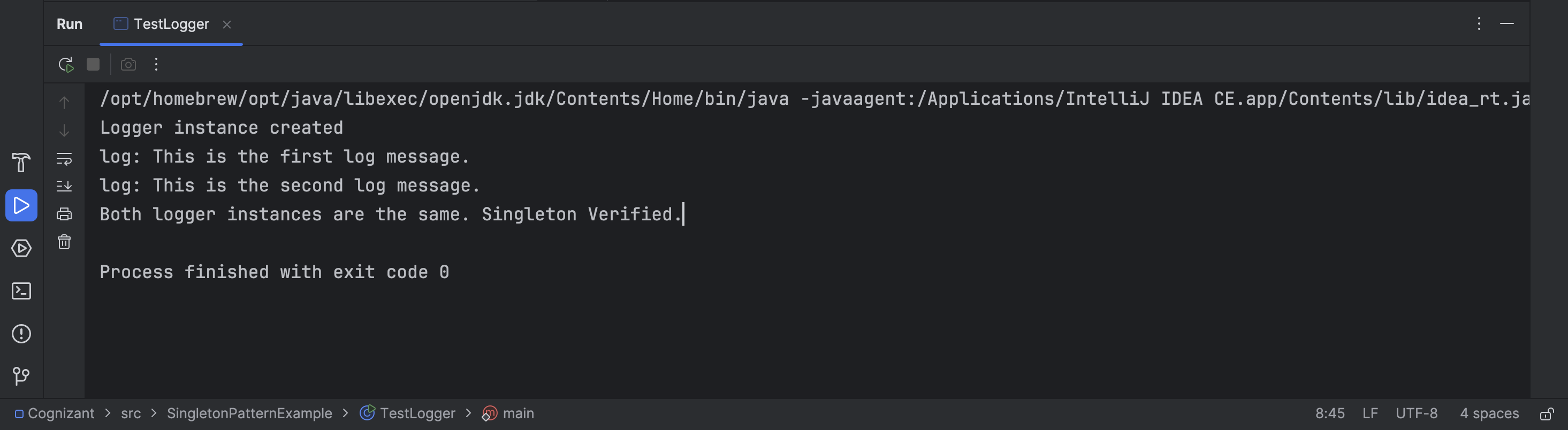
System.out.println("Logger instances are different.Singleton failed.");

}

}

}

**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:**

**1.Create a New Java Project**

**2.Define Document Classes**

package FactoryMethodPatternExample;

public interface Documents {

void open();

}

**3.Create Concrete Document Classes**

package FactoryMethodPatternExample;

public class ExcelDocument implements Documents{

@Override

public void open() {

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

}

}

package FactoryMethodPatternExample;

public class PdfDocument implements Documents{

@Override

public void open() {

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

}

}

package FactoryMethodPatternExample;

public class WordDocument implements Documents{

@Override

public void open() {

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

}

}

**4.Implement the Factory Method**

package FactoryMethodPatternExample;

public abstract class DocumentFactory {

public abstract Documents createDocument();

}

package FactoryMethodPatternExample;

public class ExcelDocumentFactory extends DocumentFactory{

@Override

public Documents createDocument() {

return new ExcelDocument();

}

}

package FactoryMethodPatternExample;

public class PdfDocumentFactory extends DocumentFactory{

@Override

public Documents createDocument() {

return new PdfDocument();

}

}

package FactoryMethodPatternExample;

public class WordDocumentFactory extends DocumentFactory{

@Override

public Documents createDocument() {

return new WordDocument();

}

}

**5.Test the Factory Method Implementation**

package FactoryMethodPatternExample;

public class TestClass {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

Documents wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfDocumentFactory();

Documents pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelDocumentFactory();

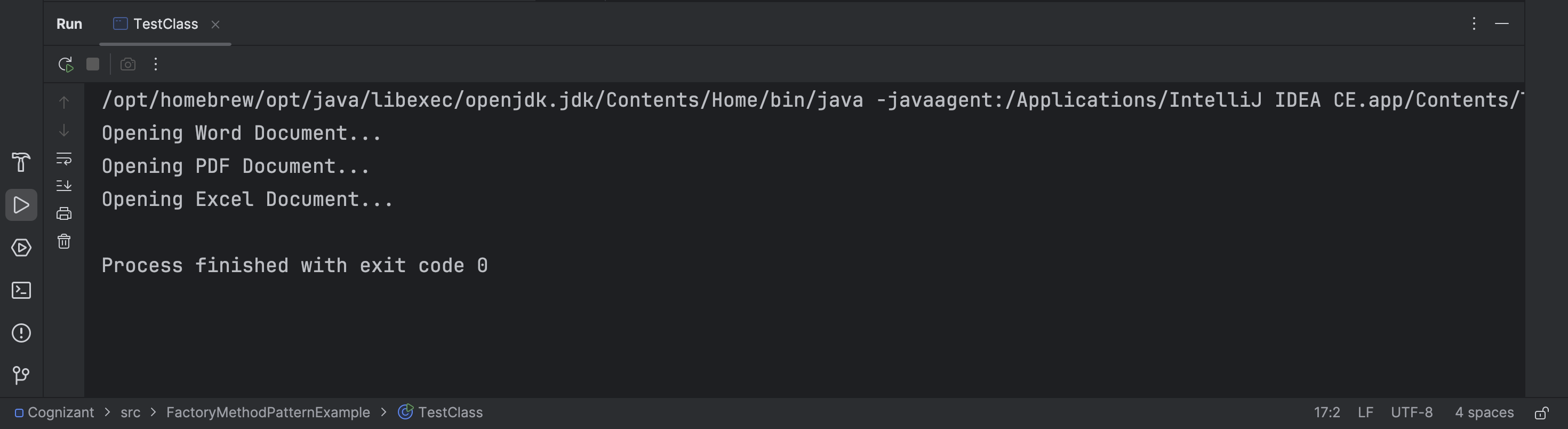
Documents excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**OUTPUT :**

****