**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:

Create a New Java Project:

Create a new Java project named SingletonPatternExample.

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.

Implement the Singleton Pattern:

Write code to ensure that the Logger class follows the Singleton design pattern.

Test the Singleton Implementation:

Create a test class to verify that only one instance of Logger is created and used across the application.

**Answer:**

**Logger.java**

package singleton;

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 singleton;

public class LoggerTest {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

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

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

if (logger1 == logger2) {

System.out.println("Both logger1 and logger2 refer to the same instance.");

} else {

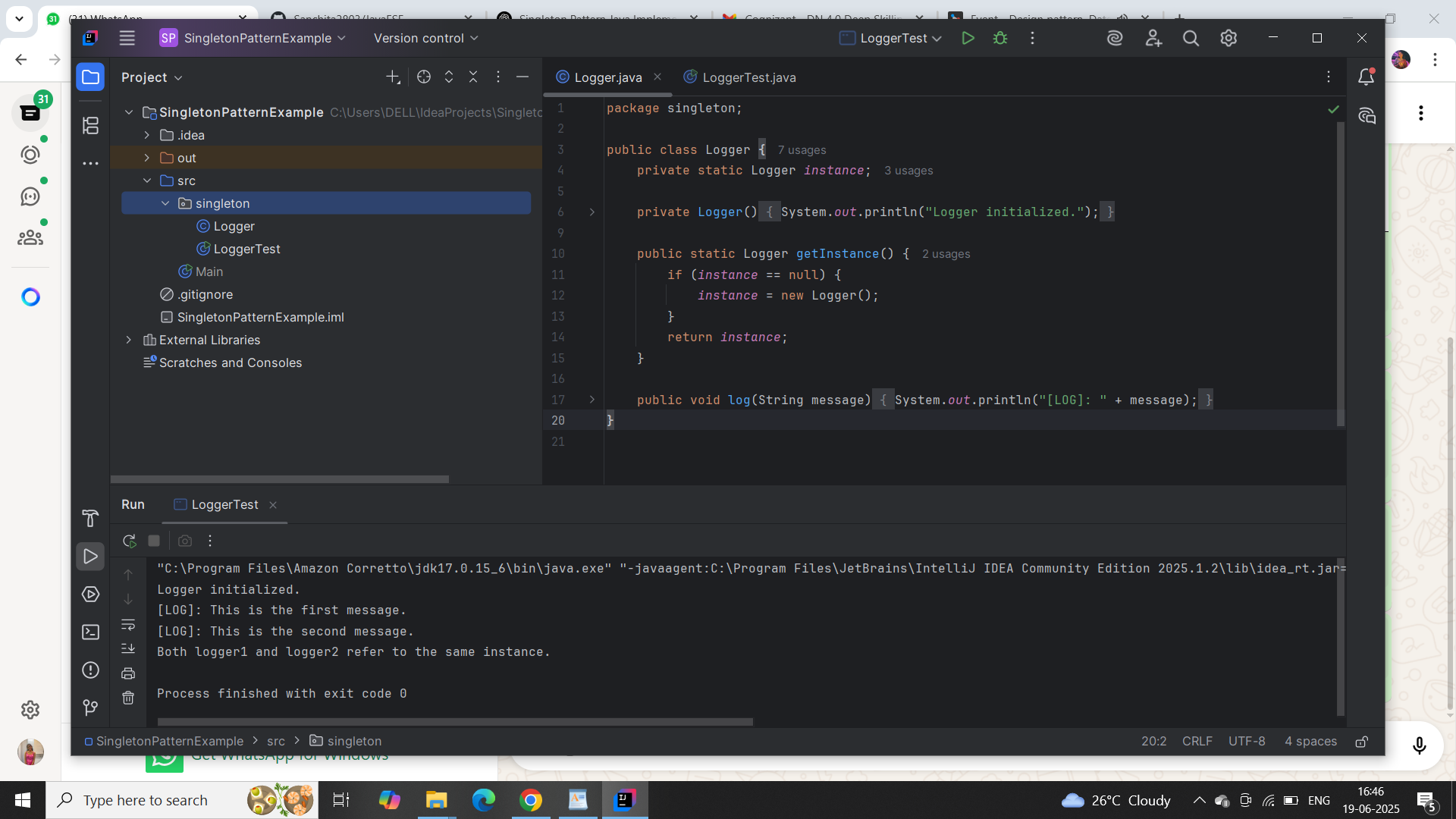
System.out.println("Different instances exist (This should not happen!).");

}

}

}

**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:

Create a New Java Project:

Create a new Java project named FactoryMethodPatternExample.

Define Document Classes:

Create interfaces or abstract classes for different document types such as WordDocument, PdfDocument, and ExcelDocument.

Create Concrete Document Classes:

Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.

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.

Test the Factory Method Implementation:

Create a test class to demonstrate the creation of different document types using the factory method.

**Answer:**

### **Document.java**

package factory;

public interface Document {

void open();

}

### **WordDocument.java**

package factory;

public class WordDocument implements Document {

public void open() {

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

}

}

### **PdfDocument.java**

package factory;

public class PdfDocument implements Document {

public void open() {

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

}

}

### **ExcelDocument.java**

package factory;

public class ExcelDocument implements Document {

public void open() {

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

}

}

### **DocumentFactory.java (abstract class)**

package factory;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

### **WordFactory.java**

package factory;

public class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

### **PdfFactory.java**

package factory;

public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

### **ExcelFactory.java**

package factory;

public class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

### **FactoryTest.java**

package factory;

public class FactoryTest {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output:**

