“DESIGN PATTERNS AND PRINCIPLES”

**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.

***SOLUTION:***

**1.Logger class**

import java.util.\*;

public class Logger {

private static Logger *instance*; //static class

public Logger() {

System.*out*.println("singleton pattern Logger instance created.");

}

public static Logger getInstance() {

if (*instance* == null) {

*instance* = new Logger(); // one instance is created

}

return *instance*;

} // A sample logging method

public void log(String message) {

System.*out*.println("Log message: " + message);

} }

**2.Logger Test Class**

public class LoggerTest {

public static void main(String[] args) {

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

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

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

if (logger1 == logger2) {

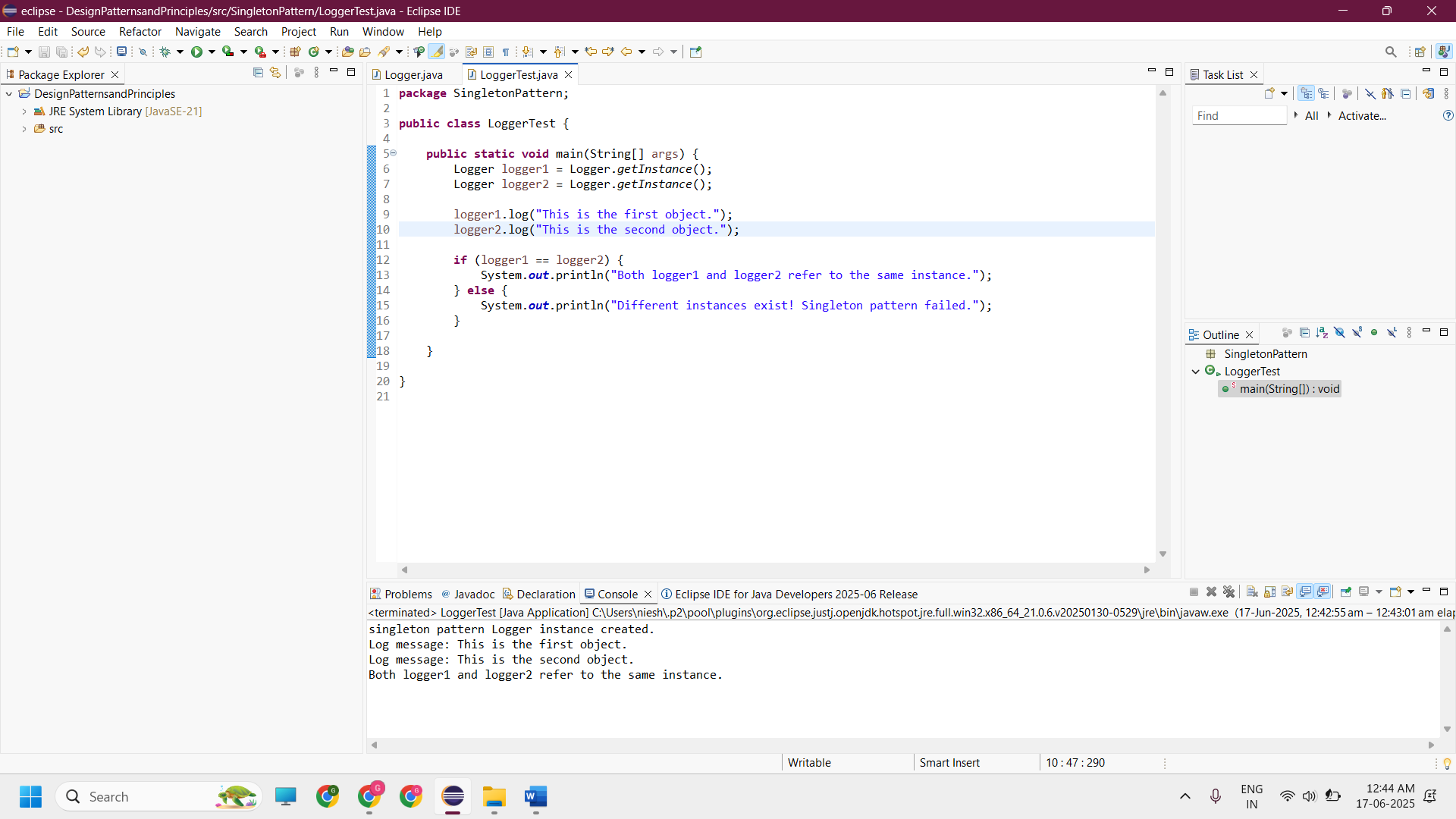
System.*out*.println("Both logger1 and logger2 refer to the same instance.");

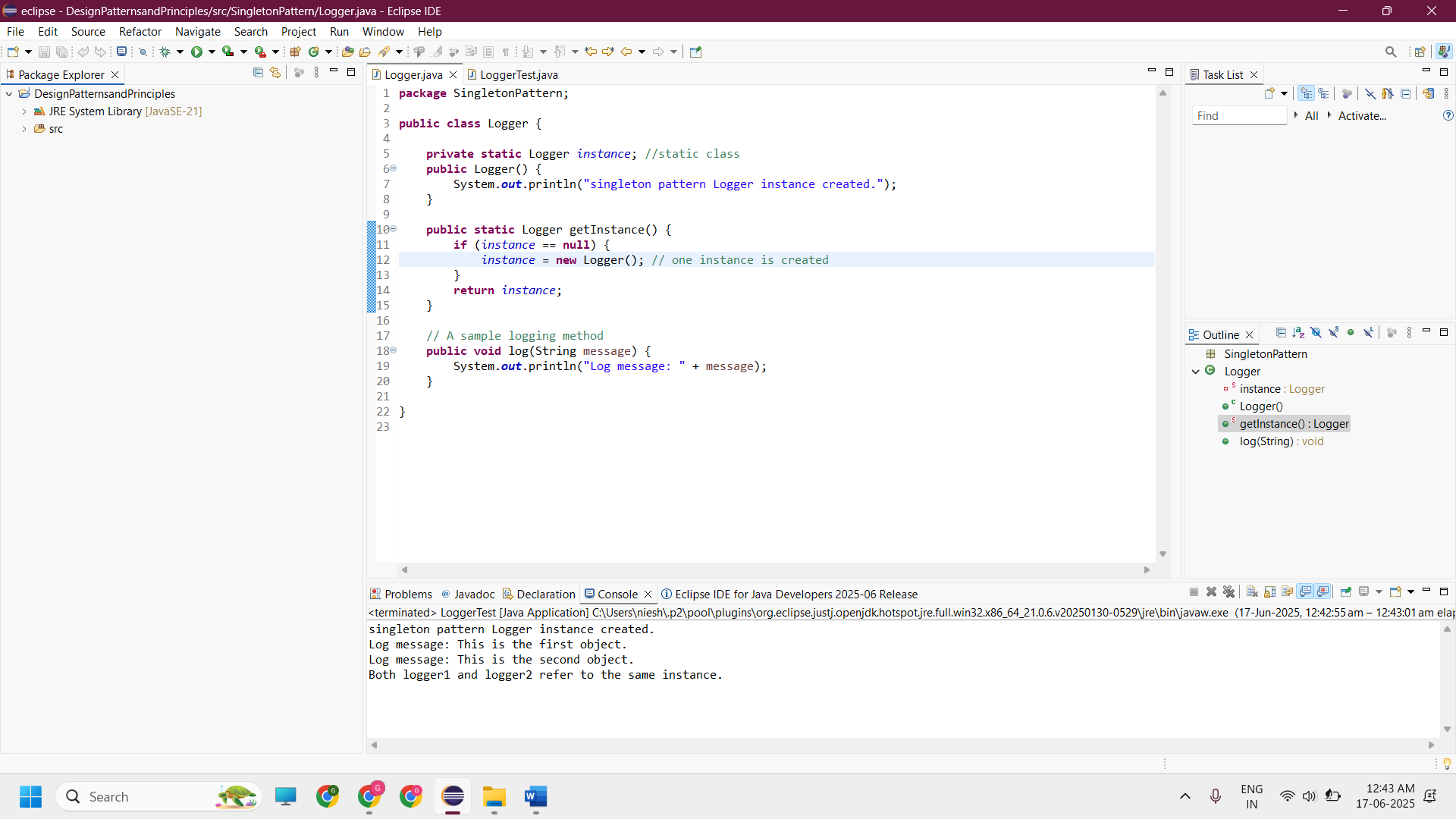
} else {

System.*out*.println("Different instances exist! Singleton pattern failed.");

}

}

}

****

**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.

***SOLUTION:***

public interface Document {

void access();

}

public class WordDocument implements Document {

public void access() {

System.out.println("Word Document is accessed");

}

}

public class PdfDocument implements Document {

public void access() {

System.out.println("PDF Document is accessed");

}

}

public class ExcelDocument implements Document {

public void access() {

System.out.println("Excel Document is accessed");

}

}

public abstract class DocumentFactory {

public abstract Document createDocument();

}

public class WordFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

public class PdfFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

public class ExcelFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

public class FactoryPatternTest {

public static void main(String[] args) {

DocumentFactory wordfactory = new WordFactory();

Document word = wordfactory.createDocument();

word.access();

DocumentFactory pdffactory = new PdfFactory();

Document pdf = pdffactory.createDocument();

pdf.access();

DocumentFactory excelfactory = new ExcelFactory();

Document excel = excelfactory.createDocument();

excel.access();

}

}

