## 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.

**Code**

**Logger.java:**

package com.cognizant.designpatterns;

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 com.cognizant.designpatterns;

public class LoggerTest {

public static void main(String[] args) {

Logger logger1 = Logger.*getInstance*();

logger1.log("Starting the application...");

Logger logger2 = Logger.*getInstance*();

logger2.log("Continuing the application...");

if (logger1 == logger2) {

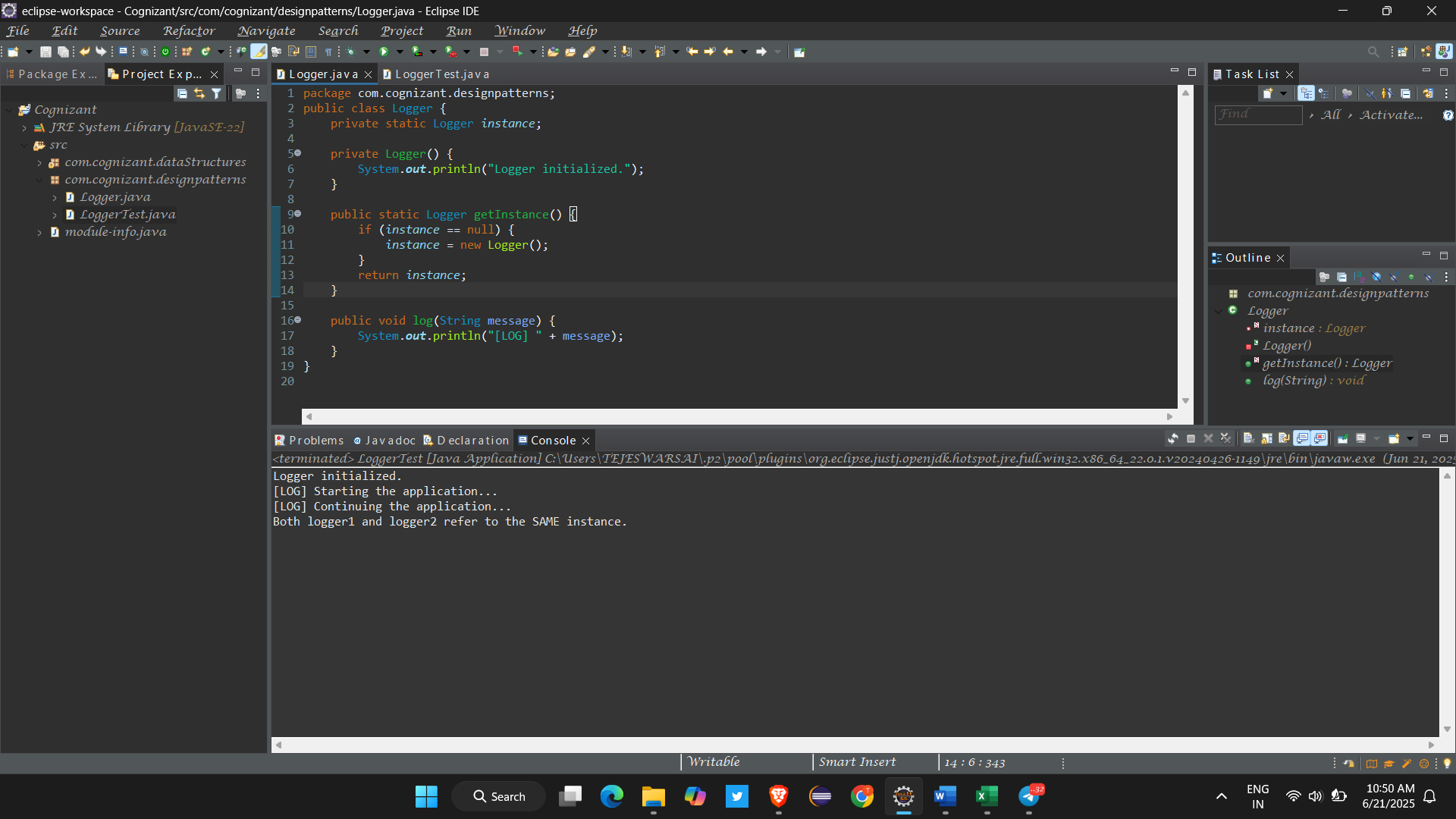
System.***out***.println("Both logger1 and logger2 refer to the SAME instance.");

} else {

System.***out***.println("logger1 and logger2 refer to DIFFERENT instances.");

}

}

}A screenshot of a computer

AI-generated content may be incorrect.

Exercise 2: Implementing the Factory Method Pattern

**Scenario:**

You are developing a document management system that needs to create different types of documents

**CODE :**

package com.cognizant.designpatterns;

public interface Document {

void open();

}

package com.cognizant.designpatterns;

public class ExcelDocument implements Document {

@Override

public void open() {

System.***out***.println("Opening an Excel document...");

}

}

package com.cognizant.designpatterns;

public class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

package com.cognizant.designpatterns;

public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PDFDocument();

}

}

package com.cognizant.designpatterns;

public class PDFDocument implements Document {

@Override

public void open() {

System.***out***.println("Opening a PDF document...");

}

}

package com.cognizant.designpatterns;

public class WordDocument implements Document {

@Override

public void open() {

System.***out***.println("Opening a Word document...");

}

}

package com.cognizant.designpatterns;

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

package com.cognizant.designpatterns;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

package com.cognizant.designpatterns;

public class FactoryTesterMain {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordDocumentFactory();

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();

}

}

