**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 :**

**Logger.java :**package singletonpatternexample;

public class Logger {

private static Logger *instance*;

private Logger()

{

}

public static Logger getInstance()

{

if(*instance* == null)

{

*instance* = new Logger();

}

return *instance*;

}

public void logger(String message)

{

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

}

}

**Main.java :**

package singletonpatternexample;

public class Main {

public static void main(String args[])

{

Logger logger1 = Logger.*getInstance*();

logger1.logger("This is the first instance");

Logger logger2 = Logger.*getInstance*();

logger2.logger("This is the second instance");

if(logger1 == logger2)

{

System.***out***.println("Both logger instances are same");

}

else

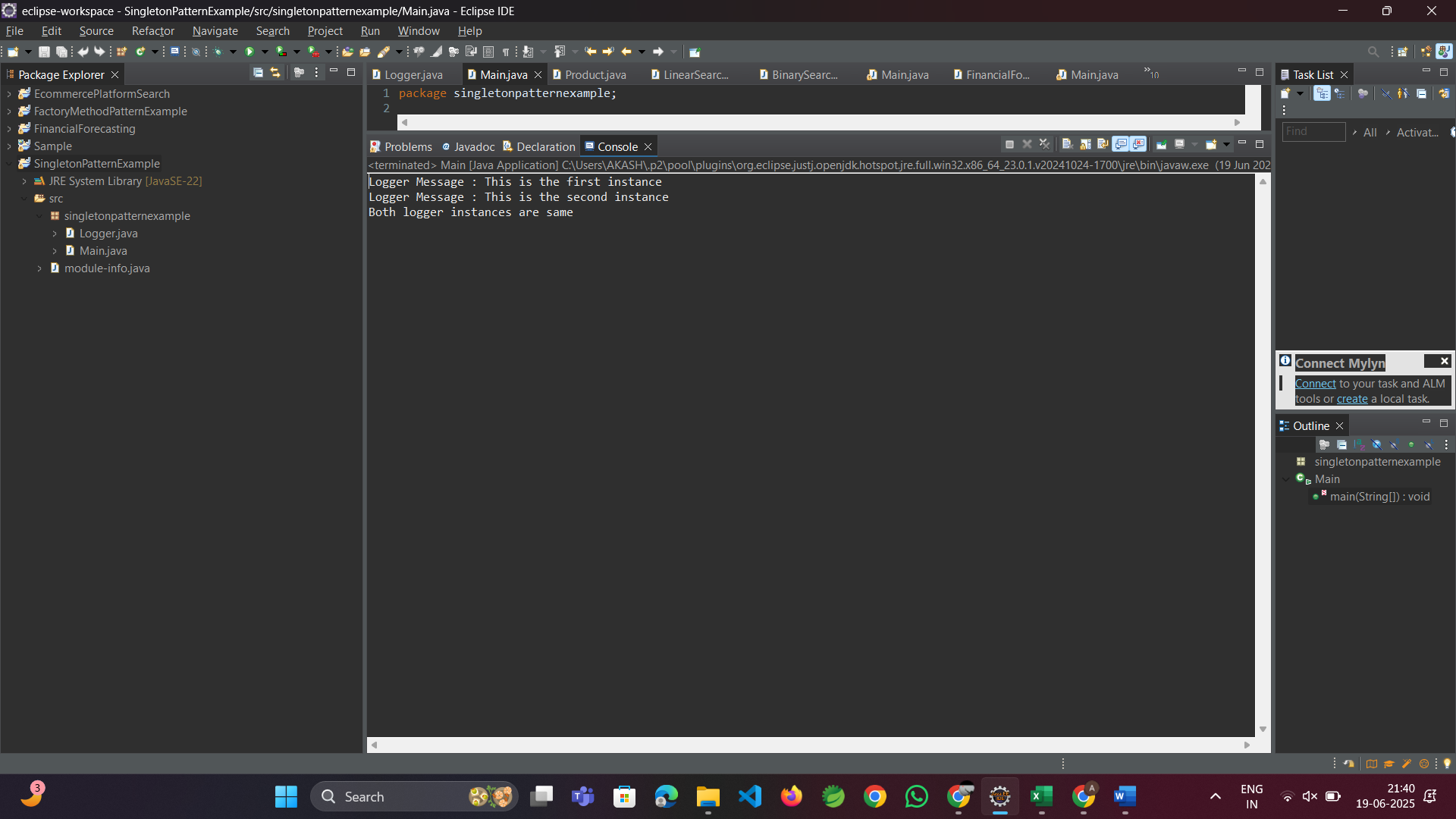
{

System.***out***.println("Both logger instances are different");

}

}

}

**Output Screenshots :   
  
**

**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 :  
  
Document.java :**package factorymethodpatternexample;

public interface Document {

public abstract void openTheDocument();

}

**DocumentFactory.java :**package factorymethodpatternexample;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**ExcelDocument.java :**package factorymethodpatternexample;

public class ExcelDocument implements Document {

*@Override*

public void openTheDocument()

{

System.***out***.println("This is an Excel Document");

}

}

**ExcelDocumentFactory.java :**package factorymethodpatternexample;

public class ExcelDocumentFactory extends DocumentFactory {

public Document createDocument()

{

return new ExcelDocument();

}

}

**Main.java :**package factorymethodpatternexample;

public class Main {

public static void main(String[] args) {

DocumentFactory wordDocumentFactory = new WordDocumentFactory();

Document wordDocument = wordDocumentFactory.createDocument();

wordDocument.openTheDocument();

DocumentFactory excelDocumentFactory = new ExcelDocumentFactory();

Document excelDocument = excelDocumentFactory.createDocument();

excelDocument.openTheDocument();

DocumentFactory pdfDocumentFactory = new PdfDocumentFactory();

Document pdfDocument = pdfDocumentFactory.createDocument();

pdfDocument.openTheDocument();

}

}

**PdfDocument.java :**package factorymethodpatternexample;

public class PdfDocument implements Document {

*@Override*

public void openTheDocument()

{

System.***out***.println("The is is a PDF Document");

}

}

**PdfDocumentFactory.java :**

package factorymethodpatternexample;

public class PdfDocumentFactory extends DocumentFactory {

public Document createDocument()

{

return new PdfDocument();

}

}

**WordDocument.java :**package factorymethodpatternexample;

public class WordDocument implements Document{

*@Override*

public void openTheDocument()

{

System.***out***.println("This is a Word Document");

}

}

**WordDocumentFactory.java :**package factorymethodpatternexample;

public class WordDocumentFactory extends DocumentFactory {

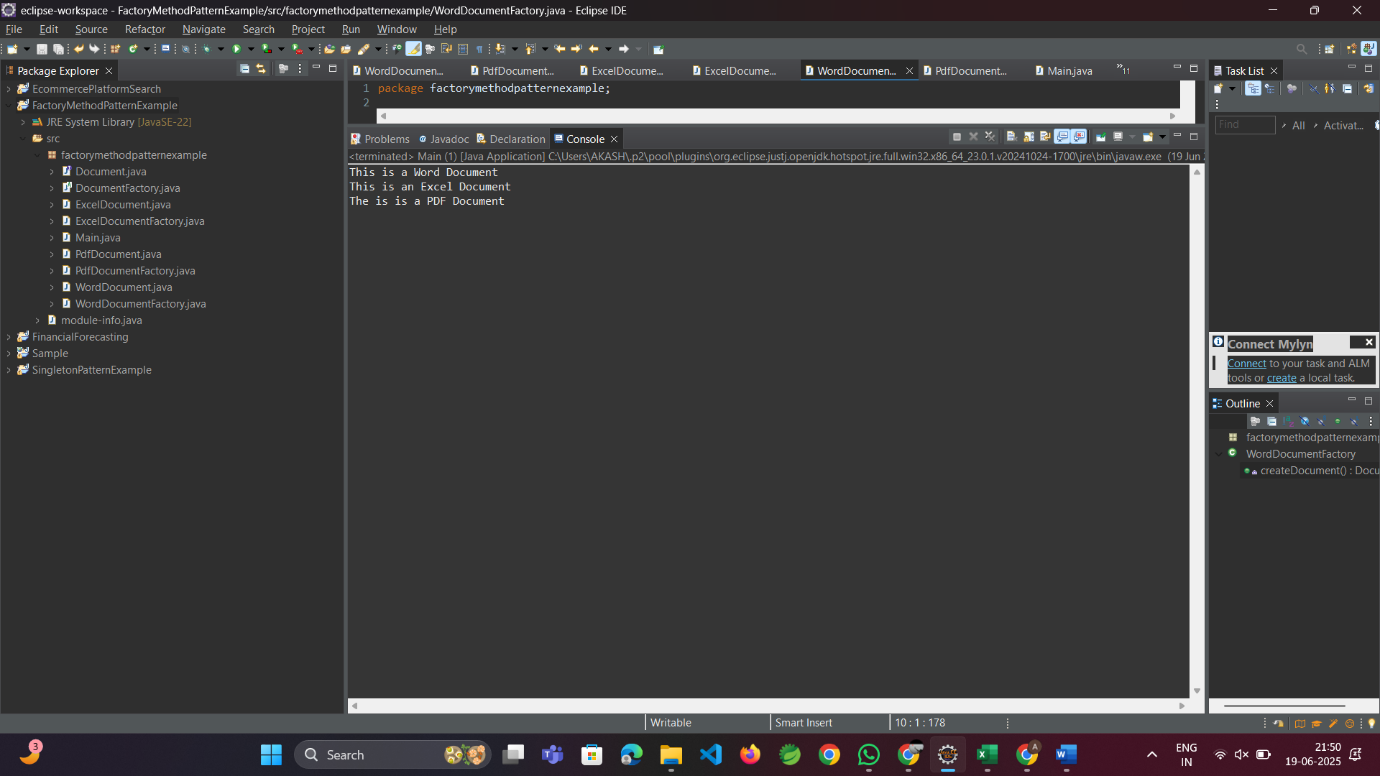
public Document createDocument()

{

return new WordDocument();

}

}

**Output Screenshots :**