**Exercise 1: Implementing the Singleton Pattern**

**Define and Implement the Singleton Class**

**File: Logger.java**

package Singleton ;

public class Logger {

// Step 1: Create a private static instance of Logger

private static Logger instance;

// Step 2: Make the constructor private

private Logger() {

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

}

// Step 3: Provide a public static method to get the instance

public static Logger getInstance() {

if (instance == null) {

instance = new Logger(); // create only once

}

return instance;

}

// Example method to simulate logging

public void log(String message) {

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

}

}

**Test the Singleton Implementation**

**File: Main.java**

package Singleton ;

public class Main {

public static void main(String[] args) {

// Get logger instance the first time

Logger logger1 = Logger.getInstance();

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

// Get logger instance the second time

Logger logger2 = Logger.getInstance();

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

// Check if both logger objects are the same

if (logger1 == logger2) {

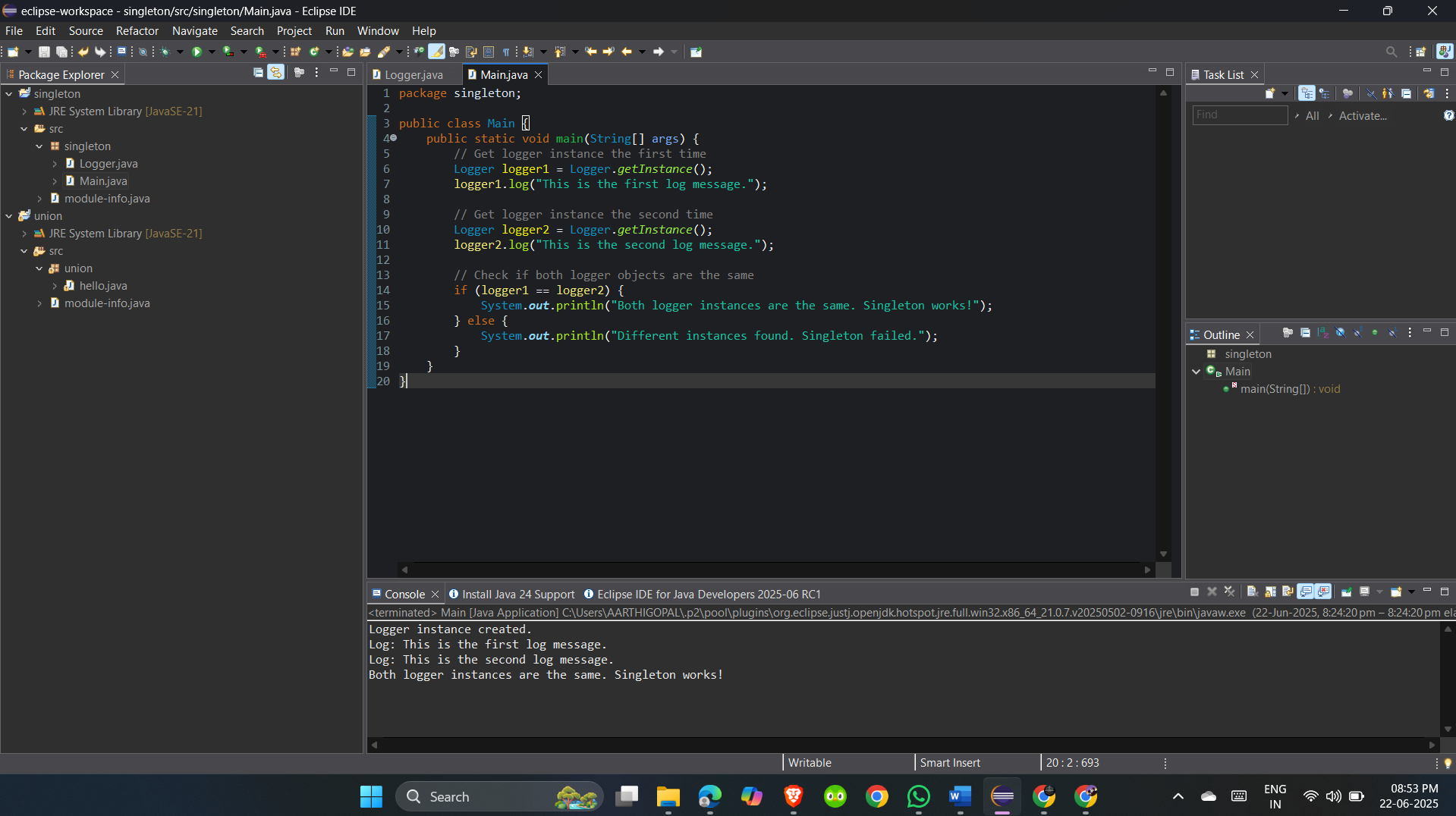
System.out.println("Both logger instances are the same. Singleton works!");

} else {

System.out.println("Different instances found. Singleton failed.");

}

}

}  


Output:  
Logger instance created.

Log: This is the first log message.

Log: This is the second log message.

Both logger instances are the same. Singleton works!

**Exercise :2  
Factory Method Pattern**

FactoryMethodPatternExample

└── src

└── factory

├── Document.java

├── WordDocument.java

├── PdfDocument.java

├── ExcelDocument.java

├── DocumentFactory.java

├── WordDocumentFactory.java

├── PdfDocumentFactory.java

├── ExcelDocumentFactory.java

└── Main.java  
**Codes:  
1. Document.java**

package factory;

public interface Document {

void open();

}

**2. WordDocument.java**

package factory;

public class WordDocument implements Document {

@Override

public void open() {

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

}

}

**3. PdfDocument.java**

package factory;

public class PdfDocument implements Document {

@Override

public void open() {

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

}

}

**4. ExcelDocument.java**

package factory;

public class ExcelDocument implements Document {

@Override

public void open() {

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

}

}

**5. DocumentFactory.java**

package factory;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**6. WordDocumentFactory.java**

package factory;

public class WordDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new WordDocument();

}

}

**7. PdfDocumentFactory.java**

package factory;

public class PdfDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new PdfDocument();

}

}

**8. ExcelDocumentFactory.java**

package factory;

public class ExcelDocumentFactory extends DocumentFactory {

@Override

public Document createDocument() {

return new ExcelDocument();

}

}

**9. Main.java**

package factory;

public class Main {

public static void main(String[] args) {

// Word Document

DocumentFactory wordFactory = new WordDocumentFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

// PDF Document

DocumentFactory pdfFactory = new PdfDocumentFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

// Excel Document

DocumentFactory excelFactory = new ExcelDocumentFactory();

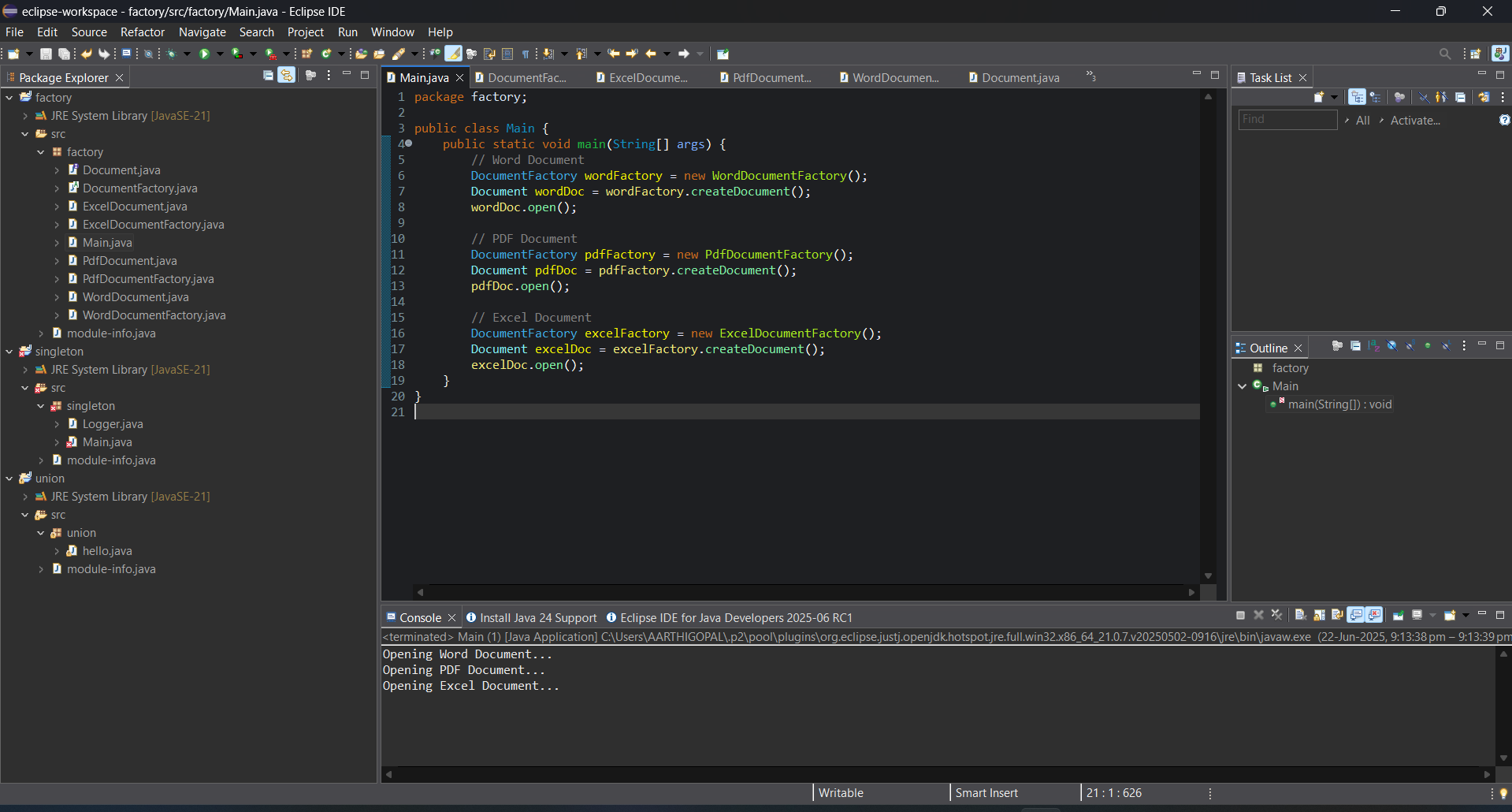
Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

Output:

  
Output:

Opening Word Document...

Opening PDF Document...

Opening Excel Document...