**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.

**Algorithm:**

1. Define Logger class with a private static instance and private constructor.

2. In constructor, print "Logger created" to show it’s called only once.

3. Implement getInstance() to return the single instance of Logger.

4. If instance is null, create it; else return existing instance.

5. Add log(String message) to print the given log message.

6. In Main, get logger instance 3 times and log different messages.

7. Compare all logger references; if equal, print "Same Logger used everywhere."

**Code:**

**Logger.java**

package mylogger;

public class Logger {

private static Logger instance;

private Logger() {

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

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

System.out.println(message);

}

}

**Main.java**

package mylogger;

public class Main {

public static void main(String[] args) {

Logger loggerA = Logger.getInstance();

loggerA.log("User clicked login.");

Logger loggerB = Logger.getInstance();

loggerB.log("Fetching user data...");

Logger loggerC = Logger.getInstance();

loggerC.log("User logged out.");

if (loggerA == loggerB && loggerB == loggerC) {

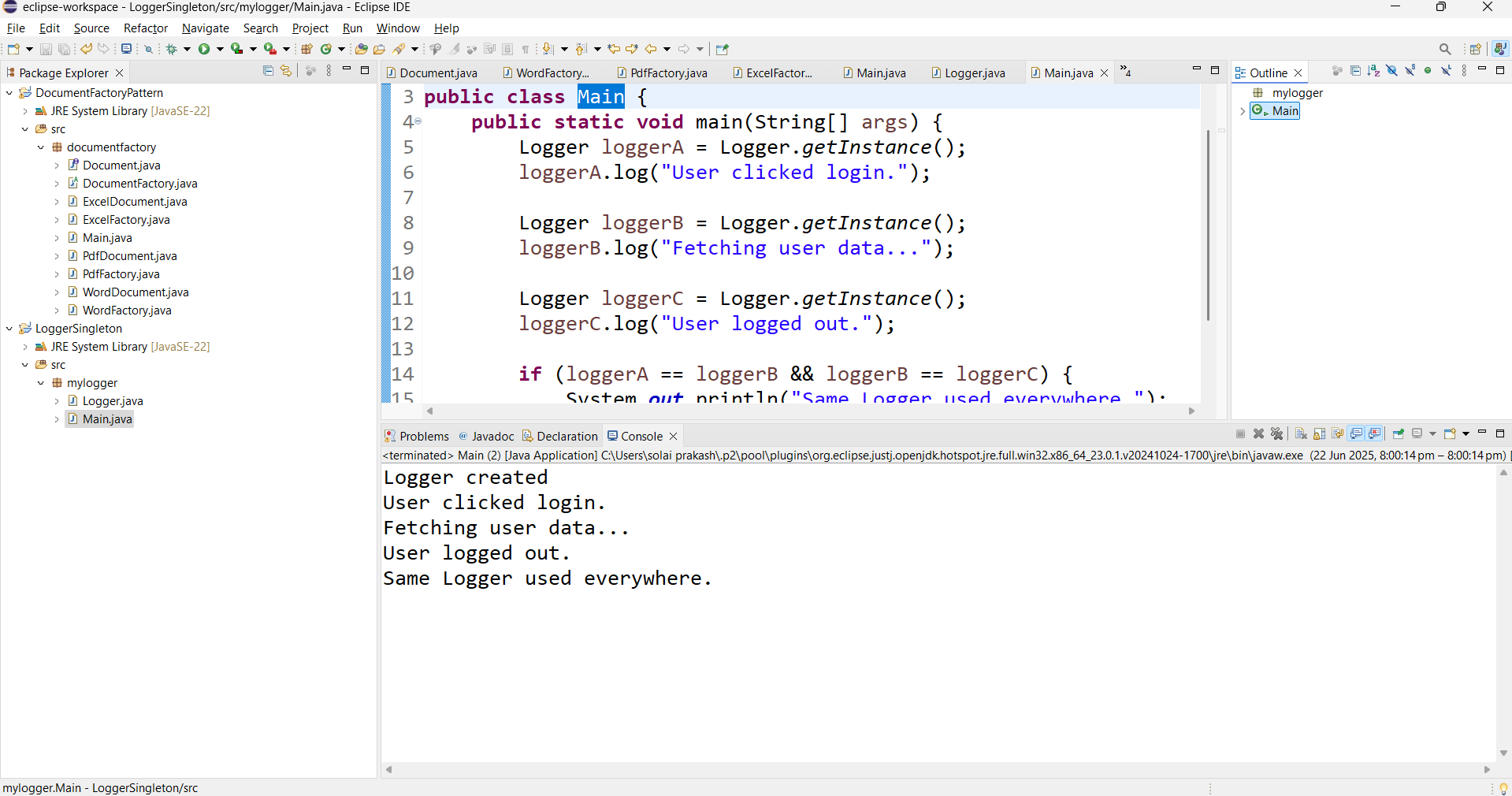
System.out.println("Same Logger used everywhere.");

}

}

}

**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.

**Algorithm:**

1. Define Document interface with open() and followUp() methods.

2. Implement WordDocument with open(): "Opening a Word document..." and followUp(): "Editing Word document...".

3. Implement PdfDocument with open(): "Opening a PDF document..." and followUp(): "Signing PDF document...".

4. Implement ExcelDocument with open(): "Opening an Excel document..." and followUp(): "Calculating in Excel document...".

5. Create abstract class DocumentFactory with abstract method createDocument().

6. Implement WordFactory that returns WordDocument from createDocument().

7. Implement PdfFactory that returns PdfDocument from createDocument().

8. Implement ExcelFactory that returns ExcelDocument from createDocument().

9. In Main, use WordFactory to create and call open() and followUp() → displays Word messages.

10. Use PdfFactory to create and call open() and followUp() → displays PDF messages.

11. Use ExcelFactory to create and call open() and followUp() → displays Excel messages.

12. Each document type displays its respective actions on the console output.

**Code:**

Document.java

package documentfactory;

public interface Document {

void open();

void followUp();

}

WordDocument.java

package documentfactory;

public class WordDocument implements Document {

public void open() {

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

}

public void followUp() {

System.out.println("Editing Word document...");

}

}

PdfDocument.java

package documentfactory;

public class PdfDocument implements Document {

public void open() {

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

}

public void followUp() {

System.out.println("Signing PDF document...");

}

}

ExcelDocument.java

package documentfactory;

public class ExcelDocument implements Document {

public void open() {

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

}

public void followUp() {

System.out.println("Calculating in Excel document...");

}

}

DocumentFactory.java

package documentfactory;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

WordFactory.java

package documentfactory;

public class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

PdfFactory.java

package documentfactory;

public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

ExcelFactory.java

package documentfactory;

public class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

Main.java

package documentfactory;

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

wordDoc.followUp();

System.out.println();

DocumentFactory pdfFactory = new PdfFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

pdfDoc.followUp();

System.out.println();

DocumentFactory excelFactory = new ExcelFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

excelDoc.followUp();

}

}

OUTPUT:

