**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:

Logger.java

**public** **class** Logger {

**private** **static** Logger *instance*;

**private** Logger() {

System.***out***.println("Only one logger is created");

}

**public** **static** Logger getInstance() {

**if** (*instance* == **null**) {

*instance* = **new** Logger();

}

**return** *instance*;

}

**public** **void** logFirstOutput(String first) {

System.***out***.println(first);

}

**public** **void** logSecondOutput(String second) {

System.***out***.println(second);

}

}

LoggerTest.java

**public** **class** LoggerTest {

**public** **static** **void** main(String[] args) {

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

logger1.logFirstOutput("This is a first logger");

logger2.logSecondOutput("This is a second logger");

**if** (logger1 == logger2) {

System.***out***.println("Yes! first and second are the same instance.");

} **else** {

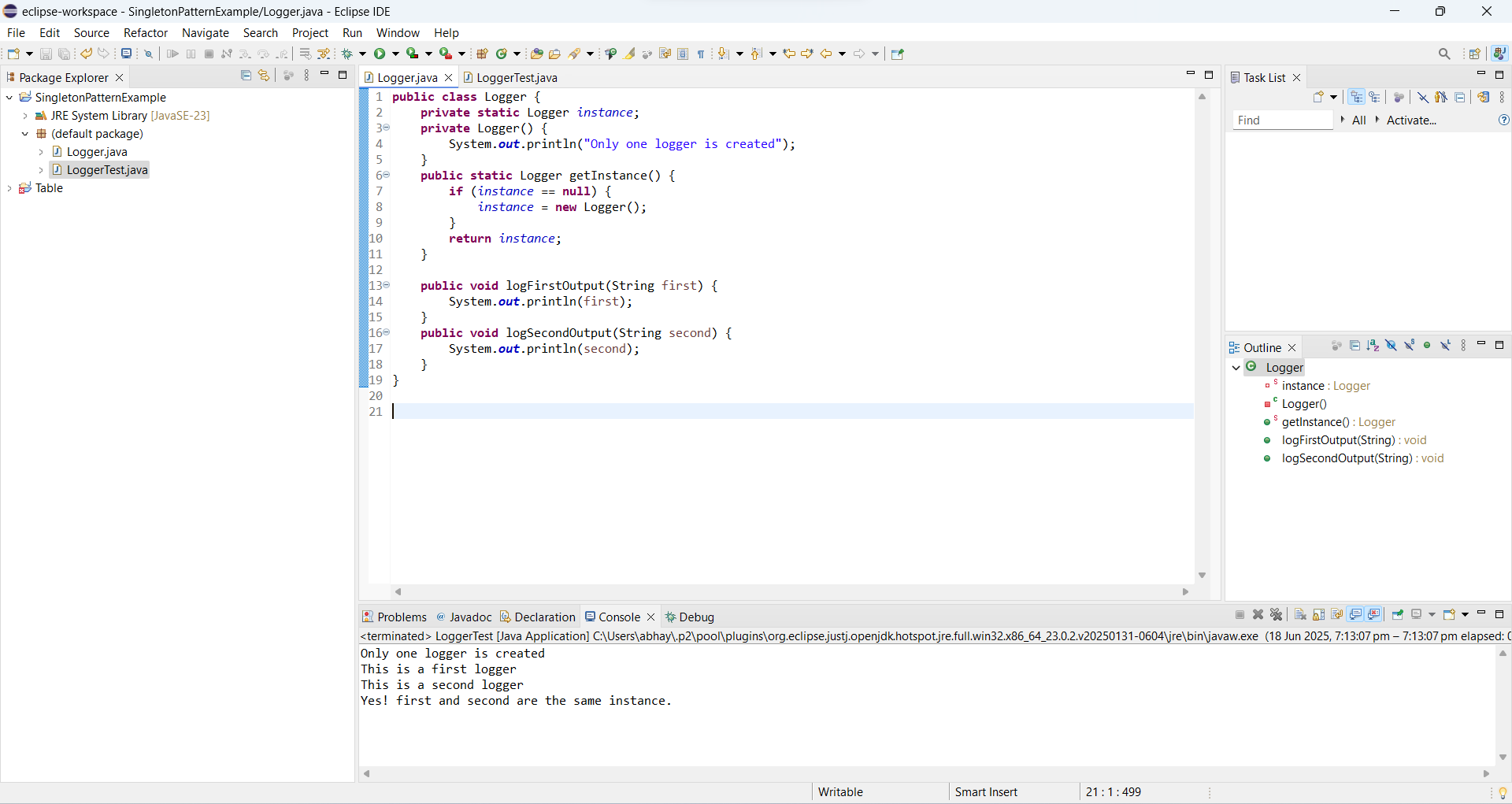
System.***out***.println("Oops! first and second are different.");

}

}

}

Outputs:



**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:

Document.java

This creates the method void which we will use for different purpose.

**package** document;

**public** **interface** Document{

**void** open();

}

DocumentFactory.java

**package** document;

**public** **abstract** **class** DocumentFactory {

**public** **abstract** Document createDocument();

}

WordDocument.java

**package** document;

**public** **class** WordDocument **implements** Document {

**public** **void** open() {

System.***out***.println("This opens word document by same method open()");

}

}

WordDocumentFactory.java

**package** document;

**public** **class** WordDocumentFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** WordDocument();

}

}

ExcelDocument.java

**package** document;

**public** **class** ExcelDocument **implements** Document {

**public** **void** open() {

System.***out***.println("This opens the excel file by using same open() method");

}

}

ExcelDocumentFactory.java

**package** document;

**public** **class** ExcelDocumentFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** ExcelDocument();

}

}

PdfDocument.java

**package** document;

**public** **class** PdfDocument **implements** Document {

**public** **void** open() {

System.***out***.println("This opens the pdf with the same method open()");

}

}

PdfDocumentFactory.java

**package** document;

**public** **class** PdfDocumentFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** PdfDocument();

}

}

Main.java

**package** document;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

DocumentFactory excelFactory = **new** ExcelDocumentFactory();

Document excel = excelFactory.createDocument();

excel.open();

DocumentFactory wordFactory = **new** WordDocumentFactory();

Document word = wordFactory.createDocument();

word.open();

DocumentFactory pdfFactory = **new** PdfDocumentFactory();

Document pdf = pdfFactory.createDocument();

pdf.open();

}

}

Output:

