**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**

**package** com.singleton.logger;

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

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

**private** Logger() {

System.***out***.println("Logger is initialized");

}

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

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

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

}

**return** *instance*;

}

**public** **void** log(String msg) {

System.***out***.println("[LOG]: " + msg);

}

}

**LoggerTest.java**

**package** com.singleton.logger;

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

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

Logger l1 = Logger.*getInstance*();

l1.log("This is the First message");

Logger l2 = Logger.*getInstance*();

l2.log("This is the Second message");

**if** (l1 == l2) {

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

} **else** {

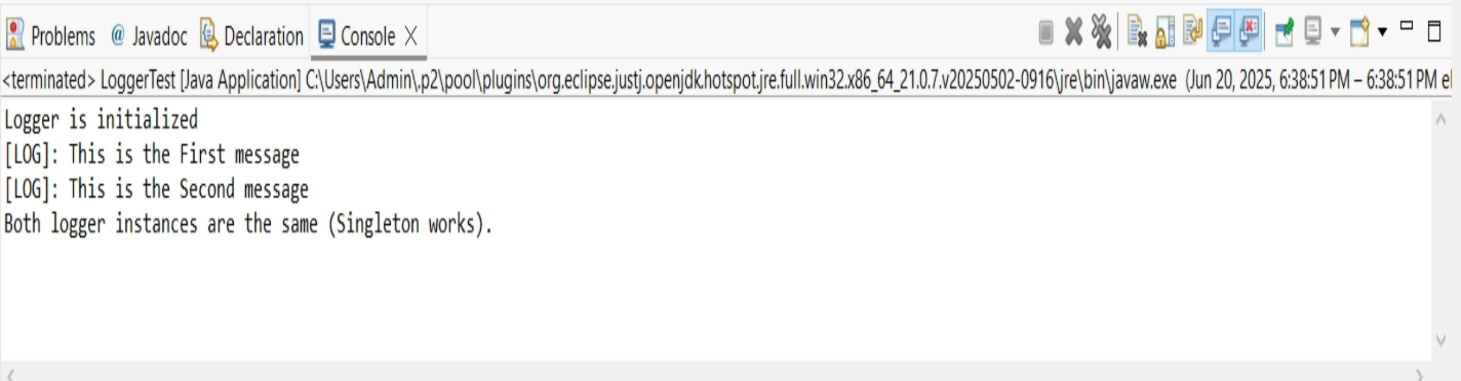
System.***out***.println("Logger instances are different (Singleton failed).");

}

}

}

**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.

**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**

**package** com.factory.documents;

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

**void** open();

}

**DocumentFactory.java**

**package** com.factory.documents;

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

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

}

**DocumentFactory.java**

**package** com.factory.documents;

**public** **class** DocumentFactoryTest {

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

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

Document word = wordFactory.createDocument();

word.open();

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

Document pdf = pdfFactory.createDocument();

pdf.open();

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

Document excel = excelFactory.createDocument();

excel.open();

}

}

**ExcelDocument.java**

**package** com.factory.documents;

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

@Override

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

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

}

}

**ExcelDocumentFactory.java**

**package** com.factory.documents;

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

@Override

**public** Document createDocument() {

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

}

}

**PdfDocument.java**

**package** com.factory.documents;

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

@Override

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

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

}

}

**PdfDocumentFactory.java**

**package** com.factory.documents;

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

@Override

**public** Document createDocument() {

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

}

}

**WordDocument.java**

**package** com.factory.documents;

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

@Override

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

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

}

}

**WordDocumentFactory.java**

**package** com.factory.documents;

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

@Override

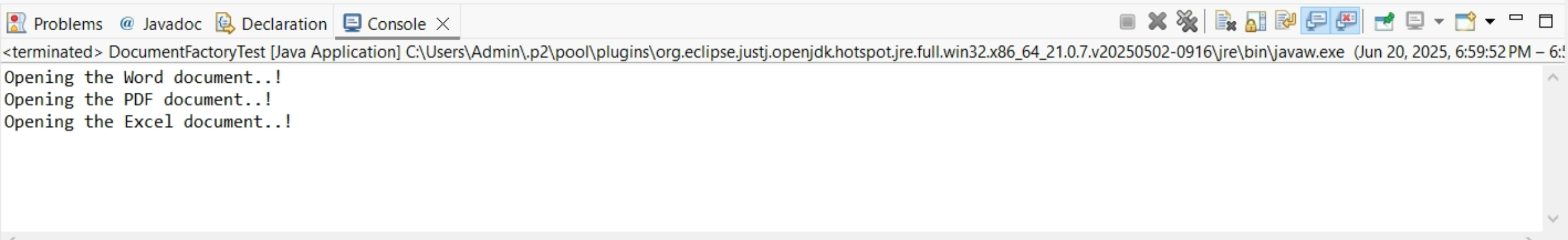
**public** Document createDocument() {

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

}

}

**OUTPUT:**

****