**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

Code:

**Logger.java**

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

// private static instance for object

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

// Step 2: private constructor to stop create a object in multiple times

**private** Logger() {

System.***out***.println("Logger Initialized");

}

// Step 3: public static method to get instance

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

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

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

}

**return** *instance*;

}

}

**Main.java**

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

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

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

System.***out***.println(logger1==logger2);

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

System.***out***.println("THIS FOLLOWS SINGLETON PATTERN");

}

**else** {

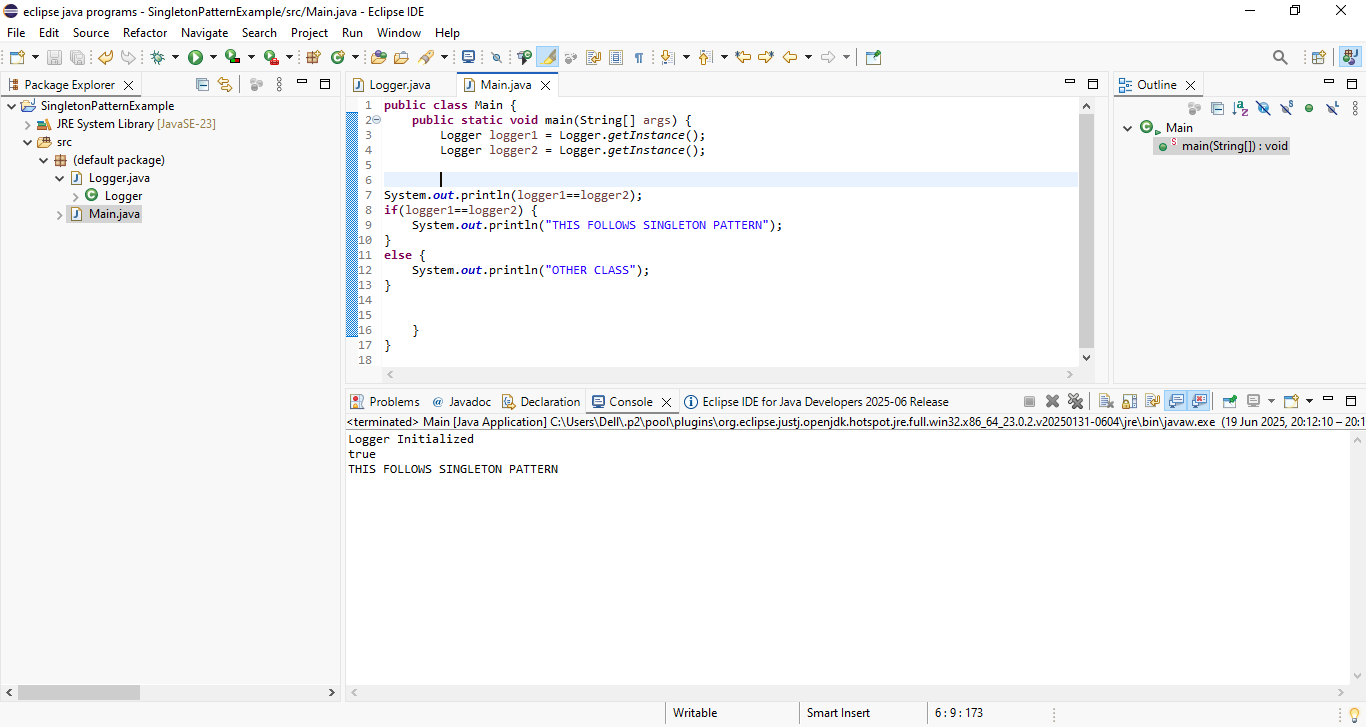
System.***out***.println("OTHER CLASS");

}

}

}

**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.

Code:

Document.java

//Product Document

**interface** Documents {

//create the template for each documents

**void** type();

}

WordDocument.java

//Concrete document

**public** **class** WordDocument **implements** Documents {

**public** **void** type() {

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

System.***out***.println("FOLLOWS FACTORY METHOD");

}

}

PdfDocument.java

//Concrete document

**public** **class** PdfDocument **implements** Documents {

**public** **void** type() {

System.***out***.println("Opening Pdf Document");

System.***out***.println("FOLLOWS FACTORY METHOD");

}

}

ExcelDocument.java

//Concrete document

**public** **class** ExcelDocument **implements** Documents {

**public** **void** type() {

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

System.***out***.println("FOLLOWS FACTORY METHOD");

}

}

Implement the Factory Method

DocumentFactory.java

//Create interface for DocumentFactory with method

**public** **interface** DocumentFactory {

Documents CreateDocument();

}

WordDocumentFactory.java

//Concrete products

**public** **class** WordDocumentFactory **implements** DocumentFactory {

**public** Documents CreateDocument() {

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

}

}

PdfDocumentFactory.java

//Concrete products

**public** **class** PdfDocumentFactory **implements** DocumentFactory {

**public** Documents CreateDocument() {

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

}

}

ExcelDocumentFactory.java

//Concrete products

**public** **class** ExcelDocumentFactory **implements** DocumentFactory {

**public** Documents CreateDocument() {

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

}

}

Main.java

**public** **class** FactoryTest {

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

DocumentFactory factory1 = **new** WordDocumentFactory();

Documents doc1 = factory1.CreateDocument();

doc1.type(); // Word output

DocumentFactory factory2 = **new** PdfDocumentFactory();

Documents doc2 = factory2.CreateDocument();

doc2.type(); // PDF output

DocumentFactory factory3 = **new** ExcelDocumentFactory();

Documents doc3 = factory3.CreateDocument();

doc3.type();

}

}

OUTPUT:

