WEEK-1 HANDS-ON

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.

**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.

PROGRAM:

**package** singletonpatternexample;

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

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

**private** Logger() {}

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

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

**synchronized** (Logger.**class**) {

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

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

}

}

}

**return** *instance*;

}

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

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

}

}

**package** singletonpatternexample;

**public** **class** SingletonTest {

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

Logger first = Logger.*getInstance*();

Logger second = Logger.*getInstance*();

System.***out***.println("Same instance: " + (first == second));

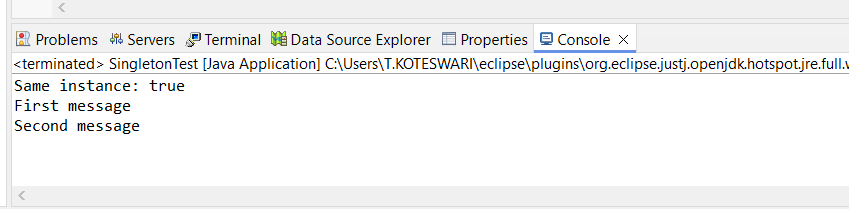
first.log("First message");

second.log("Second message");

}

}

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.

PROGRAM:

**package** factorymethodpatternexample;

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

**void** display();

}

**package** factorymethodpatternexample;

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

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

}

**package** factorymethodpatternexample;

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

**public** **void** display() {

System.***out***.println("Excel document");

}

}

**package** factorymethodpatternexample;

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

**public** Document createDocument() {

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

}

}

**package** factorymethodpatternexample;

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

**public** **void** display() {

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

}

}

**package** factorymethodpatternexample;

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

**public** Document createDocument() {

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

}

}

**package** factorymethodpatternexample;

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

**public** **void** display() {

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

}

}

**package** factorymethodpatternexample;

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

**public** Document createDocument() {

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

}

}

**package** factorymethodpatternexample;

**public** **class** FactoryMethodDemo {

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

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

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

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

Document word = wordFactory.createDocument();

Document pdf = pdfFactory.createDocument();

Document excel = excelFactory.createDocument();

word.display();

pdf.display();

excel.display();

}

}

OUTPUT:

