**Exercise 1: Implementing the Singleton Pattern**

**Title:**

Singleton Pattern – Logger Utility

**Description:**

I created a Logger utility using the Singleton Pattern to make sure that only one instance of the logger is active throughout the entire application.

**Key Notes:**

* Only one instance of the Logger class is created during runtime.
* Lazy initialization is used to create the instance when first needed.
* Verified by comparing object references to ensure the same instance is used.

**Code:**

**LoggerTest .java**

public class LoggerTest {

    private static LoggerTest instance;

    private LoggerTest() {

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

    }

    public static LoggerTest getInstance() {

        if (instance == null) {

            instance = new LoggerTest();

        }

        return instance;

    }

    public void log(String msg) {

        System.out.println("Log: " + msg);

    }

    public static void main(String[] args) {

        LoggerTest logger1 = LoggerTest.getInstance();

        LoggerTest logger2 = LoggerTest.getInstance();

        logger1.log("First log message.");

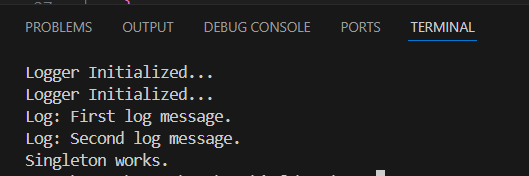
        logger2.log("Second log message.");

        System.out.println(logger1 == logger2 ? "Singleton works." : "Singleton failed.");

    }

}

**Output:**



**Exercise 2: Implementing the Factory Method Pattern**

**Title:**

Factory Method Pattern – Document Creation System

**Description:**

I developed a Document Creation System using the Factory Method Pattern to generate different types of documents like Word, PDF, and Excel, making the code flexible for future extensions.

**Key Notes:**

* Interface Document used for common structure.
* Separate classes for Word, PDF, and Excel documents.
* Factory classes handle object creation, making the main code flexible and extendable.

**Code:**

**DocumentFactoryTest.java**

interface Document {

void display();

}

class WordDocument implements Document {

public void display() {

System.out.println("This is a Word document.");

}

}

class PdfDocument implements Document {

public void display() {

System.out.println("This is a PDF document.");

}

}

class ExcelDocument implements Document {

public void display() {

System.out.println("This is an Excel spreadsheet.");

}

}

class DocumentFactory {

public static Document getDocument(String type) {

switch (type) {

case "WORD": return new WordDocument();

case "PDF": return new PdfDocument();

case "EXCEL": return new ExcelDocument();

default: return null;

}

}

}

public class DocumentFactoryTest {

public static void main(String[] args) {

Document doc1 = DocumentFactory.getDocument("WORD");

Document doc2 = DocumentFactory.getDocument("PDF");

Document doc3 = DocumentFactory.getDocument("EXCEL");

doc1.display();

doc2.display();

doc3.display();

}

}

**Output:**

