**Exercise 1: Implementing the Singleton Pattern**

public class Logger {

private static Logger instance;

private Logger() {

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

}

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

public void log(String message) {

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

}

}

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

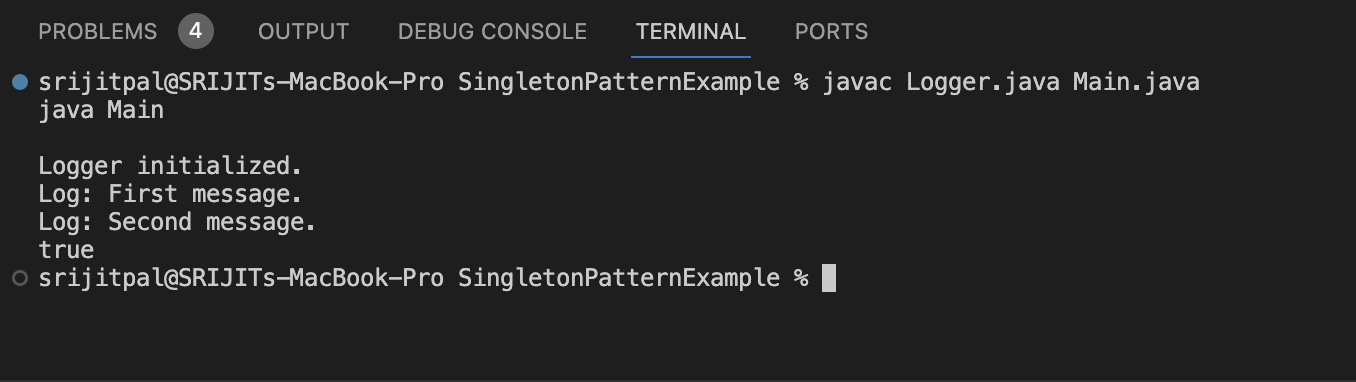
Logger logger2 = Logger.getInstance();

logger1.log("First message.");

logger2.log("Second message.");

System.out.println(logger1 == logger2); // true

}

}

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

public interface Document {

void open();

}

public class WordDocument implements Document {

public void open() {

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

}

}

public class PdfDocument implements Document {

public void open() {

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

}

}

public class ExcelDocument implements Document {

public void open() {

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

}

}

public abstract class DocumentFactory {

public abstract Document createDocument();

}

public class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

public class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

public class Main {

public static void main(String[] args) {

DocumentFactory factory = new PdfFactory();

Document doc = factory.createDocument();

doc.open();

}

}