**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.

public class Logger {

    private static Logger instance;

    // Private constructor prevents instantiation from other classes

    private Logger() {

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

    }

    // Public method to provide access to the instance

    public static Logger getInstance() {

        if (instance == null) {

            instance = new Logger();

        }

        return instance;

    }

    public void log(String message) {

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

    }

}

public class LoggerTest {

    public static void main(String[] args) {

        Logger logger1 = Logger.getInstance();

        Logger logger2 = Logger.getInstance();

        logger1.log("This is the first log message.");

        logger2.log("This is the second log message.");

        // Verifying both logger instances are the same

        if (logger1 == logger2) {

            System.out.println("Both loggers are the same instance.");

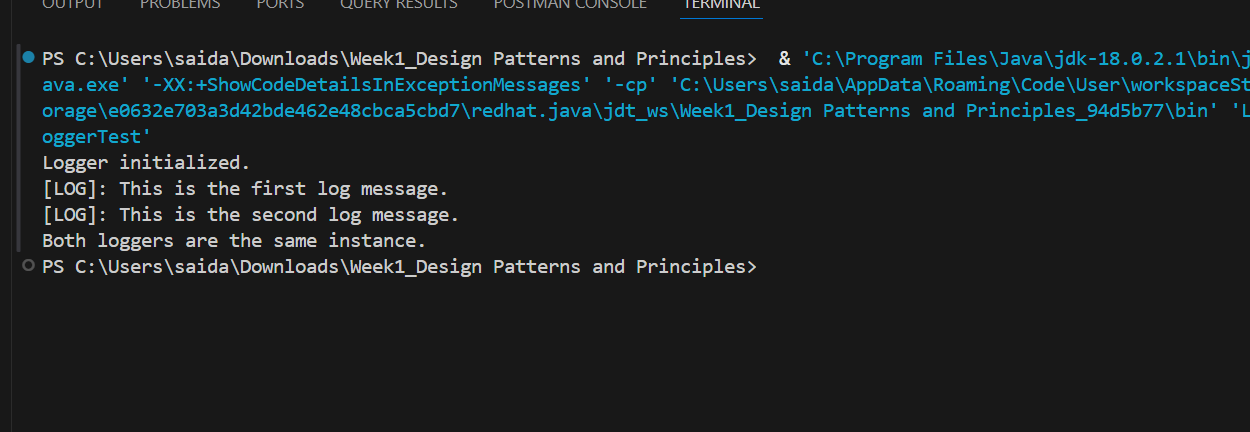
        } else {

            System.out.println("Different logger instances exist!");

        }

    }

}



**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.

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelFactory();

Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

interface Document {

void open();

}

class WordDocument implements Document {

public void open() {

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

}

}

class PdfDocument implements Document {

public void open() {

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

}

}

class ExcelDocument implements Document {

public void open() {

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

}

}

abstract class DocumentFactory {

public abstract Document createDocument();

}

class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

