**Mandatory Questions:  
  
1)Implementing the singleton Pattern  
  
Question:**You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

**Solution  
  
Code  
Logger.java**  
public class Logger {

    private static Logger l;

    private Logger() {

    }

    public static Logger getInstance() {

        if (l == null) {

            l = new Logger();

        }

        return l;

    }

    public void log(String message) {

        System.out.println(message);

    }

}

**LoggerTest.java**

public class LoggerTest {

    public static void main(String[] args) {

        Logger l1 = Logger.getInstance();

        Logger l2 = Logger.getInstance();

        if (l1 == l2) {

            System.out.println("Logger is a singleton. Both references point to the same instance.");

        } else {

            System.out.println("Logger is not a singleton. Different instances were created.");

        }

    }

}

**SingletonPatternExample.java**  
public class SingletonPatternExample {

    public static void main(String[] args) {

        Logger l1 = Logger.getInstance();

        Logger l2 = Logger.getInstance();

        System.out.println("Logger 1: " + l1);

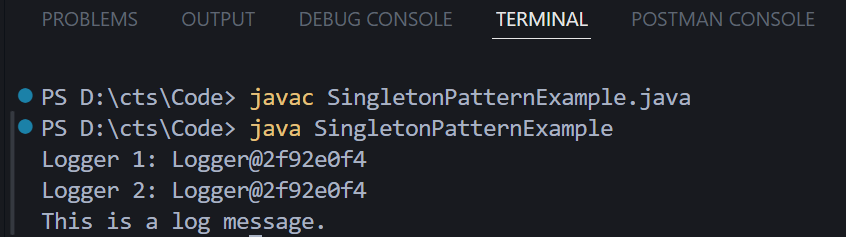
        System.out.println("Logger 2: " + l2);

        l1.log("This is a log message.");

    }

}

**Output:**

**Screenshots:  
  
**

**2)Implementing The Factory Method Pattern  
  
Question:**

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.

**Solution:  
  
Code:  
FactoryMethodPatternExample.java**  
interface Document {

    String getContent();

    String getType();

}

class ExcelDocument implements Document {

    @Override

    public String getContent() {

        return "This is an Excel document.";

    }

    @Override

    public String getType() {

        return "Excel";

    }

}

class PdfDocument implements Document {

    @Override

    public String getContent() {

        return "This is the content of a PDF document.";

    }

    @Override

    public String getType() {

        return "PDF Document";

    }

}

class WordDocument implements Document {

    @Override

    public String getContent() {

        return "This is the content of a Word document.";

    }

    @Override

    public String getType() {

        return "Word Document";

    }

}

abstract class DocumentFactory {

    public abstract Document createDocument();

}

class ExcelDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new ExcelDocument();

    }

}

class PdfDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new PdfDocument();

    }

}

class WordDocumentFactory extends DocumentFactory {

    @Override

    public Document createDocument() {

        return new WordDocument();

    }

}

public class FactoryMethodPatternExample {

    public static void main(String[] args) {

        DocumentFactory wordFactory = new WordDocumentFactory();

        Document wordDoc = wordFactory.createDocument();

        System.out.println("Created: " + wordDoc.getType() + " with content: " + wordDoc.getContent());

        DocumentFactory pdfFactory = new PdfDocumentFactory();

        Document pdfDoc = pdfFactory.createDocument();

        System.out.println("Created: " + pdfDoc.getType() + " with content: " + pdfDoc.getContent());

        DocumentFactory excelFactory = new ExcelDocumentFactory();

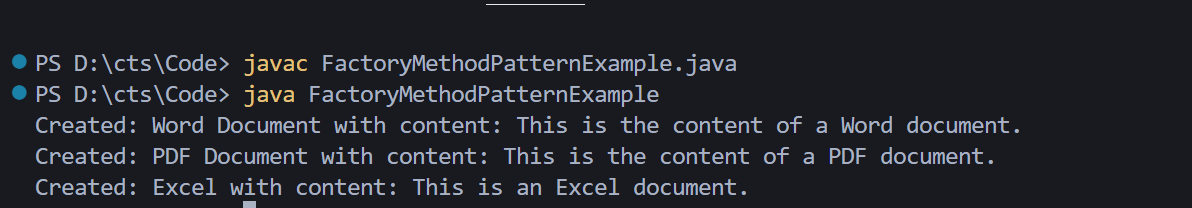
        Document excelDoc = excelFactory.createDocument();

        System.out.println("Created: " + excelDoc.getType() + " with content: " + excelDoc.getContent());

    }

}

**Output:**

**Screenshots:  
**