Design principles & Patterns:

Exercise 1: Implementing the Singleton Pattern

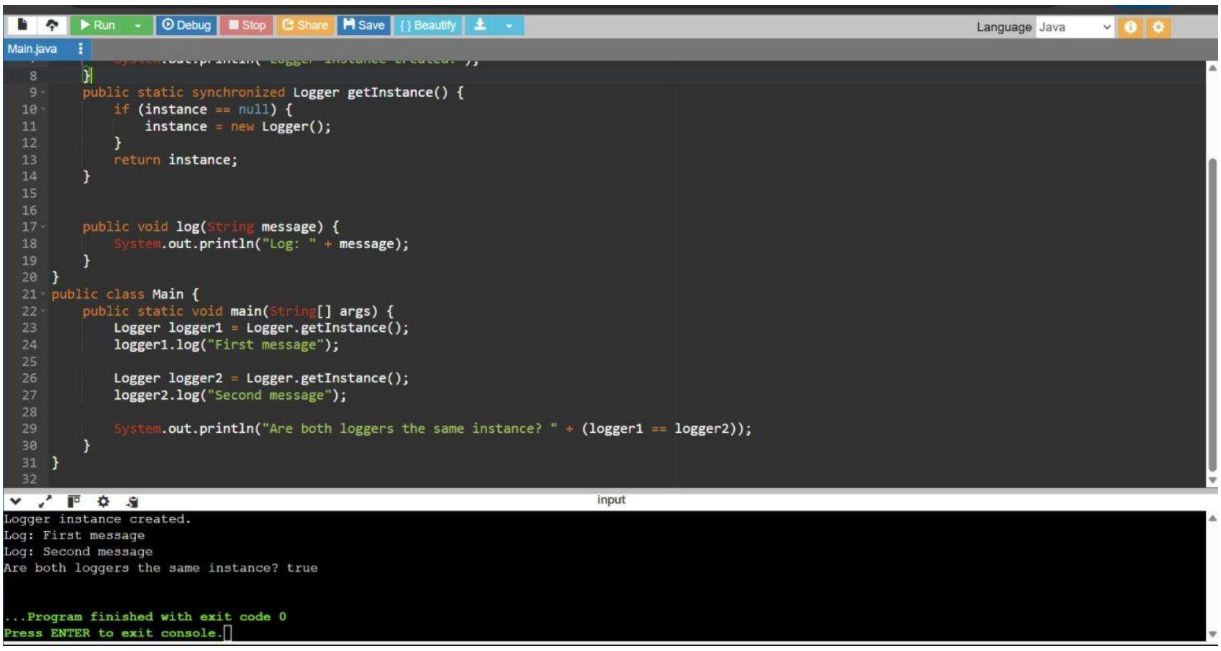
Scenario:

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

Code:

```
class Logger {
   private static Logger instance;

   private Logger() {
      System.out.println("Logger instance created.");
   }

   public static synchronized 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();
      logger1.log("First message");

      Logger logger2 = Logger.getInstance();
      logger2.log("Second message");

      System.out.println("Are both loggers the same instance? " + (logger1 == logger2));
   }
}
```

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.

Code:

```
interface Document {
    void open();
}

class WordDocument implements Document {
    @Override
```

```
    public void open() {
        System.out.println("Opening Word document.");
    }
}

class PdfDocument implements Document {
    @Override
    public void open() {
        System.out.println("Opening PDF document.");
    }
}

class ExcelDocument implements Document {
    @Override
    public void open() {
        System.out.println("Opening Excel document.");
    }
}

enum DocumentType {
    WORD, PDF, EXCEL
}

class DocumentFactory {
    public static Document createDocument(DocumentType type) {
        switch (type) {
            case WORD:
```

```
            return new WordDocument();
        case PDF:
            return new PdfDocument();
        case EXCEL:
            return new ExcelDocument();
        default:
            throw new IllegalArgumentException("Invalid document type");
    }
  }
}

public class Main {
  public static void main(String[] args) {
    Document wordDoc =
DocumentFactory.createDocument(DocumentType.WORD);
    wordDoc.open();

    Document pdfDoc =
DocumentFactory.createDocument(DocumentType.PDF);
    pdfDoc.open();

    Document excelDoc =
DocumentFactory.createDocument(DocumentType.EXCEL);
    excelDoc.open();
  }
}
```

Output:

Run Debug Stop Share Save { } Beautify Language Java

Main.java

```java
            case WORD:
                return new WordDocument();
            case PDF:
                return new PdfDocument();
            case EXCEL:
                return new ExcelDocument();
            default:
                throw new IllegalArgumentException("Invalid document type");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Document wordDoc = DocumentFactory.createDocument(DocumentType.WORD);
        wordDoc.open();

        Document pdfDoc = DocumentFactory.createDocument(DocumentType.PDF);
        pdfDoc.open();

        Document excelDoc = DocumentFactory.createDocument(DocumentType.EXCEL);
        excelDoc.open();
    }
}
```

input

```
Opening Word document.
Opening PDF document.
Opening Excel document.


...Program finished with exit code 0
Press ENTER to exit console.
```