

Design principles & Patterns:

Exercise 1: Implementing the Singleton Pattern

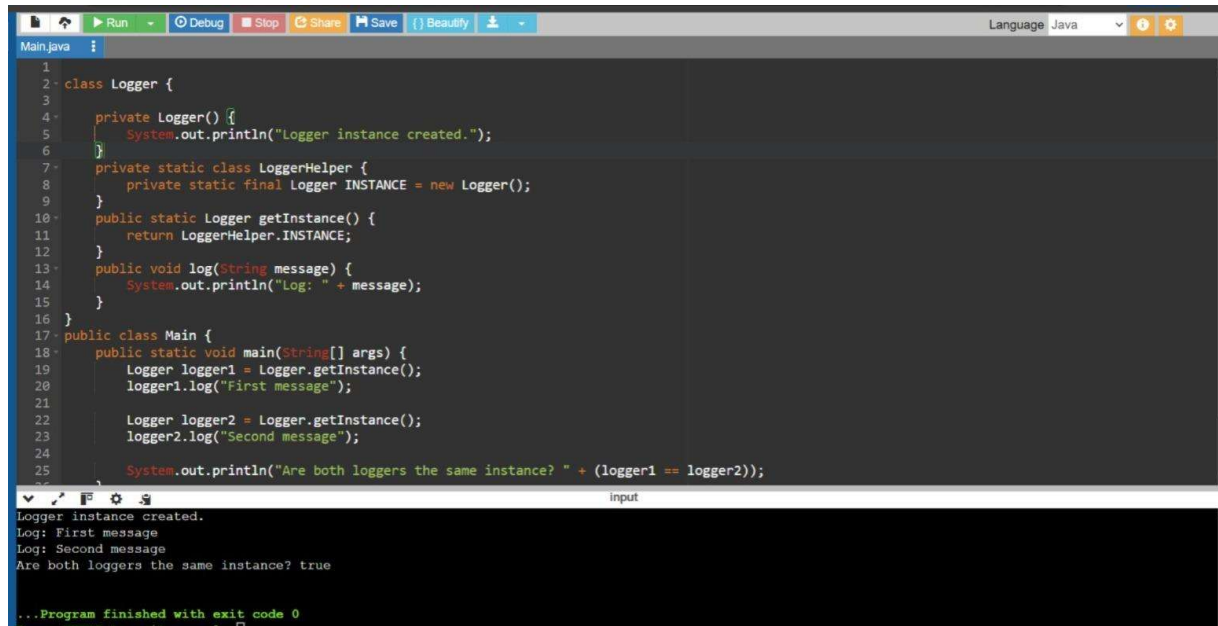
Scenario:

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 Logger() {  
        System.out.println("Logger instance created.");  
    }  
    private static class LoggerHelper {  
        private static final Logger INSTANCE = new Logger();  
    }  
    public static Logger getInstance() {  
        return LoggerHelper.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:



```
1
2 class Logger {
3
4     private Logger() {
5         System.out.println("Logger instance created.");
6     }
7
8     private static class LoggerHelper {
9         private static final Logger INSTANCE = new Logger();
10    }
11
12    public static Logger getInstance() {
13        return LoggerHelper.INSTANCE;
14    }
15
16    public void log(String message) {
17        System.out.println("Log: " + message);
18    }
19 }
20
21 public class Main {
22     public static void main(String[] args) {
23         Logger logger1 = Logger.getInstance();
24         logger1.log("First message");
25
26         Logger logger2 = Logger.getInstance();
27         logger2.log("Second message");
28
29         System.out.println("Are both loggers the same instance? " + (logger1 == logger2));
30     }
31 }

```

Logger instance created.
Log: First message
Log: Second message
Are both loggers the same instance? true
...Program finished with exit code 0

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.");  
    }  
}
```

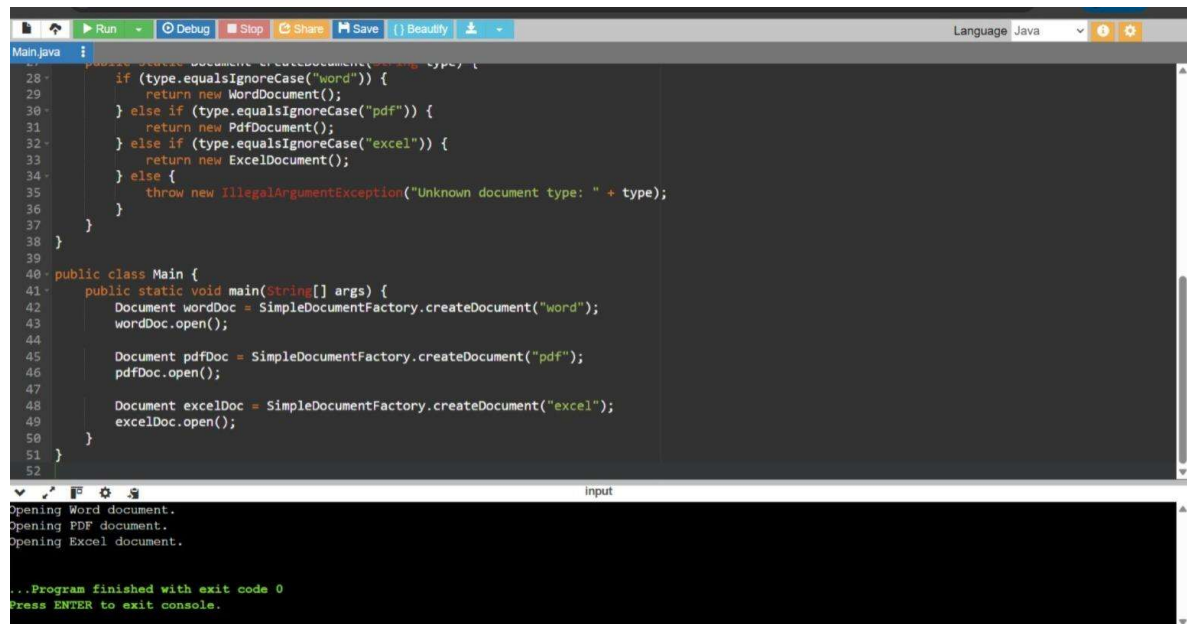
```
class ExcelDocument implements Document {  
    @Override  
    public void open() {  
        System.out.println("Opening Excel document.");  
    }  
}
```

```
class SimpleDocumentFactory {  
    public static Document createDocument(String type) {  
        if (type.equalsIgnoreCase("word")) {  
            return new WordDocument();  
        } else if (type.equalsIgnoreCase("pdf")) {  
            return new PdfDocument();  
        } else if (type.equalsIgnoreCase("excel")) {  
            return new ExcelDocument();  
        }  
    }  
}
```

```
        } else {  
            throw new IllegalArgumentException("Unknown document type: "  
+ type);  
        }  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Document wordDoc =  
SimpleDocumentFactory.createDocument("word");  
        wordDoc.open();  
  
        Document pdfDoc = SimpleDocumentFactory.createDocument("pdf");  
        pdfDoc.open();  
  
        Document excelDoc =  
SimpleDocumentFactory.createDocument("excel");  
        excelDoc.open();  
    }  
}
```

Output:



```
28- public static Document createDocument(String type) {
29-     if (type.equalsIgnoreCase("word")) {
30-         return new WordDocument();
31-     } else if (type.equalsIgnoreCase("pdf")) {
32-         return new PdfDocument();
33-     } else if (type.equalsIgnoreCase("excel")) {
34-         return new ExcelDocument();
35-     } else {
36-         throw new IllegalArgumentException("Unknown document type: " + type);
37-     }
38- }
39-
40- public class Main {
41-     public static void main(String[] args) {
42-         Document wordDoc = SimpleDocumentFactory.createDocument("word");
43-         wordDoc.open();
44-
45-         Document pdfDoc = SimpleDocumentFactory.createDocument("pdf");
46-         pdfDoc.open();
47-
48-         Document excelDoc = SimpleDocumentFactory.createDocument("excel");
49-         excelDoc.open();
50-     }
51- }
52-
input
Opening Word document.
Opening PDF document.
Opening Excel document.
...Program finished with exit code 0
Press ENTER to exit console.
```