DESIGN PATTERNS AND PRINCIPLES

MANDATORY HANDS-ON

Exercise 1: Implementing the Singleton Pattern

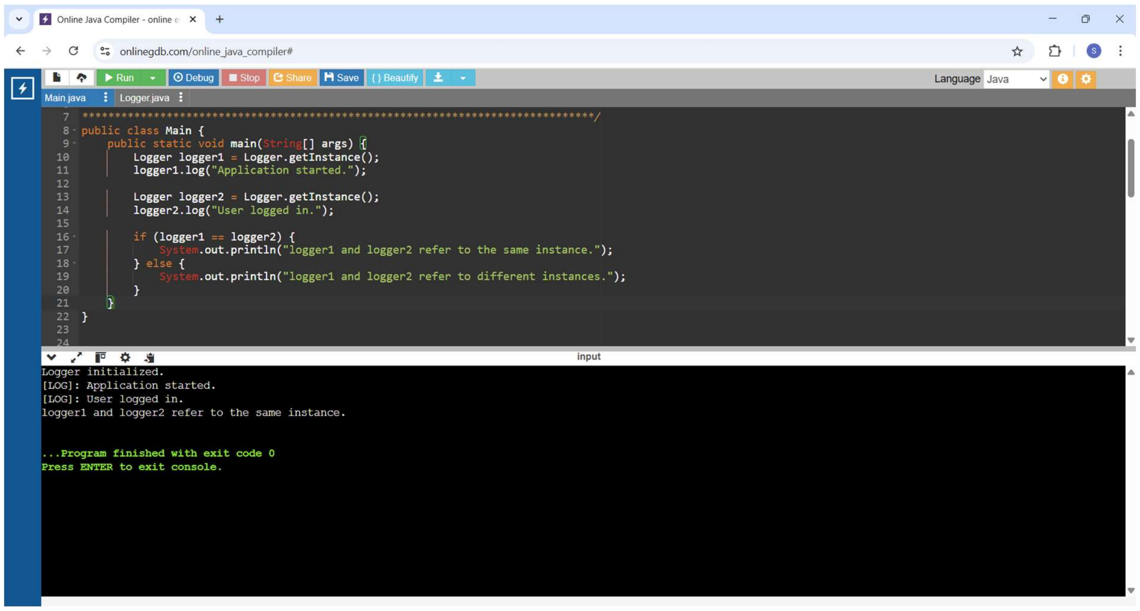
Logger.java

```
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);
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        Logger logger1 = Logger.getInstance();
        logger1.log("Application started.");
        Logger logger2 = Logger.getInstance();
        logger2.log("User logged in.");
        if (logger1 == logger2) {
            System.out.println("logger1 and logger2 refer to the same instance.");
        }
else {
            System.out.println("logger1 and logger2 refer to different instances.");
        }
    }
}
```

OUTPUT:



```
******************************************************************************/
public class Main {
    public static void main(String[] args) {
        Logger logger1 = Logger.getInstance();
        logger1.log("Application started.");

        Logger logger2 = Logger.getInstance();
        logger2.log("User logged in.");

        if (logger1 == logger2) {
            System.out.println("logger1 and logger2 refer to the same instance.");
        } else {
            System.out.println("logger1 and logger2 refer to different instances.");
        }
    }
}
```

```
Logger initialized.
[LOG]: Application started.
[LOG]: User logged in.
logger1 and logger2 refer to the same instance.


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

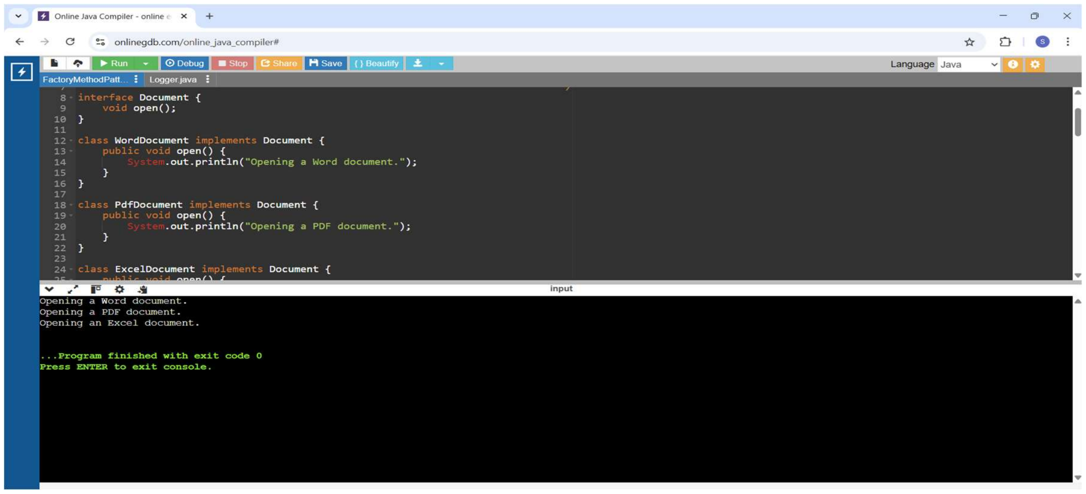
```
Logger initialized.

[LOG]: Application started.

[LOG]: User logged in.

logger1 and logger2 refer to the same instance.
```

Exercise 2: Implementing the Factory Method Pattern

FactoryPatternMethod.java

```
interface Document {
    void open();
}
class WordDocument implements Document {
    public void open() {
        System.out.println("Opening a Word document.");
    }
}
class PdfDocument implements Document {
    public void open() {
        System.out.println("Opening a PDF document.");
    }
}
class ExcelDocument implements Document {
    public void open() {
        System.out.println("Opening an Excel document.");
    }
}
abstract class DocumentFactory {
    public abstract Document createDocument();
}
class WordDocumentFactory extends DocumentFactory {
    public Document createDocument() {
        return new WordDocument();
    }
}
class PdfDocumentFactory extends DocumentFactory {
    public Document createDocument() {
        return new PdfDocument();
    }
}
class ExcelDocumentFactory extends DocumentFactory {
    public Document createDocument() {
        return new ExcelDocument();
    }
}
public class FactoryMethodPatternExample {
    public static void main(String[] args) {
        DocumentFactory wordFactory = new WordDocumentFactory();
        Document word = wordFactory.createDocument();
        word.open();
        DocumentFactory pdfFactory = new PdfDocumentFactory();
        Document pdf = pdfFactory.createDocument();
        pdf.open();
        DocumentFactory excelFactory = new ExcelDocumentFactory();
        Document excel = excelFactory.createDocument();
        excel.open();
    }
}
```

OUTPUT:



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

class WordDocument implements Document {
    public void open() {
        System.out.println("Opening a Word document.");
    }
}

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

class ExcelDocument implements Document {
```

```
Opening a Word document.
Opening a PDF document.
Opening an Excel document.


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

Opening a Word document.

Opening a PDF document.

Opening an Excel document.