

Design Patterns and Principles

Exercise 1: Implementing the Singleton Pattern

CODE:

```
J Main.java J Logger.java X
src > J Logger.java > ...
1 public class Logger {
2     // Step 1: private static instance (eager initialization)
3     private static Logger instance = new Logger();
4
5     // Step 2: private constructor
6     private Logger() {
7         System.out.println(x:"Logger instance created.");
8     }
9
10    // Step 3: public method to return the same instance
11    public static Logger getInstance() {
12        return instance;
13    }
14
15    // Logger functionality
16    public void log(String message) {
17        System.out.println("[LOG] " + message);
18    }
19 }
20
```

```
J Main.java J Logger.java X
src > J Logger.java > ...
1 public class Logger {
2     // Step 1: private static instance (eager initialization)
3     private static Logger instance = new Logger();
4
5     // Step 2: private constructor
6     private Logger() {
7         System.out.println(x:"Logger instance created.");
8     }
9
10    // Step 3: public method to return the same instance
11    public static Logger getInstance() {
12        return instance;
13    }
14
15    // Logger functionality
16    public void log(String message) {
17        System.out.println("[LOG] " + message);
18    }
19 }
20
```

Output:

```
[Running] cd "c:\Users\KIIT\Desktop\DotNet FSE\Engineering concepts\Design patterns and principles\SingletonPatternExample\src\" &&  
javac Main.java && java Main  
Logger instance created.  
[LOG] Application started.  
[LOG] Performing some operation.  
Both logger1 and logger2 are the same instance.  
  
[Done] exited with code=0 in 0.85 seconds
```

Exercise 2: Implementing the Factory Method Pattern

CODE:

```
J Main.java X
src > J Main.java > ...
1 public class Main {
    Run | Debug
2     public static void main(String[] args) {
3         // Word Document
4         DocumentFactory wordFactory = new WordDocumentFactory();
5         MyDocument wordDoc = wordFactory.createDocument();
6         wordDoc.open();
7
8         // PDF Document
9         DocumentFactory pdfFactory = new PdfDocumentFactory();
10        MyDocument pdfDoc = pdfFactory.createDocument();
11        pdfDoc.open();
12
13        // Excel Document
14        DocumentFactory excelFactory = new ExcelDocumentFactory();
15        MyDocument excelDoc = excelFactory.createDocument();
16        excelDoc.open();
17    }
18 }
19
```

```
J WordDocument.java X
src > J WordDocument.java > ...
1 public class WordDocument implements MyDocument {
2     @Override
3     public void open() {
4         System.out.println(x:"Opening a Word document...");
5     }
6 }
7
```

J PdfDocument.java X

src > J PdfDocument.java > ...

```
1 public class PdfDocument implements MyDocument {
2     @Override
3     public void open() {
4         System.out.println(x:"Opening a PDF document...");
5     }
6 }
7
```

J ExcelDocument.java X

src > J ExcelDocument.java > ...

```
1 public class ExcelDocument implements MyDocument {
2     @Override
3     public void open() {
4         System.out.println(x:"Opening an Excel document...");
5     }
6 }
7
```

J DocumentFactory.java X

src > J DocumentFactory.java > ...

```
1 public abstract class DocumentFactory {
2     public abstract MyDocument createDocument();
3 }
4
```

J PdfDocumentFactory.java X

src > J PdfDocumentFactory.java > ...

```
1 public class PdfDocumentFactory extends DocumentFactory {
2     @Override
3     public MyDocument createDocument() {
4         return new PdfDocument();
5     }
6 }
7
```

J WordDocumentFactory.java X

src > J WordDocumentFactory.java > ...

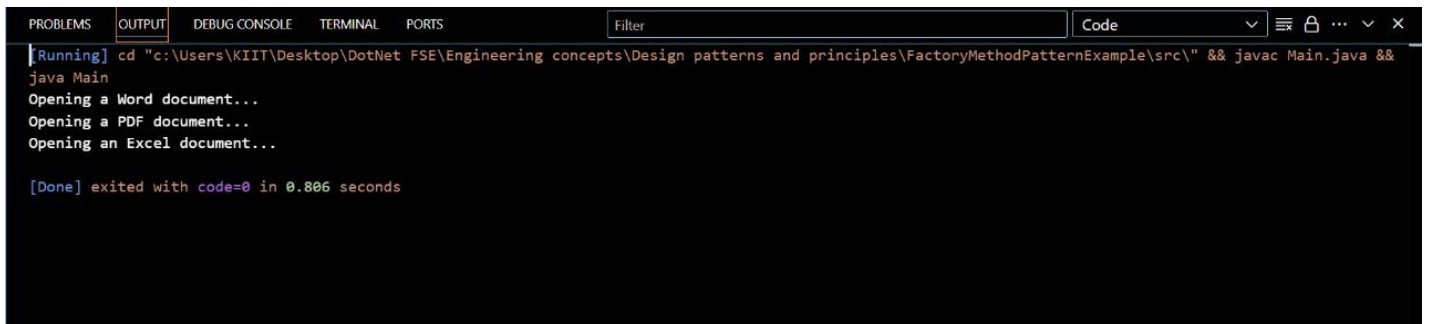
```
1 public class WordDocumentFactory extends DocumentFactory {
2     @Override
3     public MyDocument createDocument() {
4         return new WordDocument();
5     }
6 }
7
```

J ExcelDocumentFactory.java X

src > J ExcelDocumentFactory.java > ...

```
1 public class ExcelDocumentFactory extends DocumentFactory {
2     @Override
3     public MyDocument createDocument() {
4         return new ExcelDocument();
5     }
6 }
7
```

OUTPUT:



```
[Running] cd "c:\Users\KIIT\Desktop\DotNet FSE\Engineering concepts\Design patterns and principles\FactoryMethodPatternExample\src\" && javac Main.java &&
java Main
Opening a Word document...
Opening a PDF document...
Opening an Excel document...

[Done] exited with code=0 in 0.806 seconds
```