

Implementing Singleton pattern

Code: class Logger.java

```
1 package com.example.singleton;
2
3 public class Logger {
4     private static Logger instance;
5
6     private Logger() {
7         System.out.println("Logger initialized");
8     }
9
10    public static Logger getInstance() {
11        if (instance == null) {
12            instance = new Logger();
13        }
14        return instance;
15    }
16
17    public void log(String message) {
18        System.out.println(" LOG " + message);
19    }
20 }
21
```

Class Main.java

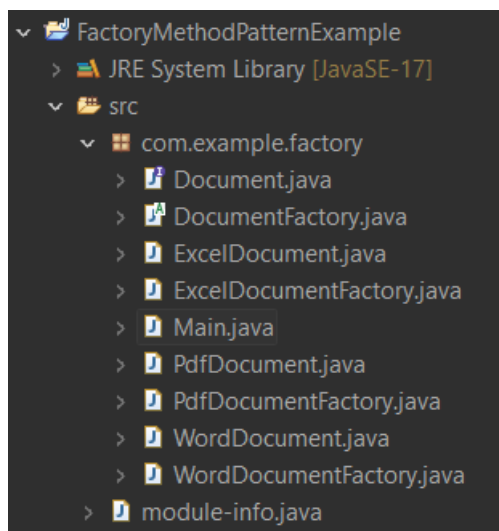
```
1 package com.example.singleton;
2
3 public class Main {
4     public static void main(String[] args) {
5         Logger logger1 = Logger.getInstance();
6         Logger logger2 = Logger.getInstance();
7
8         logger1.log("This is the first log message.");
9         logger2.log("This is the second log message.");
10
11        if (logger1 == logger2) {
12            System.out.println("Both logger1 and logger2 are the same");
13        } else {
14            System.out.println("logger1 and logger2 are different inst");
15        }
16    }
17 }
18
```

Output:

```
Problems  Javadoc  Declaration  Console  Servers  Error Log
<terminated> Main (1) [Java Application] C:\Users\sindh\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
Logger initialized
[LOG] This is the first log message.
[LOG] This is the second log message.
☒ Both logger1 and logger2 are the same instance.
```

Implementing Factory pattern

Flow:



Document and DocumentFactory.java

```
1 package com.example.factory;
2
3 public interface Document {
4     void open();
5 }
```

```
1 package com.example.factory;
2
3 public abstract class DocumentFactory {
4     public abstract Document createDocument();
5 }
```

ExcelDocument and ExcelDocumentFactory.java

```
1 package com.example.factory;
2
3 public class ExcelDocument implements Document {
4     @Override
5     public void open() {
6         System.out.println("Opening an Excel document.");
7     }
8 }
```

```
1 package com.example.factory;
2
3 public class ExcelDocumentFactory extends DocumentFactory {
4     @Override
5     public Document createDocument() {
6         return new ExcelDocument();
7     }
8 }
```

PdfDocument and PdfDocumentFactory.java

```
1 package com.example.factory;
2
3 public class PdfDocument implements Document {
4     @Override
5     public void open() {
6         System.out.println("Opening a PDF document.");
7     }
8 }
```

```
1 package com.example.factory;
2
3 public class PdfDocumentFactory extends DocumentFactory {
4     @Override
5     public Document createDocument() {
6         return new PdfDocument();
7     }
8 }
```

WordDocument and WordDocumentFactory.java

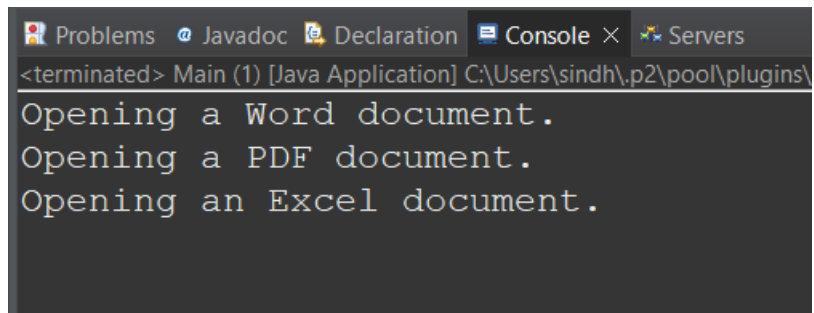
```
1 package com.example.factory;
2
3 public class WordDocument implements Document {
4     @Override
5     public void open() {
6         System.out.println("Opening a Word document.");
7     }
8 }
```

```
1 package com.example.factory;
2
3 public class WordDocumentFactory extends DocumentFactory {
4     @Override
5     public Document createDocument() {
6         return new WordDocument();
7     }
8 }
```

Main.java

```
1 package com.example.factory;
2
3 public class Main {
4     public static void main(String[] args) {
5         DocumentFactory wordFactory = new WordDocumentFactory();
6         Document wordDoc = wordFactory.createDocument();
7         wordDoc.open();
8
9         DocumentFactory pdfFactory = new PdfDocumentFactory();
10        Document pdfDoc = pdfFactory.createDocument();
11        pdfDoc.open();
12
13        DocumentFactory excelFactory = new ExcelDocumentFactory();
14        Document excelDoc = excelFactory.createDocument();
15        excelDoc.open();
16    }
17 }
```

Output:



```
Problems Javadoc Declaration Console × Servers
<terminated> Main (1) [Java Application] C:\Users\sindh\p2\pool\plugins\
Opening a Word document.
Opening a PDF document.
Opening an Excel document.
```