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.

Logger.java

```java
package singleton;

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

```java
package singleton;

public class main
{
    public static void main(String[] args)
    {
        Logger logger1 = Logger.getInstance();
        logger1.log("This is the first log message.");
        Logger logger2 = Logger.getInstance();
        logger2.log("Thus is the second log message.");
        System.out.println("Same instance?" + (logger1 == logger2));
    }
}
```

Output (VsCode Terminal):

```
\singleton> cd ..
PS C:\Users\sanga\OneDrive\Desktop\Cognizant\SingletonPattern\src> javac singleton/*.java
PS C:\Users\sanga\OneDrive\Desktop\Cognizant\SingletonPattern\src> java singleton.main
Logger Initialized
Log:This is the forst log message.
Log:Thus is the second log message.
Same instance?true
PS C:\Users\sanga\OneDrive\Desktop\Cognizant\SingletonPattern\src>
```

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.

Document.java

```java
package factory;

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

WordDocument.java

```java
package factory;

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

PdfDocument.java

```java
package factory;

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

ExcelDocument.java

```java
package factory;

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

DocumentFactory.java

```java
package factory;

public abstract class DocumentFactory
{
    public abstract Document createDocument();
}
```

WordFactory.java

```java
package factory;

public class WordFactory extends DocumentFactory
{
    public Document createDocument()
    {
        return new WordDocument();
    }
}
```

PdfFactory.java

```java
package factory;

public class PdfFactory extends DocumentFactory
{
    public Document createDocument()
    {
        return new PdfDocument();
    }
}
```

ExcelFactory.java

```java
package factory;

public class ExcelFactory extends DocumentFactory
{
    public Document createDocument()
    {
        return new ExcelDocument();
    }
}
```
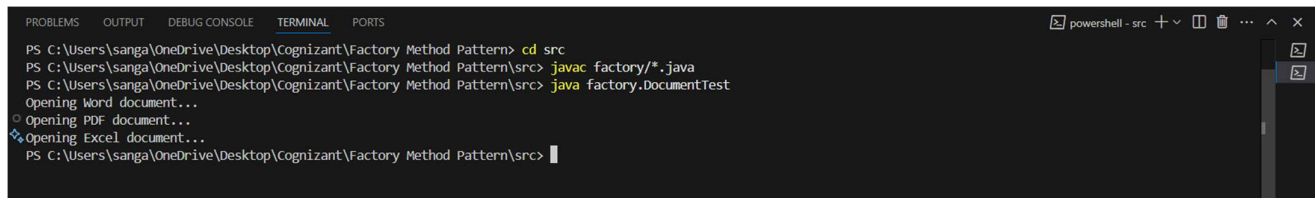
DocumentTest.java

```java
package factory;

public class DocumentTest
{
    public static void main(String[] args)
    {
        DocumentFactory wordFactory = new WordFactory();
        Document wordDoc = wordFactory.createDocument();
        wordDoc.open();

        DocumentFactory pdfFactory = new PdfFactory();
        Document pdfDoc = pdfFactory.createDocument();
        pdfDoc.open();

        DocumentFactory excelFactory = new ExcelFactory();
        Document excelDoc = excelFactory.createDocument();
        excelDoc.open();
    }
}
```

Output (VsCode Terminal):


```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\sanga\OneDrive\Desktop\Cognizant\Factory Method Pattern> cd src
PS C:\Users\sanga\OneDrive\Desktop\Cognizant\Factory Method Pattern\src> javac factory/*.java
PS C:\Users\sanga\OneDrive\Desktop\Cognizant\Factory Method Pattern\src> java factory.DocumentTest
Opening Word document...
Opening PDF document...
Opening Excel document...
PS C:\Users\sanga\OneDrive\Desktop\Cognizant\Factory Method Pattern\src>
```