# **Week-1**

**🡪Design Patterns & Principles**

**Execise-1 : Implementing Singleton Pattern**

**Code:**

1. **Defined a Singleton Class:**

**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);

}

}

1. **Below is the test class to verify that only one instance of Logger is created and used across the application.**

**Main.java:**

public class Main {

public static void main(String[] args) {

Logger logger1 = Logger.getInstance();

Logger logger2 = Logger.getInstance();

logger1.log("First log message.");

logger2.log("Second log message.");

if (logger1 == logger2) {

System.out.println("Only one instance exists.");

} else {

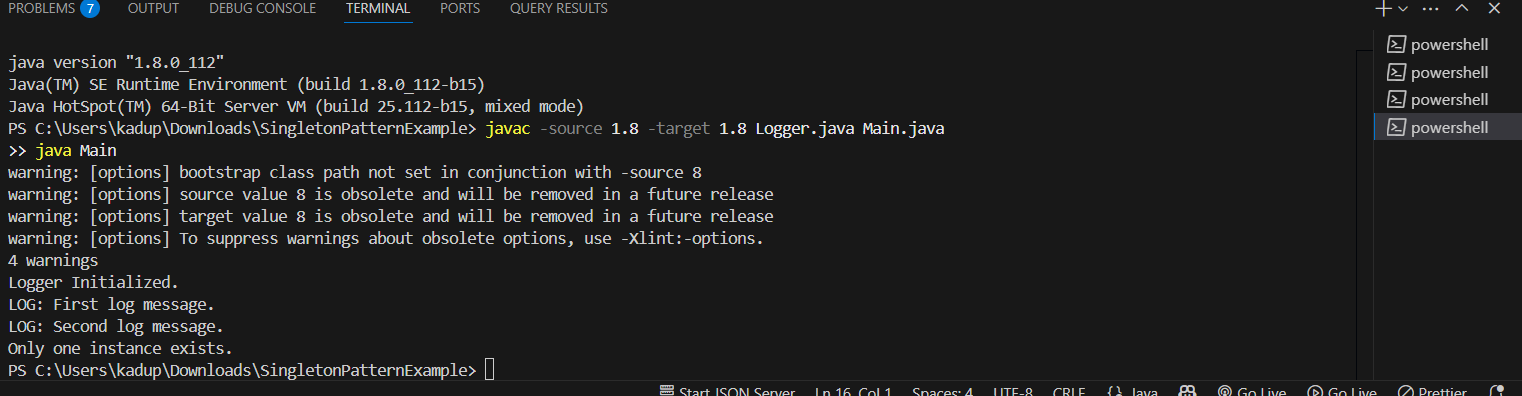
System.out.println("Different instances exist.");

}

}

}

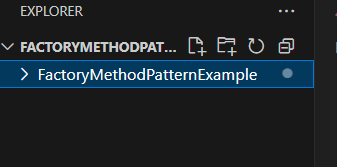
**Output:**

****

## 

**Exercise-2 : Implement Factory Method pattern**

**Step-1**: Project name: FactoryMethodPatternExample



**Step-2: Define Document Interface**

* Created a common interface named Document.
* This interface declares a method open() that will be implemented by all document types.

public interface Document {

void open();

}

**Step 3: Create Concrete Document Classes**

Implement three different document types:

1)WordDocument

public class WordDocument implements Document {

public void open() {

System.out.println("Opening Word Document...");

}

}

2) PDF Document

public class PdfDocument implements Document {

public void open() {

System.out.println("Opening PDF Document...");

}

}

3) Excel Document

public class PdfDocument implements Document {

public void open() {

System.out.println("Opening PDF Document...");

}

}

**Step 4: Implement the Factory Method**

Created an abstract class DocumentFactory with an abstract method createDocument().

public abstract class DocumentFactory {

public abstract Document createDocument();

}

Defined three concrete factory classes:

1)WordFactory:

public class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}

2)PdfFactory:

public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

3)ExcelFactory:

public class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}

**Step 5: Test the Factory Method Implementation**

Created a Main class to test how different document types can be created using factory classes.

This step demonstrates polymorphism and encapsulation of object creation logic.

**Code:**

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document word = wordFactory.createDocument();

word.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdf = pdfFactory.createDocument();

pdf.open();

DocumentFactory excelFactory = new ExcelFactory();

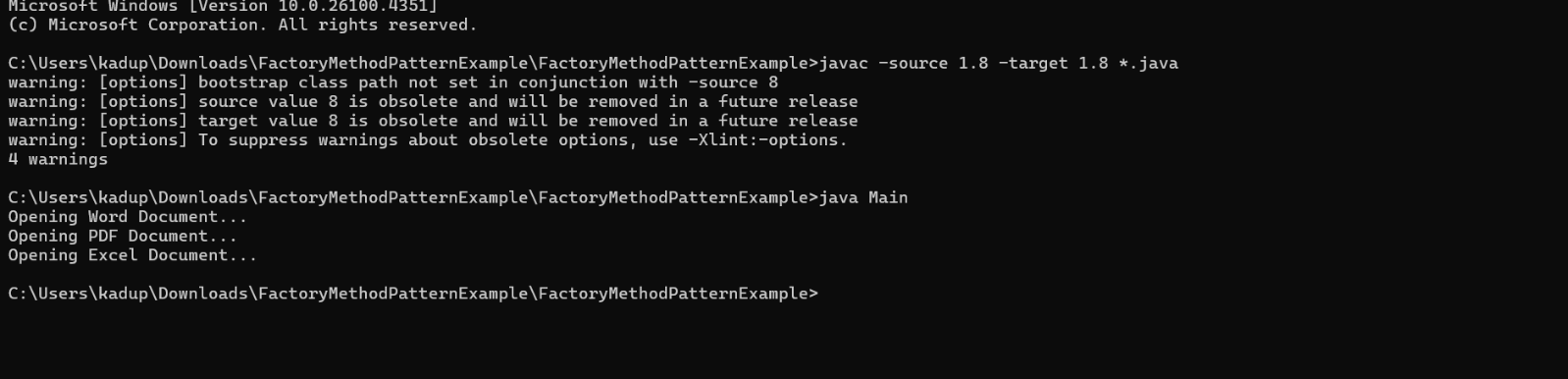
Document excel = excelFactory.createDocument();

excel.open();

}

}

**Output:**

****