**Week - 1**

**DesignPrinciples\_and\_Patterns**

**1. Singleton Pattern Implementation in Java-Logger Example**

**Project Name** : SingletonPatternExample

**Exercise**  : Implementing the Singleton Pattern

**Name**  : Alladi Manasa

**Superset Id** : 6373907

**Introduction:**

Singleton is a creational design pattern that ensures a class has only one instance and provides a global point of access to it. It’s commonly used in logging, drivers, configuration classes, etc.

**Objective:**

To implement the Singleton Design Pattern in Java to ensure only one instance of the Logger utility class is created and used throughout the application lifecycle.

**Tools Used:**

* Java
* Eclipse IDE

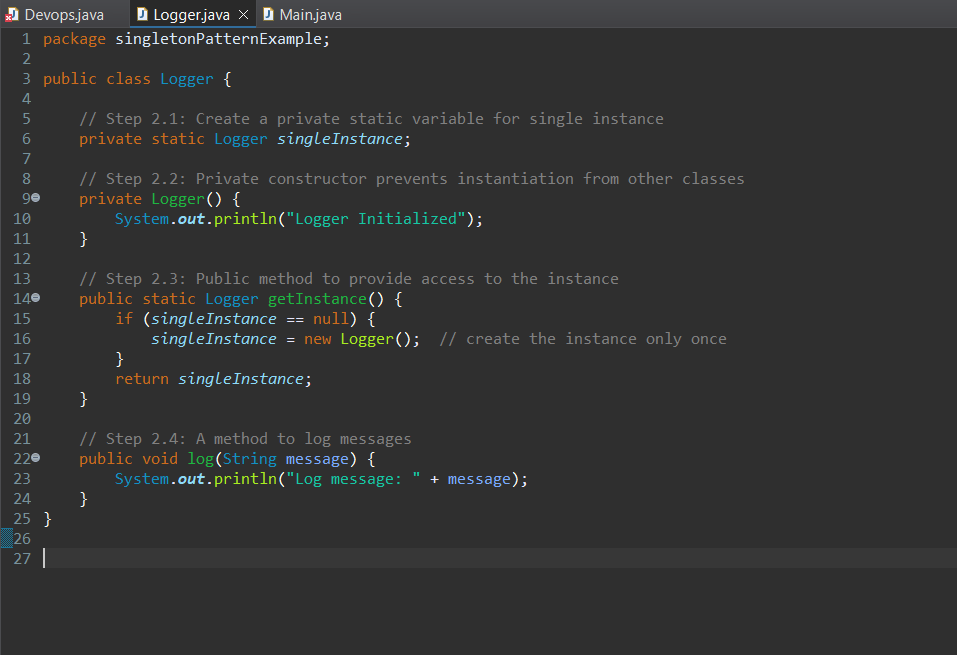
**Steps Followed:**

**Step 1:** Create a New Java Project

* Created a Java Project named SingletonPatternExample in Eclipse

**Step 2:** Created the Logger Class

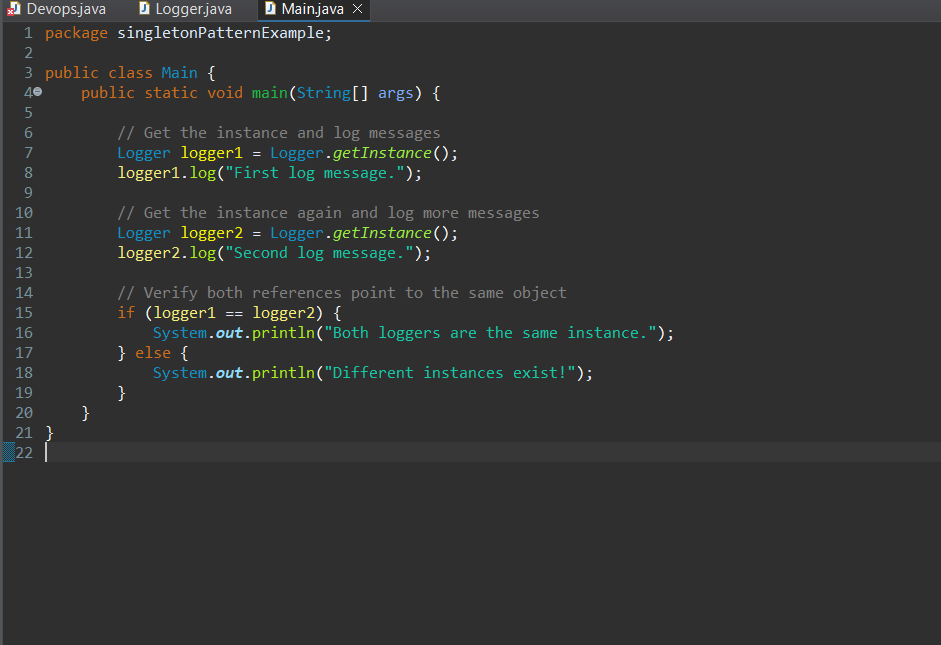
* Defined a private static variable to hold the instance.
* Made constructor private to prevent external instantiation.
* Provided a public static method getInstance() to return the single instance.
* Added a log(String message) method for demonstration.

**Logger.java:**

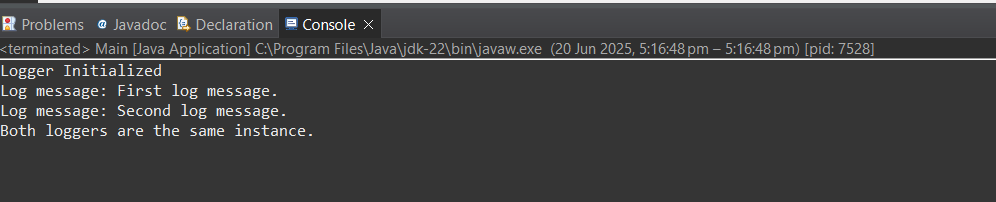
**Step 3:** Created Main Class to Test

* Retrieved Logger instance twice.
* Verified both references point to the same object.

**Main.java :**



**Output:**



**Conclusion:**

The Singleton pattern ensures only one instance of the Logger class is used throughout the application. This implementation helps maintain consistent logging and resource control.

**2. Factory Method Pattern Implementation in Java – Document Management System**

**Project Name** : FactoryMethodPatternExample

**Exercise**  : Implementing the Factory Method Pattern

**Name**  : Alladi Manasa

**Superset Id** : 6373907

**Introduction :**

The Factory Method Pattern is a creational design pattern that provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created.

In this project, the pattern is used to:

* Avoid directly instantiating document types
* Use factories to generate the appropriate document object

**Objective :**

To implement the Factory Method Design Pattern in Java to create different types of documents (Word, PDF, Excel) using factory classes.

**Tools Used :**

* Java
* Eclipse IDE

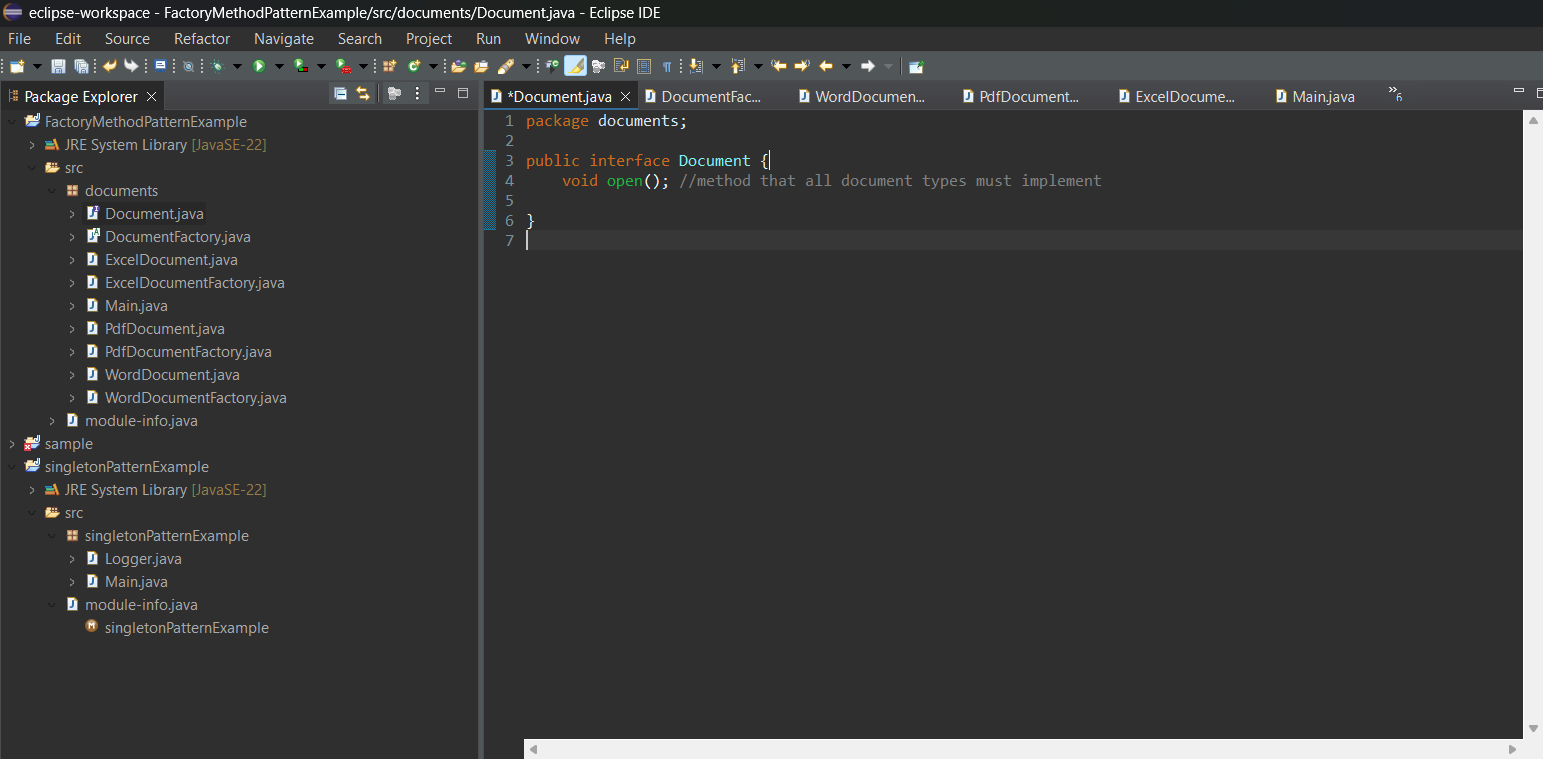
**Steps Followed:**

**Step 1:** Created Java Project

Created a project named FactoryMethodPatternExample in Eclipse.

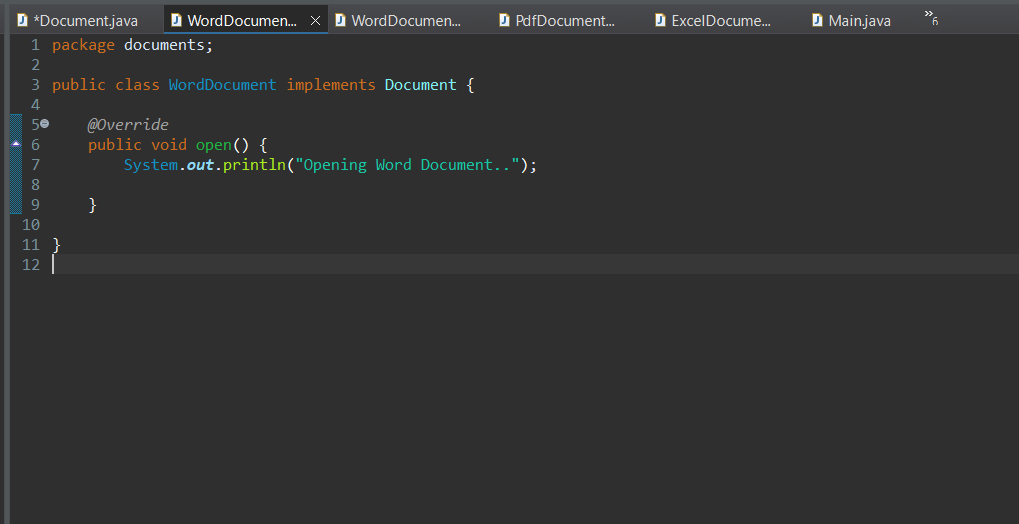
**Step 2 :** Defined the Document Interface

**Document.java**

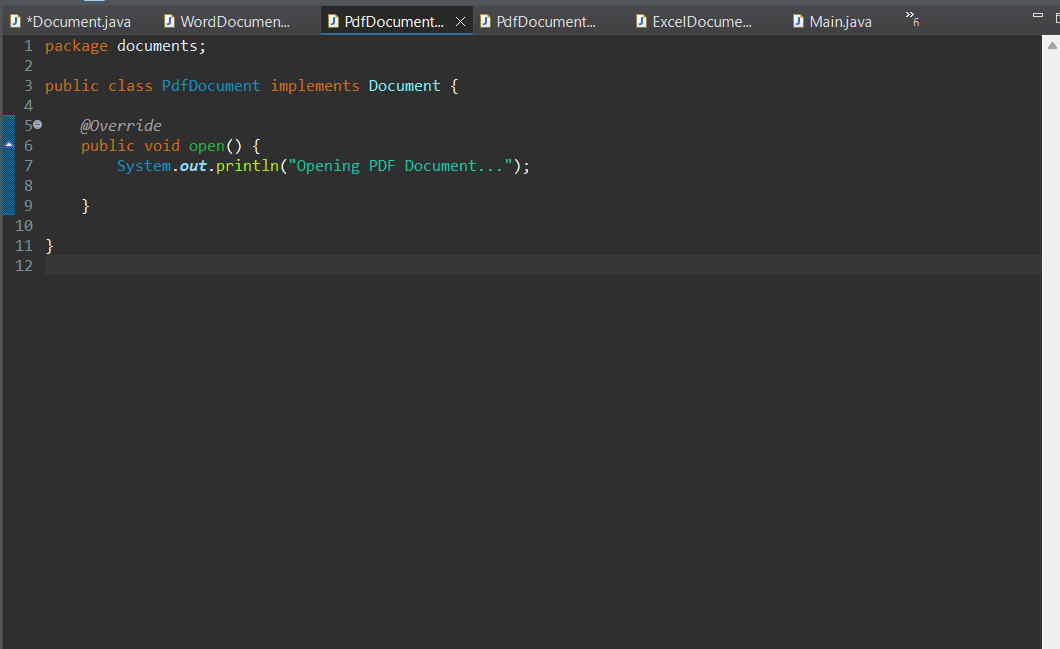


**Step 3 :** Created Concrete Document Classes

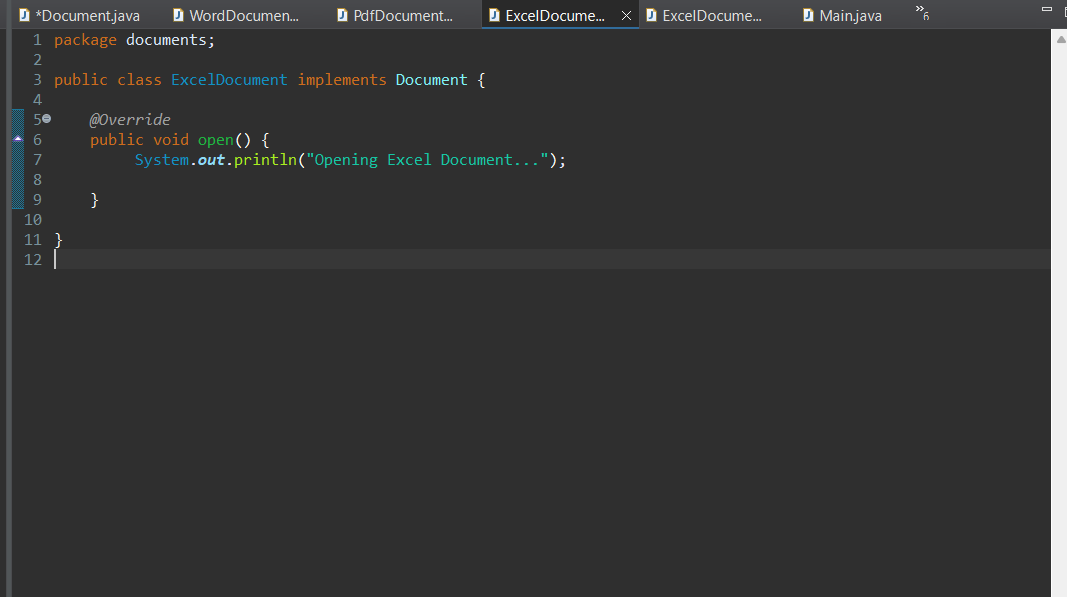
**WordDocument.java**



**PdfDocument.java**

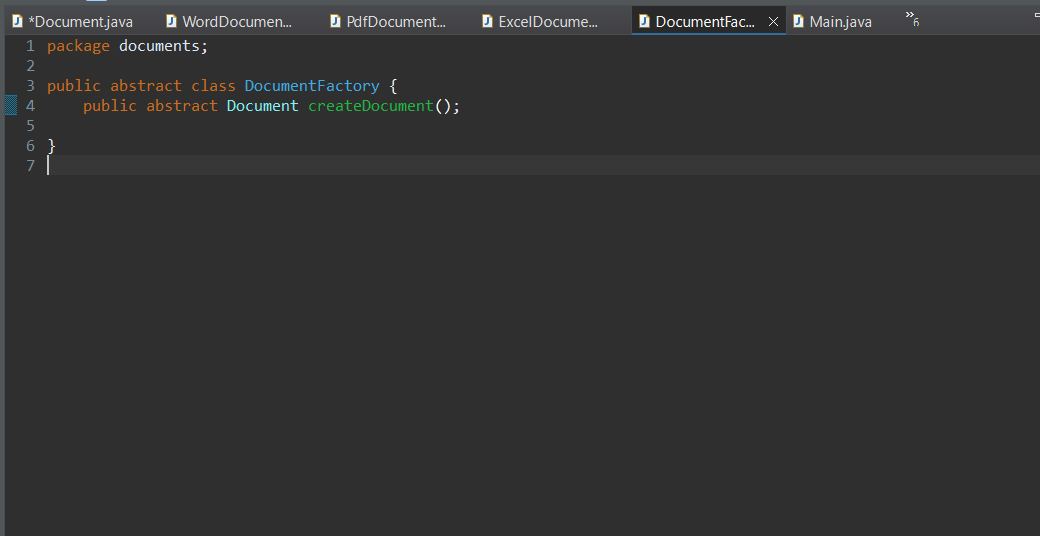


**ExcelDocument.java**

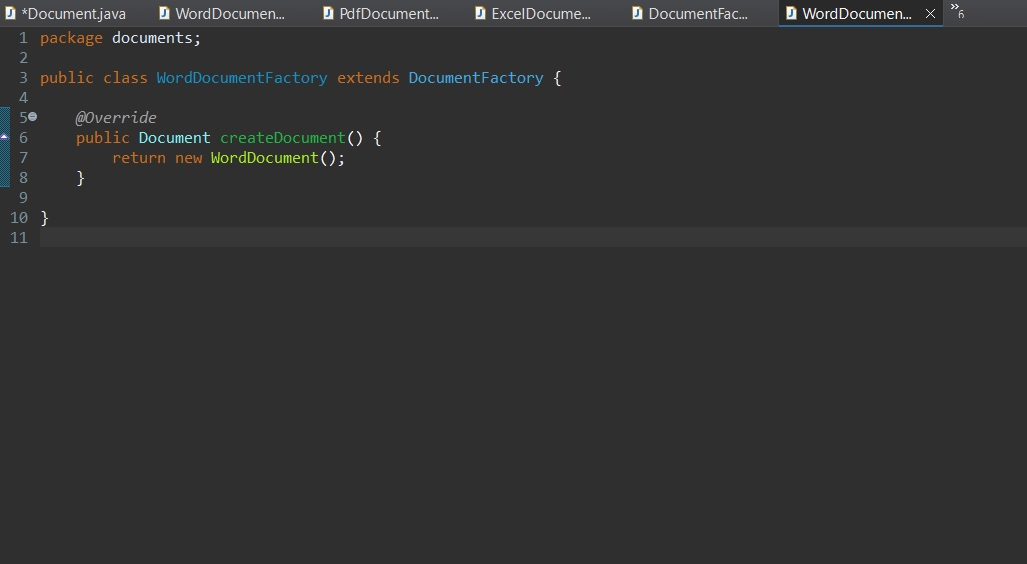


**Step 4 :** Created the Factory Classes

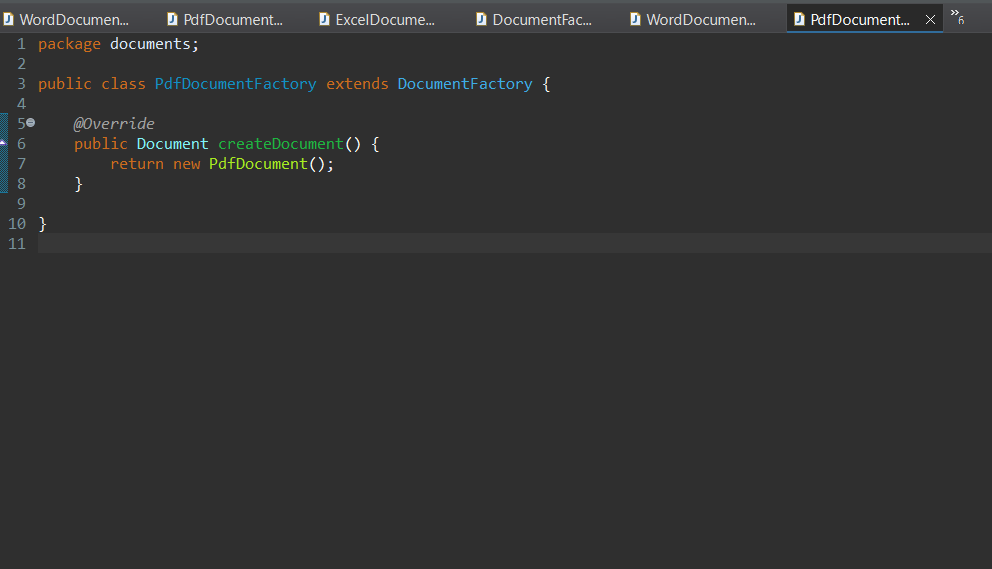
**DocumentFactory.java (Abstract Class)**



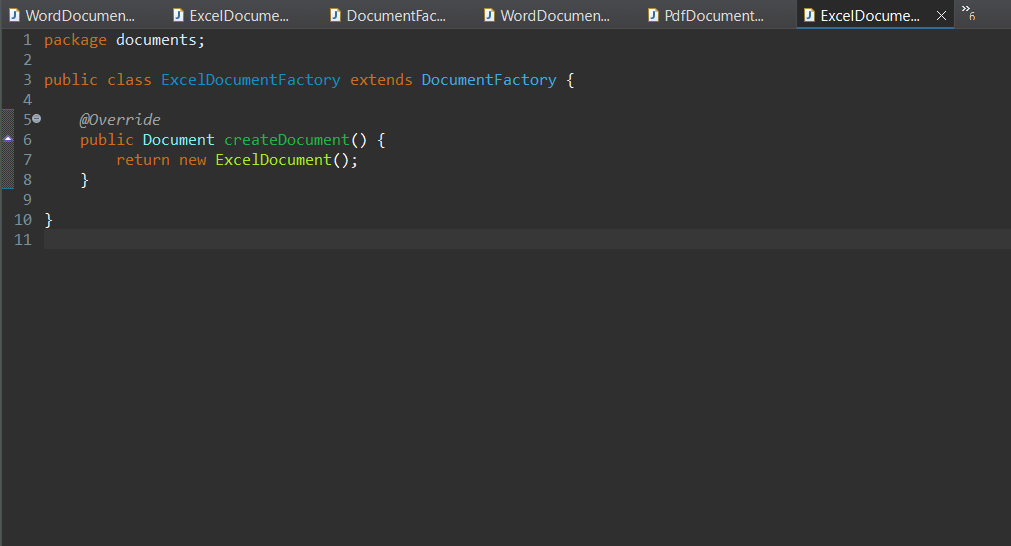
**WordDocumentFactory.java**



**PdfDocumentFactory.java**

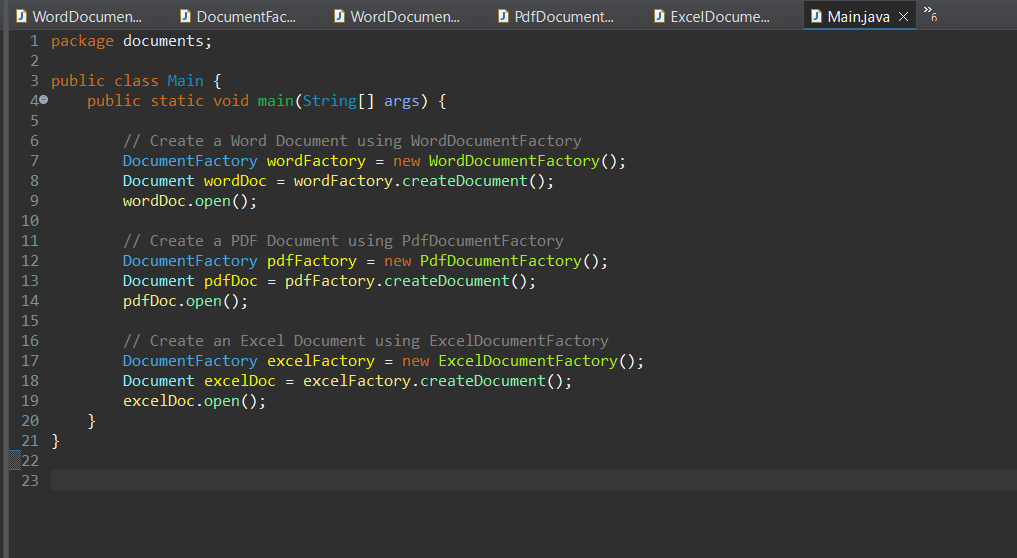


**ExcelDocumentFactory.java**

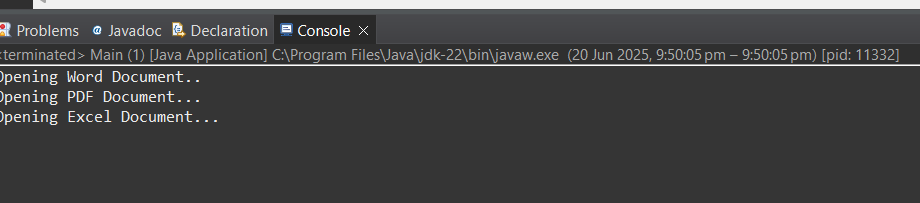


**Step 5 :** Tested the Implementation

**Main.java**



**Output :**



**Conclusion :**

This project demonstrated the implementation of the Factory Method Pattern in Java. It separates the document creation logic into dedicated factory classes, which improves modularity, scalability, and maintainability.