**WEEK-01**

**HANDS ON EXERCISES SOLUTIONS**

Design patterns and principles

**2]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.

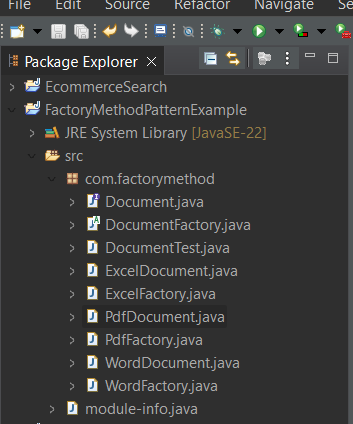
Steps:

1. Create a New Java Project:
   * Create a new Java project named FactoryMethodPatternExample.
2. Define Document Classes:
   * Create interfaces or abstract classes for different document types such as WordDocument, PdfDocument, and ExcelDocument.
3. Create Concrete Document Classes:
   * Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.
4. Implement the Factory Method:
   * Create an abstract class DocumentFactory with a method createDocument().
   * Create concrete factory classes for each document type that extends DocumentFactory and implements the createDocument() method.
5. Test the Factory Method Implementation:
   * Create a test class to demonstrate the creation of different document types using the factory method.

SOLUTION:

IMPLEMENTATION:

here is the entire file structure for the factory pattern example program.



This project demonstrates a document management system built using the Factory Method Design Pattern.

The system is designed to create different types of documents such as Word, PDF, and Excel, keeping the creation logic separate from the main application logic.

A common interface named Document was introduced to define a standard open() method.

Specific document types like WordDocument, PdfDocument, and ExcelDocument implement this interface, each providing their own version of the method.

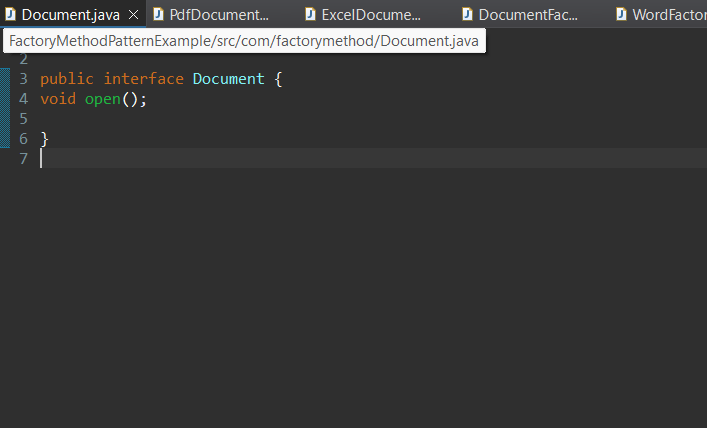
To manage object creation, an abstract factory class DocumentFactory was created with an abstract method createDocument().

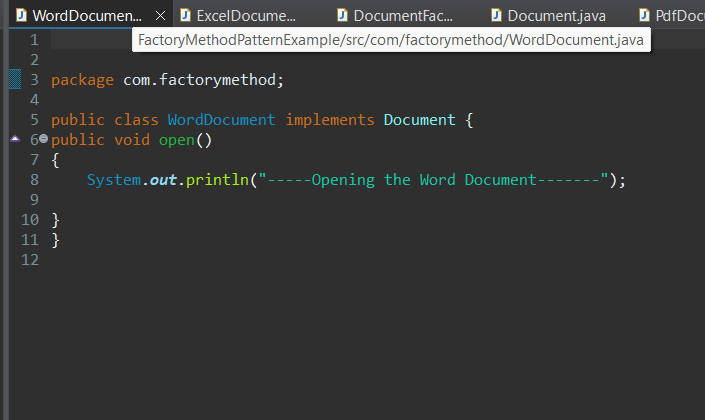
This method was then implemented in concrete factory classes like WordFactory, PdfFactory, and ExcelFactory, each responsible for returning a specific document instance.

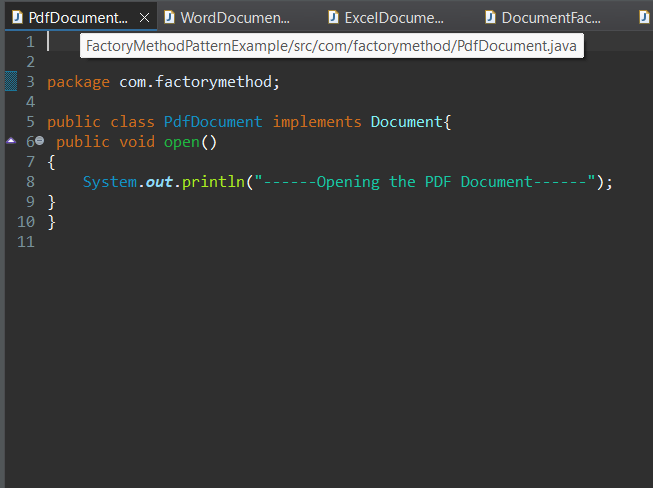
A separate test class, DocumentTest, was used to demonstrate how different document types can be created and used through their respective factories.

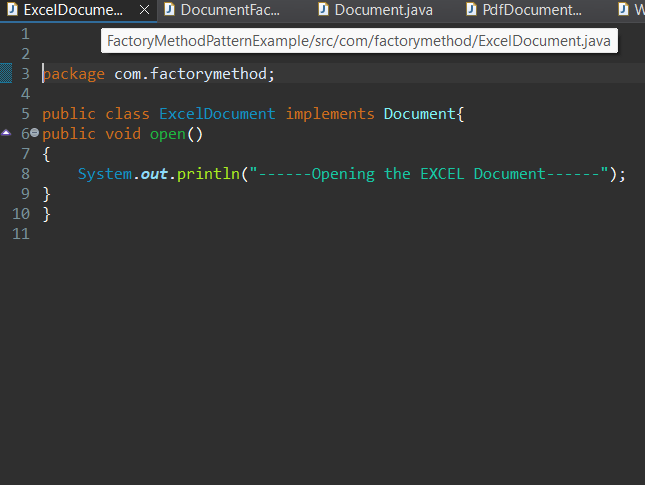
This approach promotes loose coupling, makes the system easier to extend, and adheres to object-oriented best practices by separating object creation from usage.

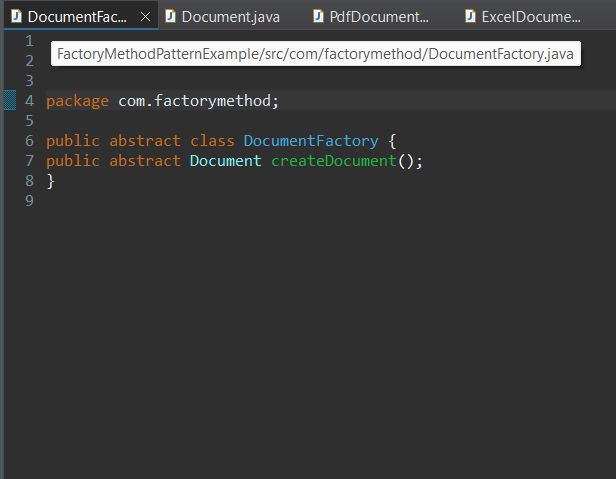
Overall, the project effectively implements the Factory Method Pattern by using abstract factories to create document objects without exposing the instantiation logic to the client.

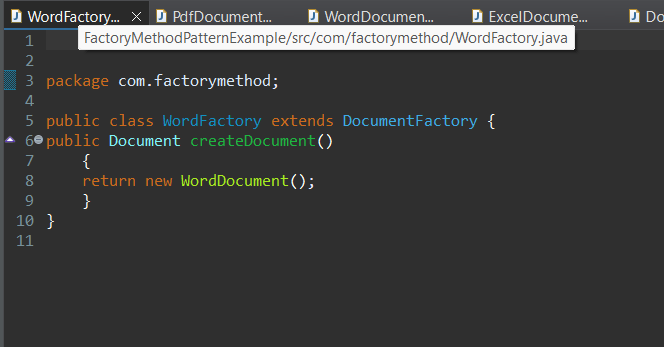


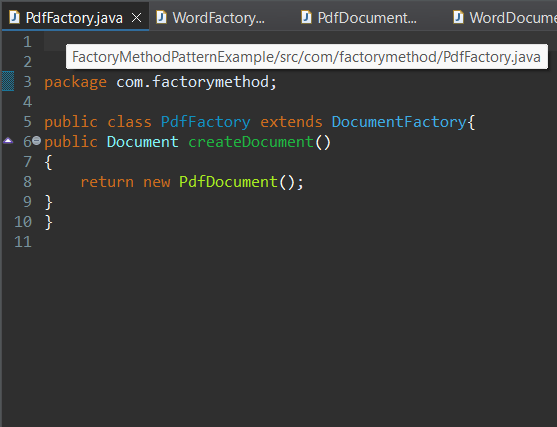


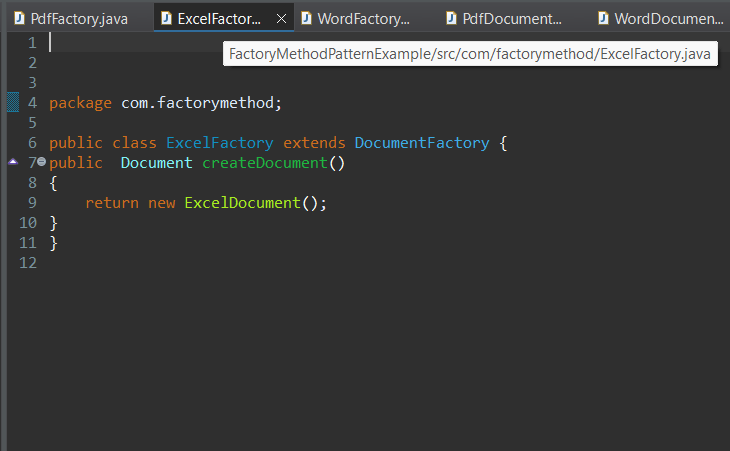


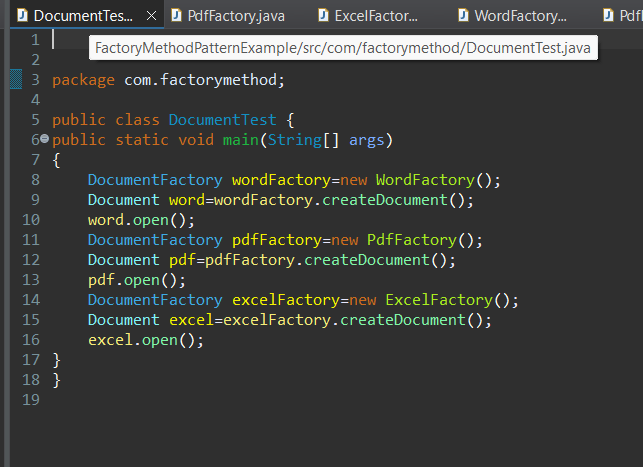


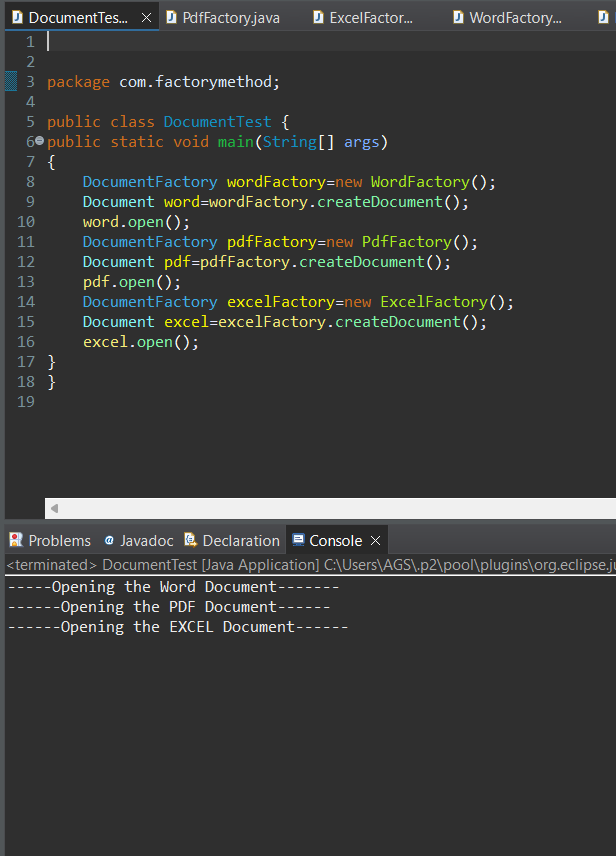












OUTPUT:

