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:

o Create a new Java project named FactoryMethodPatternExample.

2. Define Document Classes:

o Create interfaces or abstract classes for different document types such as WordDocument, PdfDocument, and ExcelDocument.

3. Create Concrete Document Classes:

o Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.

4. Implement the Factory Method:

o Create an abstract class DocumentFactory with a method createDocument().

o Create concrete factory classes for each document type that extends DocumentFactory and implements the createDocument() method.

5. Test the Factory Method Implementation:

o Create a test class to demonstrate the creation of different document types using the factory method.

**CODE SECTION**

**Document.java**  
public interface Document {

void open();

}  
  
**WordDocument.java**  
public class WordDocument implements Document {

public void open() {

System.out.println("Opening a Word document.");

}

}

**PdfDocument.java**  
public class PdfDocument implements Document {

public void open() {

System.out.println("Opening a PDF document.");

}

}  
**ExcelDocument.java**

public class ExcelDocument implements Document {

public void open() {

System.out.println("Opening an Excel document.");

}

}  
**DocumentFactory.java**  
public abstract class DocumentFactory {

public abstract Document createDocument();

}  
  
**WordFactory.java**  
public class WordFactory extends DocumentFactory {

public Document createDocument() {

return new WordDocument();

}

}  
  
**PdfFactory.java**public class PdfFactory extends DocumentFactory {

public Document createDocument() {

return new PdfDocument();

}

}

**ExcelFactory.java**public class ExcelFactory extends DocumentFactory {

public Document createDocument() {

return new ExcelDocument();

}

}  
**FactoryTest.java**

public class FactoryTest {

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();

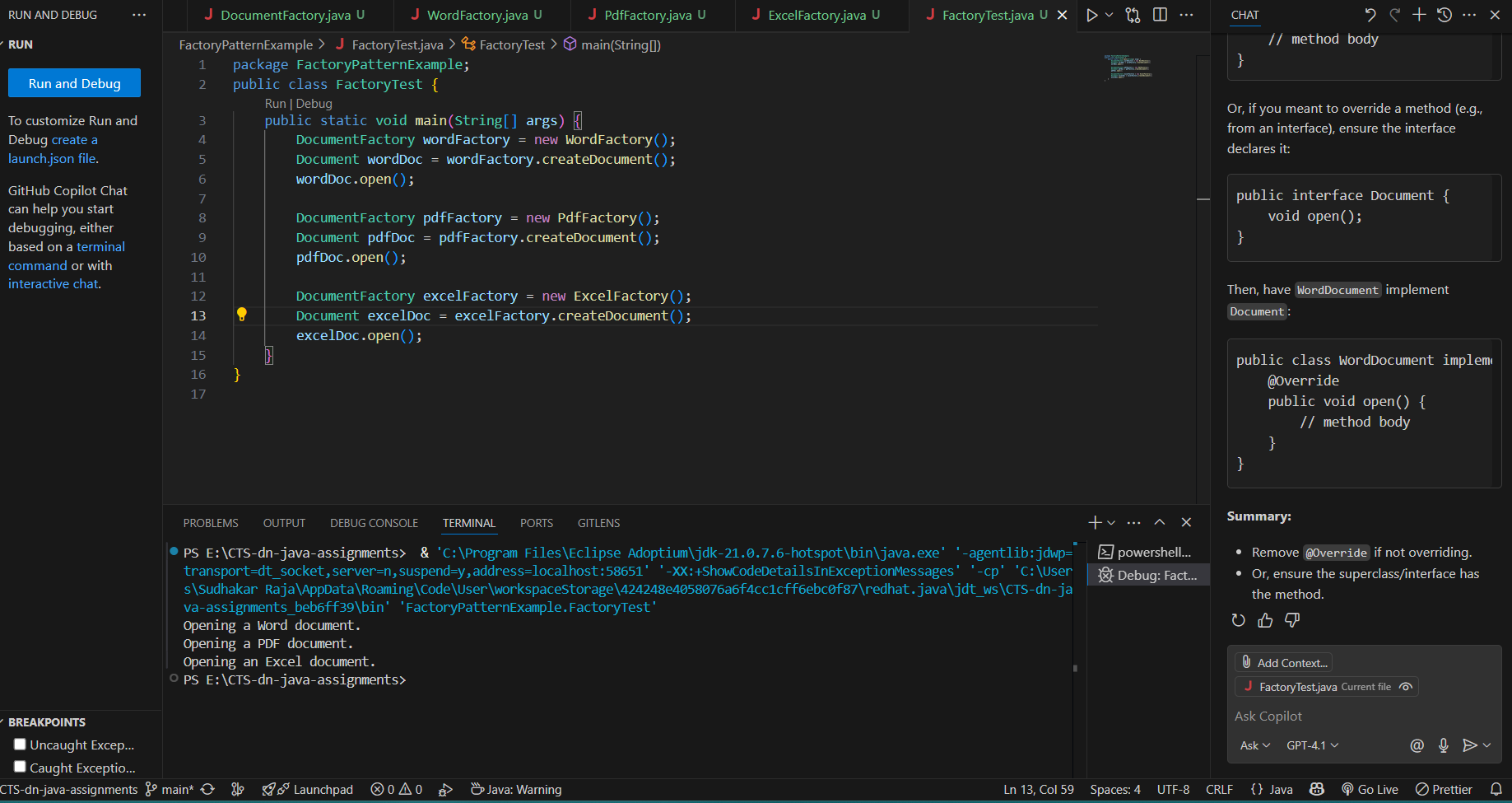
}

}  
  
**OUTPUT:**

Opening a Word document.

Opening a PDF document.

Opening an Excel document.

**OUTPUT SCREENSHOT:**   
 **Conclusion:**

The Factory Method Pattern was successfully implemented to create different types of documents (Word, PDF, Excel) in a flexible and scalable manner. By defining a common interface and using dedicated factory classes for each document type, the code achieves loose coupling and better maintainability. This approach also follows the Open/Closed Principle, allowing us to add new document types without modifying existing code. Overall, this hands-on demonstrates the effectiveness of the Factory Method design pattern in building extensible software.