

```
interface Document {  
    void open();  
}  
  
class WordDocument implements Document {  
    @Override  
    public void open() {  
        System.out.println("Opening a Word document...");  
    }  
}  
  
class PdfDocument implements Document {  
    @Override  
    public void open() {  
        System.out.println("Opening a PDF document...");  
    }  
}  
  
class ExcelDocument implements Document {  
    @Override  
    public void open() {  
        System.out.println("Opening an Excel document...");  
    }  
}  
  
abstract class DocumentFactory {  
    public abstract Document createDocument();  
}  
  
class WordDocumentFactory extends DocumentFactory {  
    @Override
```

```

    public Document createDocument() {
        return new WordDocument();
    }
}

class PdfDocumentFactory extends DocumentFactory {
    @Override
    public Document createDocument() {
        return new PdfDocument();
    }
}

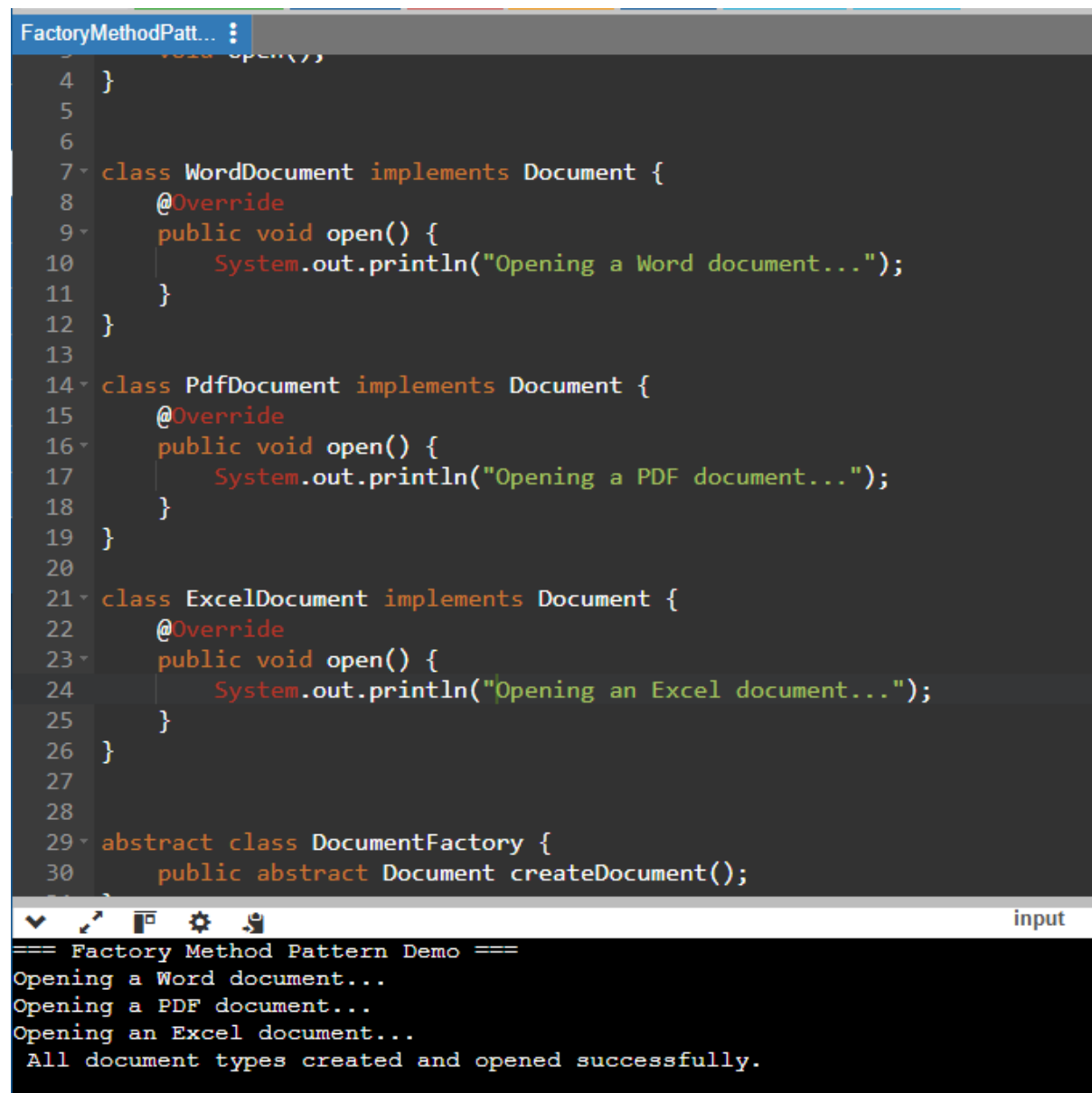
class ExcelDocumentFactory extends DocumentFactory {
    @Override
    public Document createDocument() {
        return new ExcelDocument();
    }
}

public class FactoryMethodPatternExample {
    public static void main(String[] args) {
        System.out.println("=== Factory Method Pattern Demo ===");
        DocumentFactory wordFactory = new WordDocumentFactory();
        Document wordDoc = wordFactory.createDocument();
        wordDoc.open();
        DocumentFactory pdfFactory = new PdfDocumentFactory();
        Document pdfDoc = pdfFactory.createDocument();
        pdfDoc.open();
        DocumentFactory excelFactory = new ExcelDocumentFactory();
        Document excelDoc = excelFactory.createDocument();
        excelDoc.open();
    }
}

```

```
        System.out.println(" All document types created and opened successfully.");
    }
}
```

Output:



The screenshot shows an IDE window titled "FactoryMethodPatternDemo.java". The code defines an abstract class `Document` with an `open()` method. Three concrete classes, `WordDocument`, `PdfDocument`, and `ExcelDocument`, implement this method by printing specific messages. An abstract class `DocumentFactory` defines a `createDocument()` method. The main method in the class calls `createDocument()` three times, creating and opening a Word document, a PDF document, and an Excel document. The output window at the bottom shows the execution results: "Opening a Word document...", "Opening a PDF document...", "Opening an Excel document...", and "All document types created and opened successfully."

```
4 }
5
6
7 class WordDocument implements Document {
8     @Override
9     public void open() {
10         System.out.println("Opening a Word document...");
11     }
12 }
13
14 class PdfDocument implements Document {
15     @Override
16     public void open() {
17         System.out.println("Opening a PDF document...");
18     }
19 }
20
21 class ExcelDocument implements Document {
22     @Override
23     public void open() {
24         System.out.println("Opening an Excel document...");
25     }
26 }
27
28
29 abstract class DocumentFactory {
30     public abstract Document createDocument();
31 }
32
33 public class FactoryMethodPatternDemo {
34     public static void main(String[] args) {
35         DocumentFactory factory = new DocumentFactory();
36         Document document = factory.createDocument();
37         document.open();
38         document = factory.createDocument();
39         document.open();
40         document = factory.createDocument();
41         document.open();
42         System.out.println(" All document types created and opened successfully.");
43     }
44 }
```

=== Factory Method Pattern Demo ===
Opening a Word document...
Opening a PDF document...
Opening an Excel document...
All document types created and opened successfully.

