Scenario 1: Customer Request

Customer Request:

"I need a document editor that can handle Word and PDF documents. The editor should allow me to open, save, and close these documents, but I also want to add other formats like Text and Markdown in the future. The document operations should be handled smoothly, and I don't want to worry about the internal implementation details."

Choosing the Right Pattern:

Pattern Recommendation: Factory Method Pattern

Why?

The Factory Method pattern is ideal in this situation because:

- Scalability: The customer wants to support multiple document formats, with the possibility of adding more in the future (Text, Markdown, etc.). The Factory Method allows you to easily add new document types without altering existing code.

- Decoupling: The customer doesn't want to worry about the internal implementation of how documents are handled. The Factory Method pattern decouples the creation of specific document types (like WordDocument or PDFDocument) from the client (the document editor), making it easier to manage and extend.

- Modularity: Each document type (Word, PDF) is handled by its own creator class. This modularity makes the code easier to maintain and extend as new document types are added.

Pattern Explanation: Factory Method Pattern

Key Concepts:

- Product Interface: The interface (Document) defines common operations like open(), save(), and close().

- Concrete Products: Classes like WordDocument and PDFDocument implement the Document interface and provide specific behavior for Word and PDF formats.

- Creator Class: The DocumentCreator class defines the factory method createDocument(), which subclasses implement to create specific document types.

- Concrete Creators: WordDocumentCreator and PDFDocumentCreator are subclasses that implement the factory method to return instances of WordDocument and PDFDocument.

How It Works:

- Client Code: The document editor uses a DocumentCreator to create a document. The client code doesn't need to know or care about the specific document type.

- Factory Method: The factory method createDocument() is implemented in subclasses (WordDocumentCreator, PDFDocumentCreator) to create and return specific document types.

- Extensibility: When the customer later asks for support for a new format (e.g., Markdown), you simply create a new MarkdownDocumentCreator without changing the existing code.

Implementation Example:

Here's how the implementation might look in Java:

```
// Product Interface
interface Document {
    void open();
```

```
    void save();
    void close();
}

// Concrete Products
class WordDocument implements Document {
    @Override
    public void open() {
        System.out.println("Opening Word document.");
    }
    @Override
    public void save() {
        System.out.println("Saving Word document.");
    }
    @Override
    public void close() {
        System.out.println("Closing Word document.");
    }
}

class PDFDocument implements Document {
    @Override
    public void open() {
        System.out.println("Opening PDF document.");
    }
    @Override
    public void save() {
```

```
        System.out.println("Saving PDF document.");
    }
    @Override
    public void close() {
        System.out.println("Closing PDF document.");
    }
}

// Creator Class
abstract class DocumentCreator {
    abstract Document createDocument();
    public void performOperations() {
        Document doc = createDocument();
        doc.open();
        doc.save();
        doc.close();
    }
}

// Concrete Creators
class WordDocumentCreator extends DocumentCreator {
    @Override
    Document createDocument() {
        return new WordDocument();
    }
}
```

```
class PDFDocumentCreator extends DocumentCreator {
    @Override
    Document createDocument() {
        return new PDFDocument();
    }
}

// Client Code
public class DocumentEditor {
    public static void main(String[] args) {
        DocumentCreator wordCreator = new WordDocumentCreator();
        wordCreator.performOperations();

        DocumentCreator pdfCreator = new PDFDocumentCreator();
        pdfCreator.performOperations();
    }
}
```

Key Points to Remember:

- Factory Method allows the customer to add new document formats without modifying the existing codebase.
- It encapsulates the object creation process, making it easier to manage and extend.