Scenario 18: Bridge Pattern

Customer Request:

I need a document editor that supports different formatting styles, such as applying headers, footers, and sidebars. However, I want to ensure that the implementation of these styles can vary independently of the document types (like Word, PDF, Text). The document editor should be able to combine any formatting style with any document type without tightly coupling them.

Choosing the Right Pattern:

Pattern Recommendation: Bridge Pattern

Why?

The Bridge Pattern is ideal for this scenario because:

1. Decoupling Abstraction from Implementation: The customer needs to decouple the document types (Word, PDF, Text) from the formatting styles (headers, footers, sidebars) so that they can vary independently. The Bridge Pattern separates the abstraction (document type) from its implementation (formatting styles).

2. Flexibility: The pattern allows you to combine any document type with any formatting style, providing maximum flexibility in the system.

3. Scalability: The Bridge Pattern makes it easy to add new document types or formatting styles without affecting existing code, promoting scalability and maintainability.

Pattern Explanation: Bridge Pattern

The Bridge Pattern decouples an abstraction from its implementation so that the two can vary independently. This pattern involves four key components:

1. **Abstraction**: Defines the abstract interface for the document types (e.g., Document). It maintains a reference to an implementer (e.g., DocumentFormatter).

2. **Refined Abstraction**: Extends the abstraction to include more specific document types (e.g., WordDocument, PDFDocument).

3. **Implementor**: Defines the interface for the formatting styles (e.g., DocumentFormatter).

4. **Concrete Implementor**: Implements the implementer interface to provide specific formatting styles (e.g., HeaderFormatter, FooterFormatter).

Implementation Example

```
// Implementor Interface
interface DocumentFormatter {
    void applyHeader();
    void applyFooter();
}

// Concrete Implementors
class HeaderFormatter implements DocumentFormatter {
    @Override
    public void applyHeader() {
        System.out.println("Applying header.");
    }

    @Override
    public void applyFooter() {
        System.out.println("Applying footer.");
    }
}

class SidebarFormatter implements DocumentFormatter {
```

```
    @Override
    public void applyHeader() {
        System.out.println("Applying header with sidebar.");
    }

    @Override
    public void applyFooter() {
        System.out.println("Applying footer with sidebar.");
    }
}

// Abstraction Interface
abstract class Document {
    protected DocumentFormatter formatter;

    public Document(DocumentFormatter formatter) {
        this.formatter = formatter;
    }

    public abstract void open();
    public abstract void save();
    public abstract void close();
}

// Refined Abstractions
class WordDocument extends Document {
    public WordDocument(DocumentFormatter formatter) {
```

```
        super(formatter);
    }

    @Override
    public void open() {
        System.out.println("Opening Word document.");
        formatter.applyHeader();
    }

    @Override
    public void save() {
        System.out.println("Saving Word document.");
        formatter.applyFooter();
    }

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

class PDFDocument extends Document {
    public PDFDocument(DocumentFormatter formatter) {
        super(formatter);
    }

    @Override
```

```
    public void open() {
        System.out.println("Opening PDF document.");
        formatter.applyHeader();
    }

    @Override
    public void save() {
        System.out.println("Saving PDF document.");
        formatter.applyFooter();
    }

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

// Client Code
public class DocumentEditor {
    public static void main(String[] args) {
        DocumentFormatter headerFormatter = new HeaderFormatter();
        DocumentFormatter sidebarFormatter = new SidebarFormatter();

        Document wordDocument = new WordDocument(headerFormatter);
        wordDocument.open();
        wordDocument.save();
        wordDocument.close();
```

```
        System.out.println("---");


        Document pdfDocument = new PDFDocument(sidebarFormatter);
        pdfDocument.open();
        pdfDocument.save();
        pdfDocument.close();
    }
}
```