```java
/**
 * Consider a scenario where you are developing a text editor application.
 * The application allows users to write and format text,
 * and you want to implement a feature to apply various formatting options
 * such as bold, italic, and underline to the text.
 * However, users should be able to combine these formatting options
 * in a flexible way and add new formatting options
 * in the future without modifying existing code.
 *
 * Solve this problem using Decorator Pattern with the help of following starter code.
 */

// Component interface representing the basic text functionality
interface Text {
    String getContent();
}

// Concrete implementation of the Text interface
class PlainText implements Text {
    private String content;

    public PlainText(String content) {
        this.content = content;
    }

    @Override
    public String getContent() {
        return content;
    }
}

// Decorator interface
interface TextDecorator extends Text {
    // Add Additional methods for formatting options (if necessary)
}

// Concrete decorators for different formatting options
class BoldDecorator implements TextDecorator {
    // Add member variables and functions\
}

class ItalicDecorator implements TextDecorator {
    // Add member variables and functions
}

class UnderlineDecorator implements TextDecorator {
    // Add member variables and functions
}

public class TextEditor {
    public static void main(String[] args) {
        Text plainText = new PlainText("Hello, World!");

        Text boldText = new BoldDecorator(plainText);
        Text italicText = new ItalicDecorator(boldText);
        Text finalText = new UnderlineDecorator(italicText);

        System.out.println(finalText.getContent());
        // Output: <u><i><b>Hello, World!</b></i></u>
    }
}

// Note: You are not allowed to change other part of this code.
```