Exercise 8
– Design Patterns –

1. We want to implement a text paragraph. A paragraph is a sequence of lines. Each line is represented by a string. The `Paragraph` class has to provide at least the following methods:

   String getLine(**int** i);   // *returns i−th line*
   int getCountLines();   // *returns number of lines*
   void addLine(String s); // *appends a line to the end of the paragraph*

   A text paragraph is always formatted. Whenever a line is added, it is formatted immediately. The format algorithm (e.g., left-align or centered) can be selected at runtime. It also has to be possible to add new format algorithms to the program without modifying the `Paragraph` class.

   Your task: Develop a design for `Paragraph` that satisfies the above requirements. Which design pattern could you use?

2. We want to develop a Java AWT component that can display formatted text in a flexible way. To do that, we want to reuse the `Paragraph` class. Each Java AWT component inherits from class `Component`. Our `FormattedTextArea` will inherit from the following library classes:

   public class TextComponent **extends** Component **implements** Accessible { ... }

   public class TextArea **extends** TextComponent {
     public void append(String str) { ... }
     public String getText()   { ... }
       ...
   }

   Your task: Develop a design for `FormattedTextArea` that allows us to reuse `Paragraph` and to inherit from `TextArea`. Which design pattern could you use? You should also provide implementations for the methods `append` and `getText`.

3. We want to extend our design by a character counter. This counter is a separate object that stores the number of characters in a `Paragraph` object. Whenever the `Paragraph` object is changed, the counter has to be adapted automatically.

   Your task: Develop a design for the counter. You are allowed to modify the `Paragraph` class. Which design pattern could you use?