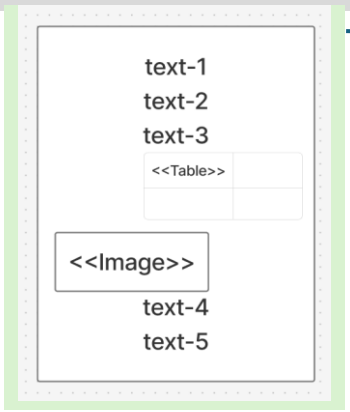
Building a Document Editor app (A Real-World LLD Project)



Design Now

Salable for support

Videos
Fonts
Drawings
Tabs
Hyperlinks
Shapes
Colors

In system design, there are two primary approaches:

Top-Down Approach	Bottom-Up Approach
• High-level view first • Break down into smaller components • Focus on interfaces and interactions	• Start with individual components • Build up to the overall system • Focus on internal behaviour

Here we are going to use ***Bottom-up*** approach. Initially we create a bad design and improve it step by step.

UML diagram:

DocumentEditor
(-) elements : List<String> (-) renderedDocument : StringBuilder
(+) addText : void (+) addImage : void (+) renderDocument : String (+) saveToFile() : Boolean

- **private elements**: it is a List of string going to contain text lines and image paths as text.
- **private renderedDocument**: it is a String which initially empty, and fills with texts and images taken from elements after calling renderDocument();
- **public addText()**: it is a method which is simply add a text line to the element list.
- **public addImage()**: it is a method which is used to add image path to the element list with starting symbol (~) for identification.
- **public renderDocument()**: it is a method which is used to render the element list and create a single string. It extracts the image path also.

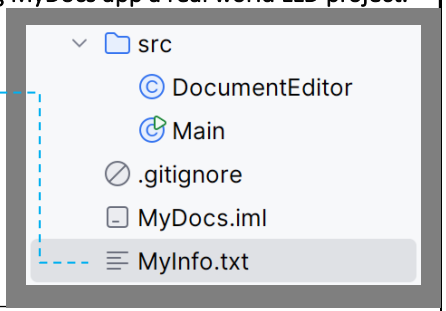
```java
import java.io.FileWriter;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import java.util.logging.Logger;

public class DocumentEditor {
    private static final Logger logger = Logger.getLogger(DocumentEditor.class.getName());

    private final List<String> element;
    private final StringBuilder renderedDocument;

    DocumentEditor() {
        this.element = new LinkedList<>();
        this.renderedDocument = new StringBuilder();
    }

    // add text to the Docs
    public void addText(String text) {
        element.add(text);
    }

    // add the image file path to the Docs
    public void addImage(String text) {
        element.add("~" + text);
    }

    // render the document by checking the type(text or image) at runtime
    public String renderDocument() {
        if (renderedDocument.isEmpty()) {
            for (String text : element) {
                if (text.charAt(0) == '~')
                    if (text.length() > 5 && (text.endsWith(".jpg") || text.endsWith(".png")))
                        renderedDocument.append("[image:").append(text.substring(1)).append("]");
                    else renderedDocument.append("[image not found]");
                else renderedDocument.append(text);
                renderedDocument.append("\n");
            }
        }
        return renderedDocument.toString();
    }
    // save as a txt file
    public Boolean saveToFile() {
        if (renderedDocument.isEmpty()) {
            renderDocument();
        }
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter File name: ");
        String name = sc.next() + ".txt";
        boolean flag = true;
        try {
            FileWriter writer = new FileWriter(name);
            writer.write(renderedDocument.toString());
            writer.close();
        } catch (IOException e) {
            System.out.println("Unknown error occurred");
            logger.severe("Something went wrong" + e.getMessage());
            flag = false;
        }
        logger.info("App ended");
        return flag;
    }
}
```

~~~~~~~~~~~~~~~~~~~~Client Code~~~~~~~~~~~~~~~~~~~~

```java
public class Main {
    public static void main(String[] args) {
        DocumentEditor editor = new DocumentEditor();

        editor.addText("Welcome to MyDocs 🥳 🎉 🎏 ");

        editor.addImage("sibasundar.jpg");

        editor.addText("Hello, I am Sibasundar Jena, and currently I am making " +
                " MyDocs app a real world LLD project.");

        System.out.println(editor.renderDocument());

        System.out.println(editor.saveToFile()); // MyInfo.txt
    }
}
```

~~~~~~~~~~~~~~~~~~Output-of-client-code~~~~~~~~~~~~~~~~~~

```
Welcome to MyDocs 🥳 🎉 🎏
[image:sibasundar.jpg]
Hello, I am Sibasundar Jena, and currently I am making MyDocs app a real world LLD project.

Enter File name:
MyInfo
Jun 19, 2025 10:22:08 PM DocumentEditor saveToFile
INFO: App ended
true

Process finished with exit code 0
```



@Sibasundarj8

Major Issues:

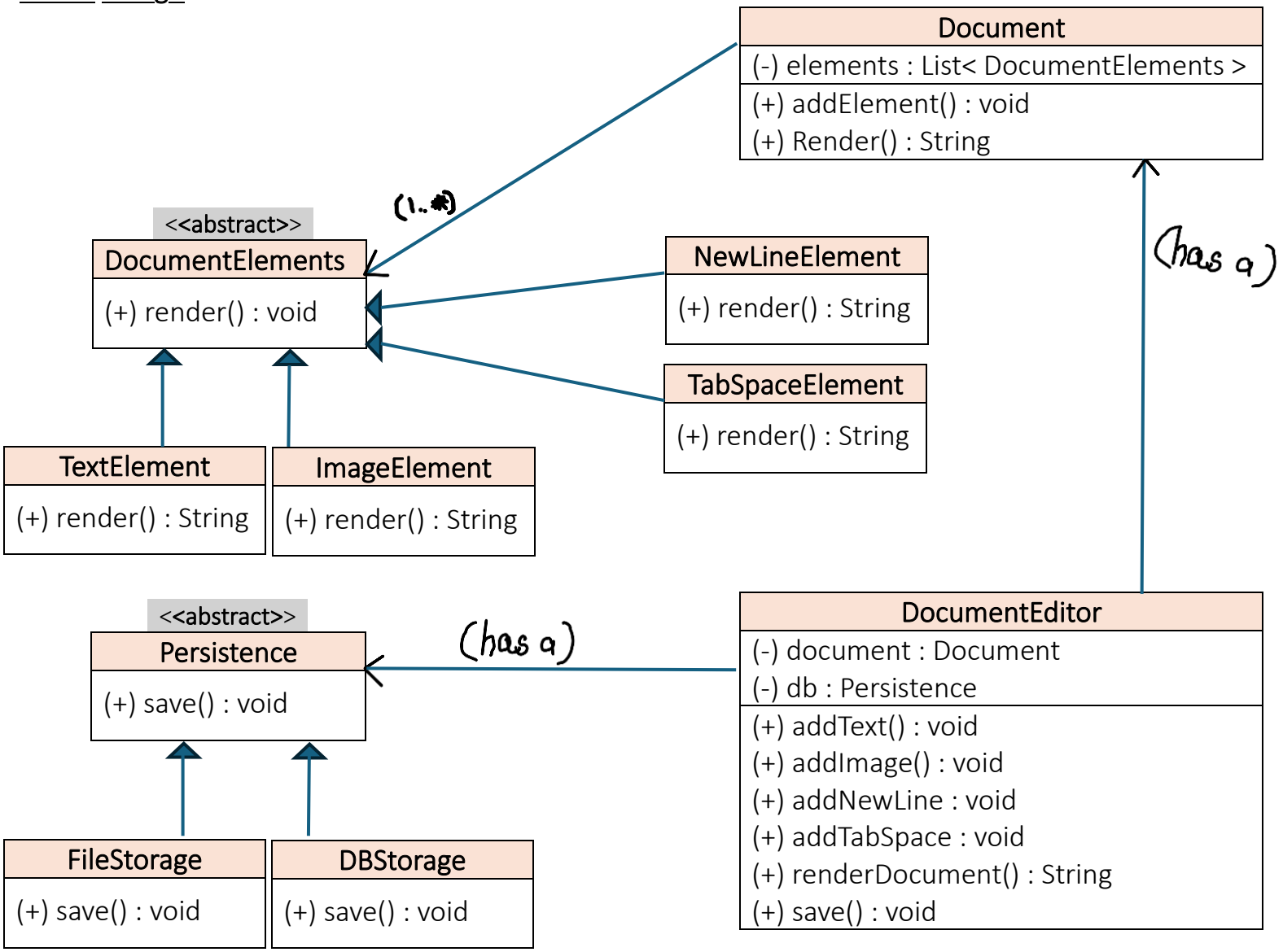
The current implementation of the DocumentEditor class ***violates*** several ***SOLID principles*** of object-oriented design:

1. Single Responsibility Principle (SRP):
   The DocumentEditor class currently handles ***multiple responsibilities*** — ***managing*** a collection of text and image elements, ***rendering*** the document, and ***saving*** it to a file. Each of these ***responsibilities*** should be ***encapsulated*** in ***separate classes*** to ensure ***well-defined responsibilities*** and ***easier maintenance***.

2. Open/Closed Principle (OCP):
   The class is ***not open*** for ***extension*** but ***closed*** for ***modification***. ***Adding new functionality***, such as support for new element types (e.g., tables, videos) or different rendering formats (e.g., HTML, Markdown), would require ***direct modification*** of the ***DocumentEditor class***, thereby ***increasing*** the ***risk*** of ***introducing errors*** and ***reducing flexibility***.

To align ***better*** with ***SOLID principles***, the ***responsibilities*** should be ***divided*** into ***distinct classes*** or ***interfaces***.

Better Design:



DocumentElements

```
public interface DocumentElements{
   String render();
}
```

TextElement

```
public class TextElement implements DocumentElements{
   private final String text;
   public TextElement (String text) {
      this.text = text;
   }
   @Override
   public String render() {
      return text;
   }
}
```

ImageElement

```
public class ImageElement implements DocumentElements {
   private final String image;
   public ImageElement(String path) {
      this.image = path;
   }
   @Override
   public String render() {
      if (this.image.endsWith(".jpg") ||
          this.image.endsWith(".png")) {
         return "[image:" + this.image + "]";
      }
      return "image not found";
   }
}
```

NewLineElement

```
public class NewLineElement implements DocumentElements {
   @Override
   public String render() {
      return "\n";
   }
}
```

TabSpaceElement

```
public class TabSpaceElement implements DocumentElements {
   @Override
   public String render() {
      return "\t";
   }
}
```

Persistence

```
public interface Persistence {
   void save (String data);
}
```

DBStorage

```
public class DBStorage implements Persistence {
   @Override
   public void save(String data) {
      // code to store in database
   }
}
```

FileStorage

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Logger;

public class FileStorage implements Persistence {
   private final String name;
   private final Logger logger = Logger.getLogger(FileStorage.class.getName());
   public FileStorage(String name) {
      this.name = name + ".txt";
   }
   @Override
   public void save(String data) {
      String msg = "successfully saved";
      try {
         FileWriter writer = new FileWriter(name);
         writer.write(data);
         writer.close();
      } catch (IOException e) {
         logger.severe("Something went wrong" + e.getMessage());
         msg = "app ended";
      }
      logger.info(msg);
   }
}
```

Document

```
import java.util.LinkedList;
import java.util.List;

public class Document {
   // has a relationship (1 to many)
   private final List<DocumentElements> elements;
   Document () {
      this.elements = new LinkedList<>();
   }
   // add new elements to the Document
   public void addElement (DocumentElements element) {
      this.elements.add(element);
   }
   // render the whole Document
   public String render () {
      StringBuilder renderedDocument = new StringBuilder();
      for (var ele : this.elements) {
         renderedDocument.append(ele.render());
      }
      return renderedDocument.toString();
   }
}
```

Dt- 22 – 06 – 2025, 02:54 AM

@Sibasundarj8

DocumentEditor

```java
public class DocumentEditor {
    private final Document document;
    private final Persistence db;
    private String renderedDocument;
    // constructor injection (dependency injection through constructor)
    public DocumentEditor(Persistence db) {
        this.document = new Document();
        this.db = db;
        this.renderedDocument = "";
    }
    public void addText(String text) {                    // add Text
        document.addElement(new TextElement(text));
    }
    public void addImage(String path) {                   // add Image
        document.addElement(new ImageElement(path));
    }
    public void addNewLine() {                            // add new line
        document.addElement(new NewLineElement());
    }
    public void addTabSpace() {                           // add Space Tab
        document.addElement(new TabSpaceElement());
    }
    public String renderDocument() {                      // render the document
        if (this.renderedDocument.isEmpty()) {
            this.renderedDocument = document.render();
        }
        return this.renderedDocument;
    }
    public void save() {                                  // save to provided database
        this.db.save(this.renderDocument());
    }
}
```

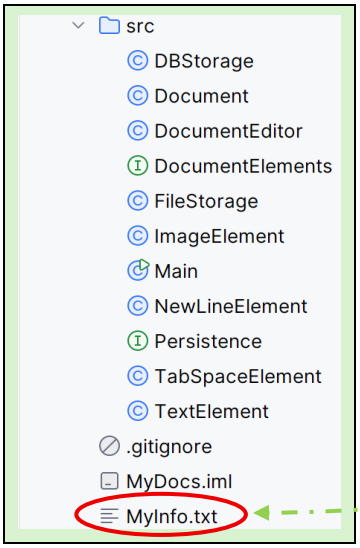
~~~~~~~~~~~~~~~~Client Code~~~~~~~~~~~~~~~~

```java
public class Main {
    public static void main(String[] args) {
        Persistence db = new FileStorage("MyInfo");
        DocumentEditor editor = new DocumentEditor(db);
        editor.addText("Welcome to MyDocs 🥳 🎉 🎊 ");
        editor.addNewLine();
        editor.addImage("sibasundar.jpg");
        editor.addNewLine();
        editor.addText("Hello, ");
        editor.addNewLine();
        editor.addTabSpace();
        editor.addText("I am Sibasundar Jena, and currently I am ");
        editor.addNewLine();
        editor.addTabSpace();
        editor.addText("making MyDocs app a real world LLD project.");
        editor.addNewLine();
        System.out.println(editor.renderDocument());
        editor.save(); // MyInfo.txt
    }
}
```

~~~~~~~~~~Output~of~client~code~~~~~~~~~~

Welcome to MyDocs 🥳 🎉 🎊
[image:sibasundar.jpg]
Hello,
I am Sibasundar Jena, and currently I am making MyDocs app a real world LLD project.

Jun 22, 2025 3:04:46 AM FileStorage save
INFO: successfully saved



- This design adheres to the SRP as each class is ***responsible*** for ***a single, well-defined task***.
- It complies with the OCP because ***new elements*** or ***storage systems*** can be added by ***extending*** existing abstractions ***without modifying*** existing code.
- The design supports the LSP since all ***subclasses*** can be ***substituted*** for their ***parent classes*** without affecting the functionality.
- It follows the ISP as no class is ***forced to implement methods*** it does not use; all interfaces are ***minimal*** and ***specific*** to the implementing classes.
- Lastly, it respects the DIP by ensuring that ***high-level modules*** (like ***Document*** or ***DocumentEditor***) ***depend on abstractions*** rather than ***concrete implementations*** (like ***TextElement*** or ***DBStorage***).