|  |  |
| --- | --- |
|  |  |
| Software Analysis |  |
|  |  |
|  | Jabber pointSoftware quality |
|  | Mahmoud HamidJoris Backis |

**Chapter 1:**

**System Overview**

**JabberPoint is a Java-based slide presentation application with a simple architecture designed for creating and displaying presentations.**

**Core Components**

1. **Presentation Management**

* Presentation class serves as the core model managing slides
* Maintains presentation title, slide list, and current slide number
* Controls navigation between slides (next/previous)

**2. Slide Structure**

* Slide class represents individual slides
* Contains title and vector of SlideItem objects
* Handles drawing and layout of slide content
* Standard slide dimensions: 1200x800 pixels

**3. Content Items**

* Abstract SlideItem base class for slide content
* Two concrete implementations:
  + TextItem: Renders text with formatting
  + BitmapItem: Displays images with scaling support

**4. User Interface**

* SlideViewerFrame: Main application window (JFrame)
* SlideViewerComponent: Rendering component for slides
* Menu system (MenuController)
* Keyboard navigation (KeyController)

**5. Storage System**

* Abstract Accessor class defining load/save operations
* Implementations:
  + XMLAccessor: Handles XML file format
  + DemoPresentation: Built-in demo content

**Technical Details**

**Data Structures**

* Slides stored in ArrayList<Slide>
* Slide items stored in Vector<SlideItem>
* Styles maintained in static array

**Style System**

* 5 predefined style levels
* Properties per style:
  + Indent
  + Color
  + Font size
  + Leading (line spacing**)**

**File Formats**

* XML-based storage format
* Tags:
  + presentation: Root element
  + showtitle: Presentation title
  + slide: Individual slides
  + item: Slide content with attributes:
    - kind: text/image
    - level: Style level

**Chapter 2:**

**Key Features**

1. **Navigation**

* Keyboard shortcuts:
  + Page Down/Enter: Next slide
  + Page Up/Up Arrow: Previous slide
  + Q: Quit
* Menu navigation options

1. **Content Support**

* Text with multiple style levels
* Image support (via BitmapItem)
* Scalable content rendering

1. **File Operations**

* Open XML presentations
* Save presentations
* Built-in demo presentation

1. **Technical Constraint**

* Fixed window size (1200x800)
* Limited style customization
* Basic image support

1. **Feature Gaps**

* No real-time editing
* Limited formatting options
* No animation support
* No multimedia support

**System Requirements**

1. **Software**

* Java Runtime Environment
* Swing GUI framework
* XML parser support

1. **File System**

* Read/Write access for XML files
* Image file access

**Chapter 3:**

**Class Descriptions and Issues**

1. **AboutBox**

What it does: Shows a pop-up window with app info (name, version, copyright).  
Problems:

* All text is hardcoded (can't change without editing code)
* Uses + to build long messages (looks messy)
* No translation support

1. **Accessor**

What it does: Base class for reading/writing presentation files.  
Problems:

* Empty constructor that does nothing
* Confusing setup for different file types (XML/demo**)**

**3. BitmapItem**

What it does: Shows images in slides.  
Problems:

* Crashes if image path is wrong (no error message to user)
* Doesn't check if images are too big for slides

**4. DemoPresentation**

What it does: Provides built-in demo slides.  
Problems:

* Contains typos (e.g., "andn" instead of "and")
* Demo content can't be changed without editing code
* Save button crashes instead of showing error

**5. JabberPoint (Main)**

What it does: Starts the program.  
Problems:

* Doesn't check if input files exist
* Uses System.exit (closes app abruptly)

**6. KeyController**

What it does: Handles keyboard shortcuts (e.g., arrow keys).  
Problems:

* Only works with specific keys (no customization)
* Uses number codes instead of clear names (e.g., VK\_PAGE\_DOWN)

**7. MenuController**

What it does: Creates the program menu (File/View/Help).  
Problems:

* Crashes if you type letters in the page number box
* Always opens "test.xml" (can't choose other files)

**8. Presentation**

What it does: Manages slides (loading, navigation).  
Problems:

* Allows invalid slide numbers (e.g., -100)
* Mixed Dutch/English comments
* Handles app closing (should be main class' job)

**9. Slide**

What it does: Represents a single slide.  
Problems:

* Fixed size (1200x800 pixels)
* Uses outdated Vector instead of modern ArrayList

**10. SlideItem**

What it does: Base class for slide content (text/images).  
Problems:

* Allows negative levels (e.g., level -5)
* No support for bullet points or text styling

**11. SlideViewerComponent**

What it does: Draws slides on screen.  
Problems:

* Page number stuck at (1100,20) (bad for small screens)
* Doesn’t call parent’s paintComponent (might cause glitches)

**12. SlideViewerFrame**

What it does: Main program window.  
Problems:

* Fixed window size (can’t resize/maximize)
* Title can’t be changed easily

**13. Style**

What it does: Controls text formatting (fonts/colors).  
Problems:

* Only 5 styles (can’t add more)
* Always uses Helvetica font
* No warning if you request non-existent styles

**14. TextItem**

What it does: Displays text in slides.  
Problems:

* Shows "No Text Given" if empty
* Complicated text-wrapping code
* Slow with very long paragraphs

**15. XMLAccessor**

What it does: Reads/writes XML files.  
Problems:

* Doesn’t check if XML files are valid
* Shows errors in console instead of pop-ups
* Might save corrupted files if interrupted

**Common Issues to Notice:**

1. Magic Numbers: Direct use of numbers like 1100 or 20 (hard to understand)
2. Crash Risks: Many operations can crash instead of showing errors
3. Hardcoded Values: Text/sizes written directly in code
4. Outdated Code: Uses old Vector/AWT instead of modern Java features
5. No Input Checks: Assumes users will always enter valid data

**Future Enhancement Possibilities**

1. **Technical Improvements**

* Add unit testing
* Introduce design patterns
* Modern UI framework
* Reduce constants
* Improve error handling

**Chapter 4:**

**Showcasing the diagrams:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Class diagram**

**A diagram of a presentation

AI-generated content may be incorrect.**

**Use case diagram**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A white rectangular object with yellow text

AI-generated content may be incorrect.**

**Conclusion**

JabberPoint is a Java-based presentation tool that allows users to create and display slide presentations. It manages slides and their content, supports basic text and image items, and provides a simple user interface for navigation. The application uses XML for storing presentations and includes a built-in demo presentation. While functional, it has limitations such as fixed window size and basic style options. Future improvements could address these limitations and add more features.