architectural problems

Jabberpoint OOD

Ian Donker 4629981

29-01-2024

# Introduction

In this report we go over the problems found with Jabberpoint. Jabberpoint is a primitive tool for presentations written in Java. The application contains a few basic functions and can read an external, static, presentation. These presentations come in the form of a XML file. Jabberpoint, unfortunately, is not a modern application and contains a lot of questionable design choices

# Found problems

## Accessor class

1. The accessor class should be an interface as it only has two abstract classes.
2. The interface itself is not suitable since not every implementation needs to be saved and loaded.

### proposed fix accessor class

Change the accessor class to an interface.

The loadable and saveable can become two interfaces from the singular interface. This will allow the implementation of either one or both. This will increase the applications adaptability.

## Style class

1. There are levels for style creation which act as headers. There is however no way to create separate styles for the levels. The style class has too many responsibilities.
2. The fields in style are also all public, this will result in security problems. Also the font field is not final, but should be final as it does not change.
3. There is no communication with the user whenever a style is created which results into confusion and insufficient use quality.

### proposed fix style responsibilities

Create a StyleFactory which uses an Enum for the levels. Whenever a new style needs to be created, a new enum can be created and adjusted with the createStyle() method. This will increase the adaptability and maintainability of the application.

### proposed fix security

Set the fields in the Style class to private and create getters and setters for the fields which will increase the security of the application.

Make the font field final, remove the fontsize field as it can be obtained from the font itself. This will increase the maintainability of the application.

### Proposed fix user communication

Throw an exception when a new enum is applied but not yet defined in code. This will increase the maintainability, testability and usability of the application.

## Classes not ordered

The classes are all within the src folder which makes the application cluttered.

### Proposed fix

Create packages for the classes, which increases the maintainability of the application.

## XML Accessor

1. The XML accessor class makes an instance of the class, this is not it’s responsibility.

### Proposed fix

Create a factory for the XMLAccessor class.

## Demopresentation

1. The same problem occurs with the DemoPresentation class, it makes an instance but it’s not it’s responsibility.
2. DemoPresentation has a large method.

### Proposed fix

Create a factory for DemoPresentation class.

Divide the loadFile() method into smaller methods. This will increase the testability and maintainability.

## ABOUT BOX

1. The AboutBox class is not abstract though it doesn’t need to be instantiated.

### proposed fix

Make AboutBox abstract.

## keycontroller

The KeyController class makes a new instance while this is not it’s responsibility.

### Proposed fix

Create a factory for the KeyController class.

## Menucontroller

1. The MenuController class has the same issue as the KeyController class.
2. It has a big constructor which can be separated into smaller methods.
3. MenuController saves and loads files using an accessor of the datatype XMLAcessor
4. MenuController always sets the slideNumber to 0 when loading a file, these two actions should always occur together.

### proposed fix

Create a factory for MenuController.

Divide the constructor into smaller methods, this will increase testability and maintainability of the application.

The accessors need to be provided with SaveAble and LoadAble which makes it so that other other accessors can be used in the future. This will increase the adaptability of the application.

Move it’s responsibility of setting the slideNumber to loadFile() method so that it will always occur at the same. This will increase the reliability and usability of the application.

## Presentation and Slideviewercomponent

1. Between these two classes is a circular relation, but it should be the other way around.
2. SlideViewerComponent updates the frame everytime a slide changes, this however has no impact on the frame.

### Proposed fix

Make it so that the SlideViewerComponent class inherits and controls a Presentation, not the other way around. Moving the methods within the Presentation class that deal with individual slides to the SlideViewerComponent. Replace all the presentation independities to SliderViewerComponent in the other classes of the application. This will increase the maintainability and useability of the application.

Move and divide the update() method from SlideViewerComponent to the classes that need to inherit the class, so that they can update individually. This will improve the performance of the application.