Student information

Cursuscode IM0102

Scenario JabberPoint with themes

Name A. Slomp

Student number 838768442

Name W. van der Weij

Student number 851988675

1 Introduction

This is our report of the final assignment of the Design patterns course of the Open Universiteit. In this report we describe our solution to the redesign of the JabberPoint application.

This report is structured as follows. The report starts with a description of the way we worked together on the assignment. The problem analysis describes the goal, concepts with commonality and variability analysis, application functionality and the assumptions during the assignment. [add more structure later]

2 Project execution

Here we describe how we handled the assignment together. The numbers indicate the explicit ordering of the activities.

1. We both studied the assignment separately to prepare for our first meeting.

2. After a couple of days we met at Wilkos place for the first time to discuss the assignment. Together we went through the assignment, the program code, the running application and we discussed possible solutions on conceptual level as well on implementation level. Just exchanging ideas, brain storming and discussing about possible solutions. We also discussed the tools to use (Eclipse with Papyrus plugin) and the format of the report (Latex). Furthermore we looked together at the git repository, the usage of github.com and how to operate it locally. We also discussed the planning of the assignment.

3. We both started working on the assignment separately. We both agreed to do most of the work separately, but based on our discussions and compare solutions. From our discussions we decided to have a different focus, but still do the work both. Arend focusing on the class diagrams and Wilko focusing on the problem analysis.

4. About a week after the first meeting, we met again to discuss our progress. This time at Arends place. We still had a lot to discuss. We went through the code, annotating it with TODOs at places with strange code or remarkable design. We went through two more documents from you learn: *Guidelines redesign* and *Beoordeling Design patterns*. The work to be done seemed more than the expectations we had after the first meeting. Again we agreed to do

both the work with focus on different items. Arend puts together both reports and looks at the design, while Wilko does the CVA, choices and description of design patterns to be used.

3 Problem analysis

The assignment is to redesign the existing application JabberPoint to support multiple themes. The feature request is to let the user pick one of the available themes and apply the new styles to the content of the presentation. The selected theme should be applied to the current application and the current slide.

Besides the addition of the feature request, the application is redesigned further. The goal is to improve the quality of the application overall and to achieve a better user experience, increased flexibility and easier maintainability.

Out of scope are at this moment:

- editing presentations
- editing themes

3.1 Application functionality

The JabberPoint application is started with the internal demo presentation or a presentation loaded from a file specified by the command line parameter. The first slide of the presentation is displayed. JabberPoint accepts one optional parameter during startup: a presentation file. An invalid parameter value results in an application without any presentation loaded.

The view of the application consists of a program window with a menubar at the top. The application is operated by a pointing device and a keyboard.

Application and file actions:

- Open
- New clears the current presentation
- Save saves the presentation (writes theme?)
- Exit

Slide actions:

- Next slide menu, key
- Previous slide menu, key
- Goto slide menu, shortcut

New action:

- Select theme

The presentation content could be read from a file or otherwise a demo presentation from within the application is read

JabberPoint does not support editing of presentations.

3.2 Concepts

A *presentation* consists of a number of sequential slides, which a user can step through forward and backwards. The presentation displays information to the audience during a talk or speech. In the context of this document a presentation is the content that the JabberPoint application shows and it is not the speech or talk to an audience itself.

A *slide* is a page in a presentation that contains display elements like text and graphics. The display elements are displayed according to predefined styles. Every slide has a title.

A slide contains *item*s such as text and graphical elements. All these items have varying and some overlapping behaviours, such as: font type, text size, background colour and size. The items on a slide have a indentation, which is also related with the appearance or style of the item.

*Theme*s arange the displaying of the elements of a presentation. A theme consists of styles that control the display behaviour such as background colour, text colour and font size. The styles within the theme are stacked and the levels are leading when a theme is applied to a slide.

A *style* prescribes the displaying behaviour of items on slides.

The *view* of the application consists of a window that has a menubar at the top left. The remainder of the window is used to render the slides of a presentation.

The application is able to read a XML file that contains a presentation according to a defined structure.

3.3 Commonality and Variability analysis

[cva and or following the video] Presentation: multiple slides, not changing.

Slide
Slide items
Theme
Style
Reader

3.4 Assumptions

Assumption is that altering the supported file format is allowed, as long as the application is backwards compatible.

Assumption is that bug fixes are appreciated as long as functionality is not decreased. The behaviour of the next, previous and goto functionality is buggy. Several scenario's result in failures. For example the goto functionality does not validate input, resulting in errors and faulty behaviour for non-numerical and out of range values. Hitting next or previous slide quickly after each other let the presentation disappear (in MacOS with the menu options shortcuts).

4 Design

4.1 Guidelines and principles

Design decisions during the design process are guided by design principles and guidelines that we describe briefly. We have a couple of design principles:

- Single responsibility principle. Do one thing and do it well, without side effects. Separate responsibilities lead to high cohesion. An class should be responsible for itself.
- Open-closed principle. Open for extension, closed for modification. Make future changes without breaking existing classes.
- Don't repeat yourself principle. There must be a lack of redundancy. Rules should be implemented once. One rule, one place.
- Liskov substitution principle. Subtypes should be substitutable for base types. Well designed inheritance.

Other guidelines or best practices are:

- Prefer delegation over inheritance.
- Program to interfaces or abstract classes.
- Each interface should have one reason to change.

These principles and guidelines should lead to clarity, high cohesion and low coupling. These are ingredients for a flexible and easy to extend application. We think that is exactly what JabberPoint needs to be. We also incorporated the principle of healthy skeptism.

4.2 Design patterns

4.3 Model-View-Controller (MVC)

The MVC pattern separates the model, the controller and the view.

- The model manages the data, logic and rules of the application.
- The controller accepts input and converts these to commands for model or view.
- The view is responsible to display the information from the model to the user and respond to user actions.

Especially removing the view dependency from the model promotes flexibility of this design pattern. This prepares for changing or adding views without changes in the model.

4.4 Observer pattern

The Observer pattern decouples the JabberPoint model from the view. This is part of the MVC pattern. The view is registered as a listener, which is notified when the model changes.

4.5 Factories

The design consists of factories to have a single point for object creation for each type of object. This supports the *program to interface* idiom, because the factories create the concrete implementations, while the application only depends on the abstract concepts. The factory classes are named XxxxFactory and are put in the factory package to make them recognizable.

5 Choices

The report and documentation are in English, not Dutch. English is the obvious choice as software development is often cross-border. Main reason: flexibility regarding the future developments of the product without much initial investments. Besides this, it is a good exercise for students.

Support of the current file format is continued. The application will display the test.xml file correctly as well as files that are in the wild. Customer trust and satisfaction are essential.

The file format is allowed to alter, but the implementation must be backwards compatible. This allows the introduction of new features while still supporting all presentations for all users.

Bugs are fixed. These refers to clear bugs that can be fixed without big risks of introducing unexpected behaviour. Quality of the application and user experience are important.

We redesign the application according to the

The themes are implemented in an XML file or hard-coded? TODO

6 Sourcecode