University of Vienna
Faculty of Computer Science
Research Group Software Architecture

Task 2

Design Patterns in Practice 1

2017S – 051050 VU – Software Engineering 2

April 10, 2017

General Remarks

- This document covers the assignment of **Task 2**.
- You can achieve a total of **18 points** for this task.
- The **deadline** for this **Task 2** is **25.05.2017 at 23:59**. No deadline extensions are given.
- This is a **group work**. You and your work group members are allowed to work together in solving this assignment. Be aware that **all** group members shall contribute to the task and document their contribution.
- If you copy code or other elements from sources other than the lecture slides, please provide a reference to it in a comment above the corresponding entry.
- If you encounter problems, please post your question in the Moodle[1] discussion page, as it probably is also of interest to other colleagues. Alternatively, you can contact the tutors via se2.tutor@swa.univie.ac.at. As a last resort you can contact the course supervisor directly via se2@swa.univie.ac.at.

Submission Guidelines

All files required by this assignment have to be submitted to our GitLab[2] server into the proper project (repository) in the Submission and Feedback System[3]. **Be aware** to push your solutions into the correct submission **branch**, which is for this task **2017s_se2_task2**.

As discussed in the Git[Lab Submission] Tutorial[4] the submission branch is created for you. For any questions regarding the **GitLab**-based submission please refer to the Git[Lab Submission] Tutorial.

[1]https://moodle.univie.ac.at/course/view.php?id=61845
[2]https://gitlab.swa.univie.ac.at
[3]https://gitlab.swa.univie.ac.at/submission
[4]https://gitlab.swa.univie.ac.at/submission/tutorial

Task 2: Design Patterns in Practice 1

Task 2 aims to provide a practical exercise on the topics covered in the lecture part of the Software Engineering 2 course.

Description

The goal of the practical exercise is the development of a spreadsheet application in Java using the Swing API[5] for the graphical user interface.

Functional Requirements (FR)

FR1 The application must be able to **read and write CSV** (Comma-separated values) files. To test this functionality use the open data sets from https://www.data.gv.at/ given below:

- Dataset A[6]
- Dataset B[7]
- Dataset C[8]

FR2 Spreadsheets can be stored in files. Since a spreadsheet might contain not only data in tabular form but also plots of the data, the application must support a data format that is able to store **both**. You must use either a **standard format** (e.g., OpenDocument[9]) or your **own proprietary format**.

FR3 A **cell** must support at least three kinds of data:

- **Text**
- **Number**
- **Formula**
  - A cell that starts with the equals sign (=) is interpreted as a formula.
  - A formula may **reference cells**.
  - A formula must support **elementary arithmetic operations** (i.e. addition, subtraction, multiplication, and division).
  - A formula must support the following **functions**:
    `SUM( FROM_CELL : TO_CELL )` Addition of all numbers in a range of cells.
    `MEAN( FROM_CELL : TO_CELL )` Arithmetic mean of the numbers in a range of cells.
    `COUNT( FROM_CELL : TO_CELL )` Counts the number of cells in a range.

[5] https://docs.oracle.com/javase/7/docs/technotes/guides/swing

[6] https://www.data.gv.at/katalog/dataset/stadt-wien_viebevlkerungseit1869wien/resource/f55512e6-81ef-4fa9-a01f-19f9c5f838c2

[7] https://www.data.gv.at/katalog/dataset/stadt-wien_viebevlkerungnachgeburtslandseit2011wienerzhlbezirke/resource/b46ba313-cc13-488d-aace-d19c29439cdc

[8] https://www.data.gv.at/katalog/dataset/stadt-wien_vieregisterzhlung2011wienerzhlbezirke5/resource/4e15b341-99f0-4592-b3c8-e8241c7aa549

[9] http://www.opendocumentformat.org/

FR4 The application must support (near) real-time visualization of the data for a range of cells with at least two types of **charts**:

- Line chart (e.g. Dataset[10])
- Bar chart (e.g. Dataset[11])

Non-Functional Requirements (NR)

NR1 **Comment** your code and provide JavaDocs.

NR2 Your implementation must be in compliance with the **Google Java Style Guide** [12] and other **common coding practices** (see lecture slides).

NR3 Use **defensive programming** (see lecture slides).

NR4 Use **key design principles** (see lecture slides).

NR5 Make sure that your implementation works properly by **testing** your implementation thoroughly (e.g., JUnit).

NR6 The solution **must use** the following design **patterns** whenever it is appropriate and at least once overall:

- **Observer Pattern**
- **Strategy Pattern**

The following optional patterns should be used whenever beneficial:

- **Abstract Factory Pattern**
- **Decorator Pattern**
- **Factory Method Pattern**

Additional Requirements (AR)

AR1 **Keep records** of your activities whenever you work on the project. That includes every activity that is directly related to this assignment. Important points that must be covered are:

- **Who**? Did you work alone or together with team members?
- **When** and for how long? Did you work in several iterations on a problem?
- **Why**? What is the purpose of your activity?
- **What** was achieved?

AR2 How and to what extent have you considered **coding practices**? Discuss and show examples from your code.

[10] https://www.data.gv.at/katalog/dataset/monatsmitteltemperaturen-ab-1981/resource/f064ce04-8b36-4ff1-aa95-dd2a921a940b

[11] https://www.data.gv.at/katalog/dataset/stadt-linz_flchederstatistischenbezirke/resource/9607bd41-792b-4f27-8c6d-a411a89be003

[12] https://google.github.io/styleguide/javaguide.html

AR3 How and to what extent have you considered **defensive programming**? Discuss and show examples from your code.

AR4 How and to what extent have you considered **key design principles**? Discuss and show examples from your code.

AR5 Where do required design patterns occur in the solution? Discuss at least one occurrence for each required **design pattern** in the code in detail. Support the use of a specific pattern by arguments. Create UML diagrams to illustrate how the pattern has been applied.

Submission

Deadline for this **Task 2** is on **25.05.2017 at 23:59**. Only material that has been submitted in time (pushed to your GitLab Work Group Project in the suitable branch) and following the naming conventions will be taken into consideration:

- Records of your work in a single file named `team_records.pdf` in the root directory.
- A discussion on how you applied coding practices in a file named `coding_practices.pdf` in the root directory.
- A discussion on how you applied defensive programming in a file named `defensive_programming.pdf` in the root directory.
- A discussion on how you applied design patterns in a file named `design_patterns.pdf` in the root directory.
- Your implementation (including test cases etc.) in the folder `implementation` in the root directory. Please **do not** bundle your code in archives like `.zip`, `.7z`, `.rar`, etc.
- Your documentation on how the application is to be installed, initialized, and tested in a file named `howto.pdf` in the root directory

Examination

Each team will be assigned a time slot to present the solution. Each group member should have knowledge of the **entirety** of the submitted solution, not simply the part he or she has worked on, and be able to explain the design process, decisions, and the rationale behind them. No elaborate presentation material (PowerPoint slides etc.) is required apart from the material you submitted as your solution.