|  |  |
| --- | --- |
| http://www.7wandtattoos.de/media/catalog/product/cache/1/image/9df78eab33525d08d6e5fb8d27136e95/w/a/wandtattoo-zahnraeder-inhalt.jpg  Group Project - TDDT | Group: NULL  Programmierpraktikum  Project 7 |

# Title page

Title Group Project - TDDT

Author Christian Karamann (24 21 98 8)

Marvin Krause (24 14 15 7)

Elmedin Turic (24 21 50 6)

Patrick Remy (23 85 48 0)

Pascal Fischer (23 84 94 4)

Learning Arrangement Programmierpraktikum

Institute Heinrich Heine Universität

Location of Institute Düsseldorf, Germany

Learning Arrangement Computer Science

Study Phase 2. Semester

Academic Year 2016

Date of Completion 14.07.2016

Language English

Table of Contents

[Title page I](#_Toc453894928)

[1. Open Questions: 1](#_Toc453894929)

[1.1 Highscore: 1](#_Toc453894930)

[1.2 Size: 1](#_Toc453894931)

[1.3 Still running: 1](#_Toc453894932)

[1.4 Mines: 1](#_Toc453894933)

[1.5 Clone: 1](#_Toc453894934)

[2. Project: 2](#_Toc453894935)

[3. Classes 2](#_Toc453894936)

[4. Critical Appraisal: 4](#_Toc453894937)

# Open Questions:

*To begin with, open questions that appeared doing the process of creating this program, should be addressed. On top of that, specific aspects need to be discussed, that influence the way, the project has been designed.*

## Highscore:

To begin with, the highscore menu has not been clearly defined. It is not mentioned, whether it should be linked to the actual game or not. If somebody would exploit this assignment, he or she would simply create a text-file with 3 names and their imaginary highscore in it. To sum it up, the highscore system could have been exploited easily and should have been defined more precisely.

In the end, the author decided to create 3 text files, one for each level of difficulty. As soon as the player wins the game, he needs to type in his name and, in the background, the program calculates, on which position he got. Due to time issues, the author decided to leave out an in-game menu, where the player could check the highscore-table. However, this could be easily implemented in the future.

## Size:

The size of the board is the next aspect that needs to be discussed. The author decided to create a size\*size field for each level of difficulty. The size is fixed and can only be changed by changing the difficulty.

## Still running:

The author did not understood, what was mentioned with the task to show “if a game is currently running”. This does not make sense, because as long as you can click on the board, the game is going on. The author doubts that anyone misunderstands this. Therefore, the author decided to leave it out of the project.

## Mines:

Like the size, the mines need to be discussed. It is important to determine beforehand, how many mines are going to be implemented on every board. Hereby, the author decided to go with a formula the leads to a fair amount of mines on every level of difficulty. The formula can be found in the Game.java file.

## Clone:

It is mentioned, that the author should create a minesweeper “clone”. However, in how far does the clone need to be like the original? The author decided to implement as many things as possible, to create a “clone”. To begin with, all the images that represent the board are from the original minesweeper game. On top of that, the same restart-button has been implemented as well. The flag that gets set by clicking the right mouse button is implemented as well. However, the author was not able to implement the style of the counter and the buttons, in the same way they were in the original minesweeper game.

# Project:

*After clarifying the open questions, it is necessary to explain the purpose and goal of this project. On top of that, it is crucial to describe the repository and its structure.*

## Project description:

*To begin with, it is important to clarify the general goal of this project, as well as its purpose and function.*

The goal of this project is to write an application with a GUI where students would be able to learn how to program on test-driven basis. Therefore, the students would get together in groups of 3 to 5 people and work on it for 4 weeks. The application should be build in gradle and support travis, two tools which should help the students in their project. On top of that, the project should contain a manual and a report, which describes the whole project.

Now, the general idea of this project is going to be explained. Like mentioned before, the application should help the user understand and learn how to program in a test-driven way. Therefore, first, the user should be able to choose between different exercises. Here, the teacher sends the student a XML-file, which works like a catalog containing different exercises. From this catalog, the student can choose an exercise and start working on it.

In total, there are 3 phases which form a cycle:

First, the user needs to write exactly one test that fails. He is not allowed to write multiple failing tests or else he will not be able to continue. Second, the user needs to write code in order to pass that one failing test. After passing the test, the user gets into the third phase, where the user is allowed to improve his written code. When this code also passes the tests, the user is allowed to write the next failing test and so on, until the whole program succeeds.

On top of all that, the students could decide for two out of three possible features, which the application should have. The first one is called Babysteps and it limits the time, the user has to finish one phase. The second one is called Tracking, which should show a graph of how much time the user spent in every single phase. The final feature is called acceptance testing. Hereby, the user should write a test before going into the first phase. This test should stay unfulfilled until a feature has been implemented completely. It can be seen as a final test for an implemented feature.

For this project, the group decided to implement the features Babysteps and Tracking.

To sum it up, the group should write an application with which the first semester students would learn how Test-Driven-Development works.

## Repository description:

*After explaining the general idea of this project, it is crucial to describe the groups’ repository and its structure.*

To start, the repository is called “programmierpraktikum-abschlussprojekt-null”. It mainly contains two folders and several files, which are going to be described.

To start, the folder called “Project 7” contains all relevant files that are necessary to execute the program. This includes Gradle- , Java- , Git- and XML-files. In this folder, you can find several other folders and files that are necessary to run the project. The only files that are not included in this folder, are files that are necessary for the project, but not relevant for the execution of the final application.

The repository includes a travis-file as well, which is used to check for compile-errors and notification purposes.

On top of all that, there is a folder that contains all protocols which have been documented during the process of this project.

A text-file which contains the MIT-license is also included in this repository.

In the end, there is the README-file, the user manual and the report.

To sum it up, the repository is divided into general files and files that are necessary for the program. The ones that are required to run the final program are located in “Project 7” whereas the general ones are located directly in the repository. Now, the files inside the folder “Project 7” need to be analyzed, because they are the core-files of the application.

# Classes

*Now, that the open questions have been clarified and the repository has been explained, it is necessary to talk about all the different classes, which are located in the “Project 7” folder.*

To begin with, the folder “Project 7”, can also be divided into sub-packages and files. The first file that needs explanation is called “.classpath” which handles errors, bug fixes and moves resources.

Next, the file “.gitignore” ignores certain files from the storage, which have been selected by the group.

The next two files are handling the building process for gradle. The two main files are called “.project” and “build.gradle”. The first file integrates gradle in eclipse and the second one builds it.

The final file is called “default.xml” which contains example-exercises for this project. It is used as a testing file for the application.

Now, only three folders are left. To start, the “Storage”-folder contains only one class, namely “Catalog.xml”. Later on, the teacher should put their exercises in this file.

The folder “.settings” merges the gui into the xmlParser and the “src/main” folder has all the different imports.

The “main” folder can also be divided into “java” and “gui”. In “gui”, there are all the resources located that are necessary to show the different scenes. It includes, icons, styles and different fxml-sheets which represent the different scenes.

On the other hand, the “java”-folder contains all java-files. To start, the “Main” file is necessary as a starting position for the program. It also initializes the GUI. Next, the files can be divided into four categories: “gui”, “models”, services” and “xmlParser”.

First, the “gui” folder is going to be described. The purpose of this folder is, to control and change the GUI of the program. It consists of controllers and views. There are different controllers and views for every part of the program. Hereby, the controller “controls” the corresponding fxml-file, which is located in the “resources/gui” folder. This includes the three phases the program is running through, as well as the overall menu. The controllers for the phases Red to Blue can be found in the sub-package called “cycle”.

Second, the “models” folder needs to be explained. Here, all the classes are located, that get created via the XML-Parser. Every exercise that gets created contains a class-, config-, test-, and trackingData-file.

Third, the “services” folder is going to be illustrated. In this folder, all the implemented features are located. To begin with, the BabystepsService.java file should implement the feature “Babysteps” into the program. Next, CompileService.java combines both, CompilerResult and TestResult, which has been given to the group. In the end, there is also a StorageService.java file which should handle storing the input, done by the student.

Fourth, the “xmlParser” folder contains all classes relevant for the XML-Parser.

To sum it up, the whole project has been divided into several minor packages in order to keep a good overview of all the different files which are necessary to make the program run. To close, the whole structure of the project and the functions of the different classes have been explained briefly.

# Critical Appraisal:

*This chapter contains some personal points that I want to address.*

To begin with, I would like to apology to whoever is going to work with this program. More specifically, I would like to talk about the class Game.java and explain, why it got almost 400 lines of code.

This was the first time, for me, working with javafx. So, when I started working on this project, I put everything into one class, out of simplicity. However, it got more and more and I continued putting most of the code in one class because when I tried to split it up in different classes, I got errors that I did not know how to fix them at that point of time.

Now, after working approximately 60 hours on this project, I would do many things differently. I probably would be able to split up the Game.java class in smaller classes.

Nevertheless, I think that I learned a lot during this project. By now, I know how to work with javafx and it was a good exercise to train recursive functions as well. On top of that, I think that it is clearly visible that I learned something during this project and that I wrote this project on my own.

On top of all that, it was my first time, working with Eclipse. At the beginning, it was hard to get to know it because javafx was not installed. So, Eclipse and I were off to a rough start. However, I quickly learned how to work with Eclipse and it made things much easier.

Even though it probably sounds like it, I do not try to come up with excuses why my Game.java class is such a mess. I just want to state that I learned a lot during these 1.5 weeks that I worked on this project and that it was a good experience (Even though I screamed at my PC sometimes).