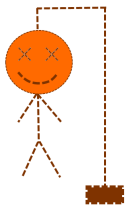
HANGMAN THE GAME

SOFTWARE DEVELOPMENT PROJECT

RUSLAN ABDULIN
written by: Ruslan Abdulin
personal code: ra222rt
email: ra222rt@student.lnu.se

GitHub Link: https://github.com/ruslanabdulin1985/ra222rt_1dv600

1 feb 2019 - 22 mar 2019



Contents

1 Revision History

Date	Version	Description	Author
17 mar 2019	2	revision before iteration 4	R.Abdulin

2 General Information

Project Summary	
Project name	**Project id**
HANGMAN THE GAME	ra222rt_1dv600
Project manager	**Main client**
Ruslan Abdulin	Old-fashion game fans
Key Stakeholders	
project director, end-customer	
Executive Summary	
Hangman The Game is a computer program representing a text game where player tries to guess as many secret words as possible earning scores. The purpose of the project is to create old-fashion style yet fancy and attractive game with low requirements	
GitHub page	
https://github.com/ruslanabdulin1985/ra222rt_1dv600	

3 Vision

The software project "Hangman" is a game in text based fashion. The basic idea of Hangman is that the player is going to guess a word by suggesting letter after letter.

3.1 Motivation

The main purpose of developing the game is to create attractive old-fashion style computer game which can be run on old computers. There is a huge community of gamers and collectors who buy old computers and try their best to make it working. Those people strive for new programs which can be run on these computers. Developing the game may interest the community because of new ideas in the well-known game.

3.2 The game process

Once started a new game the player is presented with the number of letters in the word but for every wrong guess the game is building a part of a man getting hanged. The player will loose if the hangman is completed.

3.3 Vision reflections

Vision is an idealistic view at the outcome of a project which visualize the main idea or main benefits of the project at the very early stage. The idea behind the vision may be both to engage people in the project and to show how important the project is. Also it is a basis for future discussions about the project. Vision doesn't focuses on specific details, technologies or people involved.

4 Project Plan

4.1 Introduction

The main purpose of the project is to develop a text-based guessing game with low requirements working in command prompt. The game should be designed in 4 iteration, be tested and well documented. Overall time to develop the game almost 2 months.

4.2 Justification

The purpose of the creation the game is to fulfill the need of good new games for the community of collectors of old vintage computers. As there is lack of modern programs for old computers, Hangman the game may interest the community. They may find fascinating the idea to play a brand new game on their old machines experiencing new features in obsolete environment.

4.3 Stakeholders

- project director - wants the project to be well-documentated and easy to undestand
- end-customers - community of old computers owners want the program to be easy to understand with competition features like different players and highscores.

4.4 Resources

1 programmer
1 poject manager
1 tester

1 laptop for programming, designing and keeping track of the project
1 computer for testing the result always on Internet connection
20 hours of the project manager to design a project
45 hours of the programmer to develop the game
15 hours of the tester to test the product
10 hours of the programmer to write documentation

4.5 Hard- and Software Requirements

To develop:
Processor Intel Core i5
Memory 6Gb of RAM
Solid State Drive 256Gb
Operating system Windows 7 or higher
Sofwtware:
Java 8 SDK
Libre Office
Notepad ++
Eclipse IDE

To run:
RAM: 128 MB
Disk space: 124 MB for JRE; 2 MB for Java Update
Processor: Minimum Pentium 2 266 MHz processor

4.6 Overall Project Schedule

- 1 - 6 feb - creating the project pan
- 6-8 feb - creating skeleton of code Basic functions: Text-based interface Test data base of words Picking up words and display the number of dashes Drawing the hangman Right-Wrong guessing Points calculating
- 8 feb - deadline for delivering release 1
- 9-12 feb - modelling new features using UML Features: Menu Registration Name checking Difficulty options High-Score database
- 12 feb - 17 feb Implementing the features
- 18 feb- 21 feb - Creating documentation
- 21 feb deadline for delivering release 2
- 21feb - 25feb - code review

- 26feb - 28feb - planing tests
- 29feb-6mar perform testing
- 7 mar - documenting tests
- 8 mar deadline for delivering release 3
- 12 mar - project plan review
- 15 mar - minor bugfix
- 18 mar -final testing
- 20 mar - complete documentation
- 22 mar - deadline for delivering release 4

4.7 Iterations

Plan for four iterations. There are 4 iteration during the development process. Iterations are connected to each other and should be done one by one. Skipping the iterations may lead to developing wrong product which means waste of sources.

4.7.1 Iteration 1

The first iteration in this project is planing and basic implementation. The plan should consist of General Information, Vision, Overall schedule, Iterations tasks and Time logs. Also it must have reflections about all that parts.

The outcome of this iteration is the plan and a basic version of the game stored in the GitHub account(
doc).

List of activities

A	B
action	estimate time
Collecting the information	
Write down a TimeLog	30 min
Write down a Vision	2 hours
Write down a Project plan	3 hours
Write down Risk Analisys	2 hours
Implement the basis of the game	5 hours
Check and Correct everything	2hours
Make a release	15 min

4.7.2 Iteration 2

The second iteration is modeling. The model of the project should be made using UML language. State machine and Class diagram creating must take place and Use Cases must be described. Also the implementation should start according to the model.

Outcome of this iteration is UML diagrams(State Machine Diagram and Class Diagram) and a pdf file with Use Cases stored in the GitHub account(
doc).

List of activities

Activity	Estimated Time
Study the book chapters 4-5	2 hours
Study the book chapters 6-7	2 hours
Create Use Case Model	4 hours
Draw State Machine Diagram	2 hours
Draw Class Diagram	2 hours
Implement The Game / Refactor	6 hours

4.7.3 Iteration 3

The third iteration is testing. The testing should be planned. Use cases must be derived into Test Cases. Test Cases must be executed. The entire code must be inspected commented and automaticaly tested. All this parts should be included into test report.

Outcome is a pdf file with the report stored in the GitHub Accont(
doc).

List of activities

Activity	Estimated Time
Implement The Game / Refactor	6 hours
Study the book chapters 8	2.5 hours
Make code Inspection	1.5 hours
Create Test Plan	2 hours
Perform Manual Test	4 hours
Write JUnit5 Auto test	2 hours
Run JUnit5 Auto test	15 min
Write Test report	2 hours

4.7.4 Iteration 4

The outcome of this iteration is the complete project. Reiterate the steps in iteration 1 – 3 for a set of new features but also remember to see the project as a whole, not only its parts.

Activity	Estimated Tir
Reiterate planning	3 hours
Reiterate UML Diagrams	3 hours
Finish implementing the game	5 hours
Reiterate testing	2 hours
Make a final release	1 hour

4.8 Scope, Constraints and Assumptions

Main features to be implemented:

- Main game proccess - guessing a word letter by letter - should be reflected in obvious way.
- Drawing the hangman in text mode

- Alphabet shows what the letters were tried- to make the guessing process visually simple
- Messages to user if a letter was correct or not – deleted after the 3rd iteration as unnecessary as it is obvious from the game process
- Main Menu with game options should be presented
- High score table - - moved from additional to main features after the second iteration

Additional features:

- Player registration - to personalize the game process
- Choosing a player – added after the second iteration
- Saving/Loading the game - to be able to left the game and return back
- Settings - diffuculty - - removed as low priority function after 3rd iteration
- Short word deffinition or topic - - removed as low priority function after 3rd iteration

Constraints:

- The process of the development is divided into 4 iterations.
- GitHub must be created and users dntoll, tobias-dv-lnu, DTAG-lnu, marietherese and kennyek must be added as coloborators.
- All the files should be prepared and release should be made before the deadline.
- All the code should be documented/commented as fully as possible.
- The structure should be simple and understandable. All the files should have descriptive names.
- Every iteration should contain a time-log.

- Diagrams and figures should go in files with descriptions and not to be stand-alone.
- The project - plan should be full and up-to-date
- all the releases should be made strict before the deadline.

Assumptions:

To run the game file HangmanCmd should be run in Java Virtual Machine environment. The program runs in standard way, NO additional options or libraries is required.

4.9 Project Plan Reflections

A Project Plan is a major document in plan-driven development. The plan sets goals and restrictions and marks up the resources available to the project, the work breakdown, and a schedule for carrying out activities. The details of project plans vary depending on the type of project and organization.

This project plan is basically what should be done within the project and why. The plan may change as new circumstances may occur or some risks couldn't be avoid. Therefore, sticking to the plan is the main priority.

5 Risk Analysis

Raw lists are sections 6.1 and 6.2 whereas the table which gather them presented in section 6.3

5.1 List of Risks

1. Key staff are ill at critical times in the project and aren't able to work on the project for a while - Low probability - Tolerable effects
2. Huge load on other projects which make it difficult to work on this project - Medium probability - Serious effects
3. The project or its requirements have changed requiring major design rework - Medium probability - Catastrophic effects
4. The laptop is broken or The information is lost - Low probability - - Catastrophic effects
5. The Internet isn't available - Low probability - Insignificant effects
6. It is difficult/time consuming to implement additional features - Medium probability - Tolerable effects
7. The time required to develop the software is underestimated - Medium probability - Serious effects
8. The code generated by software code generation tools is Inefficient - Low probability - Insignificant effects

5.2 Strategies

1. Plan and control activities in advance
2. Check the list of deadlines to estimate time better; Plan other activities in advance
3. Check the requirements, arrange tutoring sessions to make sure that everything is correct. Follow latest news in MyMoodle
4. Keep everything backed up, commit to GitHub after every significant change

5. There is free WiFi in the Library and in other building over Campus
6. Review the requirements and the list of additional features
7. Start implementing as soon as possible to understand better. Reduce the number of additional features
8. Rewrite code manually

5.3 table of risks

Risk	Probability	Effects	Strategy
Key staff are ill at critical times in the project and aren't able to work on the project for a while	Low	Tolerable	Plan and control activities in advance
Huge load on other projects which make it difficult to work on this project	Medium	Serious	Check the list of deadlines to estimate time better; Plan other activities in advance
The project or its requirements have changed requiring major design rework	Medium	Catastrophic	Check the requirements, arrange tutoring sessions to make sure that everything is correct. Follow latest news in MyMoodle
The laptop is broken or The information is lost	Low	Catastrophic	Keep everything backed up, commit to GitHub after every significant change
The Internet isn't available	Low	Insignificant	There is free WiFi in the Library and in other building over Campus
It is difficult\time consuming to implement additional features	Medium	Tolerable	Review the requirements and the list of additional features
The time required to develop the software is underestimated.	Medium	Serious	Start implementing as soon as possible to understand better. Reduce the number of additional features
The code generated by software code generation tools is Inefficient.	Low	Insignificant	Rewrite code manually

*the original table in doc folder

5.4 Risk Analysis Reflections

Risk management is one of the key process in designing software undertaken to identify, evaluate, and treat potential exposure to reduce the effects of damages or loss. Underestimated risks may lead to expenses rising, to delays or even to entire closure of the project.

Timelog

action	estimate time	realtime time
Collecting the information		
Write down a TimeLog	30 min	30 min
Write down a Vision	2 hours	3 hours
Write down a Project plan	3 hours	4 hours
Write down Risk Analisys	2 hours	1hour 30min
Implement the basis of the game	5 hours	5 hours 30 min
Check and Correct everything	2hours	3hours
Make a release	15 min	15 min
Total	14 h 45 min	17 h 45 min

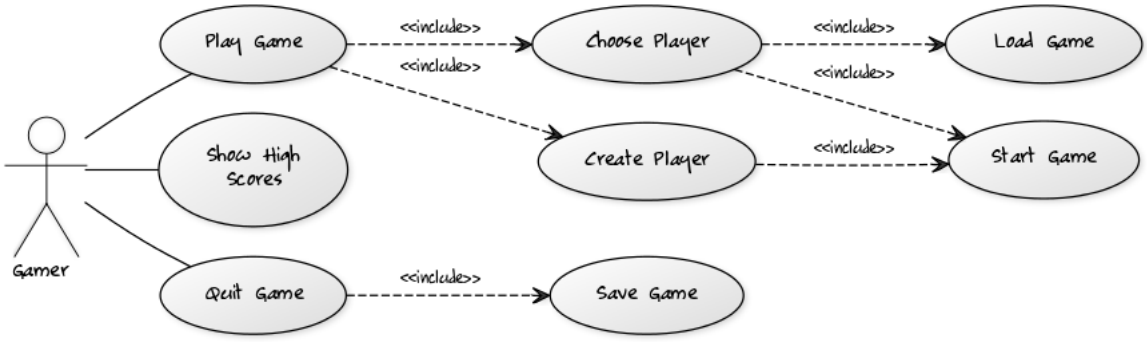
6 Modeling

Modeling is a necessary part of software development as it reflects what the product is going to be built. This project has three models stored in doc folder:

- Use Case Model
- Behavioral Stat Machine
- Class Model

6.1 Use Case Model

Use Case Model it is a model of how users interact with the system. It has Use Case Diagram which reflects different parts of iteration in a graphical way. The leads from user to activities reflects his/her intentions. These activities include other activities needed to be passed in order to fulfill the intended one.
Use Case Diagram:

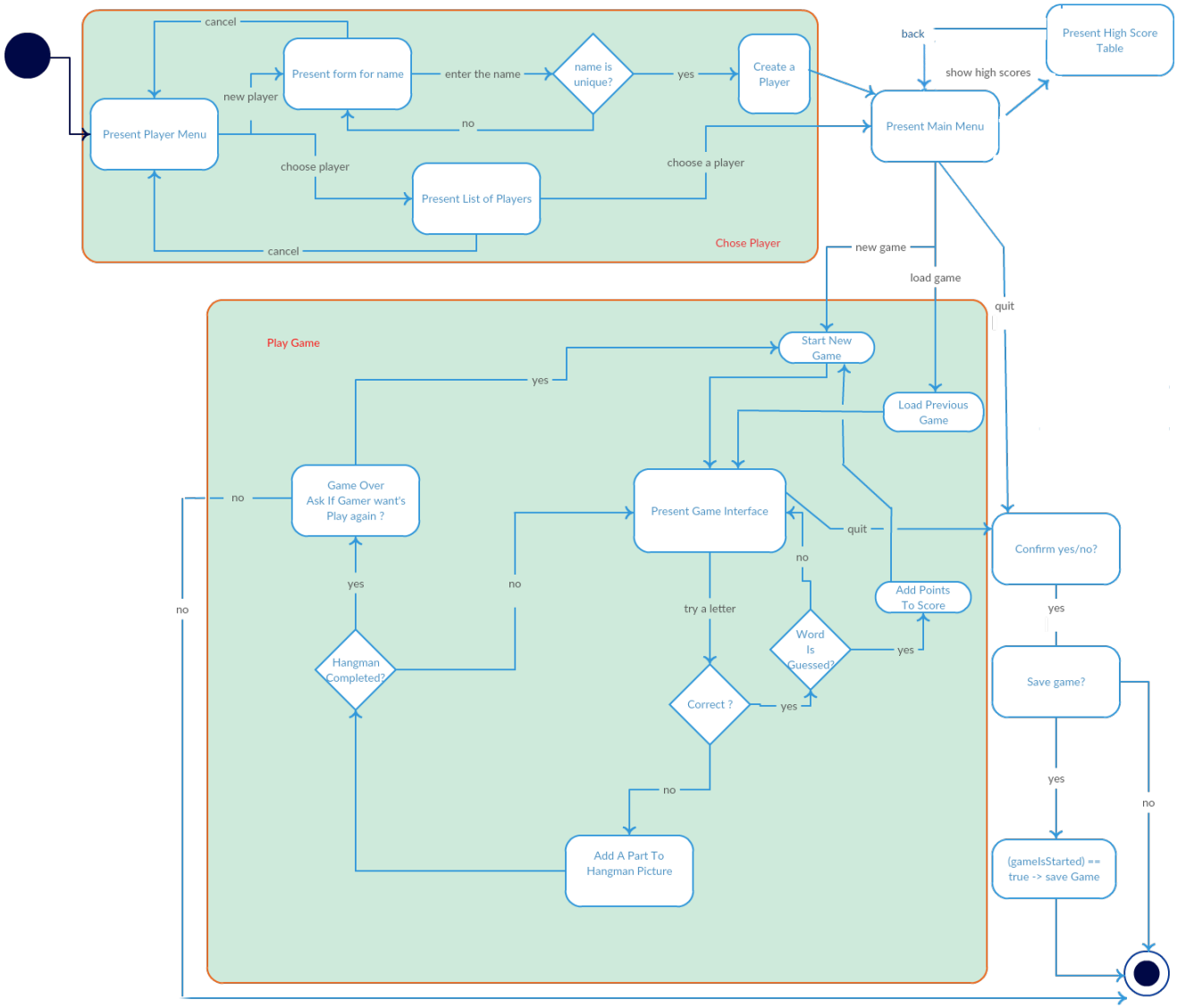


Also, the Model has descriptions for 8 Use Cases represented as scenarios of interaction between a user and the Program. These description are separated in a file USE CASE MODEL.pdf.

6.2 Behavioral State Machines

Behavioral State Machine is a model which shows behavior of a designed system. It presented as State Machine Diagram(SMD) divided into blocks. SMD reflects what exactly happens at certain point of a program running. The process starts at choosing player and proceeds to main menu where such option as Play Game, Load Game High Scores and Quit Game available.

State Machine Diagram:

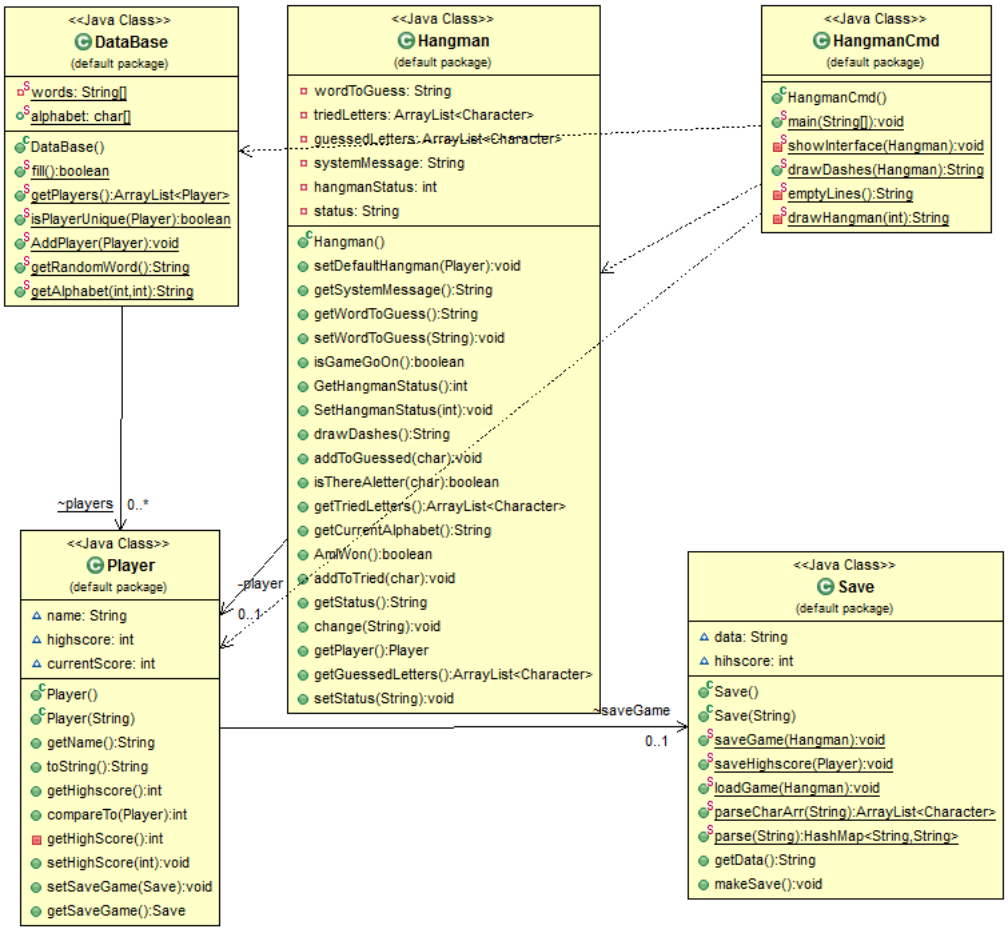


6.3 Class Model

Class Model illustrates how a program works from programmers point of view. It has Class Diagram which shows how the classes connected to each other and what methods they use for it. The program consists of 5 classes. 'HangmanCMD' is a class which

represents User Interface. Class 'Hangman' represents the main back end process of the game. It takes all the needed information from 'DataBase' class representing data base filled with information. Sometimes it refers to class 'Player' to get or set information about players available.

Class Diagram:



Timelog

Activity	Estimated Time	Real Time
Study the book chapters 4-5	2 hours	2.5 hours
Study the book chapters 6-7	2 hours	1.5 hours
Create Use Case Model	4 hours	8 hours
Draw State Machine Diagram	2 hours	3 hours
Draw Class Diagram	2 hours	1 hour
Implement The Game / Refactor	6 hours	8 hours
Total	18 hours	24 hours

Creating models is supposed to be rather simple process. However, deriving Use Cases turned out to be much more time consuming than it was expected. State Machines also took more time than expected. As the result time spent on the modeling was significantly underestimated.

7 Implementation and Run

Implementation part focuses on coding. The program is implemented according to the model. Run part has instructions how to run the code.

7.1 Implementation

To implement the project Java Programming Language is used. As a development platform Eclipse IDE is chosen. Classes are made one by one first to fulfill main requirements and then to implement additional features. During the implementation Object Oriented Approach is taken and encapsulation is obeyed. Code is written and commented as fully as possible. All the main features and nearly all additional features were implemented before 3rd iteration whereas Save/Load feature was implemented before 4th iteration. In order to make Save/Load possible all the activities were reiterated again. The code can be found in src folder.

7.2 Run

To run the game file HangmanCmd should be run in Java Virtual Machine environment. The program runs in standard way, NO additional options or libraries is required.

7.3 Timelog

- Estimated Implementation time: 19 hours
- Real Implementation time: 26 hours
- Comments: at some points programmer had been stacked with problems he couldn't solve instantly. It took some time to find the solution in the internet or come up with an idea. In fact implementation time was underestimated.

8 Testing

Testing is a process of software verification and validation. The aim of testing is to be confident in the software we are creating. Entire process of testing is described in detail in Test Report Updated.pdf in doc folder.

8.1 Interface Testing

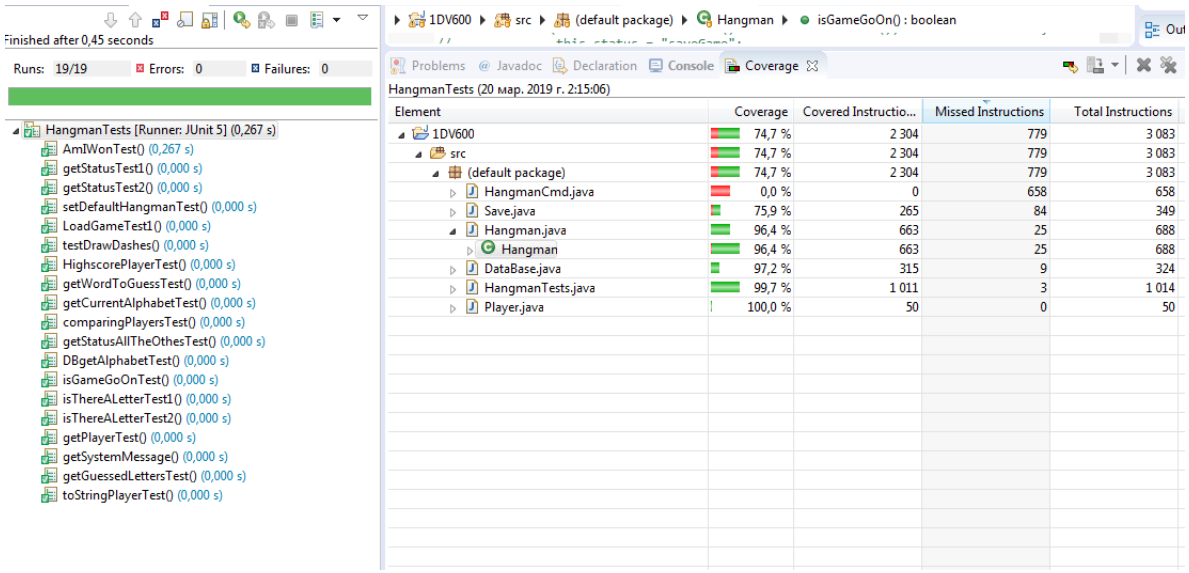
Test cases were derived from use cases and run. Initially, not all the tests has passed. However, refactoring and another successful testing took place before the 4th iteration. Table of manual testing:

Use Case ID	Use Case Name	TestCase ID	Status
UC1	Start Game	TC1.1	Passed
UC1	Start Game	TC1.2	Passed
UC2	Play Game	TC2.1	Passed
UC2	Play Game	TC2.2	Passed
COVERAGE & SUCCESS			4of 4 Passed / 100%

8.2 Source Code Testing

Source code was statically inspected and commented where needed. Also, automated regressive tests were created and run before the 3th and 4th releases to make sure that everything works as it supposed to work. A coverage tool was used during automated testing in order to make sure that tests cover almost all the code in 4 back end classes. Source code

testing found some bugs which was fixed before the 4th iteration. SourceCode test cover-



age:

8.3 Timelog

Activity	Estimated Time	Real Time
Implement The Game / Refactor	6 hours	8 hours
Study the book chapters 8	2.5 hours	3 hours
Make code Inspection	1.5 hours	3 hours
Create Test Plan	2 hours	1.5 hours
Perform Manual Test	4 hours	5 hours
Write JUnit5 Auto test	2 hours	9 hours
Run JUnit5 Auto test	15 min	15 min
Write Test report	2 hours	2.5 hours
Total	19 hours 15 min	32 hours 15 min

Comments: Writing JUnit automated tests took a way more time than it supposed to take. It is connected with unpredictably large amount of work to achieve more than 95 per cent of code coverage. In general, testing time was underestimated.

9 Final Release

Before the final release all the steps in development process were reiterated. Function Save/Load was planned, modeled, added and regressive tested. Bugs were fixed and code was completed. Documents were updated to reflect changes.

9.1 timelog

Activity	Estimated Ti	Real Time
Update plan report – Add SaveLoad Feature	3 hours	3 hours
Update UML Diagrams – Add SaveLoad Feature	3 hours	2 hours
Finish implementing the game - Add SaveLoad Feature	5 hours	4 hours
Test everything and update Testing report	2 hours	2 hours
Make a final release	1 hour	30 min
Total	14 hours	11 hours 30 m

Comments: Reiterating took less time than expected for two reasons: First almost everything was ready, only minor changes were required. Second - experience has been gained from previous iterations.

10 Conclusion

The Hangman Project has been finished successfully. Although the total development time was underestimated, other sources such as human sources and equipment were predicted correctly. Risks was analysed and reanalysed which helped to improve preventing strategies and complete the project in time. The development process went through all the iterations according to the Project plan and was successfully tested in the end. The final program meets its requirements and ready for deploying.