HANGMAN GAME

Leonardo Rochinha Pedro

Linnaeus University

16/03/2019

Contents

1. Revision History

Date	Version	Description	Author
8/02/2019	1.0	Initial version/sketch	Leonardo Pedro
18/02/2019	1.2	System Requirements	Leonardo Pedro
22/02/2019	2.0	Game code finished (Java)	Leonardo Pedro
5/03/2019	2.1	Game animations (JavaFx)	Leonardo Pedro
12/03/2019	2.2	“Hints” section added	Leonardo Pedro
15/03/2019	2.3	Clean up code and final Testing	Leonardo Pedro
22/03/2019	3.0	Final version	Leonardo Pedro

2. General Information

Project Summary	
Project name	Project ID
HangMan Game	lr222qp_1DV600
Project Manager	Main Client
Leonardo Rochinha Pedro	Children
Key stakeholders	
Leonardo Rochinha Pedro Tobias Andersson Gidlund Tobias Olsson Daniel Toll	
Executive Summary	
An easy game, known and classic, the hangman game demands of the player's attention and knowledge of the language, besides strategies to avoid defeat. And that's where the project is based: a simple, fast and fun game with both entertainment and learning purposes.	

3. Vision

This document works as a guideline for the application called Hangman game, which consists in a guessing game for one or more players where the computer randomly chooses a target string which must be guessed by the player and asks them to suggest letters that occur in the string.

Then the player will guess a letter. If that letter is in the word, then the program will write the letter at everyplace it appears and leaves the other spaces still unrevealed. If the letter isn't in the secret word, then the user loses one life and starts hanging the man.

After each guess, the program shows the user a version of the target string that replaces letters that have not been guessed with underscores (_) along with the lives the user still has left until the Man hangs, the program also shows the user which letters he already tried and were wrong in previous guesses, so the user doesn’t have to remember all the letters he tried before and it gets easier to exclude letters from the secret word.

The game is over either when the user wins by correctly guessing the desired string or loses by making more than 6 incorrect guesses, where in that case the Man hangs.

Reflection

The vision is a key part in this project because its used as a guideline to the whole procedure of the game, so every member responsible for the program has a clear idea of the game, and so nothing gets forgotten and no features of the game get missed understood when developing the application.

4. Project Plan

4.1 Introduction

The hangman game is a guessing game for one or more players where the user needs to find the secret word by suggesting letters, and if at the end of 7 wrong guesses the user loses.

4.2 Justification

This hangman application should be made to improve our coding skills in Java, for evaluation purposes since it's an assignment and finally for the user entertainment.

4.3 Stakeholders

- Leonardo Rochinha Pedro
- Tobias Andersson Gidlund
- Tobias Olsson
- Daniel Toll

4.4 Resources

JDK11, Stackoverflow, IntelliJ.

4.5 Hard- and Software Requirements

Specifications	Developed	Required
Operating System	Windows 10, 64-bit	Windows XP/Vista/7/8/9/10 (32/64 bit)
Processor	Intel® Core™ i7-8750H	Intel® Core™ i3 or above
Memory RAM	8GB	2048MB
Graphic Card	Nvidia GeForce GTX 1050	1GB

4.6 Overall Project Schedule

Iteration 1 - Friday, 8 February 2019, 23:55

Iteration 2 - Thursday, 21 February 2019, 12:00

Iteration 3 - Friday, 8 March 2019, 23:55

Iteration 4 - Friday, 8 February 2019, 13:55

4.7 Scope Constrains and Assumptions

Scope: (What is implemented)

- GUI
- A place to write the guesses
- The wrong guesses should be showed on the screen
- A hint with the topic of the secret word should be presented on the screen.
- A graphical representation of the Hangman. The graphical representation should change every time that the player guesses a wrong letter.
- Keep track of the wrong guesses as lives go down as the player gets wrong guesses.
- A pop-up window with the result (win/lose).
- That same pop-up with the options to play again or exit the game.

Out of Scope: (Features that didn't make into the final version)

- Multiplayer Mode.
- ".exe" version of the game.

Constrains:

- The Game runs into an IDE.
- Only 1 player can play.

Assumptions:(Instructions that we assume the player should run before play the game)

The player needs to do all the things below to be able to play the game:

- Download all the files for the game from GitHub.
- Put all these files into an IDE.
- Run the main class.

5. Iterations

All iterations presented below are the core parts of this project, as it shows the four stages of this game development.

5.1 Iteration 1

In this first iteration of the game, it was started by planning what features we would include in the game, start thinking on documentation and start with a sketch of skeleton code.

From this iteration, the user can already try the game, as we present a raw game concept, coded in java and so it can be tested by running it in console using IntelliJ or Eclipse.

5.2 Iteration 2

The second iteration brings more detail than the previous iteration, where is presented UML diagrams and more in-depth documentation.

This iteration focuses on the Use Case Model, Diagram class, State Machine Diagram and a Fully Dressed Case for the play game.

5.3 Iteration 3

This is iteration is crucial to ensure that the application is running without any problems, where this is focused on the testing part of the program, where all the testing procedure is planned and executed.

Here is presented a clear version of the code where the code itself is organized and improved with methods so the Junit testing becomes more efficient.

In this iteration GUI is also introduced where the game starts having visual animations to become the game more interactive and more fun for the player.

This iteration focuses on Test planning, Manual Testing and Junit testing.

5.4 Iteration 4

This is the most wanted iteration where all the pieces come together, and the final game is done.

In this iteration all the documentation was completed all the updates of new features added into the game were updated and tested properly to deliver the best project possible.

6. Risk Analysis

To avoid getting potential issues that could negatively impact this project in this section I present possible risks that can occur in the making of this game.

6.1 List of risks

This table takes in consideration possible risks relating personal and technical problems.

Risk	Probability	Impact
Getting stuck	Easy	Slow down the pace of development
Procrastinating / Sloppiness	Easy	Deliver a poor version of the project
GitHub /Upload	Medium	Slow down the pace of development
Sickness	Difficult	Slow down the pace of development
Losing the whole project file	Very Difficult	Start the whole project again

6.2 Strategies

Risk	Strategy
Getting stuck	Search for help within your friend group + Teachers on slack + Google
Procrastinating / Sloppiness	Make sure to make a work schedule and choose a workspace that don't allow for distractions
GitHub /Upload	Submit all your work with extra time before the deadline, so you still have time to find a solution if any issue appears. (Teachers + Friends + Google)
Sickness	Healthy lifestyle, if you get sick take medicine and try to get back on track.
Losing the whole project file	Submit all your iterations on GitHub/Moodle and always save your work.

7. Time Log

Iteration 1

Task to do	Time Estimated (min)	Time Taken
Planning	90	90
Implementation	90	60

Iteration 2

Task to do	Time Estimated (min)	Time Taken
Class Diagram	20	10
Fully Dressed Case	60	60
Use Case Diagram	45	30
State Machine	45	90
Implementation	60	70

Iteration 3

Task to do	Time Estimated (min)	Time Taken
Test Plan	25	40
Manual Testing	30	30
JUnit Testing	60	120
GUI	180	240

Iteration 4

Task to do	Time Estimated (min)	Time Taken
Last features	120	60
GUI	60	120
Final Testing	45	20