SOFTWARE DEVELOPMENT
Project plan project Hangman

Marcus Gärdskog Hill Start date : 24-01-2019

Contents Page

Revision History

Date	Version	Description	Author
02/03/19	1	First draft	Marcus Gärdskog Hil
02/11/19	2	Addin use case/ scenarios	Marcus Gärdskog Hill
02/12/19	3	Moved "make system work with minimal features to week 8 according to deadline from school	Marcus Gädskog Hill
02/12/19	4	Added to timelog and expanded use case scenarios	Marcus Gädskog Hill
02/22/19	5	Added fully dressed usecase, class diagram , use case diagram and state machine to document for handin	Marcus Gädskog Hill

2 General Information

Project Summary

Project name ID	Java Hangman
Project Manager	Marcus Gärdskog Hill
Main client	Lnu.se 1dv600
Key Stakeholders	Marcus Gärdskog Hill
Executive Summary	

In this project I will create a console/text based Hangman game in Java. The application will use the java reserved keywords as the default words for the game. The default words will be read from the site https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html.
If there is time there will be an added option to read the words from a wikipedia site selected by the user.

Hangman is a simple guessing game where the user guesses one letter at the time and for every wrong guess, (guessing a letter that is not part of the word) the image of the man getting hanged gets one more piece added to it. If the user cannot guess the complete word before the image of the man is full, the user loses.
https://en.wikipedia.org/wiki/Hangman_(game)

The system will be build as a part of a software engineering course (1dv600) at Linné university in Kalmar, Sweden.

3 Vision

1 Introduction

1.1 Overview

The system is a textbased version of the classic game Hangman written in Java, they system will read the words for the game from the web and present the gameplay in the console. Version controll will be done by git bash. There will be no other GUI then the terminal inside the java IDE

1.2 Purpose

The main purpose of this project is to create a system that will fill the reqirements for a Hangman game and be able to read information from webpages using Java. The purpose is also for me to learn how to plan a software engineering project properly for the course 1dv600 at the Linné univeristy in Kalmar, Sweden.

1.3 System usage

The system will only be used inside ecplipse or a similar IDE and only for testing by me and the teachers of this course. No future use is anticipated and the system will not be maintenanced once the course is complete.

1.4 System requirements

Functional requriments, Feature	**Functional requriments, Rationale**
Enter a username	The user need to enter a username so that highscore can be keept. (If time)
User input	The system will scan input from the user to make the game interactive.
Text output to console	The game will be presented in text in the console.
Reading from the web	To extend the number of words and easily add more the system will be able to read information from the web and save it as words for the game.
Game play	The system will ask the user to guess a letter and check if this letter is in the current word. If not the hangman image will be one more step to completion and if the user is correct, the letter will be saved in the correct spot(s). Example: Wrong guesses: 3 Right guesses: 2 ___ | o | /|\ R . . E . . E .

Amount of Guesses allowed	9 wrong guesses is the max.
Error handling	If the user enters something other than aA – zZ.
Make the "read from the web" class reusable	I want to be able to reuse the class reading from the web in other projects
Non functional requirments, Feature	**Non functional requirments, Rational**
Speed, Performance	The user shall not have to wait for more than 10 seconds for the game to load or fetch the words from the web.
Operational	Any Java EDI is needed to run the system
Modifiability	The system shall be devided into minium 3 classes to make modifying easy. One class reading from the web, one or more classes with game logic and one class containing main.
Security	The system will be designed to not have any sensitive information.
Usability	Anyone with basic Java knowledge shall be able to run the system.
Platform	Any system that can run a Java IDE or a working modern browser shall be able to run the system localy or at a online IDE
Time	The system will have to be done in 7 weeks when the course deadline is up.

1.5 User requirements

User requirement	Supporting feature
The user will need to have a basic knowledge of eclipse or other Java EDI and the language of Java in order run the application	There is no supporting feature from the development team to teach the user Java

4 project plan

4.1 Introduction

This project plan spans for 7 weeks and will produce a textbased version of the classic game Hangman in Java. This project is a part of course 1dv600 at linné university, Kalmar, Sweden. The first part of the system to be built will be the class that read words from the web, this function is something I don't know how to do so it has the highest risk. Once this is done the next class will be the class holding the game logic wich I feel confident I can handle without problems. The testing will be carried out through out the whole process but intensify in the last few weeks.

4.2 Justification

This system will be build because it is required to pass the software engineering course. The system will be able to read from the web because this is something I want to learn how to do.

4.3 Stakeholders

The key stakeholder for this project is me, Marcus and I will me planning an programming this system. The teacher are:

Tobias Andersson Gidlund

Daniel Toll

Tobias Olsson

4.4 Resources

Computer, Eclipse and internet connection

4.5 Scope, constrains and assumptions

The system will be textbased single player with a highscore and it will read the words for the gameplay from the web. Constrains for the system is that it should be text based according to the instructions and I have 7 weeks to finish it. I assume that I have the ability to read the words from the web, if not I will work around this problem as suggested in the risk analysis. The game will be played directly inside the IDE so the user will need basic knowledge about Java and some IDE.

4.6 Hard-and software requirements

The system needs a Java IDE to run and a computer capable of running such IDE.

4.7 Overall Project Schedule

Week	Activity
5 and 6	Create and hand in the first version of the project plan and skelleton code with as much information as possible during the first increment.
7	Learn UML and implement it in the project plan where it is applicable
7	Develop class to read words from the web
7	Design the game with minimal features.
9	Add highscore, and 1 dd username
10	develop testplan
10	Start testing the system.

10	Finish the project
11	Finilize and present the Git-repo
12	Hand in the project

5 Iterations

5.1 Iteration 1:

When:	To do:	Time Estimates:	Recources:
week 5-6	**Create the first draft of the project plan.**	8 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 5-6	**Create Skeleton Code**	1 hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 5-6	**Hand in first draft and skeleton code**	15 minutes	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)

5.2 Iteration 2:

When:	To do:	Time Estimates:	Recources:
week 7	**Add to project plan what is discovered to be missing from Iteration 1**	5 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 7-8	**Create class that reads the words from the web and save them in an ArrayList.**	10 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
Week 7-8 **(MOVED TO ITERATION 3)**	**Develop test plan**	5 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 7-8	**Add UML to project plan**	5 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
Week 7	**Make game work whit minimum features.**	10 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)

5.3 Iteration 3:

When:	To do:	Time Estimates:	Recources:
week 9-10	**Add to project plan what is discovered to be missing from Iteration 2**	3 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 9-10	**Create the user ID class whit minimum features.**	2 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 9-10	**Create the highscore class whit minimum features.**	10 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
Week 9-10	**Develop test plan**	5 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)

5.4 Iteration 4:

When:	To do:	Time Estimates:	Recources:
week 11	**Add to project plan what is discovered to be missing from Iteration 3**	2 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 11-12	**Add extended Highscore**	3 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 11-12	**Add timer**	3 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 12-13	**Add option read words from different site**	3 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 12-13	**Comprehensive testing**	12 Hours	Eclipse,Oracle website, Internet connection, lectures, book

			Sommerville (2015)

6 Risk Analysis

6.1 Risks and strategies

Project risks:	**Bussines risks:**	**Product risks:**
I might have to work overtime during this course and this would put me behind schedual Probability: 5 impact: delays Avoidance strategy: Tell boss that I rather not work extra during this project. **Contingency plan: Work on project during work hours.**	**I might burnout** Probability: 1 impact: catastrophic. Avoidance strategy: start work in time and get lots of sleep.	**Reading words from the web might be harder than anticipated.** Probability: 6 Impact: Delays or change of specifications. Contingency plan: manually program the requried words or read in different words.
The time estimates might be off and therefore some extra features will not be implemented. Probability: 5 impact: less complete system		**Game logic might not be as easy as I anticipate.** Probability: 2 impact: Delays and possible F in course.

7 Timelog

To do:	Estimated Time	Accual time
Create first draft of project plan and skeleton code. 19-01-24 / 19-02-03	9 hours	12 hours
Refine the project plan. 19-02-06	1 hours	1 hours
Read chapter 6 in Patternsbook	1 hour	5 Hours
Create text based use cases	2 Hours	2 hours
Create basic use case diagram	8 hours	3 hours
Expand the use case model	3 hours	2 hours
Fully dressed use case	3 hours	2 hours
Model behavior with UML state machine	6 hours	2 hours
Play game state machine	2 hours	0.5 hours
Implement basic version	8 hours	10 hours
Create Class diagrams	6 hours	4 hours

8 Files related to project:

File:	Description:
ProjectPlanHangman.odt	Full project plan
Full project plan PDF VersionProjectPlanHangman.PDF	
HangmanLogic.java	Class containing Game logic
ReadWordsFromWeb.java	Class reading words from the web
HangmanMain.java	Hangman program Main class

9 Fully dressed use case diagram

Use case: Playing the game

Main Scenario:

Pre-condition: The user has an java IDE open with the code for this system.

1.User wants to play the game.
2.System displays how long the word is . . . and ask user to guess a letter.
3.User guesses letter B.
4.System tells user where in the word B appears "B . ."
5.User makes second guess of C.
6.System tells User there is no C in the word, creates the top of the Hangman and saves the guess.
7.User guesses letter I.
8.System tells user where in the word I appears "BI ." .
9.User guesses letter T.
10.System tells user where in the word T appears. And lets the user know he has won the game with a total of four guesses he has guessed the word BIT.
11.User press enter to get back to main menu.

Post-condition: The game is back at its starting point and user can choose to start again, change adress to fetch words from or quit game.

Extentions:
Getting words from wikipedia.
Post condition: game starts with words from selected site.
1.1 User wants to fetch words from specific wikipedia page.

1.User types in the specific english wikipedia adress from where he wants to fetch the words
2.System cannot fetch words from site, tells the user and ask him to try again.
3.User chooses another site.
4.System fetches words from site and starts game.

2.1 user wants to quit.

1. System asks user if he wants to terminate the application.
1.1 User says yes.
1.2 System terminates.
2.1 User says no.
2.2 System continues as nothing happened.

3.1 User inputs other character than letter aA-zZ.

1. System let user know that the guess have to be a letter but does not count the guess.

5.1 User wants to quit

1. System asks user if he wants to quit.
1.1 User says yes.
1.2 System terminates
2.1 User says no.

2.1 System continues as nothing happened

7.1 User wants to quit

1. System asks user if he wants to quit.

1.1 User says yes.

1.2 System terminates

2.1 User says no.

2.1 System continues as nothing happened

9.1 User wants to quit

1. System asks user if he wants to quit.

1.1 User says yes.

1.2 System terminates

2.1 User says no.

2.1 System continues as nothing happened

10.1 User guesses wrong 9 times until the Hangman is complete.

1. System tells user he has lost, display the correct word and sends user back to main menu.

10.2 User guesses a letter that is already in the wrong guesses list

1. System counts guess as a wrong guess and adds the letter again to the wrong guesses list.

11.1 User wants to quit

1. System asks user if he wants to quit.

1.1 User says yes.

1.2 System terminates

2.1 User says no.

2.1 System continues as nothing happened.

11.2 User want to play with the java reserved keywords.

1.System fetches the words from the oracle website.

1.1 System fails to fetch words and tells user to try again.

2. User want to play with specific wikipedia page.

2.1 System fetches words from the site and starts the game.

2.2 System fails to fetch words and tells the user to try again.

11.3 User wants to quit

1. System asks user if he wants to quit.

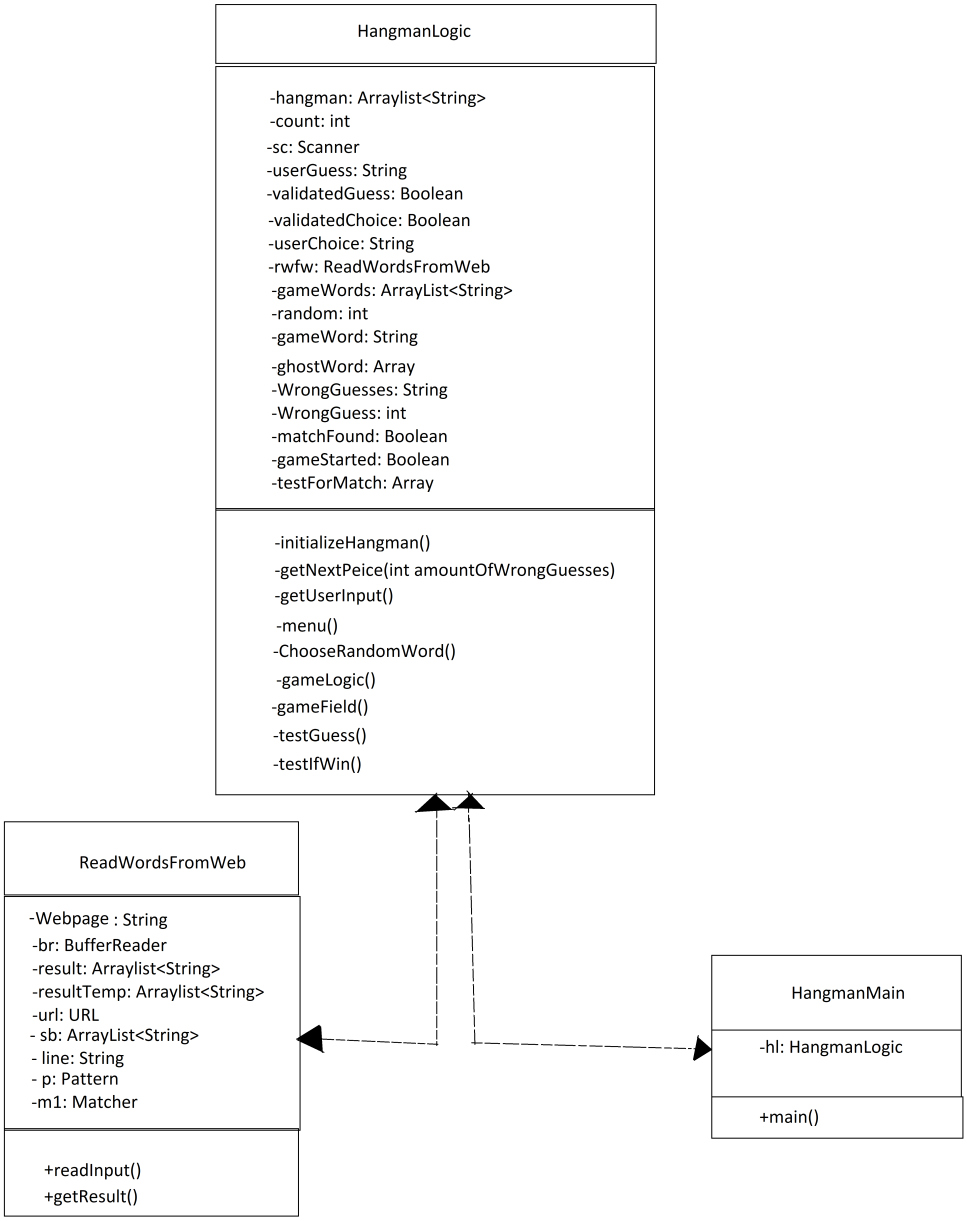
1.1 User says yes.

1.2 System terminates

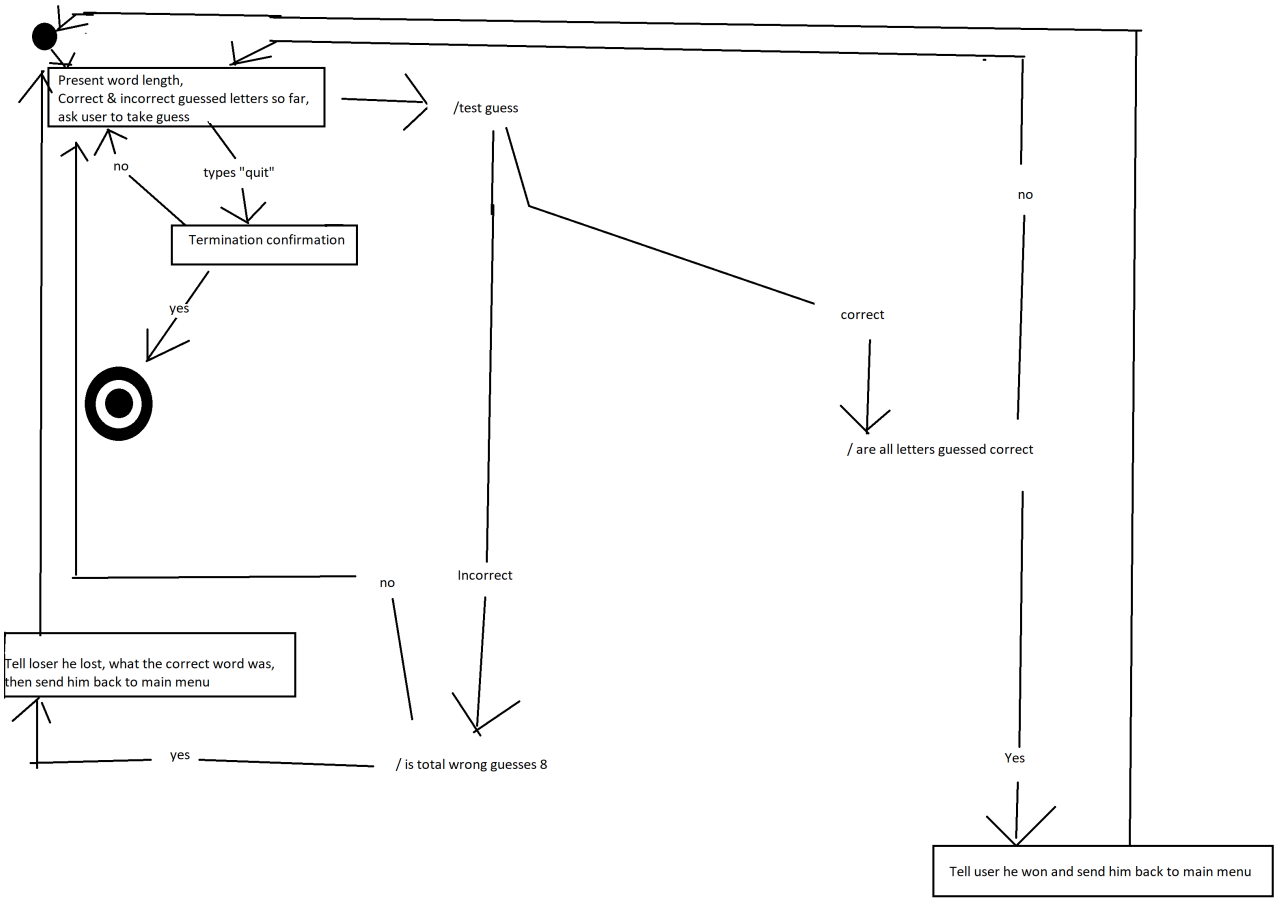
2.1 User says no.

2.1 System continues as nothing happened.

10 Class Diagram



11 State machine (Play)



12 Use case Diagram