SOFTWARE DEVELOPMENT
Project plan project Hangman

Marcus Gärdskog Hill Start date : 24-01-2019

Contents Page

1 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 hand in	Marcus Gädskog Hill
03/10/19	6	Updated: General Information Vision Timelog according to Kenny Ek's comments	Marcus Gärdskog Hill
03/11/19	7	Updated: Project plan Risk Analysis	Marcus Gärdskog Hill
03/20/19	8	Updated project plan with highscore/nickname	Marcus Gärdskog Hill
03/21/19	9	Updated UML and use cases according to feedback from Tobias Ohlsson	

2 General Information

Project Summary

Project name ID	Java Hangman
Project Manager	Marcus Gärdskog Hill
Main client	Marcus Gärdskog hill
Key Stakeholders	Marcus Gärdskog Hill (Developer),Tobias Andersson Gidlund (Teacher/end user), Daniel Toll(Teacher/end user), Tobias Olsson (Teacher/end user) Kenny Ek (Teacher assistant/end user), Therese Forsberg(Teachers assistant/end user).
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

I will create a simple, textbased version of the classic game Hangman in Java. The game will be console/textbased and thus it will be played inside any Java IDE. No other UI will be provided so the end-user will need knowledge of Java.

The game will fetch the words for the game by scraping websites and then using these words for the game. The user will be able to choose what Wikipedia page to scrape from or alternativly play the game with the default words which are the Java reserved keywords. The Java reserved keywords gets scraped straight from the oracle website so if the words are updated the game will reflect this.

The game is not meant for the public and will only ever be played by the key stakeholder: Marcus and possibly the other stakeholders Tobias Andersson Gidlund, Daniel Toll, Tobias Olsson, Kenny Ek, Therese Forsberg.

The whole process will be documented and I will track time and what is done in each iteration inside this document. The system will be build iterativly during 9 weeks, starting at week 4 and finishing with a fully functional system in week 12.

3.1 Reflections

The main function of the system is very easy to build and there is no doubt in my mind that I can get it working but the additional functions are dependent on things outside my control and experience, for example: Oracle and wikipedia needs to allow me to scrape their sites. I dont know what policies these sites have but I expect that there will be no problem and I won´t have to deal with "429" errors or similar problems. Worst case scenario I will hard code the words in the game. I'm also very inexperienced documenting a software project in this way and I suspect that this will be the biggest challenge for the weeks to come.

4 project plan

4.1 **Introduction**

This project will be executed iteratively over a 9 week period to produce a hangman game in Java. The final product should be a playable, textbased version of the game hangman played inside any Java IDE. Week 5 and 6 will focus on creating a first draft of the project plan and the skeleton code for the game. Week 7 and 8 will produce a minimal viable product and a test plan for this. Week 9 and 10 will be focused on fine tuning the project, documentation and testing. The project will be finalized during the three last weeks 11 and 12 and ready for "production".

4.2 Justification

This system will be built because it will be a good learning experience for me to code, document and test a system in a somewhat professional manner. 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 be planning and developing this system The other stakeholders are:

Tobias Andersson Gidlund (Teacher / end user)

Daniel Toll (Teacher / end user)

Tobias Olsson (Teacher / end user)

Kenny Ek (Teachers assistant / end user)

Therese Forsberg (Teachers assistant / end user)

4.4 Resources and requirements:

Required Staff:

Developer: Marcus Gärdskog Hill
Project Manager: Marcus Gärdskog Hill
Tester: Marcus Gärdskog Hill

Avaliable Time:

9 weeks

Required hardware:

1 development/Test system.
Minimum 1GB ram
Minimum 20 MB free diskspace
Minimum 1,5GHz CPU

Required software:

Eclipse or similar IDE.
Git bash
Github Account
Slack
Open office / Windows office
Windows, MacOS or linux

Connection:

Internet connection > 1mb/s.

Theory:

Hortsmann Big Java.(Book)
Lectures: all lectures from 1dv600
Finance: 0 sek.

Communication:

Dialog on slack with teacher assistants (Kenny Ek and Therese Forsberg) and teachers, (Tobias Andersson Gidlund, Daniel Toll and Tobias Olsson)

4.5 Scope

This project will identify requirements, design, develop and document the hangman project consisting of a basic but playable version of the hangman game. There will be no UI other than the console inside the IDE and the user will only have text and no graphics to aid the gameplay. If there is time at the end of the 9 week period, extra features will be added, such as highscore and nickname.

The goal of this project is thus to create a functional and playable software that satisfies the requirements stated above. All activities will be documentet inside this document and updated throughout the 9 week period.

Tasks needing to be performed to create the system are identified in this project plan with information of what needs to get done during each iteration and the lenght of each iteration.

4.6 Constrains

Deadline for the project is 22 of march 2019 at 00:00 with possible extended deadlines 19 of April 2019 at 00:00 and 23 of August 2019 at 00:00.
The software created in this project is intendent for personal educational purposes, focusing on documentation, design and testing in a iterativ manner. The software will therefore not be built with the wider market or multiple customers in mind. The implementation will be a version playable for someone who got at least a litte bit of java knowledge. The whole project got a 9 week window from start to finish. (Week 4 to Week 13).

4.7 Assumptions

A few assumptions where made before the development process got started. I assume I have the ability to scrape websites for words to be used inside the game and I assume that the websites will let me scrape them. I have limited experience with webscraping but ways around this problem is descriped in the risk analysis.

4.8 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 9 weeks when the course deadline is up.

Get name and score of best game	The user should be able to retrive the name and the score of the best game

4.9 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

4.10 Over all schedual

Week	Activity	Dead line
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.	To hand in the first iteration. First try: 2019/02/08, 23:55 Second try:2019/04/19, 23:55 Third try: 2019-08-23, 23:55
7	Learn UML and implement it in the project plan where it is applicable	To hand in the second iteration. First try: 2019/02/22, 15:99 TBA
7	Develop class to read words from the web	To hand in the second iteration. First try: 2019/02/22, 15:99 TBA
7	Design the game with minimal features.	To hand in the second iteration. First try: 2019/02/22, 15:99 TBA
9	develop testplan	To hand in the third iteration. First try: 2019/03/08, 23:53 TBA
10	Start testing the system.	To hand in the third iteration. First try: 2019/03/08, 23:53 TBA
12	Add highscore, and nickname	To hand in the forth iteration. First try: 2019/03/22, 00:00 Second try: 2019/04/19, 00:00 Third try: 2019/08/23, 00:00
12	Add highscore and nickname to testplan	To hand in the forth iteration. First try: 2019/03/22, 00:00 Second try: 2019/04/19, 00:00 Third try: 2019/08/23, 00:00
12	Create tests for highscore and untested methods	To hand in the forth iteration. First try: 2019/03/22, 00:00 Second try: 2019/04/19, 00:00

		Third try: 2019/08/23, 00:00
12	Finish the project	To hand in the forth iteration. First try: 2019/03/22, 00:00 Second try: 2019/04/19, 00:00 Third try: 2019/08/23, 00:00
12	Finilize and present the Git-repo	To hand in the forth iteration. First try: 2019/03/22, 00:00 Second try: 2019/04/19, 00:00 Third try: 2019/08/23, 00:00
12	Hand in the project	To hand in the forth iteration. First try: 2019/03/22, 00:00 Second try: 2019/04/19, 00:00 Third try: 2019/08/23, 00:00

4.11 Reflections

Every little choice i make constrains and narrows the scope of the project in a intresting way. I did not think about this to much beforehand but it has become an obvious biproduct of any choice made inside a software project, or any project for that matter. This project is heavily constrained by the nature of the instructions and the narrow timeframe given. I realized while writing the assumptions part of the document that I had made some pretty bold assumptions in my vision without any real knowledge of how to scrape websites for information. If this project in any way was on comission this would have been quite irresponsible without proper testing or knowledge about web scraping.

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(have not got answer from teacher in time, moved until response is given)**	3 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
week 9-10	**Create the user ID class whit minimum features. (Class merged with highscoreclass in iteration 4)**	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)
Week 10	**Hand in testplan**	5 minutes	Internet connection

5.4 Iteration 4:

When:	**To do:**	**Time Estimates:**	**Recources:**
19/03/10	**Update project plan with new highscore class to include nickname**	2 hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
19/03/11	**Update test plan with addhighscore test, getHighScore test and checkNickname test.**	2 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
19/03/12	**Update Class diagram, use case diagram and statemachine with highscore and**	2 hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)

	nickname.		
19/03/12	**implement new highscore class with nickname**	2 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
19/03/12	**Write tests for Highscore and nickname**	2 hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
19/03/12	**Update testplan with code and coverage for highscore/nickname**	1 hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
19/03/13	**Update project plan according to feedback from iteration 1**	4 hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
19/03/19	**Proof read the document, instructions and slack and see if anything is missing**	5 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
19/03/21	**Finialize the project. Save as PDF, hand in.**	12 Hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)
19/03/16 – 19/03/22	**Add what ever turns out to be missing from iteration 2 (When ever it gets returned from teachers)**	?? hours	Eclipse,Oracle website, Internet connection, lectures, book Sommerville (2015)

Reflections Iteration 4:

The last iteration changed alot since the first draft, I have made it way more fined grained according to comments from my first iteration. I don't have the exact dates saved from the previous iterations so instead och just making those dates up I leave them as is although I know they should also be more fine grained. As my final feature I choose to include nickname in the highscore class and rewrite and implement the class in a better way using hashmap. I choose to write automatic Junit tests for the highscore and nickname tests since they have simple and testable outputs and inputs it is easy to automate.

The process I followed to implement the last feautre is described in the iteration above. First I updated the project plan to include Highscore and nickname as the same class since this seemed like a better design decison, no real reason to separate these two things. I did not have to create any new UML but I added the highscore class to the state machine for play game since it is a part of the state at the end of a winning game, I also added it as a use case in the use case diagram and added the class to the class diagram.

Then I updated the testplan, implemented the class and the created the tests for the class and added printscreens of code and coverage to the testplan since they replaces older tests that no longer was in use.

6 Risk Analysis

6.1 Risk impact

Area affected	C	B	A
Schedule	Delay > 0 days	Delay > 2 days	Delay > 5 days
Scope	Barley noticeable decreace in scope	Minor areas of scope affected	Major scope reduction.
Quality	Barley noticeable decreace in quality	Quality reduction does affect not functionality for end user	Quality reduction noticeable for end user

6.2 Probable risks

Event	Probability	Impact rating	Area affected	Avoidance strategy	Contingency plan
I might have to work overtime during this course and this would put me behind schedual	50.00%	A	Schedual: Heavy delays to be expected. Quality: Cutting corners to make project in less time.	Tell boss that I rather not work	Work on project during work hours.
I might burnout	5.00%	A	Schedule: Project might be put on hold. Scope: Scope might have to become MVP to get passing grade. Quality: Quality might also be reduced to MVP for	Start work on time. Don´t procrastinate and get lots of sleep	Finish the course on the last retake in august

			passing grade		
Reading words from the web might be harder than anticipated.	80.00%	B	Quality: Words will be hardcoded and not read from webb. End user wont know much of a difference	Start with this class to make sure that it can be done. And if not it can be thrown away early with no code depending on it.	Hardcode the words into the system.
The time estimates might be off and therefore some extra features will not be implemented.	50.00%	B	Schedule: If estimates is of the whole schedual will be off and need rework. Scope: Scope might have to be reduced to save time. Quality: Corner might have to be cut to save time	Take extra care on time estimates and add some extra slack just in case.	If time estimates are to off, aim to get done at the second or third retake.
Game logic might not be as easy as I anticipate.	20.00%	B	Schedule: It might take longer than anticipated because of some factor in the code I did not take into concideration but the logic will still get produced with slight delays.	Plan properly	Put down the extra work required to get the logic working.

6.3 Reflections

This project is so small and still there is several risks to take into account and I really see whats to gain from doing so. Risk analysis, contingency plans and avoidance strategies seems like a very useful tool in software projects and life in general. Many of my minor private projects have run into some kind of wall because I did not spend time to think about the different risks that might be associated with what ever I'm trying to do or what I'm supposed to do if some kind of problem turn up (as they always do). The risk analysis lends a peace of mind that I am equipped with some extra tools in case/when problems show up. They help me to keep the problems in mind and steer away from them whenever possible. I feel like this will really help my programming practices going forward.

7 Timelog

To do:	Estimated Time	Accual time	Discussion
Create first draft of project plan and skeleton code. 19-01-24 / 19-02-03	9 hours	12 hours	I underestimated the amount of reading and searching that would be required.
Refine the project plan. 19-02-06	1 hours	1 hours	Though 1 hour would be much but turned out to be about right
Read chapter 6 in Patterns book	1 hour	5 Hours	Under estimated how soul draining someone could make a book. Very hard to read but alot of information.
Create text based use cases	2 Hours	2 hours	Straightforward good information in the lectures to.
Create basic use case diagram	8 hours	3 hours	I imagined there would be alot more drawing involved.
Expand the use case model	3 hours	2 hours	Under estimated how many possible paths there accually are in such a small system.
Fully dressed use case	3 hours	2 hours	Under estimated how many possible paths there accually are in such a small system.
Model behavior with UML state machine	6 hours	2 hours	I was done in about 2 hours but found the exercise very beneficial and keept refining it for a few hours.
Play game state machine	2 hours	0.5 hours	This was basicly done in the step above and not much time was needed.
Implement basic version	8 hours	10 hours	I'd imagined it take the whole night and it did. (one full nightshift at work). I got done 2

			hours early but it was about what I imagined.
Create Class diagrams	6 hours	4 hours	This entailed alot more reading and watching videos than anticipated.
Creating testplan	4 hours	10hours	Alot more reading, watching examples and trying to decide on good methods to test. Realized I had not build a very testable system.
Manual testing for fetching words from wikipedia	2 hours	1 hours	I had already tested this manually a lot and I knew the steps inside and out.
Manual testing for quit game	2 hours	0.5 hours	This use case had fewer possible out comes than anticipated.
Adding what was missing in iteration 1	3 hours	4 hours	This time estimation is harder than anticipated.
Rewriting iteration 4 plan according to make it more fine grained.	4 hours	5 hours	Alot to think about trying to change the last iteration to fit with what I have learned in the course
Update testplan with highscore and nickname	2 hours	2 hours	I felt like I had all the knowledge needed to do this and no surprises turned up so I was right on time estimate for once
Update UML with new highscore and nickname	2 hours	0,5 hours	I did not have to re do it all only simple additions
Added highscore and nickname	3 hours	1 hours	Made one use case of the two and saved alot of time. Save name and highscore is done at the same time into a hashmap. Did not know about hashmaps when I made time estimate.
Write tests for Highscore and nickname	1 hours	1 hours	Straightforward just small changes to old test.

Update project plan according to feedback from iteration 1	4 hours	5 hours	It was not totaly clear to me what I had to do so had to ask in slack. How ever I should have just read the instructions more closley. And saved 3 hours.
Updated document with feedback from iteration 2	3 hours	2 hours	There was not as much missing as I suspected
Proof read the document, instructions and slack.	4hours	3 hours	Alot to read but I knew most of it so it went a little bit faster.
Create PDF and Hand in	5 minutes		

9 Fully dressed use case diagram

Use case: Playing the game

Main Scenario:
Pre-condition: The user has started the system and is connected to the internet.
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 a letter.
4.System tells user where in the word the letter appears"
5.User guesses a letter.
6.System tells User this letter is not in the word, creates the top of the Hangman and saves the guess.
7.User guesses a letter.
8.System tells user where in the word the letter appears.
9.User guesses a letter
10.System tells user where in the word the letter appears. And lets the user know he has won the game with a total of four guesses and displays the word and amount of guesses made.

Post-condition: The game is back at main menu.

Extentions:
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.

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.

Use case: Quit

Main Scenario:

Pre-condition: The user has started the system.

1. User wants to quit game.
2. System asks user to confirm that he wants to quit.
3.User says yes.
4. System terminates

Extentions:

3 User says no.
3.1 System continues as nothing happened.

Use case: Getting words from wikipedia.

Main scenario:

Pre-condition: The user has started the system and is connected to the internet.
1.User wants to fetch words from specific wikipedia page
2.System fetches the words from the page and starts the game.

Extentions:

2.System cannot fetch words from site, tells the user and ask him to try again.
2.1.User chooses another site.
2.2.System fetches words from site and starts game.

Use case: Save highscore

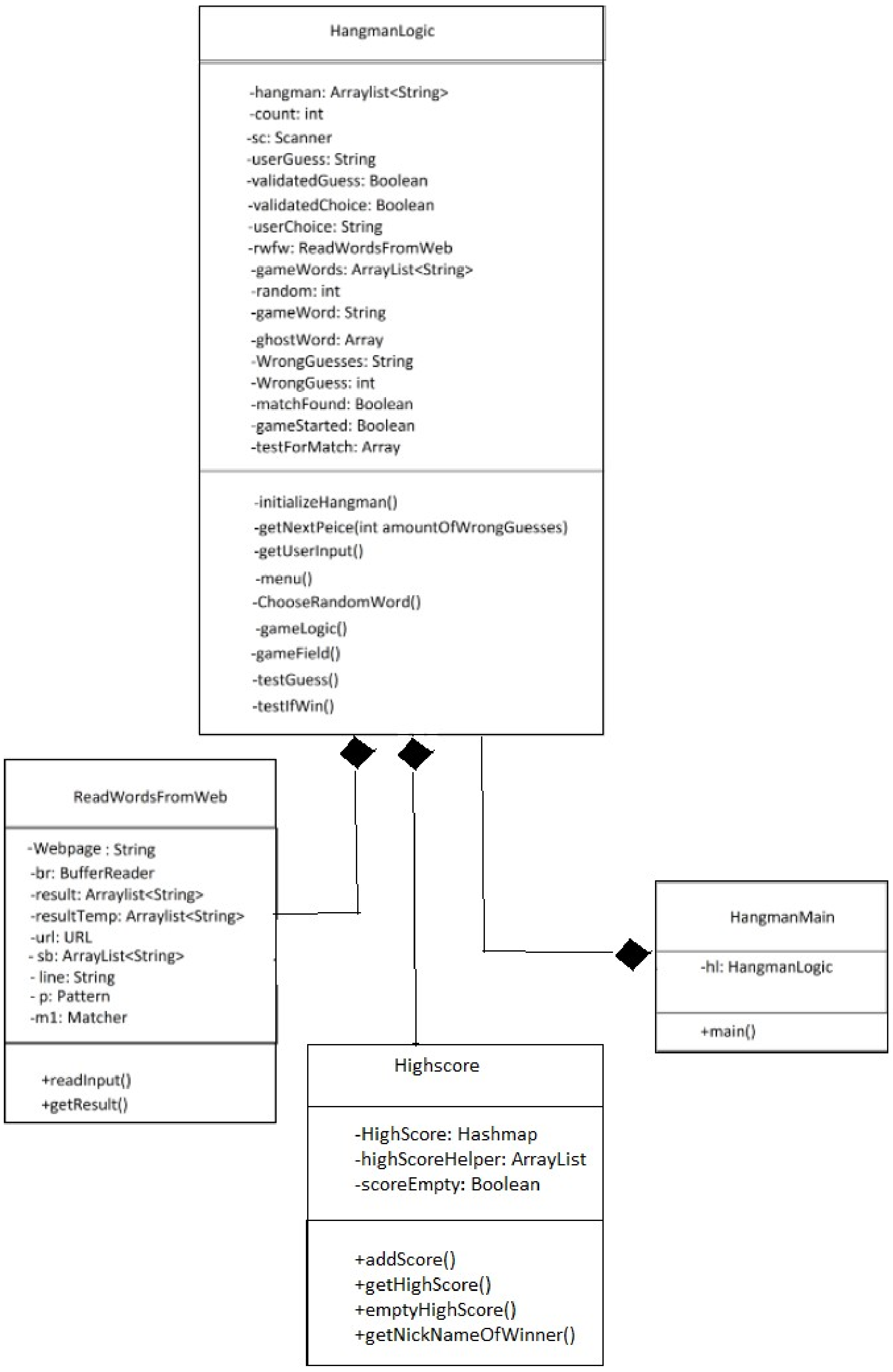
Main scenario:

Pre-condition: The user has finished a game and guessed all the letters correct.
1.User wants to save score
2.System ask user to enter nickname
3.User enters nickname.
4.System saves nickname and score.

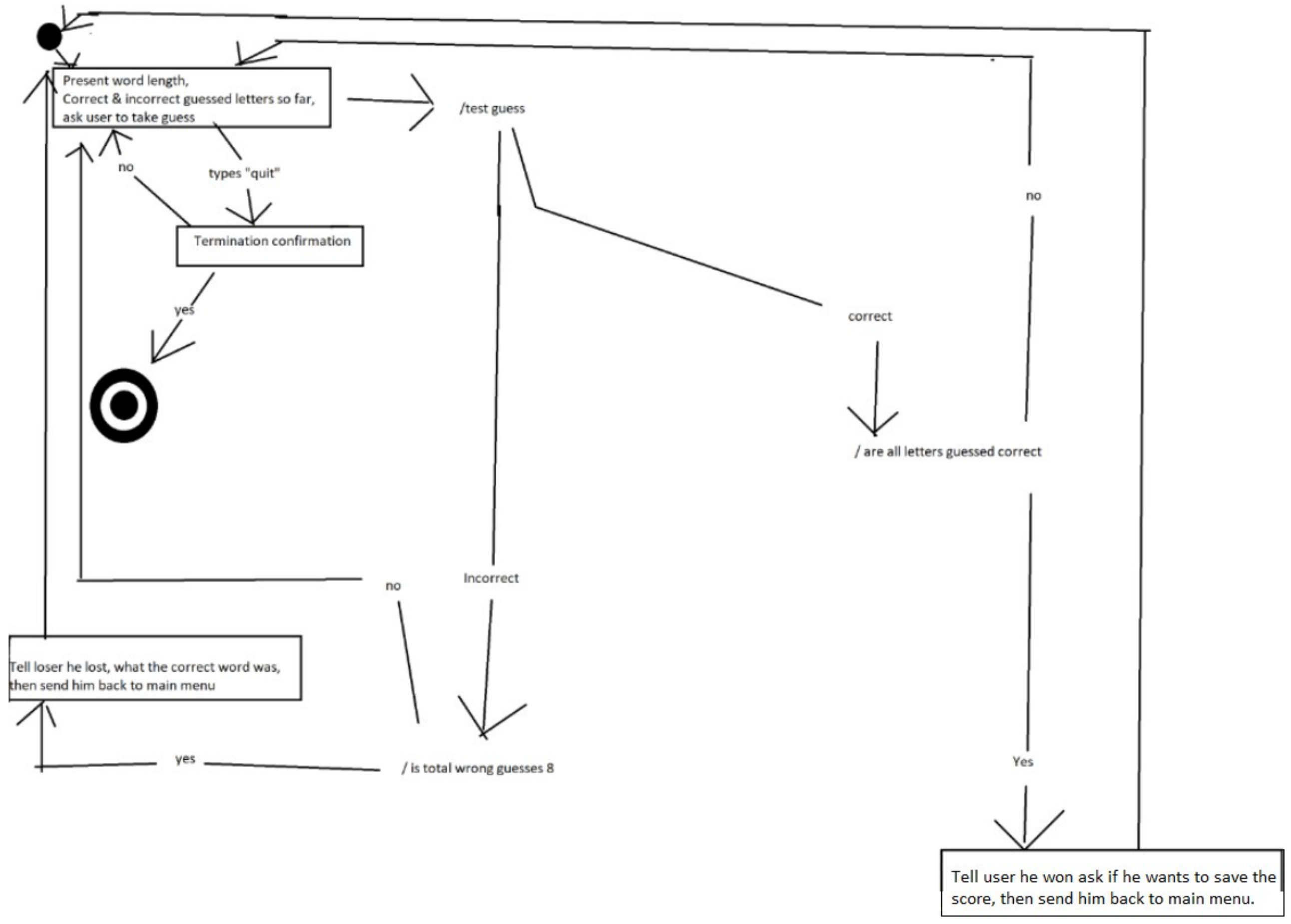
Extentions:

1. User don´t want to save nickname
1.1 System asks if user wants to save name
1.2 User don't want to save
1.3 System discards score.

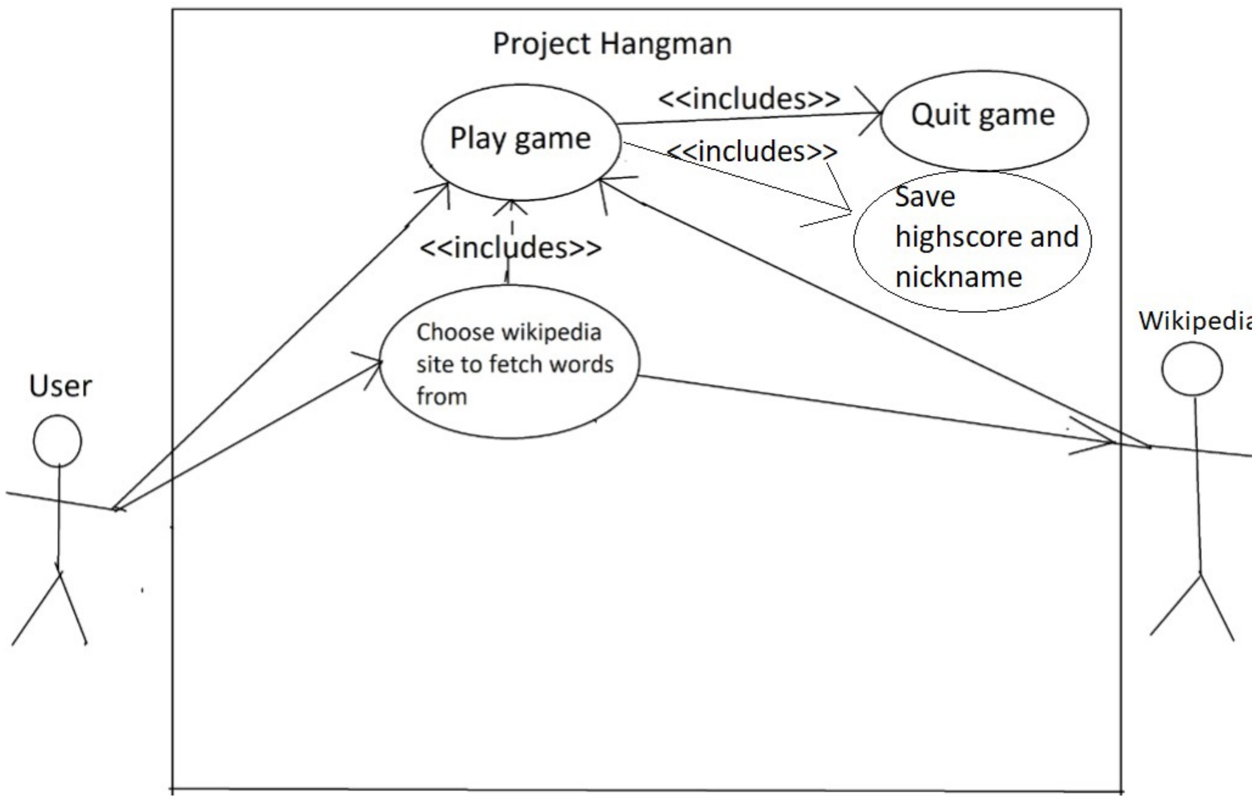
10 Class Diagram



11 State machine (Play)


Present word length,
Correct & incorrect guessed letters so far,
ask user to take guess
/test guess
no
types "quit"
Termination confirmation
yes
correct
no
/ are all letters guessed correct
no
Incorrect
Tell loser he lost, what the correct word was,
then send him back to main menu
yes
/ is total wrong guesses 8
Yes
Tell user he won ask if he wants to save the
score, then send him back to main menu.

12 Use case Diagram


Project Hangman
Play game
<<includes>>
Quit game
<<includes>>
Save highscore and nickname
<<includes>>
Choose wikipedia site to fetch words from
User
Wikipedia

Test Plan

Project hangman

Objective

The objective is to test part of the code written so far for the project hangman application according to the instructions at https://github.com/dntoll/1dv600/tree/master/A3.

What to test and how

All tests in this report is dynamic and done by the author, first I will manually test the "Choose wikipedia site to fetch words from" refered to as “wikipedia” in this report and the "quit game" use case. The wikipedia use case has a lot of moving parts and is best tested manually so that I can make sure that I have internet connection and if it takes a lot of time I can manually check what might be going on. Also I am scraping wikipedia and I don't want to automate it and accidentally make to many calls and get my IP blocked. The quit game use case will also be manually tested because it is a easy enough test to do manually, there is not to many possible outcomes and It won't take to much time.

Then I will write two Junit tests each for two methods. The first method is addHighScore() in the HighScore class to check that the highscores get added and sorted properly so that the highscore can be displayed in the correct way. This method works well for automated testing as it got simple inputs and is easy to check if it performed correctly. The second method that I will create Junit tests for is initializeHangman() in the gameLogic class. The tests will make sure that the hangman get initialized in the correct form and look exactly the same every time. This method is also well suited for automated testing since I might miss small changes in the hangman "image" by just looking at it but the automated test will find any deviation. Lastly one automated Junit test called checkNickNameTest() that will detect a failure in a incomplete method for checking if a nickname is saved.

Time plan

Task	Estimated time	Actual time
Manual test case	2 hours	8 hours
Unit test	3 hours	9 hours
Running manual test	1 hours	1 hours
Create test report	5 hours	15 hours

Manual Test-Cases

TC1 Choose wikipedia site to fetch words from

Use case: **Choose wikipedia site to fetch words from**

What's tested: The applications ability to fetch words for the game from a English wikipedia site.

Scenario: Successfully fetch words from chosen wikipedia

Precondition: Active internet connection and application started

Test steps

1. Run the app in IDE

System shows:

"--welcome, to start game type start and press Enter

--if you want to choose what wikipedia site to fetch your words from type the address

and press enter to start the game. Type quit at any time terminate the application

Example https://en.wikipedia.org/wiki/Stockholm"

2. Enter the address " https://en.wikipedia.org/wiki/Stockholm" and press enter

Game starts and shows:

amount of wrong guesses = 0

    your wrong guesses =

    ....

    Take a guess!

3.Push q and hit enter

Repeat step 3 changing q for w,t,p,s,d,g,h,k,l,z,x,c,v,b,n,m,until game finish

Game shows:

```
  ____

   |    o
   |   /|\
   |    |
   |   / \
   |_
 |   |______
 |          |
 |__________|
GAME OVER!, hit Enter to restart application, the answer was
continental
```

4. Copy the word after "the answer was " and go to https://en.wikipedia.org/wiki/Stockholm

5. crl + F Search for the word to see that the word got pulled from the page.

Expected

The crl + F search should find the word on the page

Game asks user to hit enter to restart application

Test successfull	Test unsuccessfull
OK	

Note:
Test was successfull every time it was conducted

TC1.2 invalid adress prompt user for new adress

Use case: **Choose wikipedia site to fetch words from**

What's tested: The applications ability recognize an incomplete address and tell the user to try again.

Scenario: unsuccessful to fetch words from the address given.

Precondition: user is at the main menu.

Test steps

1.Run the app in IDE

•System shows:

```
"--welcome, to start game type start and press Enter


--if you want to choose what wikipedia site to fetch your
words     from type the adress
and press enter to start the game. Type quit at any time
terminate     the application
Example https://en.wikipedia.org/wiki/Stockholm"
```

2.Enter the adress “ `ht://en.wikipedia.org/wiki/nothingheretosee`” and press enter

•**Expected**

•The system should show the text "`the URL does not look right, try again and follow the format https://en.wikipedia.org/wiki/Stockholm`"

•System shows main menu again and waits for input.

Test successfull	Test unsuccessfull
OK	

Note:
Test was successfull every time it was conducted

TC2 Quit game

Use case: Quit game

What's tested: The users ability to terminate the application from the main menu.

Scenario: terminate the application from main menu

Precondition: user is at the main menu.

Test steps

1. Run the app in IDE

•System shows:

```
"--welcome, to start game type start and press Enter


--if you want to choose what wikipedia site to fetch your
words     from type the adress
and press enter to start the game. Type quit at any time
terminate      the application
     Example https://en.wikipedia.org/wiki/Stockholm"
```

2.Enter “quit” and hit enter

3.systems shows “Are you sure you want to terminate? Yes/no”

4.Enter “yes” and hit enter

Expected

•The system should terminate the application.

Test successfull	Test unsuccessfull
OK	

Note: This should work at any time during the game and does not have to be done at main menu.

TC2.1 Quit game, change mind.

Use case: Quit game.

Whats tested: Terminate the application from the main menu but change mind.

Scenario: starting termination of the application from main menu when writing “quit” and hitting enter then typing “no” and hitting enter to return to main menu.

Precondition: user is at the main menu.

Test steps

1.Run the app in IDE

•System shows:

```
"--welcome, to start game type start and press Enter


--if you want to choose what wikipedia site to fetch your
words     from type the adress
and press enter to start the game. Type quit at any time
terminate      the application
     Example https://en.wikipedia.org/wiki/Stockholm"
```

2.Enter “quit” and hit enter

3.systems shows “Are you sure you want to terminate? Yes/no”

4.Enter “no” and hit enter

Expected

•The system should return to main menu and await input.

Test successfull	Test unsuccessfull
OK	

Note:

1.Anything but “yes” will return user to main menu.

2.This will work at any time during the game and does not have to be done at main menu.

Test Report

Test traceability and success matrix

Test	UC Wiki	UC Quit
TC1.1	OK	
TC1.2	OK	
TC2.1		OK
TC2.2		OK
COVERAGE & SUCCESS	2/OK	2/OK

Automated unit test coverage and success

Test	Method tested	Coverage & success
InitializeHangmanTest	initializeHangman()	100/OK
testAddHighScore	HighScore.addHighScore()	100/OK
testgetHighScore	HighScore.getHighScore()	100/OK
checkNickNamesTest	checkNickNames	100/OK

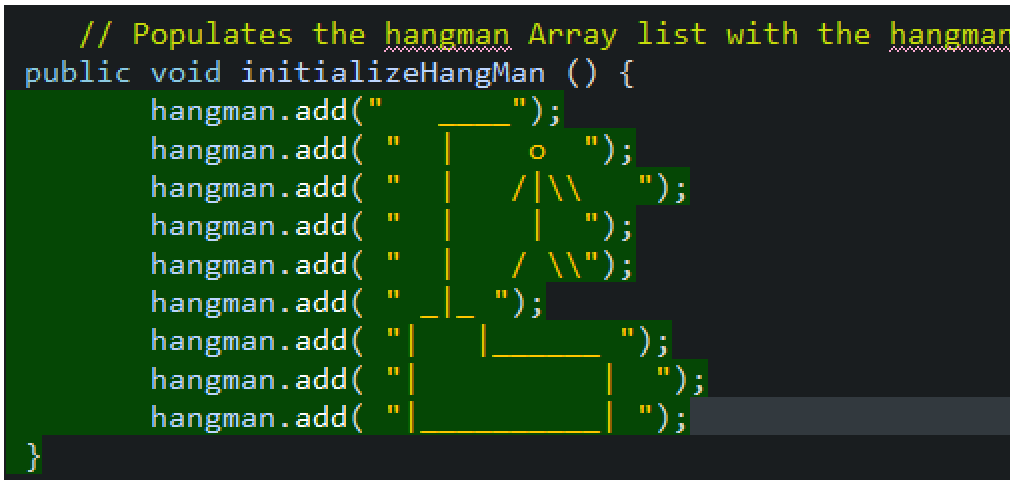
Comment

All tests that tested complete methods pass, next iteration I need to focus on extending use cases and if needed refactor the code.

Automated Junit test cases

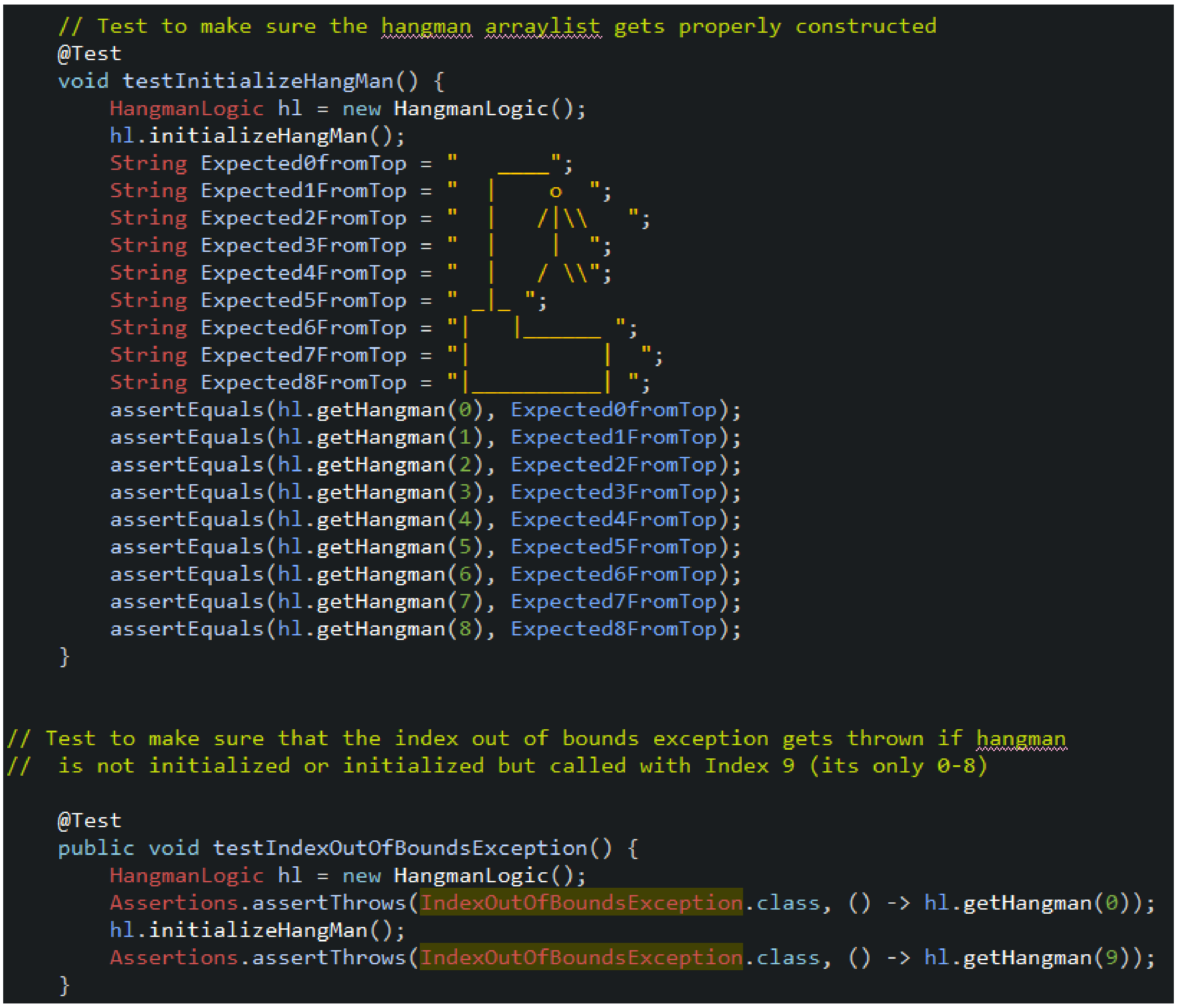
Test case 1:
Whats tested: The applications ability to create the hangman in the right porportions
Precondition: application started

Coverage, initializeHangmanTest():



```
    // Populates the hangman Array list with the hangman
 public void initializeHangMan () {
        hangman.add("    ____");
        hangman.add( "  |    o  ");
        hangman.add( "  |   /|\\   ");
        hangman.add( "  |    |  ");
        hangman.add( "  |   / \\");
        hangman.add( " _|_ ");
        hangman.add( "|   |______ ");
        hangman.add( "|          |  ");
        hangman.add( "|__________| ");
 }
```

Test code, initializeHangmanTest():

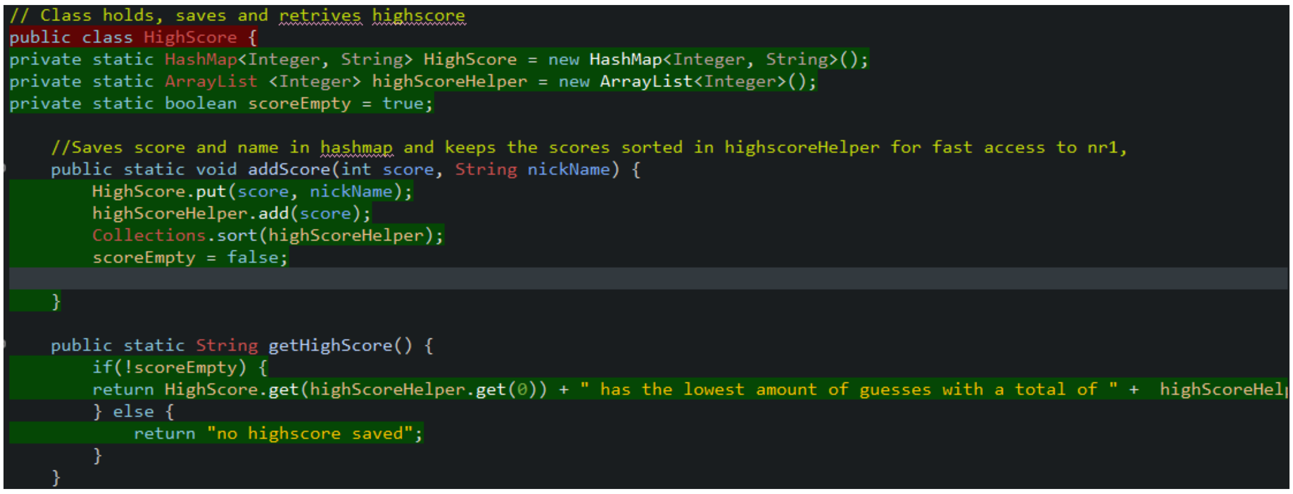


```java
    // Test to make sure the hangman arraylist gets properly constructed
    @Test
    void testInitializeHangMan() {
        HangmanLogic hl = new HangmanLogic();
        hl.initializeHangMan();
        String Expected0fromTop = "   ____";
        String Expected1FromTop = "  |    o  ";
        String Expected2FromTop = "  |   /|\\   ";
        String Expected3FromTop = "  |    |  ";
        String Expected4FromTop = "  |   / \\";
        String Expected5FromTop = " _|_ ";
        String Expected6FromTop = "|   |______ ";
        String Expected7FromTop = "|          |  ";
        String Expected8FromTop = "|__________| ";
        assertEquals(hl.getHangman(0), Expected0fromTop);
        assertEquals(hl.getHangman(1), Expected1FromTop);
        assertEquals(hl.getHangman(2), Expected2FromTop);
        assertEquals(hl.getHangman(3), Expected3FromTop);
        assertEquals(hl.getHangman(4), Expected4FromTop);
        assertEquals(hl.getHangman(5), Expected5FromTop);
        assertEquals(hl.getHangman(6), Expected6FromTop);
        assertEquals(hl.getHangman(7), Expected7FromTop);
        assertEquals(hl.getHangman(8), Expected8FromTop);
    }


// Test to make sure that the index out of bounds exception gets thrown if hangman
//  is not initialized or initialized but called with Index 9 (its only 0-8)

    @Test
    public void testIndexOutOfBoundsException() {
        HangmanLogic hl = new HangmanLogic();
        Assertions.assertThrows(IndexOutOfBoundsException.class, () -> hl.getHangman(0));
        hl.initializeHangMan();
        Assertions.assertThrows(IndexOutOfBoundsException.class, () -> hl.getHangman(9));
    }
```

Test case 2:

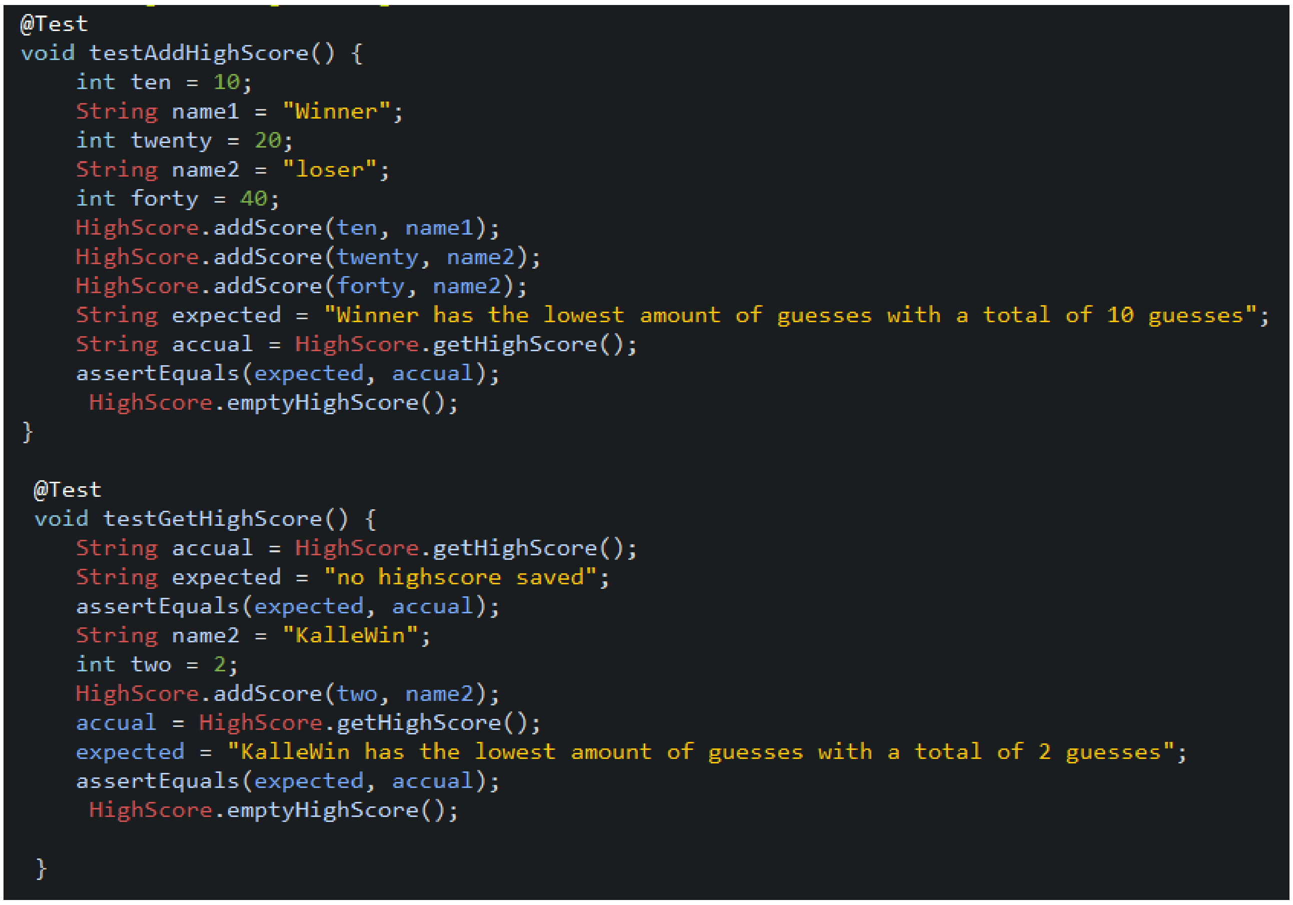
Whats tested: To add to highscore and keep highscore sorted
Precondition: application started

coverage, testAddHighScore():



```
// Class holds, saves and retrives highscore
public class HighScore {
private static HashMap<Integer, String> HighScore = new HashMap<Integer, String>();
private static ArrayList <Integer> highScoreHelper = new ArrayList<Integer>();
private static boolean scoreEmpty = true;

    //Saves score and name in hashmap and keeps the scores sorted in highscoreHelper for fast access to nr1,
    public static void addScore(int score, String nickName) {
        HighScore.put(score, nickName);
        highScoreHelper.add(score);
        Collections.sort(highScoreHelper);
        scoreEmpty = false;

    }

    public static String getHighScore() {
        if(!scoreEmpty) {
        return HighScore.get(highScoreHelper.get(0)) + " has the lowest amount of guesses with a total of " +  highScoreHel
        } else {
            return "no highscore saved";
        }
    }
```

Test code part 1, testAddHighScore():



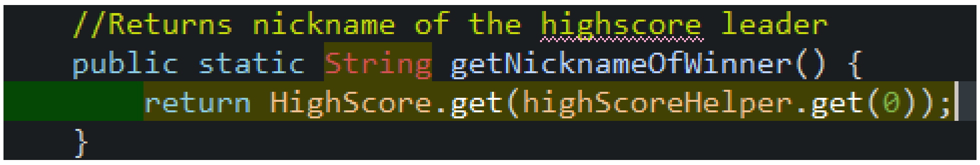
```
@Test
void testAddHighScore() {
    int ten = 10;
    String name1 = "Winner";
    int twenty = 20;
    String name2 = "loser";
    int forty = 40;
    HighScore.addScore(ten, name1);
    HighScore.addScore(twenty, name2);
    HighScore.addScore(forty, name2);
    String expected = "Winner has the lowest amount of guesses with a total of 10 guesses";
    String accual = HighScore.getHighScore();
    assertEquals(expected, accual);
     HighScore.emptyHighScore();
}

 @Test
 void testGetHighScore() {
    String accual = HighScore.getHighScore();
    String expected = "no highscore saved";
    assertEquals(expected, accual);
    String name2 = "KalleWin";
    int two = 2;
    HighScore.addScore(two, name2);
    accual = HighScore.getHighScore();
    expected = "KalleWin has the lowest amount of guesses with a total of 2 guesses";
    assertEquals(expected, accual);
     HighScore.emptyHighScore();

 }
```

Test case 3:

Whats tested: the functionality to retrive the name of the winner
Precondition: application started

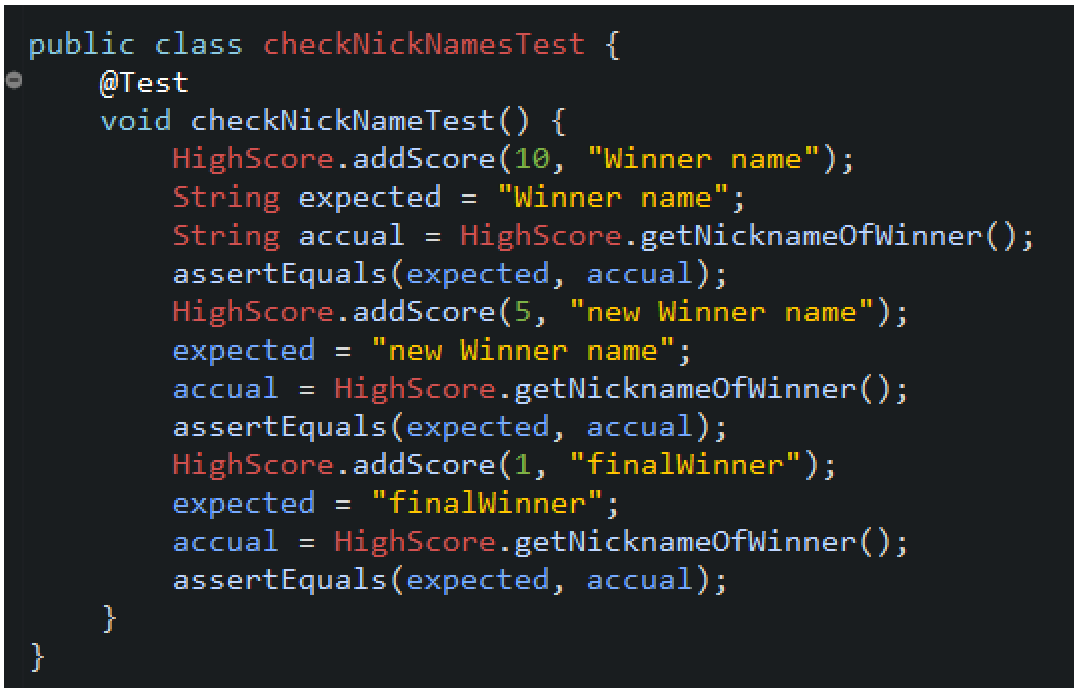
checkNickNamesTest:

coverage:



```
//Returns nickname of the highscore leader
public static String getNicknameOfWinner() {
    return HighScore.get(highScoreHelper.get(0));
}
```

Test code:



```java
public class checkNickNamesTest {
    @Test
    void checkNickNameTest() {
        HighScore.addScore(10, "Winner name");
        String expected = "Winner name";
        String accual = HighScore.getNicknameOfWinner();
        assertEquals(expected, accual);
        HighScore.addScore(5, "new Winner name");
        expected = "new Winner name";
        accual = HighScore.getNicknameOfWinner();
        assertEquals(expected, accual);
        HighScore.addScore(1, "finalWinner");
        expected = "finalWinner";
        accual = HighScore.getNicknameOfWinner();
        assertEquals(expected, accual);
    }
}
```

Reflection:

Before this iteration the word testing sounded really boring to me and I thought it was something I would not like, how ever this iteration has been eye opening for me. My code in this project is not very great. It's not written to be tested since I have never really tested before I did not think about it. I have gained a lot of understanding of what I need to do differently when I code and I'm looking forward to writing code with testing in mind since it will be so much better and I will have alot more confidence in the code I produce.