CS12420 Individual Assignment - A Simple Teaching Game

Alexander David Brown (adb9)

March 12, 2010

Contents

Part I

Brief

0.1 Given Brief

Problem

This assignment was suggested by Keith Lucas. Keith works with mentally handicapped people covering a range of ability. Some already have a degree of literacy and others the ability to learn simple written communication. The computer can help.

Basic Requirements

- There are two users – the builder and the player.
- The game will usually be played by a client with minimal staff intervention. It will be simple in presentation. The player sees several words on the screen including a target word.
  - The object of the game is to click the copy of the target word.
  - If a word other than a target copy is clicked then a negative message will be read and a suitable sound will be produced.
  - The game is ended by choosing all the correct words or by closing the application.

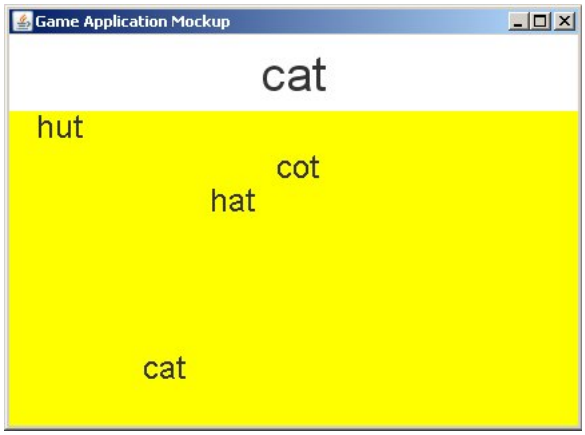


Figure 1: Mock up of the Game Screen

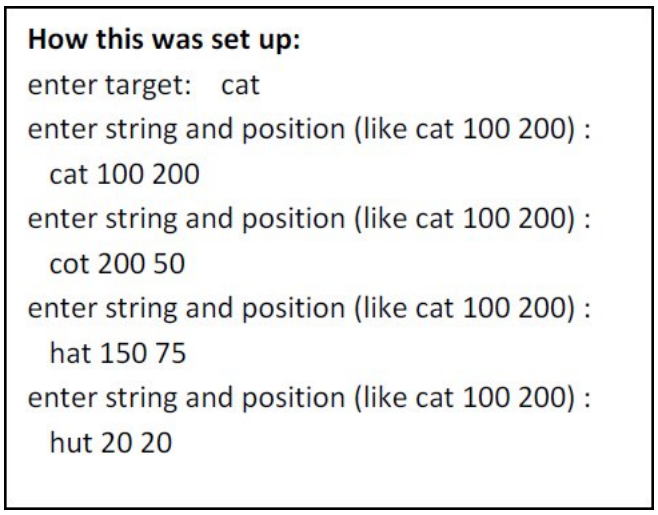


Figure 2: Mock up of the Editor Screen

In this simple version of the game, the Builder started the game and typed in just a single target word followed by four words (including the target word) and their coordinates in the window.

The playing screen then appeared and the game switched to Player mode. When the player clicked on hut the siren went off. When the player clicked on cat a nice noise was produced and a pop up saying that the player got the word right was displayed and the game completed.

Your version of the game must have a more sophisticated Builder part:

- The builder must be able to design, save and load a game, and then:
  - Set a target word, e.g. cat
  - Set other words (similar in appearance) e.g. cat, cot, hat, hut with associated positioning (perhaps with a click).
  - **This should also be done via a GUI, but to split the problem up you may want to begin with a simple text interface as illustrated above.**

Nice extra features:

- Words could appear in different fonts and sizes, which are chosen when the game is designed.
- The builder could set the game so that the player could play it several times with different words.
- Having a version where a picture (icon) is clicked instead of a word.
- Being able to drag the words and place them over the target instead of just click on them would be a really nice (but non-trivial) modification.
- Ultimately, Keith would like the ability to design and tailor lots of games, not just to have this one game. (Then different games could be designed at will.) Were really not sure how to do that, but you could give it some thought.

0.2 Requirements

0.2.1 Requirement A - General Requirements

A01 Must support two users – a player and a builder.

A02 Game must be played by a client with minimal staff intervention.

A03 The Game must run without problems.

0.2.2 Requirement C - Coding Standards

C01 The code must be properly commented.

C02 The code must use correct JavaDoc comments.

0.2.3 Requirement E - Editor Requirements

E01 The Builder must be able to design a game.

E02 the Builder must be able to load a game.

E03 The Builder must be able to save a game.

E04 The Builder must be able to set a target word.

E05 The Builder must be able to set the other words.

0.2.4 Requirement G - Game Requirements

G01 Several words must be printed on the screen, including a target word.

G02 If the wrong word is clicked then a negative message is shown.

G03 If the wrong word is clicked then a negative sounds is shown.

G04 The game ends when all the correct word(s) are clicked.

0.2.5 Requirement P - Presentational Requirements

P01 Must be simple in presentation.

P02 Several words must be printed on the screen, including a target word.

P03 The Game window should look similar to the mock-up.

P03 The Editor must have a Graphical User Interface.

Part II

Design

0.3 Definition

0.3.1 Text Definition

- The **Application** has both a **Game** and an **Editor**.
- Every **Game** has several **Words**.
- Every **Editor** has several **Words**.

Therefore there must be at least four Classes:

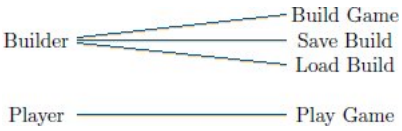
Application The main Class which will contain the main running methods.

Game The Class which runs the playing side of the Application.

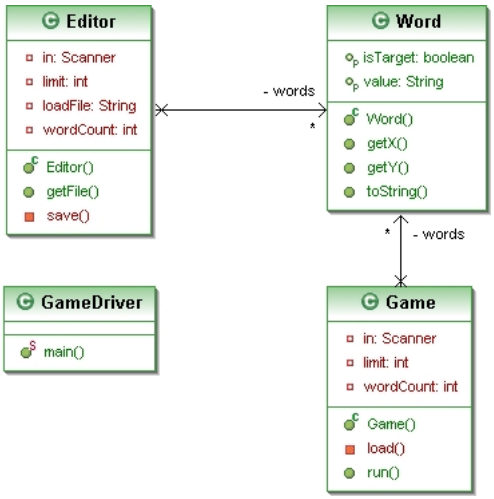
Editor The Class which creates Games.

Words The Content of the Application.

0.3.2 Use Case Diagram



0.3.3 Simple Class Diagram



0.3.4 Interface Design

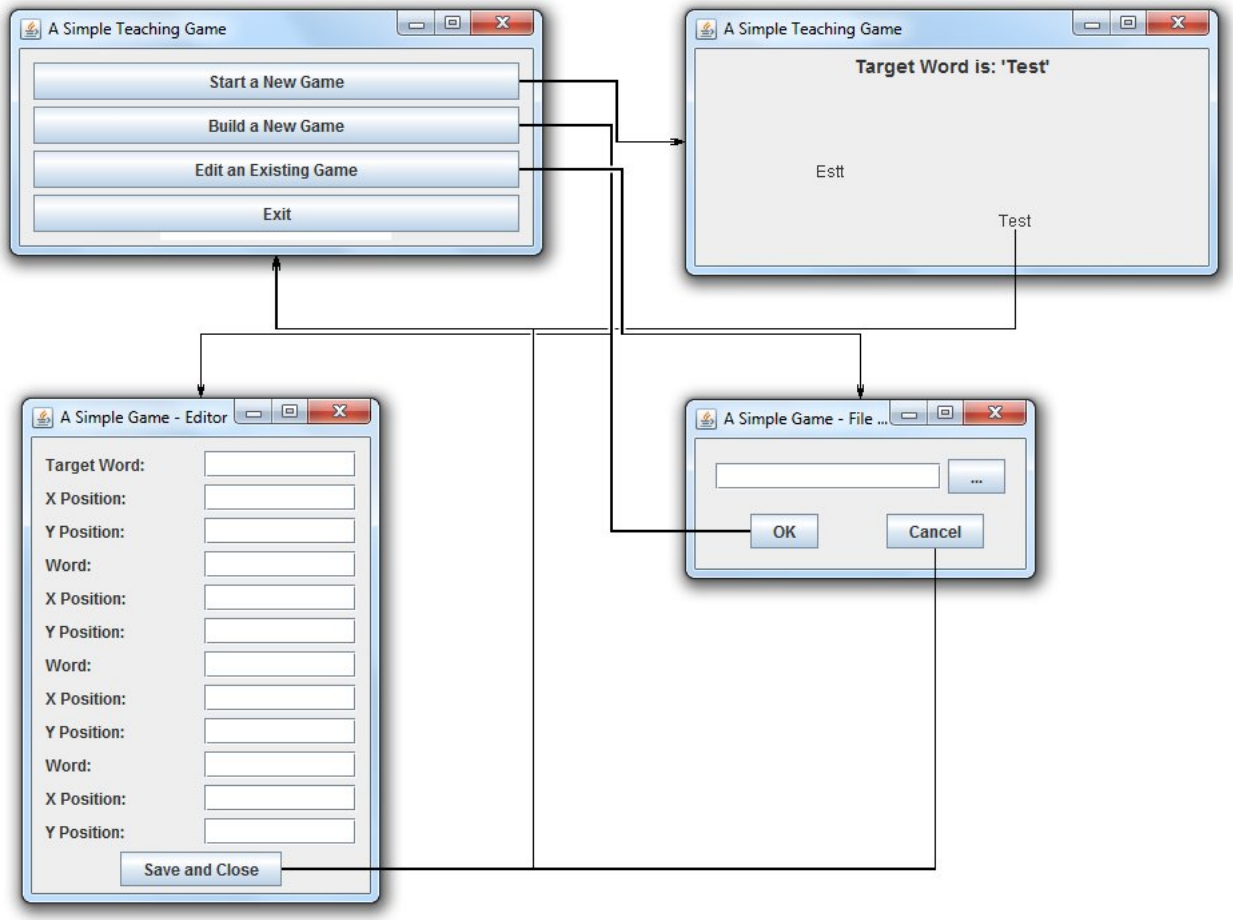
Main Menu The main menu of the application. Allows users to:

- Start a new Game.
- Build a new Game.
- Edit an existing Game.
- Exit.

Game Window The window in which the Game is played. Displays the Target Word in the top of the pane, and the Words in their respective positions in the center of the pane.

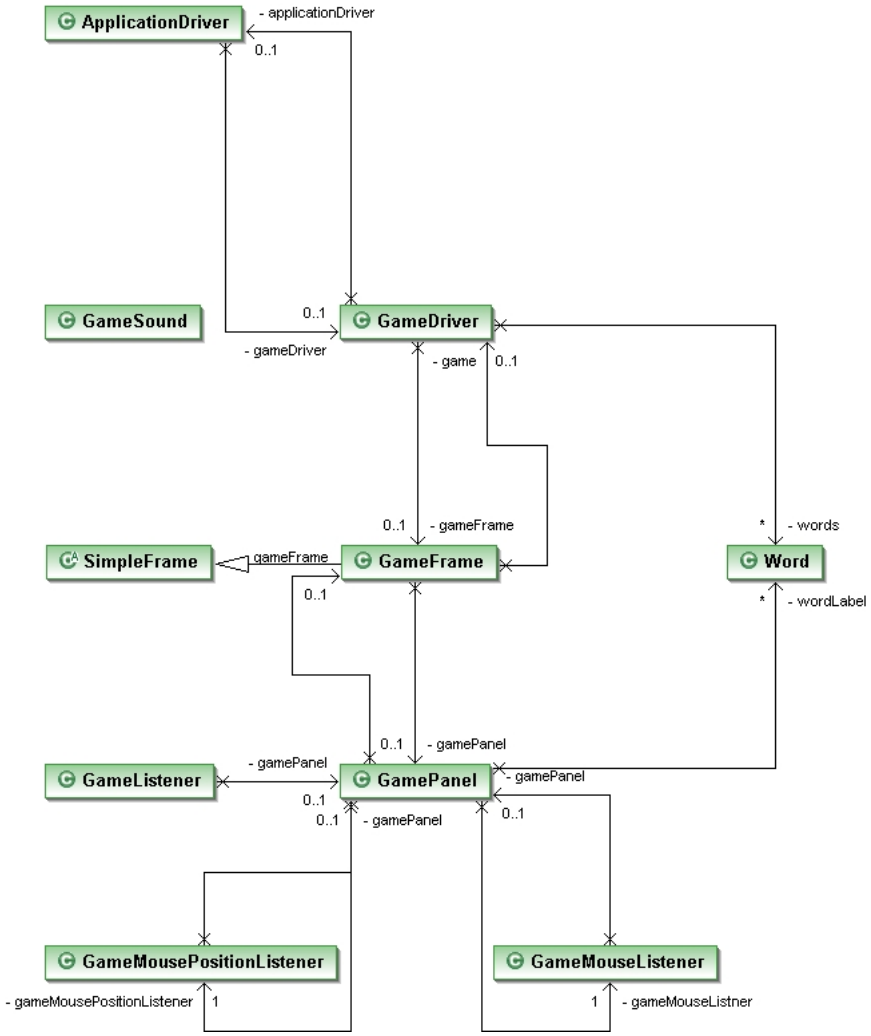
Editor Window Allows Games to be Edited. At first perhaps just a set number of Words all displayed, but later from one up to about twenty.

Load Window Allows a filename to be Entered and Loaded, which can then be passed into the Editor or Game.

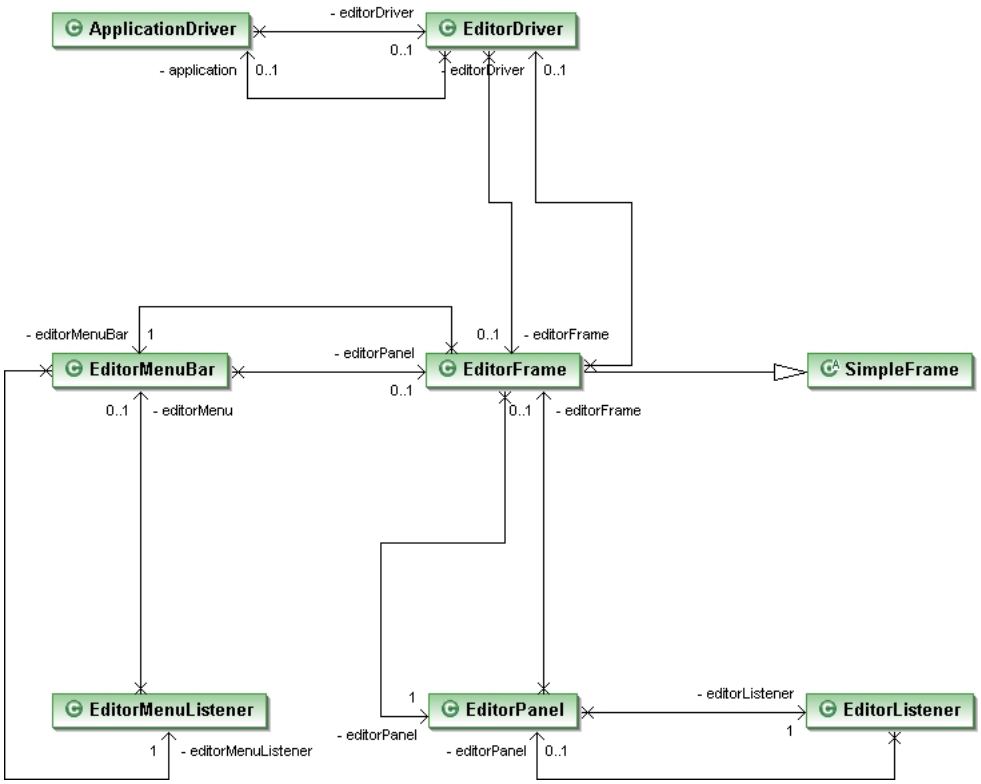


0.3.5 Class Diagram including G.U.I.

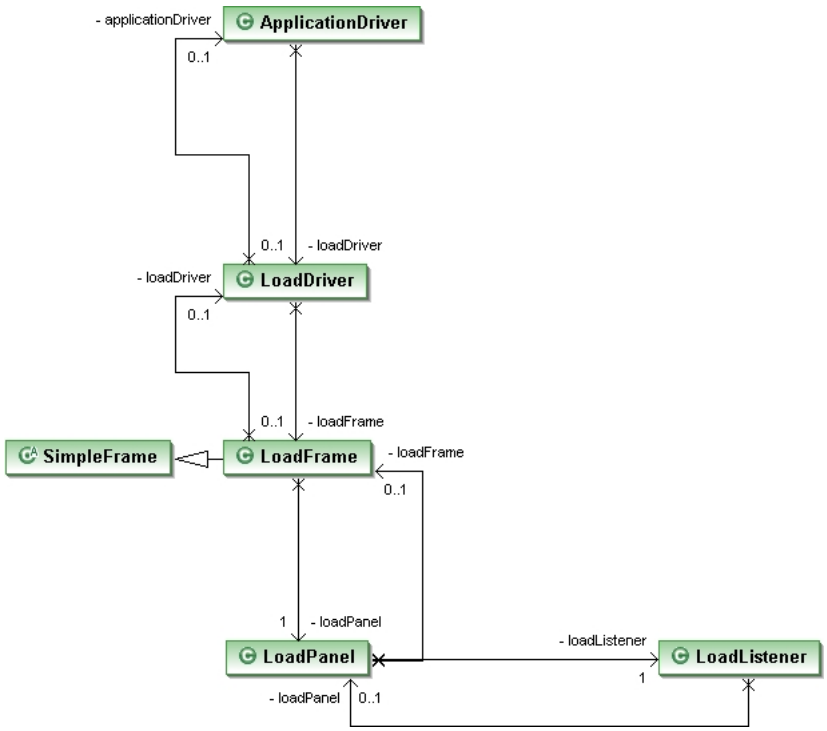
Game



Editor



Loader



0.3.6 Loading/Saving

File Extension

This application will use the extension `.asgf`, standing for 'A Simple Game File'. It is unused by any other commercial program.

File Format

Brackets '()' and their contents are not included in the file.

```
4 (Number of words)
Cat (Word 0 Value)
10 (Word 0 X Coordinate)
10 (Word 0 Y Coordinate)
true (Word 0 is the target word)
Dog (Word 1 Value)
50 (Word 1 X Coordinate)
25 (Word 1 Y Coordinate)
false (Word 1 isn't the target word)
(etc.)
```

G.U.I. Class Diagram

0.4 Algorithm Design

Most of the algorithms used are pretty simple and don't need designing. With the G.U.I. it seems to be more passing methods from class to class rather than having complex methods.

0.4.1 Drag and Drop

```
boolean isPickedUp = false;
Dimesnion startPosition;

dragAndDropRoutine()
{
    Word word;

    if(mouse pressed over a word)
    {
        word = getWord(mouse.getPosition());
        startPosition = word.getPosition();
        isPickedUp = true;
    }

    while(isPickedUp)
    {
        if(mouse moves)
       {
           drag(word);
        }
        else if(mouse goes outside of Frame)
        {
            isPickedUp = false;
            word.setPosition(startPosition);
        }
        else if(mouse is released)
        {
            isPickedUp = false;
            word.setPosition(mouse.getPosition));
            checkResult(word);
        }
    }
}

drag(Word word)
{
    word.setPosition(mouse.getPosition());
}

checkResult(Word word)
{
    if(word.isOverDropZone()
    {
        if(word.isTargetWord())
        {
            win();
        }
        else
        {
            notCorrect();
        }
    }
}
```

Part III

Version Planning

Every time a large change is made a new version should be made, begining at 0.1.0.

Every time a large change is met (e.g. meeting every requirement in the brief), the version should be increased by 1

Every time a medium sized change is met (e.g. adding an interface for the Game), the version should be increased by 0.1

Every time a small change is met (e.g. adding adding a menu to the Editor), the version should be increased by 0.0.1

Whenever a version moves to the next it should be duplicated, one kept as a backup and the other continued upon. Also any decimal points below that of the one increased should be reset to 0 (i.e. 0.3.3 would go to after a medium change 0.4.0).

0.5 Road map

These are the revisions I am planning on including and the details on each revision.

0.5.1 Versions 0.x - Development Versions

These versions begin at the Text Based Interface and finish at a version which should include every requirement in the brief.

Version 0.1.0 Command Line Interface (C.L.I.) based version.

Version 0.2.0 Start adding Graphical User Interface (G.U.I.) for the Game side to match the given mock-up.

Version 0.3.0 Positioning of words in the Game Panel, this should now fully match the given mock-up. Should still use a C.L.I. Editor.

Version 0.3.1 Mouse highlighting events.

Version 0.3.2 Error Pop-ups.

Version 0.3.3 Sound inclusion.

Version 0.4.0 Main Menu added.

Version 0.4.1 Loading of a game (not running through editor every time).

Version 0.4.2 Loading G.U.I.

Version 0.5.0 Simple G.U.I. Editor.

Version 0.5.1 Choice of number of words.

Version 0.6.0 Loading into Editor.

Version 0.6.1 Full menu system in Editor.

Version 0.6.2 Loading into Editor from Main Menu.

0.5.2 Version 1.0 - Beta Version(s)

These versions are the untested basic version (Version 1.0). These versions will be throughly tested.

Version 0.7.0 (a.k.a. 1.0-beta0) Untested, fully working version.

Possible other beta versions due to bug fixing.

0.5.3 Version 1.0 - Stable Version

This is the completed basic version, with no (or at least none which will cause the game to fail) Software Faults.

It should also have complete documentation in JavaDoc.

Version 1.0.0 Tested, fully working stable version. JavaDoc completed.

0.5.4 Version 1.x - Development Versions

These are the versions with the advanced features suggested.

Version 1.1.0 Addition of drag & drop editor.

Version 1.1.1 Drag & drop in Game too.

Version 1.2.0 Font customisation.

Version 1.2.1 Random font generation.

Version 1.3.0 Pictorial Version.

Version 1.4.0 Multiple games in a row.

Version 1.4.1 Related words.

Version 1.5.0 Interface design for similar games.

0.5.5 Version 2.0 - Beta Version(s)

The untested versions of the advanced version (Version 2.0). Which will be tested thoroughly.

Version 1.6.0 (a.k.a. 2.0-beta0) Untested development version.

Possible other beta versions.

0.5.6 Version 2.0 - Stable Version

The completed advanced version, fully tested and free of Software Faults.

Version 2.0.0 Tested, fully working version with advanced features. JavaDoc up to date.

0.5.7 Version 2.x - Development Versions

Potentially other versions with even more features. Very time dependent.

Version 2.1.0 Unknown.

See roadmap.txt and changelog.txt for more information of how this changed over the course of the project.

Part IV

Testing

0.6 Testing Tables for Version 0.7.0 (1.0-beta0)

See Table 1 and Table 2 in Part VI.

0.7 Issues from the testing of Version 0.7.0 (1.0-beta0)

0.7.1 Loader Issues

The Loader needs to be a lot stricter, it should only accept valid arguments and should not throw Exceptions. Use of error messages should be used to counter this.

This will be fixed and retested in Version 0.7.1 (1.0-beta1).

0.7.2 Editor Issues

The Editor also needs to be stricter and should not throw Exceptions. Again this should use error messages.

This will be fixed and retested in Version 0.7.2 (1.0-beta2).

0.7.3 Game Issues

Sound has not yet been implemented in the Game, also there is no 'Good' message.

This will be fixed and retested in Version 0.7.3 (1.0-beta3).

0.7.4 Final Checks

Once the above issues have been fixed the whole application will be tested under Version 0.7.4 (1.0-beta4). If this passes it will be allowed to become Version 1.0.0.

0.8 Testing Tables for Version 0.7.1 (1.0-beta1)

See Table 3 in Part VI.

0.9 Testing Tables for Version 0.7.2 (1.0-beta2)

See Table 4 in Part VI.

0.10 Testing Tables for Version 0.7.3 (1.0-beta3)

See Table 5 in Part VI.

0.11 Testing Tables for Version 0.7.4 (1.0-beta4)

See Table 6 and Table 7 in Part VI.

0.12 Issues from the testing on Version 0.7.4 (1.0-beta4)

0.12.1 Editor Issues

Test Editor still allows non-alphabet characters to be input as the value of a Word. This is caused by the way Java String works and will take some complex methods to fix.

For now these two errors will not be fixed, they may be completed as part of the 1.x Versions.

Part V

Evaluation

0.13 Overall Evaluation

This project has been a huge undertaking in comparison to any other project I have worked on, especially comparing back to the CS12230 assignment of last semester.

However, the actual difficulty of the project was about on par with anything I had already done, some things took some research to implement, but for the most part everything was fairly straight forward. The difficulty came with the number of Classes I had used in the project, despite using meaningful names and comments, both general and JavaDoc, it becomes an exercise of memory and not coding ability.

To begin with I took the advice given - to create the G.U.I. and the implement from that. However after a hideously complex start, I went back to basics, produced the Roadmap and started just getting the application to run. From there I slowly built upwards, using the road map as a general guide and keeping older versions in packages (towards the end I was taught to proper method of using packages and converted my old version packages to the proper format).

By keeping each version separately it gave me a nice safety net. I could mess up the current version and know that if the worst happened I would only be one square back (and not back to square one).

Sound was possibly the most difficult Version 0.x feature to implement. I did have to use the links emailed to work out how it might be possible, however I did write the actual code myself though it was difficult not to tread on the toes of the author of the code as much of the implementation was always going to be the same, hence why I included a comment about it in the JavaDoc for that method.

Drag and Drop was much easier to implement that I had thought and though my psuedocode was way off, it did provide a nice framework from which I could work out the correct implementation.

Images proved trickier to implement, mainly because I had an 'or' statement the wrong way around

```
if(word.getValue().equals("...") || word==null)
{
}
```

Was how I originally had the statement. Of course this was throwing Null Pointer Exceptions, which in turn were being caught by the try..catch block I had also included. A simple flip of the argument meant that it would use the null before it could be caught.

Due to the nature of the assignment, I really didn't feel much algorithm design was needed, most methods I could work out in my head and get down easily thanks to Eclipse.

I have improved my use of JavaDoc with this assignment and even including it within my testing tables. Finding and using the @see notation was especially useful due to the number of method being passed from Class to Class.

So in conclusion, the most difficulty I had with this project was remembering where everything was, not the implementation. Having recently learnt about packages hopefully this should be slightly less of an issue. I'm a little disappointed I didn't get to finish everything I wanted to do, but baring in mind the short period of time in which I had to do this assignment the resulting application is very good.

I have given the ability to make other similar games some thought - unfortunately I haven't been able to act upon this, but it would require interface version of the ApplicationDriver, GameDriver and EditorDriver classes, and very possibly their respective GUI classes.

0.14 Expected Mark

85/100

0.14.1 Reasoning

I have poured a lot of effort and time into this assignment, despite not being able to do everything I wanted to do with the project, I have included the main 'wow factor' feature - drag and drop, in both the Game and Editor.

Part VI

Tables

0.15 Testing Tables

ID	Requirement	Description	Inputs	Expected Outputs	Pass/Fail	Comments
A1.1	A01	The Game and the Editor can be accessed easily	N/A	N/A	P	Both accessible from the main menu.
A1.2	A02	The Game can be played without intervention	N/A	N/A	P	
A1.3	A03	Load a Game from a file which exists	Test.asgf	Screen shot SS01 is displayed	P	
		Load a Game from a file which does not exits	Fail.asgf	Screen shot SS02 is displayed	F	Exception thrown in Console.
		Load a Game from a file without an extension	Test	Screen shot SS03 is displayed	F	Exception thrown in Console.
		Load a Game from a file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	F	Exception thrown in Console.
A1.4	A03	Create a new Editor file which does not exist	New.agsf	Screen shot SS04 is displayed	P	
		Create a new Editor file which does exist	Test.asgf	Screen shot SS05 is displayed	F	Overwrites the file.
		Create a new Editor file without an extension	Test	Screen shot SS03 is displayed	F	Creates the file without the extension.
		Create a new Editor file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	F	Creates the file with the incorrect extension.
A1.5	A03	Load an Editor from a file which exists	Test.asgf	Screen shot SS06 is displayed	P	
		Load an Editor from a file which does not exits	Fail.asgf	Screen shot SS02 is displayed	F	Exception thrown in Console.
		Load an Editor from a file without an extension	Test	Screen shot SS03 is displayed	F	Loads the file without the extension (if it exists). As A1.4[a]
		Load an Editor from a file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	F	Loads the file without the extension (if it exists). As A1.4
A1.6	A03	Target word is edited to a non-null String	TestNew	Screen shot SS07 is displayed	P	
		Target word is edited to a null		Screen shot SS08 is displayed	F	Saves as ".asgf"
		Target word is edited to a non-String	123	Screen shot SS08 is displayed	F	Expected fail, due to Java String.
		Target word is edited to a value of another Words	Word1	Screen shot SS09 is displayed	F	Accepts normally.
A1.7	A03	Other word is edited to a non-null String	TestOther	Screen shot SS10 is displayed	P	
		Other words is edited to a null		Screen shot SS08 is displayed	F	As A1.6
		Other word is edited to a non-String	123	Screen shot SS08 is displayed	F	As A1.6
		Other word is edited to a value of another Words	Estt	Screen shot SS09 is displayed	F	As A1.6
A1.8	A03	X Position of the Target Word is edited to an integer	10	Screen shot SS11 is displayed	P	
		Y Position of the Target Word is edited to an integer	10	Screen shot SS12 is displayed	P	As Above.
		X Position of the Target Word is edited to a null		Screen shot SS13 is displayed	F	Exception thrown in Console.
		Y Position of the Target Word is edited to a null		Screen shot SS13 is displayed	F	As Above.
		X Position of the Target Word is edited to a non-integer	String	Screen shot SS13 is displayed	F	Exception thrown in Console.
		Y Position of the Target Word is edited to a non-integer	String	Screen shot SS13 is displayed	F	As Above.
		X and Y Position of the Target Word is edited to be that of another word	0, 0	Screen shot SS14 is displayed	F	Accepts Normally.

Table 1: Testing table for requirement A for Version 0.7.0 (1.0-beta0)

[a]As ID means the test is running through the same method and should generate the same result.

ID	Requirement	Description	Inputs	Expected Outputs	Pass/Fail	Comments
C1.1	C01	The code is properly commented	N/A	N/A	-	Not tested due to number of Fails. Will be tested in Version 0.7.4 (1.0-beta4).
C1.2	C02	Every Class has the relevant JavaDoc	N/A	N/A	-	Not tested due to number of Fails. Will be tested in Version 0.7.4 (1.0-beta4).
E1.1	E01	The Editor allows a game to be designed	N/A	N/A	P	
E1.2	E02	The Editor allows a game to be saved	N/A	N/A	P	(See A1.9)
E1.3	E03	The Editor allows a game to be loaded	N/A	N/A	P	See A1.4 and A1.5
E1.4	E04	The Editor allows a target word to be set	N/A	N/A	P	See A1.6 and A1.8
E1.5	E05	The Editor allows other words to be set	N/A	N/A	P	See A1.7 and A1.8
G1.1	G01	Words are printed on the Game Screen	Test.asgf	Screen shot SS01 is displayed	P	
G1.2	G02	Wrong word is chosen	Estt	Screen shot SS15 is displayed 'badsound.wav' is played	F	Sound is not played .
G1.3	G03	Target word is chosen	Test	Screen shot SS16 is displayed 'goodsound.wav' is played	F	No good message displayed and sound is not played.
P1.1	P01	Presentation is simple	N/A	N/A	P	
P1.2	P02	Words are printed on the Game Screen	Test.asgf	Screen shot SS01 is displayed	P	See G1.1
P1.3	P03	The Game Window is similar to the mock-up			P	
P1.4	P04	The Editor must have a G.U.I.	N/A	N/A	P	

Table 2: Testing table for requirements C, E, G and P for Version 0.7.0 (1.0-beta0)

0.16 Re-test tables

ID	Requirement	Description	Inputs	Expected Outputs	Pass/Fail	Comments
A2.3	A03	Load a Game from a file which exists	Test.asgf	Screen shot SS01 is displayed	P	
		Load a Game from a file which does not exits	Fail.asgf	Screen shot SS02 is displayed	P	
		Load a Game from a file without an extension	Test	Screen shot SS03 is displayed	P	Actually shows Screen shot SS17.
		Load a Game from a file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	P	
A2.4	A03	Create a new Editor file which does not exist	New.agsf	Screen shot SS04 is displayed	P	As A2.3
		Create a new Editor file which does exist	Test.asgf	Screen shot SS05 is displayed	P	
		Create a new Editor file without an extension	Test	Screen shot SS03 is displayed	P	Actually shows Screen shot SS17. As A2.3
		Create a new Editor file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	P	As A2.3
A2.5	A03	Load an Editor from a file which exists	Test.asgf	Screen shot SS06 is displayed	P	As A2.3
		Load an Editor from a file which does not exits	Fail.asgf	Screen shot SS02 is displayed	P	As A2.3
		Load an Editor from a file without an extension	Test	Screen shot SS03 is displayed	P	Actually shows Screen shot SS17. As A2.3
		Load an Editor from a file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	P	As A2.3

Table 3: Testing table for requirements A for Version 0.7.1 (1.0-beta1)

ID	Requirement	Description	Inputs	Expected Outputs	Pass/Fail	Comments
A2.6	A03	Target word is edited to a non-null String	TestNew	Screen shot SS07 is displayed	P	
		Target word is edited to a null		Screen shot SS08 is displayed	P	
		Target word is edited to a non-String	123	Screen shot SS08 is displayed	F	Expected Fail, due to Java String.
		Target word is edited to a value of another Words	Word1	Screen shot SS09 is displayed	P	
A2.7	A03	Other word is edited to a non-null String	TestOther	Screen shot SS10 is displayed	P	
		Other words is edited to a null		Screen shot SS08 is displayed	P	As A2.6
		Other word is edited to a non-String	123	Screen shot SS08 is displayed	F	As A2.6
		Other word is edited to a value of another Words	Word2	Screen shot SS09 is displayed	P	Expected Fail, due to Java String. As A2.6
A2.8	A03	X Position of the Target Word is edited to an integer	10	Screen shot SS11 is displayed	P	
		Y Position of the Target Word is edited to an integer	10	Screen shot SS12 is displayed	P	As Above.
		X Position of the Target Word is edited to a null		Screen shot SS13 is displayed	P	
		Y Position of the Target Word is edited to a null		Screen shot SS13 is displayed	P	As Above.
		X Position of the Target Word is edited to a non-integer	String	Screen shot SS13 is displayed	P	
		Y Position of the Target Word is edited to a non-integer	String	Screen shot SS13 is displayed	P	As Above.
		X and Y Position of the Target Word is edited to be that of another word	0, 0	Screen shot SS14 is displayed	P	

Table 4: Testing table for requirements A for Version 0.7.2(1.0-beta2)

ID	Requirement	Description	Inputs	Expected Outputs	Pass/Fail	Comments
G2.1	G01	Words are printed on the Game Screen	Test.asgf	Screen shot SS01 is displayed	P	
G2.2	G02	Wrong word is chosen	Estt	Screen shot SS15 is displayed 'badsound.wav' is played	P	Sound is played
G2.3	G03	Target word is chosen	Test	Screen shot SS16 is displayed 'goodsound.wav' is played	P	Sound is played

Table 5: Testing table for requirements G for Version 0.7.3(1.0-beta3)

0.17 Testing Tables

ID	Requirement	Description	Inputs	Expected Outputs	Pass/Fail	Comments
A2.1	A01	The Game and the Editor can be accessed easily	N/A	N/A	P	Both accessible from the main menu.
A2.2	A02	The Game can be played without intervention	N/A	N/A	P	
A3.3	A03	Load a Game from a file which exists	Test.asgf	Screen shot SS01 is displayed	P	
		Load a Game from a file which does not exits	Fail.asgf	Screen shot SS02 is displayed	P	
		Load a Game from a file without an extension	Test	Screen shot SS17 is displayed	P	
		Load a Game from a file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	P	
A3.4	A03	Create a new Editor file which does not exist	New.asgf	Screen shot SS04 is displayed	P	
		Create a new Editor file which does exist	Test.asgf	Screen shot SS05 is displayed	P	.
		Create a new Editor file without an extension	Test	Screen shot SS17 is displayed	P	
		Create a new Editor file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	P	
A3.5	A03	Load an Editor from a file which exists	Test.asgf	Screen shot SS06 is displayed	P	
		Load an Editor from a file which does not exits	Fail.asgf	Screen shot SS02 is displayed	P	
		Load an Editor from a file without an extension	Test	Screen shot SS17 is displayed	P	
		Load an Editor from a file with an incorrect extension	Test.agsf	Screen shot SS03 is displayed	P	As A3.4
A3.6	A03	Target word is edited to a non-null String	TestNew	Screen shot SS07 is displayed	P	
		Target word is edited to a null		Screen shot SS08 is displayed	P	
		Target word is edited to a non-String	123	Screen shot SS08 is displayed	F	Expected fail, due to Java String.
		Target word is edited to a value of another Words	Word1	Screen shot SS09 is displayed	P	
A3.7	A03	Other word is edited to a non-null String	TestOther	Screen shot SS10 is displayed	P	
		Other words is edited to a null		Screen shot SS08 is displayed	P	As A3.6
		Other word is edited to a non-String	123	Screen shot SS08 is displayed	F	Expected fail, due to Java String. As A3.6
		Other word is edited to a value of another Words	Estt	Screen shot SS09 is displayed	P	As A3.6
A3.8	A03	X Position of the Target Word is edited to an integer	10	Screen shot SS11 is displayed	P	
		Y Position of the Target Word is edited to an integer	10	Screen shot SS12 is displayed	P	As Above.
		X Position of the Target Word is edited to a null		Screen shot SS13 is displayed	P	
		Y Position of the Target Word is edited to a null		Screen shot SS13 is displayed	P	As Above.
		X Position of the Target Word is edited to a non-integer	String	Screen shot SS13 is displayed	P	
		Y Position of the Target Word is edited to a non-integer	String	Screen shot SS13 is displayed	P	As Above.
		X and Y Position of the Target Word is edited to be that of another word	0, 0	Screen shot SS14 is displayed	P	

Table 6: Testing table for requirement A for Version 0.7.4 (1.0-beta4)

ID	Requirement	Description	Inputs	Expected Outputs	Pass/Fail	Comments
C2.1	C01	The code is properly commented	N/A	N/A	P	
C2.2	C02	Every Class has the relevant JavaDoc	N/A	N/A	P	
E2.1	E01	The Editor allows a game to be designed	N/A	N/A	P	
E2.2	E02	The Editor allows a game to be saved	N/A	N/A	P	See A3.9
E2.3	E03	The Editor allows a game to be loaded	N/A	N/A	P	See A3.4 and A3.5
E2.4	E04	The Editor allows a target word to be set	N/A	N/A	P	See A3.6 and A3.8
E2.5	E05	The Editor allows other words to be set	N/A	N/A	P	See A3.7 and A3.8
G3.1	G01	Words are printed on the Game Screen	Test.asgf	Screen shot SS01 is displayed	P	
G3.2	G02	Wrong word is chosen	Estt	Screen shot SS15 is displayed ‘badsound.wav’ is played	P	
G3.3	G03	Target word is chosen	Test	Screen shot SS16 is displayed ‘goodsound.wav’ is played	P	
P2.1	P01	Presentation is simple	N/A	N/A	P	
P2.2	P02	Words are printed on the Game Screen	Test.asgf	Screen shot SS01 is displayed	P	See G3.1
P2.3	P03	The Game Window is similar to the mock-up			P	
P2.4	P04	The Editor must have a G.U.I.	N/A	N/A	P	

Table 7: Testing table for requirements C, E, G and P for Version 0.7.4 (1.0-beta04)

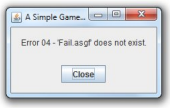
Part VII

Screen shots

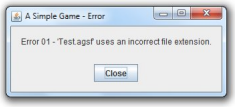
0.18 Testing Screen shots

0.18.1 SS01

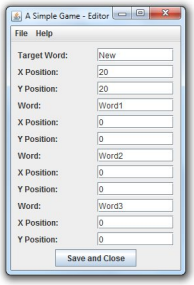
0.18.2 SS02



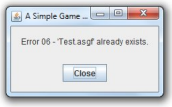
0.18.3 SS03



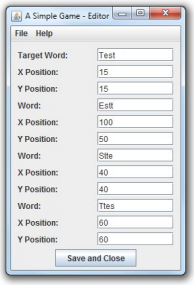
0.18.4 SS04



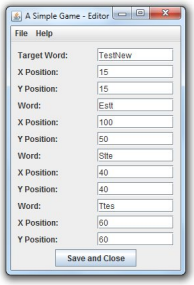
0.18.5 SS05



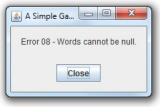
0.18.6 SS06



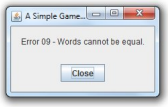
0.18.7 SS07



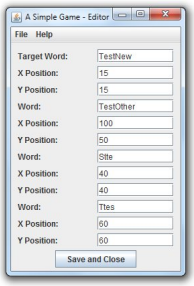
0.18.8 SS08



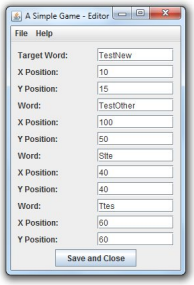
0.18.9 SS09



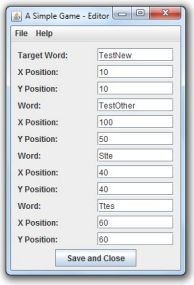
0.18.10 SS10



0.18.11 SS11

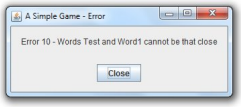


0.18.12 SS12

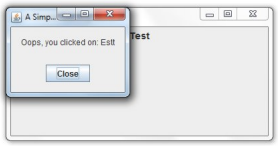


0.18.13 SS13

0.18.14 SS14



0.18.15 SS15



0.18.16 SS16

0.18.17 SS17