

Team 57-

Blackbox Studios

The System, the Process and the Final Printed Report

Name	Student Number	QUB Email Address
Brandon Smylie	40178070	bsmylie02@qub.ac.uk
Conor Clarke	40184143	cclarke44@qub.ac.uk
Edward Muldrew	40176081	emuldrew03@qub.ac.uk
Grace Turkington	40172213	gturkington05@qub.ac.uk
Jack McNaughton	40179912	jmcnaughton01@qub.ac.uk
GitLab Project	csc-lvl2-1718-57 (Team 57)	

Contents

Contents.....	2
Documented interface design.....	5
Splash Screen	5
Menu Screen.....	5
Options Screen.....	6
User Game Play Screen	6
User Place Card	7
User Passes Turn	7
AI Game Play Screen	7
Game Ending Screen	8
Class Relationship Models	9
Use Case Realisations.....	12
Screen Select.....	12
Playing a card	13
Game Progression.....	14
Test Plan.....	15
User Interface Tests	15
Splash Screen	15
Menu Screen.....	16
Options Screen Tests.....	17
Card Demo Screen Tests	18
Round & Game Base Tests	20
AI Tests.....	21
Gitlab.....	23
Appendix	25
Adherence to Process	25
Proof of Testing.....	25
Hand Class Test	25
Card Demo Screen Test.....	26
Card Class Test	27
Turn Class Tests.....	28
Ai Card Demo Screen Tests	29
Game End Screen Test	30
Score Class Test.....	31
Deck Class Tests	32

Utilities Class Test	33
Classes that match the documented design	34
Strength Card Class	35
Minion Card Class.....	36
Hero Card Class	36
Spell Card Class	37
Game Screen	38
Menu Screen.....	39
Options Screen.....	40
Card Demo Screen	41
Overworld Screen	42
Turn class	43
Utilities Class	44
Round Class.....	45
Game Record Class.....	46
AI Demo Screen Class.....	47
Game End Screen Class	48
Round End Screen	49
Team Minutes	50
Meeting 1: Sprint 4, Week 1	51
Task Reporting	51
Actions Planned	51
Meeting 2: Sprint 4, Week 2	52
Task Reporting	52
Actions Planned	53
Meeting 3: Sprint 4, Week 3	54
Task Reporting	54
Actions Planned	55
Meeting 4: Sprint 5, Week 1	56
Task Reporting	56
Actions Planned	57
Task Reporting	58
Actions Planned	59
Meeting 6: Sprint 5, Week 3	60
Task Reporting	60
Actions Planned	61

Meeting 7: Sprint 6, Week 1	62
Task Reporting	62
Actions Planned	63
Meeting 8: Sprint 6, Week 2	64
Task Reporting	64
Actions Planned	65
Meeting 9: Sprint 6, Week 3	66
Task Reporting	66
Actions Planned	67
Meeting 9: Sprint 7, Week 1	68
Task Reporting	68
Actions Planned	69
Meeting 10: Sprint 7, Week 2	70
Task Reporting	70
Actions Planned	71
Peer Assessment.....	72
Evaluation	72
Declaration.....	72

Documented interface design

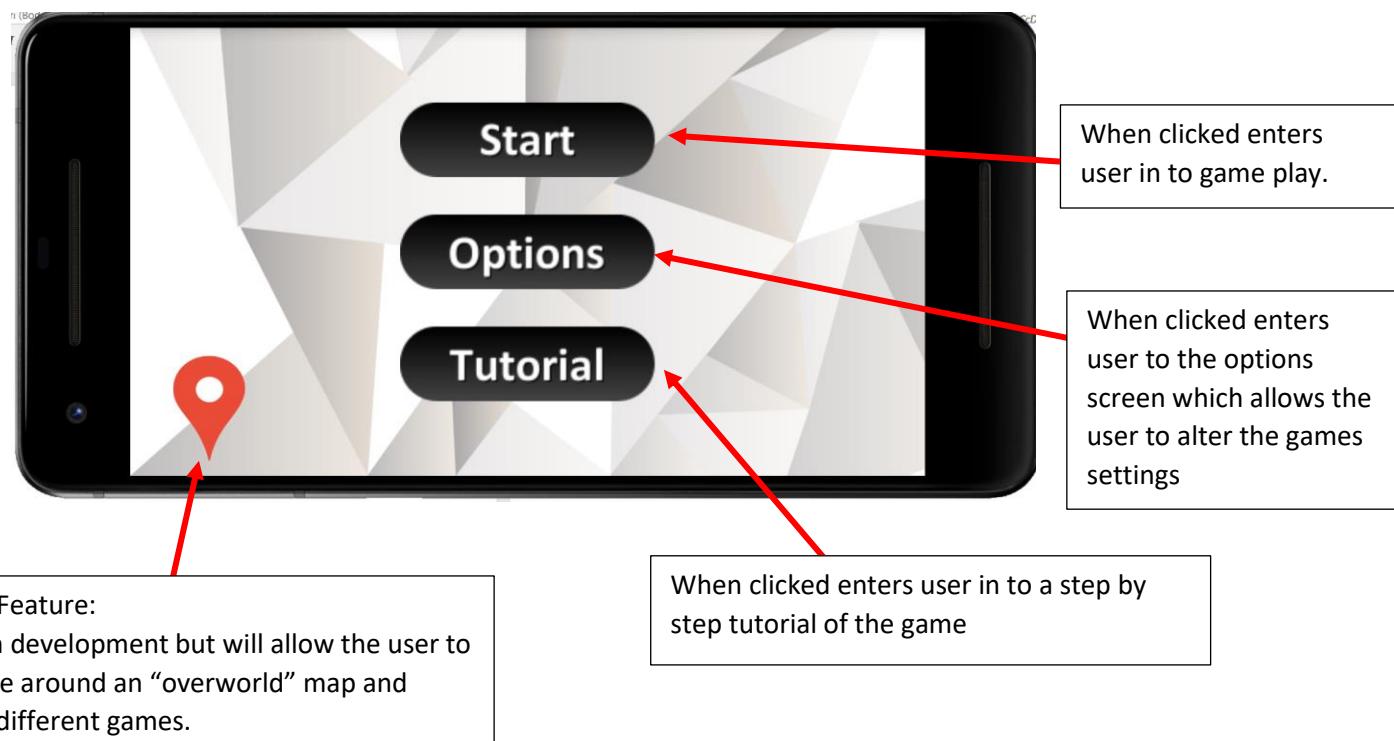
Splash Screen

The splash screen is the first screen the user will see and shows the user the developers of the game. The user simply clicks on this screen to get brought to the options screen.



Menu Screen

The Menu screen carries design consistency from the splash screen. The menu screens provides visible options for the user to alter the course of the game.

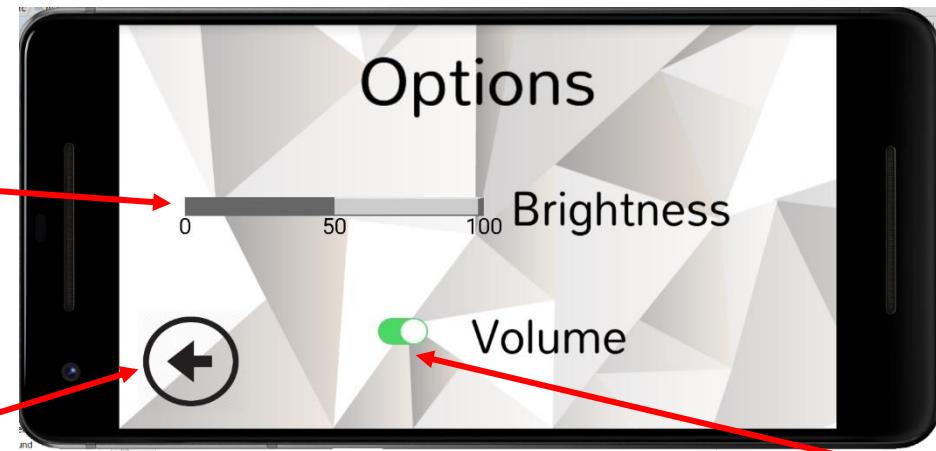


Overworld Feature:

Currently in development but will allow the user to walk a sprite around an “overworld” map and enter in to different games.

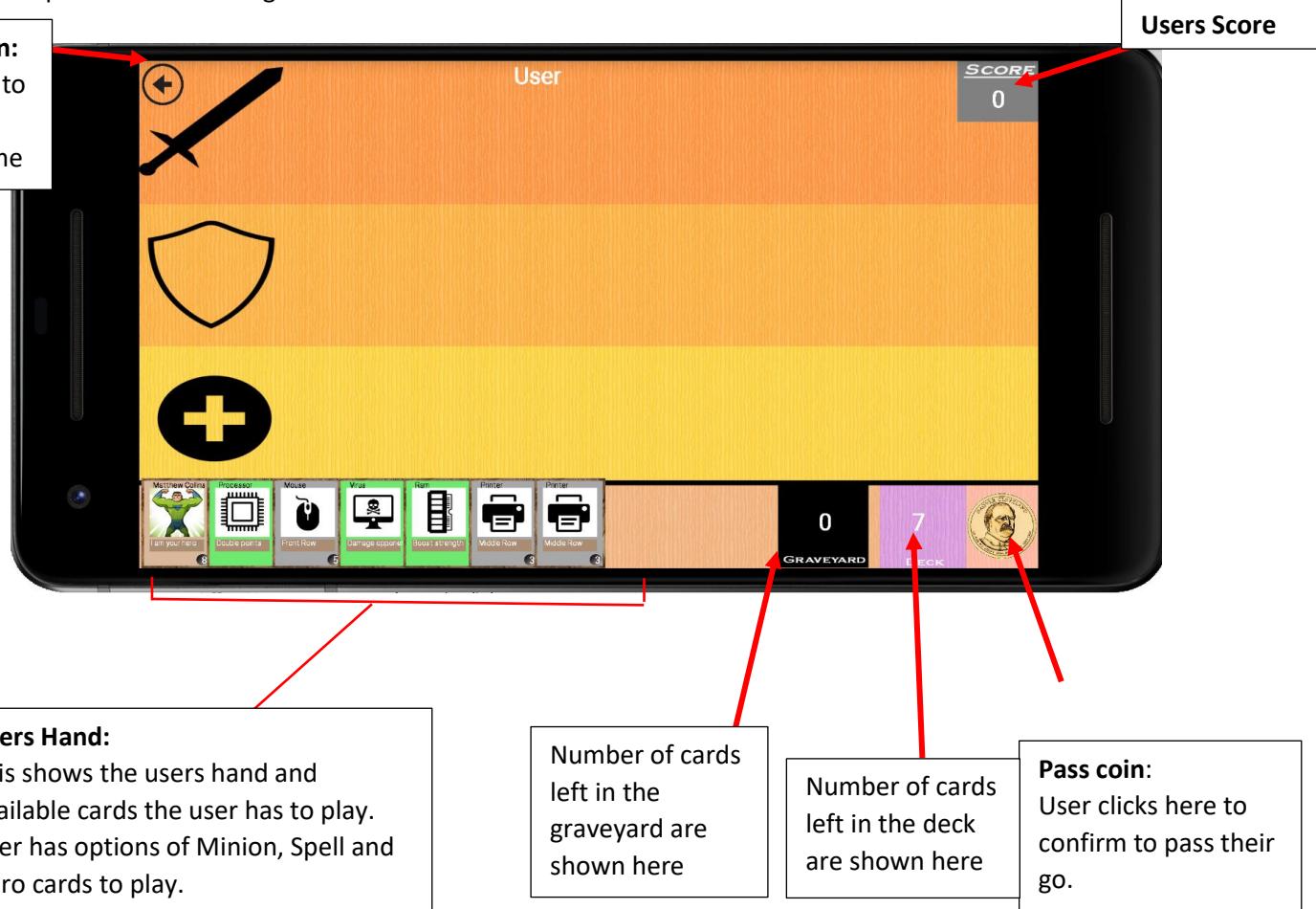
Options Screen

The options screen carries design consistency from the menu screen. The user has the option to adjust the brightness via a slider or turn on and off the volume via a toggle switch. The interface design also consists of clear readable text.



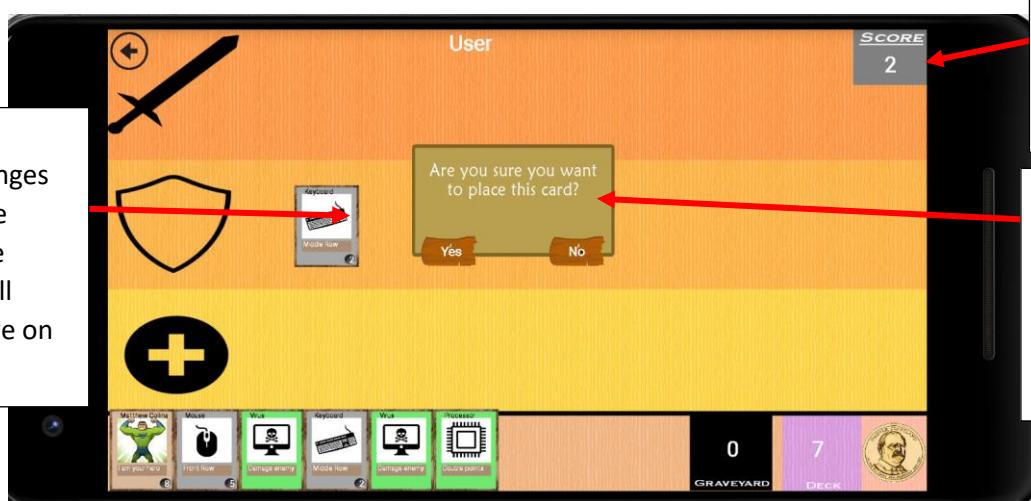
User Game Play Screen

This screen is the screen the game is actually played on. The user can choose to pass goes, place cards all on this screen. Multiple goes and iterations can take place here depending on the AI's response to the users go.



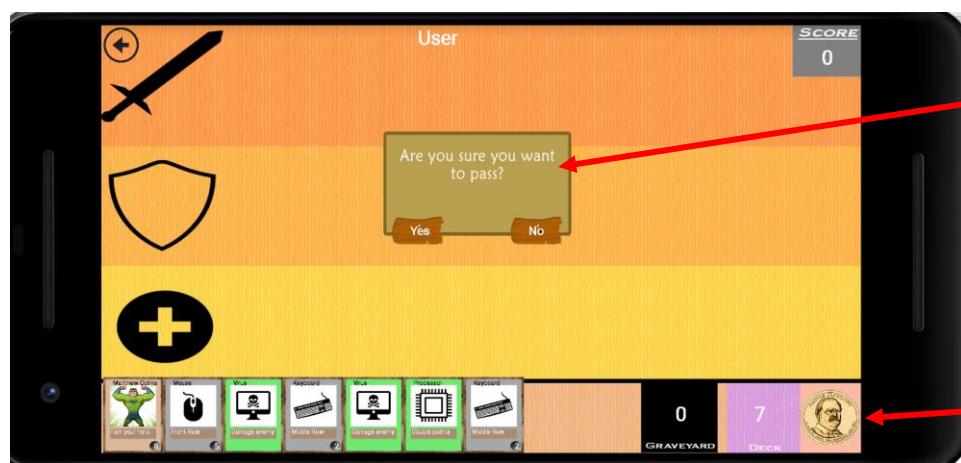
User Place Card

This screen represents what the user will see once they place their card. The score will be updated and the user will be asked if they want to confirm their selection and placement of card.



User Passes Turn

This screen shows what happens when the user chooses to click the pass coin to pass their go. This will mean the AI can choose to either play or pass if the user confirms their pass.



Users Score:

Score will change accordingly to the strength number of card

Prompt:

A prompt appears with two buttons which allows the user to either choose another card to play or to confirm their selection.

AI Game Play Screen

This screen is the AI screen the user will be able to see only the AI's hero card but not the AI's hand. They can only see the backs of the cards.-

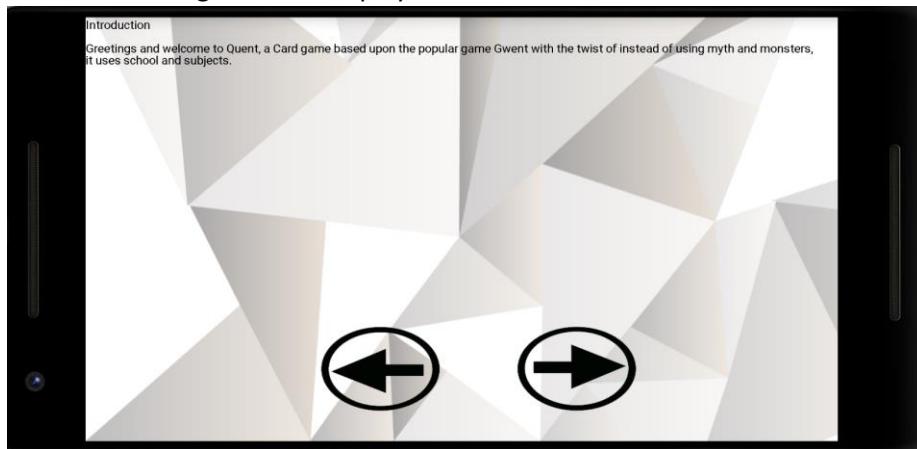


AI Hand:

This shows the users hand and available cards the ai has to play. User only sees back of cards and hero card so turn of play is not altered.

Tutorial Screen

This screen is a tutorial the user can go through by clicking on the right and left arrows. It outlines the rules of the game how to play.



Round ending screen

This screen appears once a round has finished displaying feedback to the user about who won the round and the score of the round. A button is also made available to the user to allow them to create a new round.



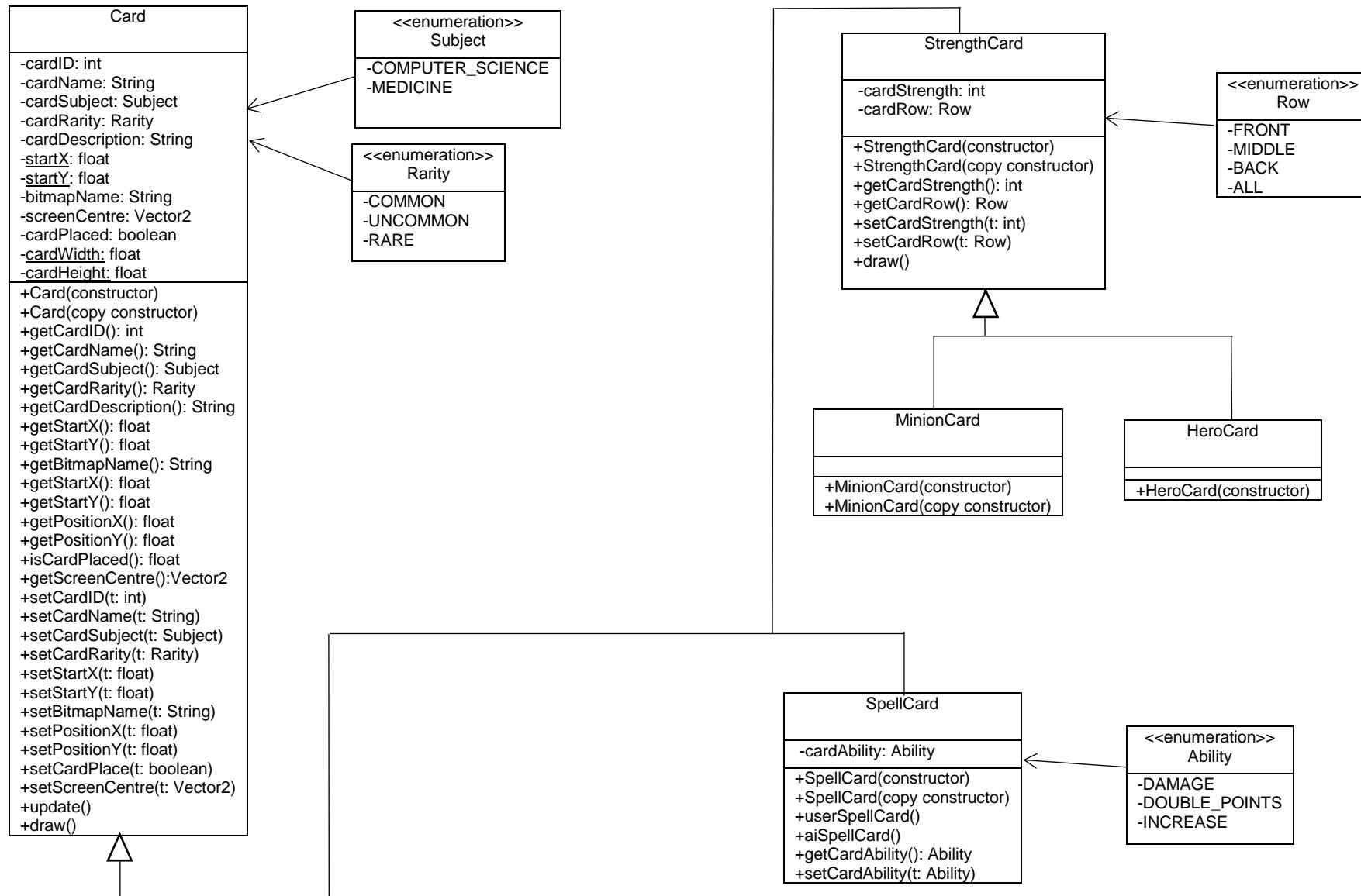
New round button:
Only available selection for the user so that either a new round is created or the game will be ended.

Game Ending Screen

The game ending screen will appear when either the user or the AI has a lead to which the opponent can not come back from. As there is a max possibility of three rounds the User has won in this instance. The game ending screen summarises the scores in each round and the overall score.



Class Relationship Models



This class relationship model represents the various classes involved in the Card superclass and its various subclasses. There are two immediate subclasses: SpellCard and the StrengthCard.

The SpellCard class is intended for cards that do not sit on the board of the game, rather they cast their effect(s) and then they are moved to the graveyard, and as such has an enum called ability attached to it for the time being although this may change to an interface.

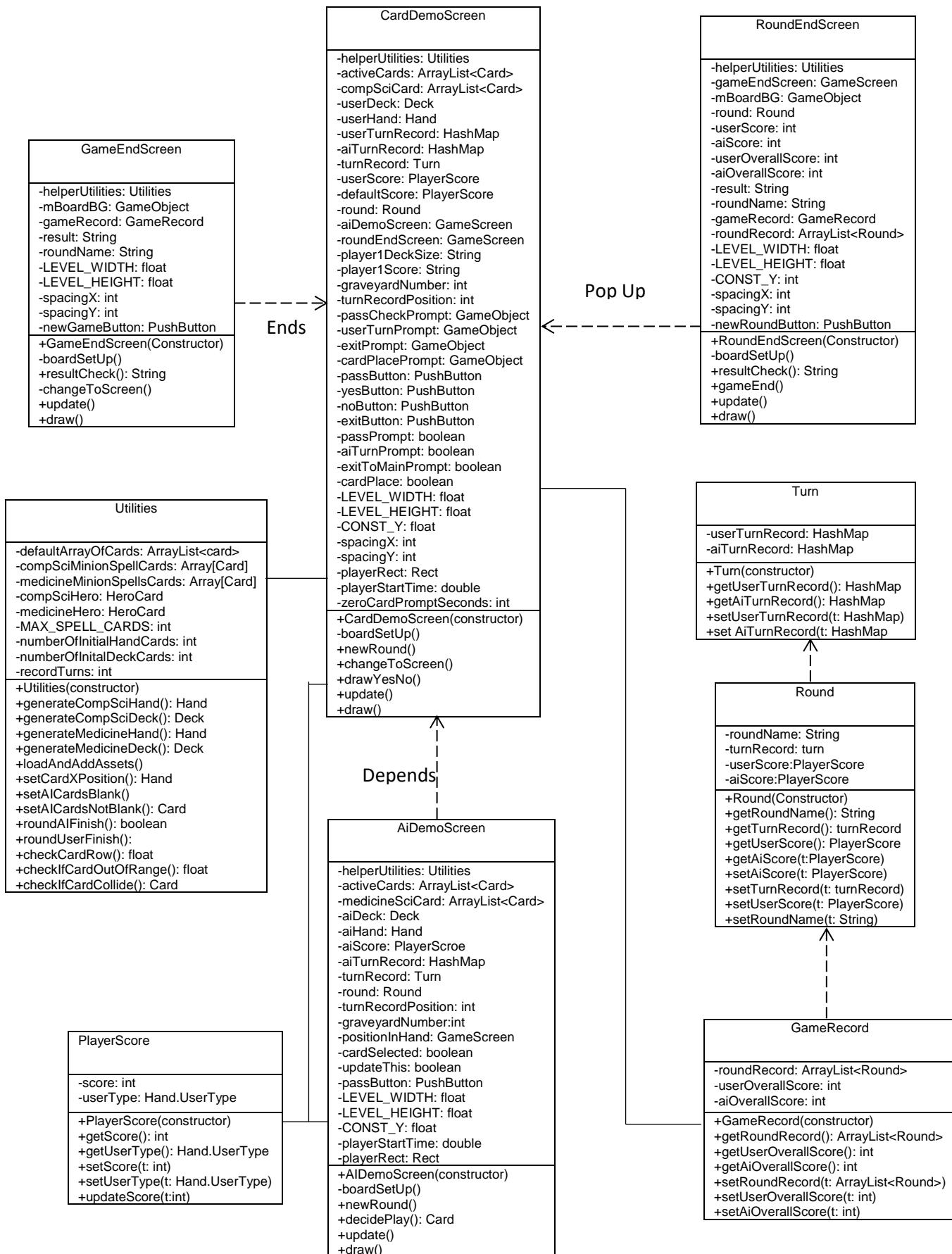
The StrengthCard subclass on the other hand is intended for cards that stay on the board when they are played, i.e. they have a strength attribute. This class contains both the strength of the card as an integer and the row that it may be played it (dictated by the “Row” enumeration). The two subclasses of StrengthCard are MinionCard and HeroCard respectively. MinionCard is a class that represents normal minions that can be found in the deck while the HeroCard class represents heroes which exist outside of the deck and instead on their own in the hero card slot on the board.

The next class relationship model represents the various screens for our actual game. This is a complex model showing the interactions between the game classes.

On the CardDemoScreen, the actual game is displayed to the player, with the board, their new hand and their hero being displayed to them. Much of the game’s interaction and change comes from this class. Various screens and classes are entirely dependent on this screen, most prominently being the AiDemoScreen, which handles the Ai’ board and behaviour. Other screens such as the RoundEndScreen and the GameEndScreen are also dependent on the CardDemoScreen, ending a round and the game respectably when their conditions are met. The PlayerScore class updates the CardDemoScreen’s and AiDemoScreen’s scores.

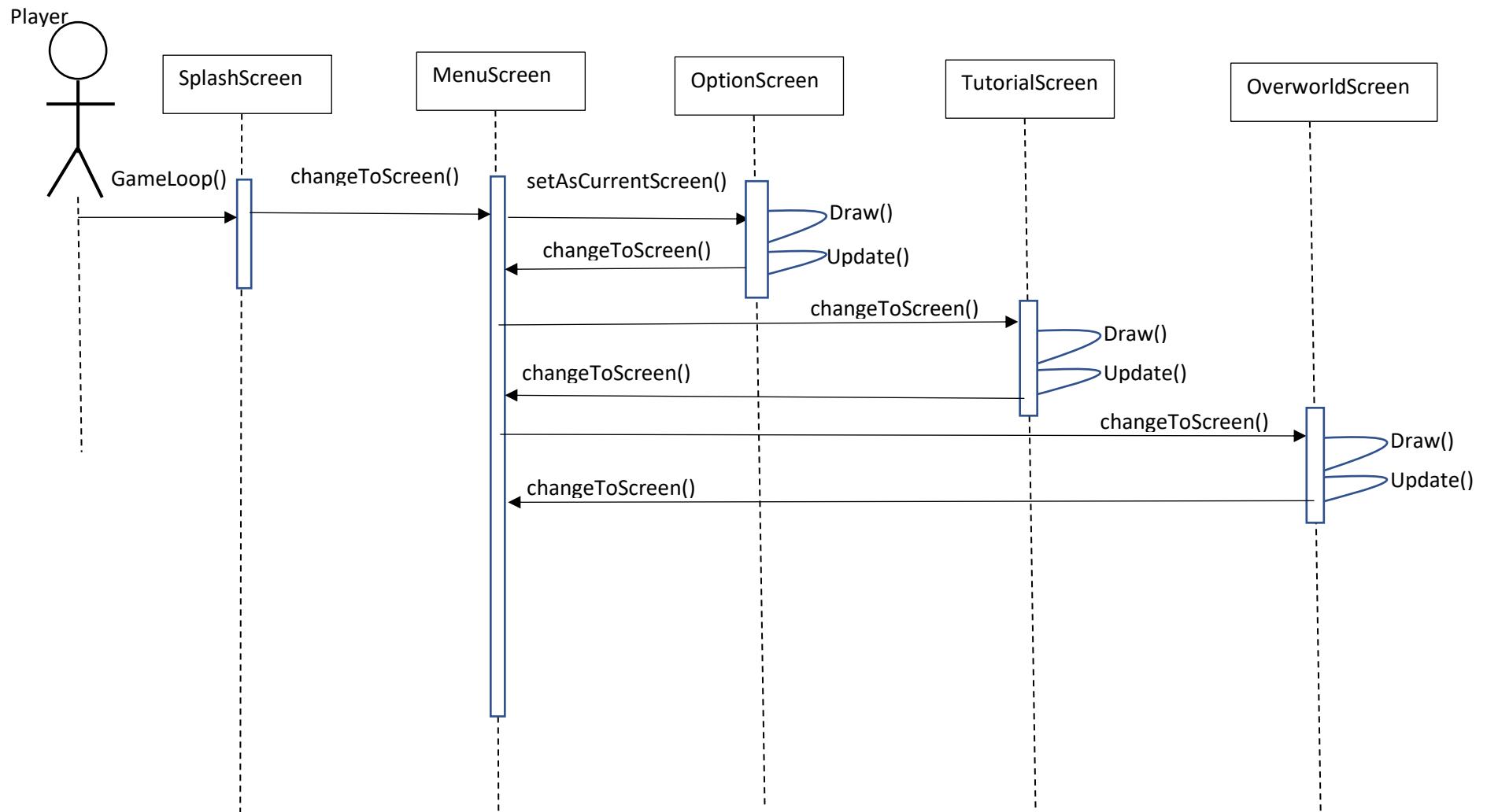
The Utilities class is heavily linked to the CardDemoScreen but is linked to most classes providing useful methods to be used.

Lastly, the GameRecord class, which is dependent on the Round class which in turn is dependent on the Turn class, handles how turns are carried out and calculates whose turn it should be and thus which board should be showing.



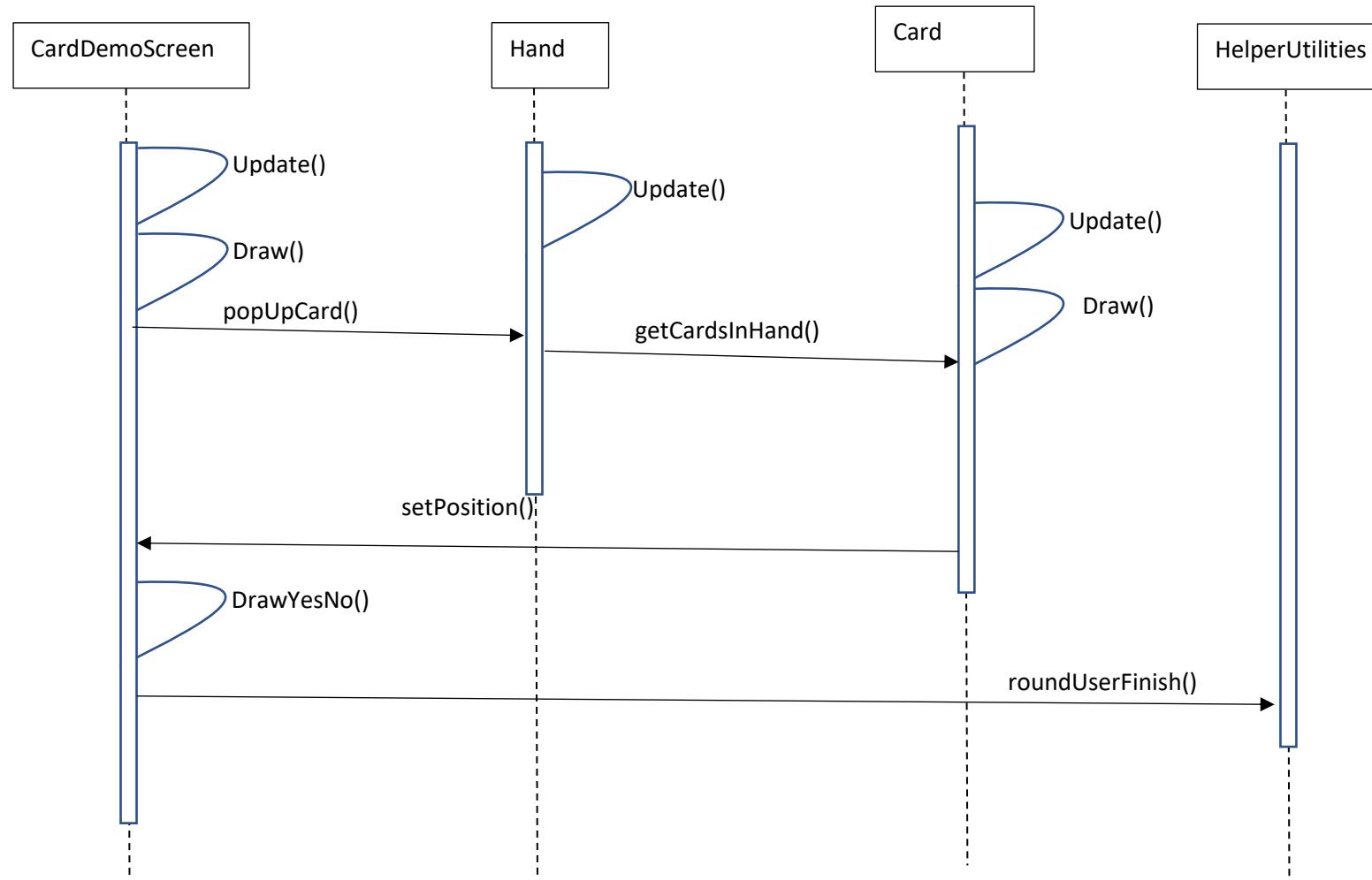
Use Case Realisations

Screen Select



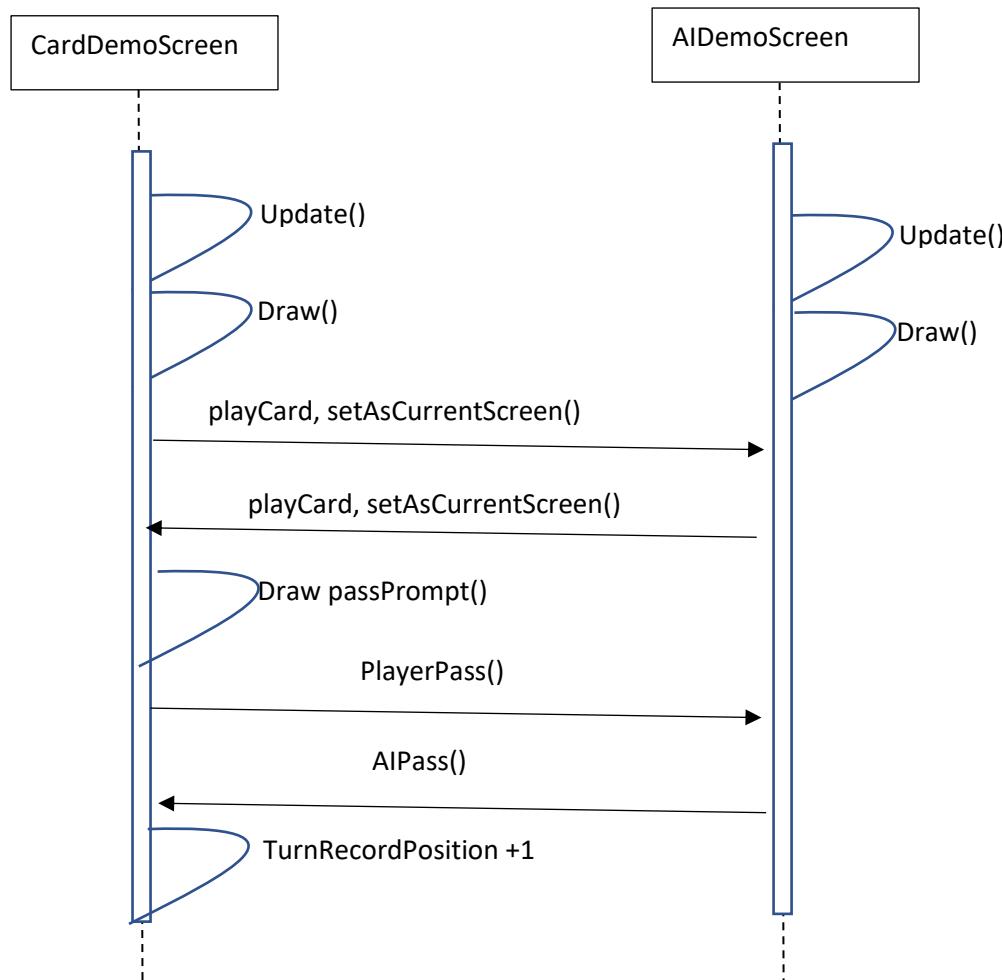
Playing a card

A player clicks on a card, then clicks where they want to place the card, removing the card from the hand and ending the turn if they select Yes when the Yes No screen is drawn by the DrawYesNo section, which is handled by the roundUserFinish method in the helper utilities.



Game Progression

When a card is placed or if the player presses the pass button their turn ends, when both sides pass a round ends, above diagram abbreviated to playCard in below realisation



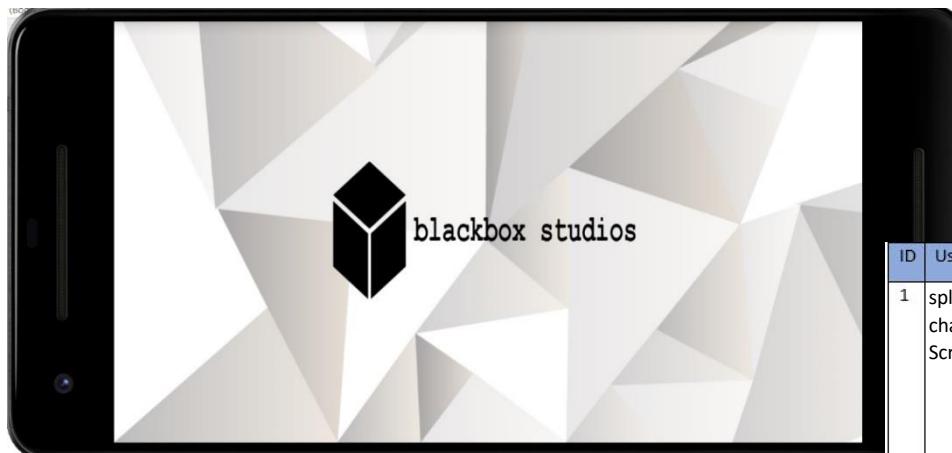
Test Plan

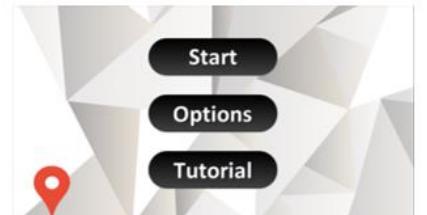
User Interface Tests

Splash Screen

This test plan is used to test if the navigation to the menu screen from the splash screen is correct. This is black box acceptance/unit testing. Unit tests for this class have also been written, as can be seen within the SplashScreenTest class in our project.

A screenshot of the splash screen is shown below, followed by the test plan for this screen.



ID	Use case ref	Description of Test	Execution Steps	Expected Results	Evidence	Passed?
1	splashScreen changeTo Screen()	Testing navigation to the menu screen	Click somewhere random on the screen	You should be directed to the menu screen		✓
2	splashScreen changeTo Screen()	Testing navigation to the menu screen	Repeat test 1 but click on a different location	You should be directed to the menu screen		✓

Menu Screen

This test plan is used to test if the navigation through the menu screen is correct. This is black box acceptance/unit testing. Unit tests for this class have also been written, as can be seen within the MenuScreenTest class in our project.

A screenshot of the menu screen is shown below, followed by the test plan for this screen.

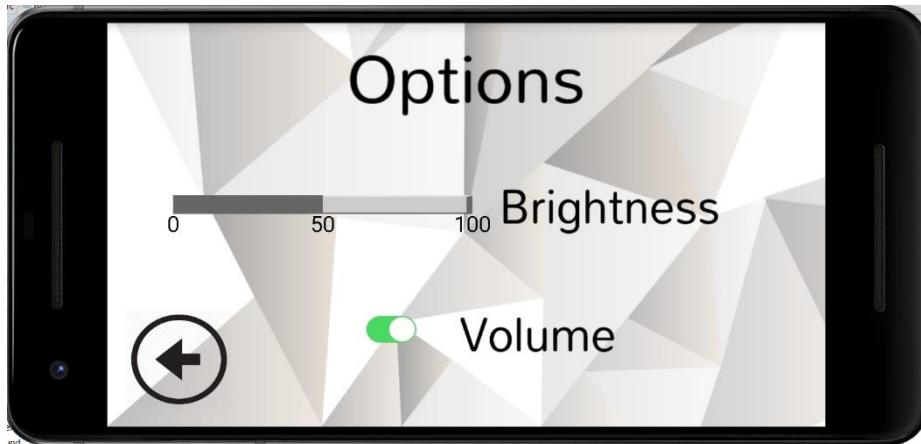


ID	Use case ref	Description of Test	Execution Steps	Expected Results	Evidence	Passed?
1	MenuScreen changeToScreen()	Testing navigation to begin the game	Click the start button	You should be taken to the game screen		✓
2	MenuScreen changeToScreen()	Testing navigation to edit the game's options	Click the options button	You should be taken to the options screen		✓
3	MenuScreen changeToScreen()	Testing navigation to view the game's tutorial	Click the tutorial button	You should be taken to the tutorial screen		✓
4	MenuScreen changeToScreen()	Testing navigation does not occur anywhere you click	Click somewhere random on the screen	You should remain on the menu screen		✓

Options Screen Tests

This test plan is used to test if the navigation and features in the options screen function correctly. This is black box testing. Unit tests for this class have also been written, as can be seen within the OptionsScreenTest and SliderTest classes in our project.

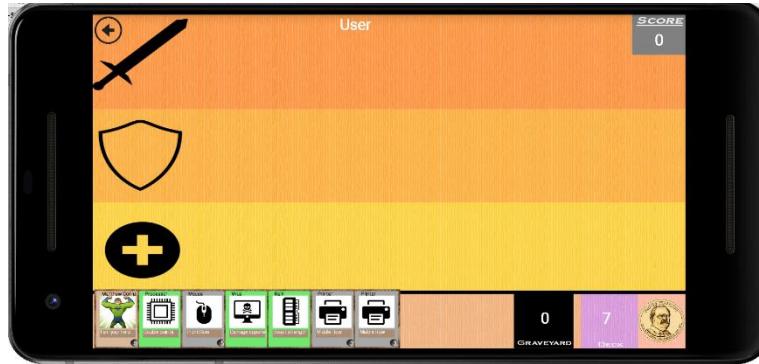
A screenshot of the options screen is shown below, followed by the test plan for this screen.



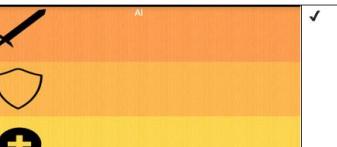
ID	Use case ref	Description of Test	Execution Steps	Expected Results	Evidence	Passed?
1	OptionsScreen changeToScreen()	Test navigation occurs to go back to the main menu	Click the back button	You should be taken to the main menu		✓
2	OptionsScreen	Test brightness slider- when scrolled up and down the brightness of the screen should increase and decrease	Drag the slider up and down	Not yet implemented-expected to fail		X
3	OptionsScreen	Test volume toggle button-when turned/left on the game music should be on	Turn/leave the slider on	Not yet implemented-expected to fail		X
4	OptionsScreen	Test volume toggle button-when turned off the game music should be off	Turn the slider off	Not yet implemented-expected to fail		X

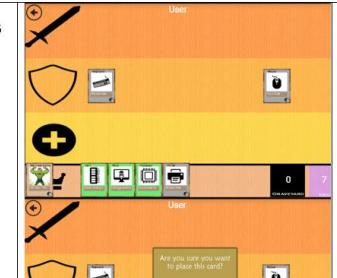
Card Demo Screen Tests

Tests based the users actions through the game screen. This test plan is used to test all the functionality the user using the Card Demo Screen. It a unit test solely testing the core functionality of this screen. Example of game screen below. I will take some extracts from my test plan and will evidence this in the adherence to process through my android tests.



CARD TESTS					
ID	Use case ref	Description of Test	Execution Steps	Expected Results	Evidence
					Passed ?
1	Select Card	Test card selection	Click on one of the available cards in Users hand	Card should 'pop' up to indicate selection has been made.	
2	Drag Card	Test card placement	Click on the board to place card	Card should move place on the board with a prompt asking the user if they wish to confirm their placement and the score should adjust to the cards strength number.	
10.	Draw Cards	Test random user hand generation.	Run the game three different times and test to see if you get a different hand every time. The users hand is randomly generated every game to ensure both user and AI have a fair chance.	The users hand should have a different hand every time consisting of different spell cards and minion cards. The hero card should always remain the same and any hand should not have more than 3 spell cards.	
11.	Draw Cards	Test random AI hand generation	Again similar to the test before. Run the game 3 times and test the AI has been given a randomly generated hand	The AI hand should again have 7 cards and a max of 3 spell cards and 1 consistent hero. The rest will be minion cards	

4.	Place Card	Confirm selection	Select yes to placing the card on the board	Card should become an active card on the board of play and stick to its position, the game should then switch to the <u>opponents</u> view.		✓
5.	Spell Animation	Play a double points spell card	Select and place a double points spell card on a board with a strength card already previously placed	The user's score should double from 3 to 6.		✓

7	Spell Card	Play a "boost" strength spell card	Select and play the "boost strength" spell card on a board with two other strength cards.	The board already has two other strength cards of 2 and 5 creating a user score of 7. The "boost strength" spell card should double the strength of all spell cards therefore their strength attributes should double to 4 and 10. As well as the overall score doubling.		✓
8	Spell Card	Play "damage enemy" spell card	Select and play the "damage enemy" spell card from the user's hand.	Once played the opponents score should minus by 3.		✓

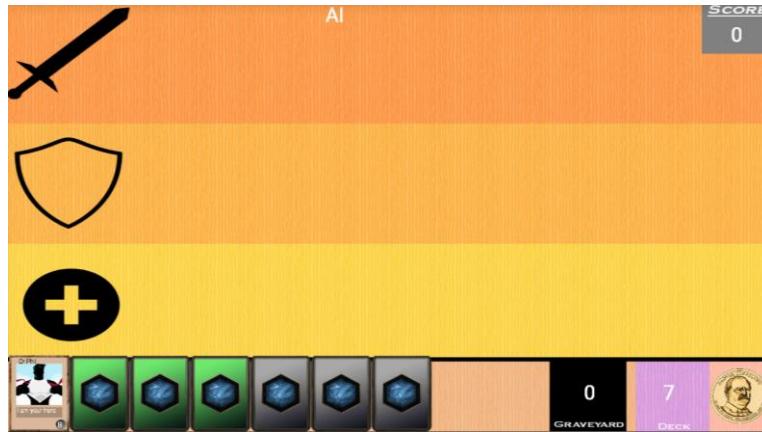
Round & Game Base Tests

Tests based off how the game calculates if a round or game is won. The next set of tests are set out to test the round & game ending mechanism. I will take some extracts from my test plan and will evidence this in the adherence to process through my android tests.

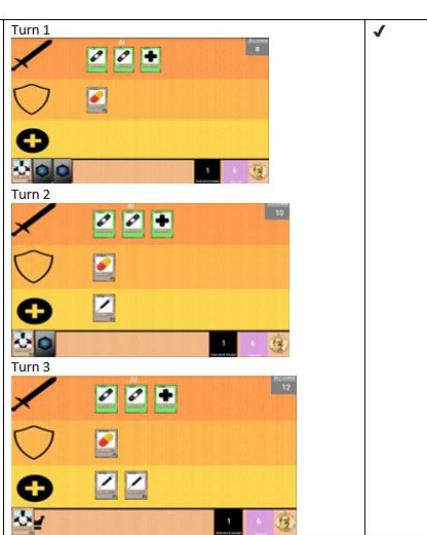
Test Round & Game Ending/Turn Based Mechanism					
ID	Use Case Ref	Test Description	Execution Steps	Expected Results	Evidence
1.	Win/ Lose Animation	User & AI pass their go every time	Ensure the AI & the User Pass every time	3 rounds should pass with 2 round end screens shown before a final game ending screen displaying the result as a tie.	
5.			Test "finish game" button	Continue to the end of the game until you reach the final screen. Press finish game.	
2.		Test new round is working correctly	Play 1 card for the user and then continue to pass for both ai and user until the round has finished and a new round is created.	Ensure User plays 1 card in the first round to test if on the new round both user & AI scores are set to 0 and each player gets a card drawn from their deck. Deck value should go from 7 to 6.	
3.			Test round 3 draws cards from deck	Similarly to the test before, continue to pass until you reach round 3.	

AI Tests

Tests based the AI's actions through the game screen. This test plan is used to test all the functionality the user using the AI Demo Screen. It a unit test solely testing the core functionality of this screen. Example of game screen below. I will take some extracts from my test plan and will evidence this in the adherence to process through my android tests.



AI Demo Screen Tests								
ID	Use Case Ref	Test Description	Execution Steps	Expected Results	Evidence	Passed?		
1.		AI Plays Minion Card first	Play a card and then the AI should play a minion card. I will also test this for a new round	The AI should always play a minion card unless it has no other cards left to play. I		✓		
3.				Test to see if a boost strength spell card manipulates the score.	The score should manipulate based off the spell cards ability/	The strength card of all placed strength cards should <u>double</u> and the score should change.		✓

10.	Test to see if any cards collide	Simulate a few turns to see if any of the placed AI cards collide or on top of each other.	All the placed AI cards should have space between each of the placed cards.		✓	<p>15.</p> <p>Test to see what happens when the AI runs out of cards to play</p> <p>Continue playing until the AI has no cards left to play</p> <p>The AI should continue to play cards until they have to pass their turn which should return to the User and then eventually lead to the round ending.</p> <p>AI plays all AI cards</p> <p>Screen is returned back to User screen</p> <p>Round ending screen appears as players have no available cards left</p> <p>ROUND ENDED IN A Loss! USER 16 AI 20 New Round</p>	✓
-----	----------------------------------	--	---	--	---	---	---

Gitlab

As evidence of a populated Gitlab repository, I have included a couple of screenshots of our commits throughout the project. We have had more commits than shown, but this is to act as an example.

The image displays four separate screenshots of a Gitlab repository interface, each showing a list of commits for a specific branch. The branches shown are:

- csc-lvl2-1718 / csc-lvl2-1718-57**: This branch contains commits related to User Stories 26, 23, 12, and 10, along with various sprint progress and merge requests. One commit notes a problem with reading text documents.
- csc-lvl2-1718 / csc-lvl2-1718-57**: This branch shows a series of commits from January 2018, primarily related to User Stories 17+18, including multiple sprint progress updates and merge requests.
- csc-lvl2-1718 / csc-lvl2-1718-57**: This branch has one commit dated November 2017, which merges the 'master' branch from a local path. It also includes several commits from Brandon James Smylie and Grace Emily Turkington.
- csc-lvl2-1718 / csc-lvl2-1718-57**: This branch shows a series of commits from March 2018, focusing on difficulty enums, unit tests for user stories, and various UI and feature edits. It includes commits from multiple team members including Brandon, Grace, and Conor.

Here is Gitlab evidence of the various branches we used throughout our project.

The screenshot shows the GitLab interface for the repository 'csc-lvl2-1718 / csc-lvl2-1718-57'. The 'Repository' tab is selected. The page displays three branches: 'AiTemp', 'master' (marked as default and protected), and 'revert-12b9e196' (marked as merged). Each branch entry includes a commit history, a merge request button, a compare button, and a delete icon. A search bar at the top right allows filtering by branch name. A green 'New branch' button is also visible.

Branch	Status	Last Commit	Merge Requests	Compare	Delete
AiTemp		5ebd340a · AI screen behaviour added, script to be added later · 4 days ago	6 1	Merge Request Compare	⋮ ⚡
master	default protected	a5600c46 · Add change to forward button to non white background · about 7 hours ago		⋮ ⚡	
revert-12b9e196	merged	29a2737d · Revert "Edits for music" · 2 weeks ago	21 0	Merge Request Compare	⋮ ⚡

Appendix

Adherence to Process

Proof of Testing

Here is some evidence of our testing that has gone on during our project which has been split up in to different test classes.

Hand Class Test

For example here is the hand class test with 7 tests passing with code coverage. These tests test card selection, placement, de-selection, removing cards, placing card with cards available and instantiation.

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Path:** csc-lvl2-1718-57 - [C:\Users\user\Documents\Queens University\Software Engineering\CSC2044-Software Development- Processes and Practice\3.Code\csc-lvl2-1718-57] - [app] - ...app\src\test\java\uk\ac\qub\eeecs\game\HandClassTest.java - Android Studio
- Code Editor:** The editor shows the `HandClassTest.java` file with several test methods and their corresponding Java code.
- Coverage Report:** A coverage tool window titled "Coverage HandClassTest" is open, showing the following data:

Element	Class, %	Method, %	Line, %
Card	100% (3/3)	38% (17/44)	53% (55/103)
Deck	0% (0/1)	0% (0/9)	0% (0/22)
G Hand	100% (2/2)	77% (14/18)	70% (77/109)
HeroCard	100% (1/1)	100% (1/1)	100% (2/2)
MinionCard	100% (1/1)	100% (2/2)	100% (4/4)
SpellCard	66% (2/3)	44% (4/9)	29% (8/27)
StrengthCard	100% (2/2)	55% (5/9)	45% (11/24)
- Run Tab:** Shows the results of the test run: All 7 tests passed - 187ms.
- Bottom Status Bar:** Displays "Tests Passed: 7 passed (moments ago)" and other system information like CPU, RAM, and Git status.

Card Demo Screen Test

Here is another example of testing from the CardDemoScreenTest with 8 tests passing and code coverage shown. These tests check that all AI cards are blank at the start, check if cards are aligned in the user hand correctly, tests turns are managed correctly, checks if appropriate prompts appear when they should, etc.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** Shows the project structure with packages like `app`, `src`, `test`, `java`, `uk.ac.qub.eecs.game`, and `game`. Under `game`, there are several test classes: `RoundClassTest`, `ScoreClassTest`, `ScreenManagerTest`, `SliderTest`, `TurnClassTest`, `UtilitiesClassTest`, and `ScreenTests` which contains `AiDemoScreenTest`, `GameEndScreenTest`, `MenuScreenTest`, `OptionsScreenTest`, `RoundEndScreenTest`, `SplashScreenTest`, `CardDemoScreenTest` (the current file), and `HandClassTest`.
- Coverage Report:** A coverage report titled "Coverage CardDemoScreenTest" is displayed on the right. It shows 52% classes, 32% lines covered in package 'cardDemo'. The report includes a table with columns: Element, Class, %, Method, %, and Line, %. Key data points include:

Element	Class, %	Method, %	Line, %
AI	0% (0/2)	0% (0/16)	0% (0/29)
Cards	85% (12/14)	46% (44/95)	50% (158/310)
Containers	0% (0/10)	0% (0/48)	0% (0/129)
Options	0% (0/2)	0% (0/16)	0% (0/152)
AiDemoScreen	100% (1/1)	20% (1/5)	40% (31/77)
CardDemoScreen	100% (1/1)	28% (2/7)	33% (58/174)
GameEndScreen	100% (1/1)	16% (1/6)	26% (18/68)
PlayerScore	100% (1/1)	50% (3/6)	58% (7/12)
RoundEndScreen	100% (1/1)	16% (1/6)	32% (29/88)
Turn	100% (1/1)	60% (3/5)	71% (10/14)
TutorialScreen	0% (0/1)	0% (0/11)	0% (0/145)
Utilities	100% (1/1)	90% (9/10)	94% (128/136)
- Run Tab:** Shows the test results for `CardDemoScreenTest`:
 - All 8 tests passed - 337ms
 - Test cases: `checkIfAICardsBlank` (142ms), `checkIfDeckSize` (24ms), `checkTurnWhenUserPasses` (27ms), `checkIfCardsArePositionedCorrectly` (23ms), `checkIfYesButtonPushed` (25ms), `checkUserTurnTrueWhenCardPlaced` (22ms), `checkIfPassPromptsUser` (24ms), and `checkIfNoButtonPushed` (50ms).
 - Output log:


```
"C:\Program Files\Android\Android Studio\jre\bin\java" ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
uk.ac.qub.eecs.game\...
exclude patterns:
Process finished with exit code 0
```
- Bottom Status Bar:** Shows "Tests Passed: 8 passed (moments ago)" and the system status: 16:22 CRLF: UTF-8 Git: master Context: <no context>

Card Class Test

Here is another example of testing from the CardClassTest with 6 tests passing and code coverage shown. These tests test the inheritance hierarchy, all cards x position are equally aligned, the strength variable has been converted correctly, cards are constructed properly and bitmaps can be changed properly.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "csc-lvl2-1718-57". It contains packages "gage" and "game". Under "game", there are test classes: CardClassTest, CardDemoScreenTest, and HandClassTest. There are also utility files: .gitignore, app.iml, build.gradle, and proguard-rules.pro.
- Code Editor:** The editor shows the CardClassTest.java file with the following code:

```

CardClassTest(cardClassTestConstruction())
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151

// This test is to check if the inheritance hierarchy is working correctly that
// is an instance of the Card Class
@Test
public void checkCardIsInstanceOfParentClass() {
    assertTrue(testMinion instanceof Card);
}

@Test
public void checkIfGetXMethodWorks() {
    assertEquals(testMinion.getStartX(), actual: 20.0f, delta: 2.0f);
    testMinion.setStartX(40);
    assertEquals(testMinion.getStartX(), actual: 40, delta: 2.0f);
}

@Test
public void checkSetBitmapMethodWork() {
    // assertEquals(testMinion.getBitmap(), "PrinterMinion");
    testMinion.setImageBitmapName("MouseMinion", gameScreen);
    assertEquals(testMinion.getBitmap(), game.getAssetManager().getBitmap(assetName));
    assertEquals(testMinion.getBitmapName(), actual: "MouseMinion");
}

```

- Coverage Report:** A coverage report titled "Coverage cardDemoScreenTest" is displayed on the right. It shows 92% classes and 48% lines covered in the package "Cards". The report includes a table with columns: Element, Class, %, Method, %, and Line, %.

Element	Class, %	Method, %	Line, %
Card	100% (3/3)	60% (27/45)	51% (75/147)
Deck	100% (1/1)	22% (2/9)	31% (7/22)
Hand	100% (2/2)	65% (13/20)	55% (71/127)
HeroCard	100% (1/1)	100% (1/1)	100% (2/2)
MinionCard	100% (1/1)	100% (2/2)	100% (4/4)
SpellCard	66% (2/3)	45% (5/11)	21% (9/42)
StrengthCard	100% (2/2)	55% (5/9)	48% (12/25)

- Run Tab:** Shows the results of the test run: "All 6 tests passed - 139ms". The log output is as follows:

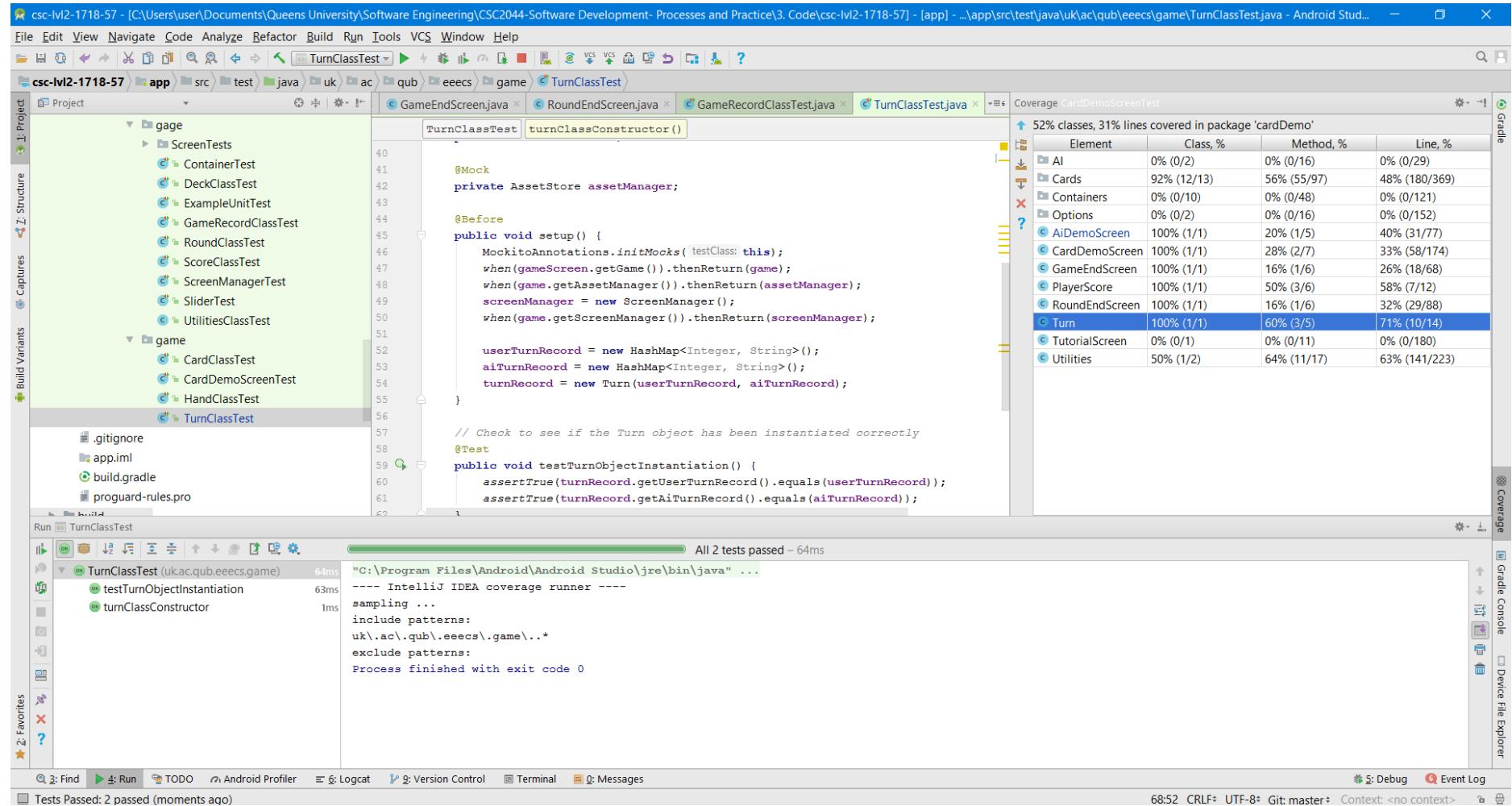
```

"C:\Program Files\Android\Android Studio\jre\bin\java" ...
--- IntelliJ IDEA coverage runner ---
sampling ...
include patterns:
uk.ac.qub.eeecs.game...
exclude patterns:
Process finished with exit code 0

```

Turn Class Tests

Here is another example of testing the turn class method with 2 tests passing and code coverage shown below. This test ensures the turn class is created and instantiated correctly.



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "csc-lvl2-1718-57". The "app" module contains "src", "test", and "java" directories. The "test/java" directory contains "uk.ac.qub.eeecs.game" and "TurnClassTest.java".
- Code Editor:** The editor shows the content of `TurnClassTest.java`. It includes two test methods: `turnClassConstructor()` and `testTurnObjectInstantiation()`. The code uses Mockito annotations like `@Mock`, `@Before`, and `@Test`.
- Coverage:** A coverage report titled "Coverage: cardDemoScreenTest" is displayed on the right. It shows 52% classes, 31% lines covered in package 'cardDemo'. The report includes a table with columns: Element, Class, %, Method, %, and Line, %.
- Run Tab:** The "Run" tab shows the results of the test run: "All 2 tests passed - 64ms". The log output shows the coverage runner command and patterns used.
- Bottom Bar:** The bottom bar includes tabs for Find, Run, TODO, Android Profiler, Logcat, Version Control, Terminal, and Messages. It also shows the status "Tests Passed: 2 passed (moments ago)" and the time "68:52".

Element	Class, %	Method, %	Line, %
AI	0% (0/2)	0% (0/16)	0% (0/29)
Cards	92% (12/13)	56% (55/97)	48% (180/369)
Containers	0% (0/10)	0% (0/48)	0% (0/121)
Options	0% (0/2)	0% (0/16)	0% (0/152)
AiDemoScreen	100% (1/1)	20% (1/5)	40% (31/77)
CardDemoScreen	100% (1/1)	28% (2/7)	33% (58/174)
GameEndScreen	100% (1/1)	16% (1/6)	26% (18/68)
PlayerScore	100% (1/1)	50% (3/6)	58% (7/12)
RoundEndScreen	100% (1/1)	16% (1/6)	32% (29/88)
Turn	100% (1/1)	60% (3/5)	71% (10/14)
TutorialScreen	0% (0/1)	0% (0/11)	0% (0/180)
Utilities	50% (1/2)	64% (11/17)	63% (141/223)

Ai Card Demo Screen Tests

This class tests the AI and has 7 tests with code coverage. The tests test switching between appropriate screens, score change, new round method, if the user passes, if a card has successfully been placed on the board, etc.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "csc-lvl2-1718-57". The "src/main/java/uk/ac/qub/eeecs/game/cardDemo" package contains several test classes under "ScreenTests": DeckClassTest, ExampleUnitTest, GameRecordClassTest, RoundClassTest, ScoreClassTest, ScreenManagerTest, SliderTest, AiDemoScreenTest, GameEndScreenTest, MenuScreenTest, OptionsScreenTest, RoundEndScreenTest, SplashScreenTest, CardClassTest, CardDemoScreenTest, HandClassTest, TurnClassTest, and UtilitiesClassTest.
- Code Editor:** The code editor shows the implementation of the `update()` method in `AiDemoScreen.java`. The code handles user turns, card placement, and round transitions.
- Coverage:** A coverage report on the right side shows the following data:

Element	Class, %	Method, %	Line, %
AI	0% (0/2)	0% (0/16)	0% (0/29)
Cards	92% (12/13)	56% (55/97)	48% (180/369)
Containers	0% (0/10)	0% (0/48)	0% (0/121)
Options	0% (0/2)	0% (0/16)	0% (0/152)
AiDemoScreen	100% (1/1)	20% (1/5)	40% (31/77)
CardDemoScreen	100% (1/1)	28% (2/7)	33% (58/174)
GameEndScreen	100% (1/1)	16% (1/6)	26% (18/68)
PlayerScore	100% (1/1)	50% (3/6)	58% (7/12)
RoundEndScreen	100% (1/1)	16% (1/6)	32% (29/88)
Turn	100% (1/1)	60% (3/5)	71% (10/14)
TutorialScreen	0% (0/1)	0% (0/11)	0% (0/180)
Utilities	50% (1/2)	64% (11/17)	63% (141/223)

- Test Results:** The test results show 7 tests passed in 340ms. The tests are:

 - checkIfYesButtonPushed (135ms)
 - testIfCardPlacedOnBoard (56ms)
 - testAINewRound (27ms)
 - testIfAIPasses (27ms)
 - checkMinionCardPlayedFirst (23ms)
 - testIfAIPlays (33ms)
 - testIfAIPlaysAllCards (39ms)

- Bottom Status Bar:** Shows "All 7 tests passed - 340ms", "Process finished with exit code 0", and other standard Android Studio status indicators.

Game End Screen Test

Here is the game end screen test which specifically tests the GameEndScreen class. The Unit test has two tests testing the game screen instantiation and testing the result check method.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "csc-lvl2-1718-57". It contains packages "uk.ac.qub.eeecs.gage" and "uk.ac.qub.eeecs.game". Under "game", there is a "ScreenTests" folder containing several test classes: "AiDemoScreenTest", "GameEndScreenTest", "MenuScreenTest", "RoundEndScreenTest", "SplashScreenTest", "CardClassTest", "CardDemoScreenTest", "HandClassTest", "TurnClassTest", and "UtilitiesClassTest".
- Code Editor:** The "GameEndScreenTest.java" file is open. It contains Java code for testing the "GameEndScreen" class. Two test methods are shown:
 - `@Test` public void testNewGameButton() { ... }
 - `@Test` public void testResultCheckMethod() { ... }
- Coverage Analysis:** A coverage report for "CardDemoScreenTest" is displayed on the right. It shows 50% classes, 28% lines covered in package 'uk.ac.qub.eeecs.game'. The report includes a table with columns: Element, Class, %, Method, %, and Line, %.
- Run Tab:** Shows the results of the run: All 2 tests passed - 97ms. The log output includes:


```
include patterns:
uk\.\ac\.\qub\.\eeecs\.\gage\..\*
exclude patterns:
Process finished with exit code 0
```
- Bottom Navigation:** Includes tabs for Find, Run, TODO, Android Profiler, Logcat, Version Control, Terminal, Messages, Debug, and Event Log.

Score Class Test

This is the score class test which is specifically testing only the score class. It has 3 tests testing the score objects instantiation, setters and score change method.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "csc-lvl2-1718-57". The "src" folder contains packages "uk.ac.qub.eeecs.gage" and "game". "uk.ac.qub.eeecs.gage" contains classes like ContainerTest, DeckClassTest, ExampleUnitTest, GameRecordClassTest, OptionsScreenTest, RoundClassTest, ScoreClassTest, ScreenManagerTest, and SliderTest. "game" contains ScreenTests with classes like AiDemoScreenTest, GameEndScreenTest, MenuScreenTest, RoundEndScreenTest, SplashScreenTest, CardClassTest, CardDemoScreenTest, HandClassTest, TurnClassTest, and UtilitiesClassTest.
- ScoreClassTest.java:** This is the active file in the editor. It contains three test methods: testScoreInstantiation, checkScoreChange, and checkSetScore. The code uses JUnit annotations (@Test) and assertions (assertEquals).
- Coverage Analysis:** A coverage report on the right side shows 52% classes and 31% lines covered. The report includes a table with columns: Element, Class, %, Method, %, and Line, %. The table lists various classes and their coverage metrics.
- Run Tab:** Shows the results of the last run: "All 3 tests passed – 91ms". The test results are listed under "ScoreClassTest (uk.ac.qub.eeecs.gage)" with times: checkSetScore (89ms), checkScoreChange (1ms), and testScoreInstantiation (1ms). The output shows include and exclude patterns, and the message "Process finished with exit code 0".
- Bottom Navigation:** Includes tabs for Find, Run, TODO, Android Profiler, Logcat, Version Control, Terminal, Messages, Debug, and Event Log.

```

59
60     testScore = new PlayerScore(score: 0, Hand.UserType.USER);
61 }
62
63 // Test to check if test score variable has been instantiated correctly using @Test
64
65 @Test
66 public void testScoreInstantiation() {
67     assertEquals(testScore.getUserType(), Hand.UserType.USER);
68     assertEquals(testScore.getScore(), actual: 0);
69 }
70
71 // Test to see if score changes when it is updated
72
73 @Test
74 public void checkScoreChange() {
75     assertEquals(testScore.getScore(), actual: 0);
76     testScore.updateScore(updatedScore: 10);
77     assertEquals(testScore.getScore(), actual: 10);
78 }
79
80 // Test to see if set score method works
81
82 @Test
83 public void checkSetScore() {
84     assertEquals(testScore.getScore(), actual: 0);
85     int scoreToBeSet = 20;
86     testScore.setScore(scoreToBeSet);
87     assertEquals(testScore.getScore(), scoreToBeSet);
88 }
89

```

Element	Class, %	Method, %	Line, %
AI	0% (0/2)	0% (0/16)	0% (0/29)
Cards	92% (12/13)	56% (55/97)	48% (180/369)
Containers	0% (0/10)	0% (0/48)	0% (0/121)
Options	0% (0/2)	0% (0/16)	0% (0/152)
AiDemoScreen	100% (1/1)	20% (1/5)	40% (31/77)
CardDemoScreen	100% (1/1)	28% (2/7)	33% (58/174)
GameEndScreen	100% (1/1)	16% (1/6)	26% (18/68)
PlayerScore	100% (1/1)	50% (3/6)	58% (7/12)
RoundEndScreen	100% (1/1)	16% (1/6)	32% (29/88)
Turn	100% (1/1)	60% (3/5)	71% (10/14)
TutorialScreen	0% (0/1)	0% (0/11)	0% (0/180)
Utilities	50% (1/2)	64% (11/17)	63% (141/223)

Deck Class Tests

This is the deck class test which specifically tests the deck class test. This unit test tests adding and removing cards to and from the deck to the hand, deck instantiation, etc

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "csc-lvl2-1718-57". It contains packages "uk.ac.qub.eeecs" and "game". Under "game", there are several test classes: "DeckClassTest", "CardClassTest", "CardDemoScreenTest", "HandClassTest", "TurnClassTest", and "UtilitiesClassTest".
- Code Editor:** The "DeckClassTest.java" file is open. It contains Java code for testing the "Deck" class. The code includes methods for checking card addition and removal from the deck.
- Coverage Analysis:** A coverage report for "CardDemoScreenTest" is displayed on the right. It shows 92% classes, 51% lines covered in package 'Cards'. The report includes a table with columns: Element, Class, %, Method, %, and Line, %.
- Run Tab:** The "DeckClassTest" test is selected, and the output shows "All 4 tests passed - 113ms" and "Process finished with exit code 0".
- Bottom Status Bar:** The status bar indicates "Tests Passed: 4 passed (moments ago)" and "108:56 CRLF: UTF-8 Git: master Context: <no context>".

```

92% classes, 51% lines covered in package 'Cards'
Element      Class, %    Method, %    Line, %
Card          100% (3/3)   62% (28/45)  57% (85/147)
Deck          100% (1/1)   88% (8/9)   77% (17/22)
Hand          100% (2/2)   77% (17/22)  49% (81/163)
HeroCard      100% (1/1)   100% (1/1)  100% (2/2)
MinionCard    100% (1/1)   100% (2/2)  100% (4/4)
SpellCard     66% (2/3)   45% (5/11)  21% (9/42)
StrengthCard  100% (2/2)   55% (5/9)   48% (12/25)

```

Utilities Class Test

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays `UtilitiesClassTest.java` with several test methods. A coverage report is open on the right side, showing the following data:

Element	Class, %	Method, %	Line, %
AI	0% (0/2)	0% (0/16)	0% (0/29)
Cards	92% (12/13)	67% (67/99)	52% (211/405)
Containers	0% (0/10)	0% (0/48)	0% (0/121)
Options	0% (0/2)	0% (0/16)	0% (0/152)
AiDemoScreen	100% (1/1)	16% (1/6)	30% (48/159)
CardDemoScreen	100% (1/1)	28% (2/7)	32% (73/225)
GameEndScreen	100% (1/1)	16% (1/6)	26% (18/68)
PlayerScore	100% (1/1)	50% (3/6)	58% (7/12)
RoundEndScreen	100% (1/1)	16% (1/6)	32% (29/88)
Turn	100% (1/1)	60% (3/5)	71% (10/14)
TutorialScreen	0% (0/1)	0% (0/11)	0% (0/180)
Utilities	100% (2/2)	76% (13/17)	68% (170/248)

The code editor shows the following Java code:

```

68     utilitiesTester.loadAndAddAssets(game);
69     cardDemoScreen = new CardDemoScreen(game, name: "cardDemoScreen");
70     game.getScreenManager().addScreen(cardDemoScreen);
71     game.getScreenManager().setAsCurrentScreen("cardDemoScreen");
72     testCard = new MinionCard( cardID: 1, cardName: "TestCard", Card.Subject.COMPUTER,
73                               Card.Rarity.COMMON, cardDescription: "TestCard", bitmapName: "syringeMinion" );
74   }
75
76   @Test()
77   public void testRoundUserFinish() {
78     assertEquals(game.getScreenManager().getCurrentScreen().getName(), actual: "Turn");
79     turnRecord.getUserTurnRecord().put(0, "true");
80     turnRecord.getAiTurnRecord().put(0, "false");
81     turnRecord.getUserTurnRecord().put(1, "false");
82     utilitiesTester.roundUserFinish(turnRecord, 1, game);
83     assertEquals(game.getScreenManager().getCurrentScreen().getName(), actual: "RoundEndScreen");
84   }
85
86   @Test()
87   public void testCardRowChangeMethod() {
88     assertEquals(testCard.getPositionY(), setPositionY, delta: 1.0);
89     float returnedY = utilitiesTester.checkCardRow(testCard, testCard.getPositionX());
90     // As test card is a minion card with there row set to middle the returned
91     // position is 155
92     assertEquals(returnedY, actual: 155, delta: 1.0);
93   }
94 }

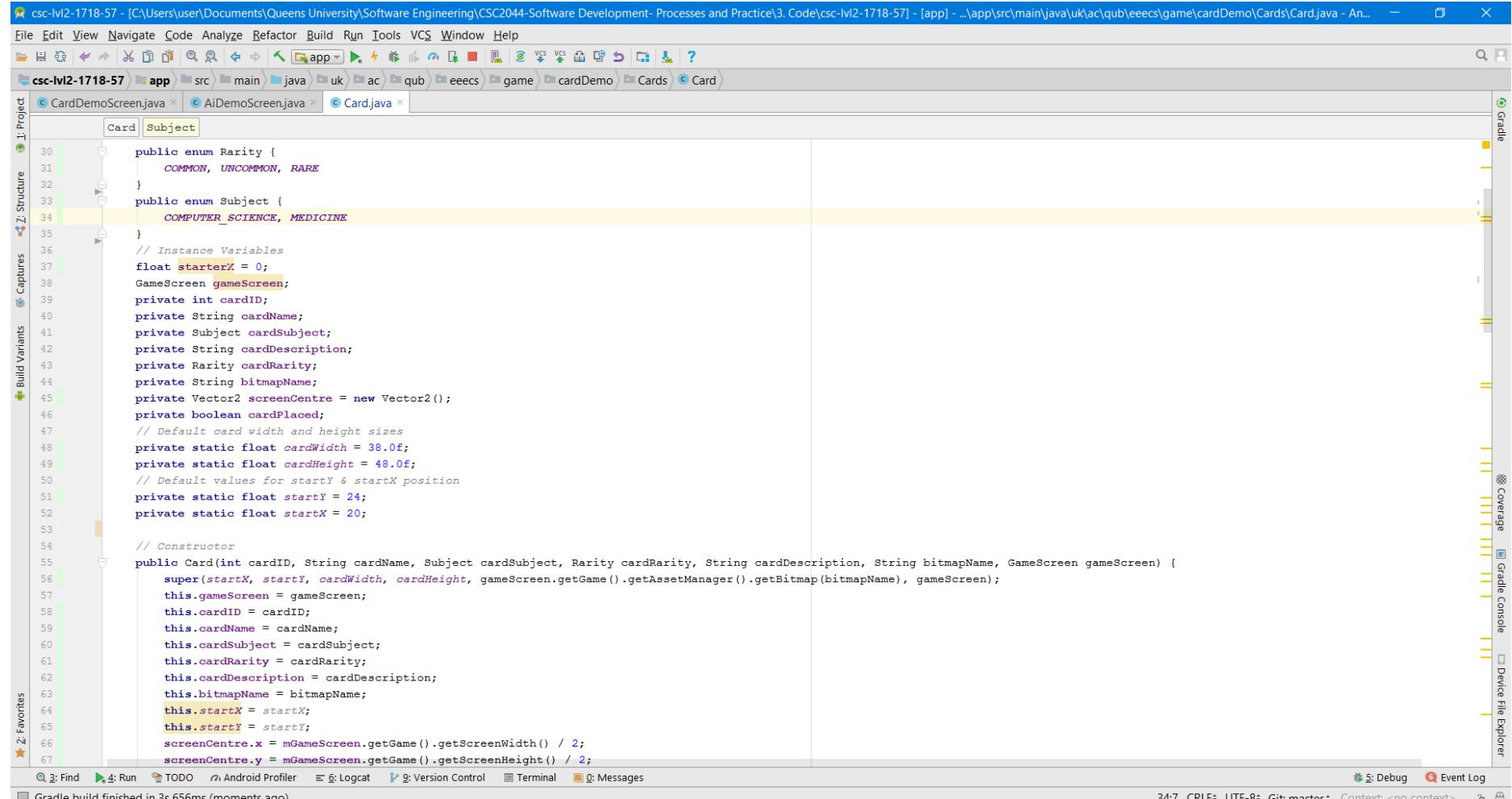
```

The bottom of the screen shows the run results: All 2 tests passed - 234ms.

Classes that match the documented design

Here is evidence of our class relationship model which represents the various classes involved in the Card superclass and its various subclasses

Card Class



The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it, the code editor displays the `Card.java` file under the `CardDemoScreen.java` tab. The code defines a class `Card` with enumerations for Rarity and Subject, and several private instance variables. It includes a constructor that initializes these variables and sets up the card's position and appearance. The code editor has syntax highlighting and code completion features visible.

```

public enum Rarity {
    COMMON, UNCOMMON, RARE
}

public enum Subject {
    COMPUTER_SCIENCE, MEDICINE
}

// Instance Variables
float starterX = 0;
GameScreen gameScreen;
private int cardID;
private String cardName;
private Subject cardSubject;
private String cardDescription;
private Rarity cardRarity;
private String bitmapName;
private Vector2 screenCentre = new Vector2();
private boolean cardPlaced;

// Default card width and height sizes
private static float cardWidth = 38.0f;
private static float cardHeight = 48.0f;
// Default values for startY & startX position
private static float startY = 24;
private static float startX = 20;

// Constructor
public Card(int cardID, String cardName, Subject cardSubject, Rarity cardRarity, String cardDescription, String bitmapName, GameScreen gameScreen) {
    super(startX, startY, cardWidth, cardHeight, gameScreen.getGame().getAssetManager().getBitmap(bitmapName), gameScreen);
    this.gameScreen = gameScreen;
    this.cardID = cardID;
    this.cardName = cardName;
    this.cardSubject = cardSubject;
    this.cardRarity = cardRarity;
    this.cardDescription = cardDescription;
    this.bitmapName = bitmapName;
    this.startX = startX;
    this.startY = startY;
    screenCentre.x = mGameScreen.getGame().getScreenWidth() / 2;
    screenCentre.y = mGameScreen.getGame().getScreenHeight() / 2;
}

```

Strength Card Class

The screenshot shows the Android Studio interface with the file `StrengthCard.java` open in the editor. The code implements a `StrengthCard` class that extends `Card`. It includes an enum `Row` with values `Front`, `Middle`, `Back`, and `All`. The class has private instance variables `cardRow` and `cardStrength`, and constructors for creating new cards and copying existing ones. It also provides getters and setters for these variables.

```

11 /**
12 * Created by Edward Muldrew on 27/11/2017. ~ User Story 12: Develop & Implement Inheritance Hierarchy
13 * The StrengthCard extends the card class and has the cardRow and cardStrength instance variable with getters and setters as well as the updateScore method
14 */
15
16 public class StrengthCard extends Card {
17     //Enums
18     public enum Row {
19         Front, Middle, Back, All
20     }
21
22     // Instance Variables
23     private Row cardRow;
24     private int cardStrength;
25
26     // Constructor
27     public StrengthCard(int cardID, String cardName, Subject cardSubject, Rarity cardRarity, String cardDescription,
28                         String bitmapName, GameScreen gameScreen, Row cardRow, int cardStrength) {
29         super(cardID, cardName, cardSubject, cardRarity, cardDescription, bitmapName, gameScreen);
30         this.cardRow = cardRow;
31         this.cardStrength = cardStrength;
32     }
33     // Copy Constructor
34     public StrengthCard(StrengthCard copy, String bitmapName, GameScreen gameScreen) {
35         super(copy, bitmapName, gameScreen);
36         this.cardRow = copy.cardRow;
37         this.cardStrength = copy.cardStrength;
38     }
39
40     // Getters
41     public Row getCardRow() { return cardRow; }
42
43     public int getCardStrength() { return cardStrength; }
44
45     // Setters
46     public void setCardStrength(int cardStrength) { this.cardStrength = cardStrength; }
47
48     public void setCardRow(Row cardRow) { this.cardRow = cardRow; }

```

The code editor shows syntax highlighting and several code completion suggestions (light blue boxes) for methods like `super`, `this`, and `copy`. The bottom status bar indicates a successful Gradle build and the current time as 33:24.

Minion Card Class

```

1 package uk.ac.qub.eeecs.game.cardDemo.Cards;
2
3 import ...
4
5 /**
6  * Created by Edward Muldrew on 27/11/2017 ~ User Story 12: Develop & Implement Inheritance Hierarchy
7  * This card inherits all methods from strength card
8 */
9
10 public class MinionCard extends StrengthCard {
11     // Constructor
12
13     public MinionCard(int cardID, String cardName, Subject cardSubject, Rarity cardRarity, String cardDescription, String bitmapName, GameScreen gameScreen, Row cardRow, int cardStrength) {
14         super(cardID, cardName, cardSubject, cardRarity, cardDescription, bitmapName, gameScreen, cardRow, cardStrength);
15     }
16
17     public MinionCard(StrengthCard copy, String bitmapName, GameScreen gameScreen) {
18         super(copy, bitmapName, gameScreen);
19     }
20 }
21
22

```

Hero Card Class

```

1 package uk.ac.qub.eeecs.game.cardDemo.Cards;
2
3 import ...
4
5 /**
6  * Created by Edward Muldrew on 27/11/2017 ~ User Story 12: Develop & Implement Inheritance Hierarchy
7  * This card inherits all methods from strength card. The only difference being the hero card can be placed on any row on the board.
8 */
9
10 public class HeroCard extends StrengthCard {
11     // Constructor
12     public HeroCard(int cardID, String cardName, Subject cardSubject, Rarity cardRarity, String cardDescription, String bitmapName, GameScreen gameScreen, Row cardRow, int cardStrength) {
13         super(cardID, cardName, cardSubject, cardRarity, cardDescription, bitmapName, gameScreen, cardRow, cardStrength);
14     }
15 }
16
17

```

Spell Card Class

```

14 public class SpellCard extends Card {
15     // Enums
16     public enum Ability {
17         DAMAGE, DOUBLE_POINTS, INCREASE
18     }
19
20     // Instance Variables
21     private Ability cardAbility;
22
23     // Constructor
24     public SpellCard(int cardID, String cardName, Subject cardSubject, Rarity cardRarity, String cardDescription, String bitmapName, GameScreen gameScreen, Ability cardAbility) {
25         super(cardID, cardName, cardSubject, cardRarity, cardDescription, bitmapName, gameScreen);
26         this.cardAbility = cardAbility;
27     }
28
29     public SpellCard(SpellCard copy, String bitmapName, GameScreen gameScreen) {
30         super(copy.bitmapName, gameScreen);
31         this.cardAbility = copy.cardAbility;
32     }
33
34
35     public void undoIncreaseSpell() {
36     }
37
38     // Methods
39     public void userSpellCard(ElapsedTime elapsedTime, Round round, ArrayList<Card> activeCards) {
40         int total = 0;
41         switch (this.cardAbility) {
42             case INCREASE:
43                 for(Card selectedCard: activeCards) {
44                     if(selectedCard.getClass() == MinionCard.class || selectedCard.getClass() == HeroCard.class) {
45                         ((StrengthCard)selectedCard).setCardStrength(((StrengthCard) selectedCard).getCardStrength() * 2);
46                         total += ((StrengthCard)selectedCard).getCardStrength();
47                         round.getUserScore().setScore(total);
48                     }
49                 }
50             break;
51
52             case DAMAGE:
53         }
54     }
55 }

```

The screenshot shows the SpellCard.java file in the Android Studio code editor. The code implements a SpellCard class with methods for increasing spell strength and dealing damage. The editor interface includes toolbars, a navigation bar, and various panels for project management and build logs.

Game Screen

Here is evidence of the class relationship model which represents the various screens for our project which all inherit from the abstract class GameScreen

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** csc-lvl2-1718-57 - [C:\Users\user\Documents\Queens University\Software Engineering\CSC2044-Software Development- Processes and Practice\3. Code\csc-lvl2-1718-57] - [app] - ...\\app\\src\\main\\java\\uk\\ac\\qub\\eeecs\\gage\\world\\GameScreen.java - Android...
- Toolbar:** File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
- Side Panels:** Project (GameScreen selected), Captures, Build Variants, Favorites, and Device File Explorer.
- Code Editor:** The main window displays the `GameScreen.java` file content. The code defines an abstract class `GameScreen` with methods for getting the name and game, and a constructor for creating a new game screen.

```
12  /*
13  * public abstract class GameScreen {
14  *
15  *     // Properties
16  *     // Constructors
17  *
18  *     /**
19  *      * Name that is given to this game screen
20  *      */
21  *     protected final String mName;
22  *
23  *     /**
24  *      * Return the name of this game screen
25  *      *
26  *      * @return Name of this game screen
27  *     */
28  *     public String getName() { return mName; }
29  *
30  *     /**
31  *      * Game to which game screen belongs
32  *      */
33  *     protected final Game mGame;
34  *
35  *     /**
36  *      * Return the game to which this game screen is attached
37  *      *
38  *      * @return Game to which screen is attached
39  *     */
40  *     public Game getGame() { return mGame; }
41  *
42  *     // Constructors
43  *     // Constructors
44  *
45  *     /**
46  *      * Create a new game screen associated with the specified game instance
47  *      */
48  * }
```

- Bottom Navigation:** Find, Run, TODO, Android Profiler, Logcat, Version Control, Terminal, Messages, Debug, Event Log.

Menu Screen

Options Screen

The screenshot shows the Android Studio interface with the code editor open to the `OptionsScreen.java` file. The code implements a game screen with various member variables and methods. The code is annotated with comments and TODOs.

```
public class OptionsScreen extends GameScreen {
    private final float LEVEL_WIDTH = 1000.0f;
    private final float LEVEL_HEIGHT = 1000.0f;
    private final int MIN_BRIGHTNESS = 0;
    private final int MAX_BRIGHTNESS = 100;
    private ScreenViewport mScreenViewport;
    private LayerViewport mLAYERViewport;

    //NEW UNCOMMITTED CODE
    public static boolean soundEnabled = true;
    private boolean continuePlaying;
    private MenuScreen backingMusic;
    private Paint opacity, opacity2;

    //?
    private Rect backgroundRect;
    private Rect musicOffRect;
    private Rect musicOnRect;
    private Rect volumeUpRect;
    private Rect volumeDownRect;
    private Rect menuBackButtonRect;
    private Rect creditsRect;
    private int musicVolume;
    private Paint paint;
    private Music music;
    //private boolean playSong;
    //NEW UNCOMMITTED CODE

    Context context;

    /*Background image for this screen.
    *By Grace 40172213, Sprint 4
    */
    private GameObject mOptionsBG;

    /*Text saying "Options" is saved as a button to be displayed as a title for this screen.
    */
}
```

The code editor includes standard Java syntax highlighting and annotations. The right side of the interface features toolbars for Coverage, Gradle Console, Device File Explorer, and other development tools. The bottom status bar displays build information and the current time.

Card Demo Screen

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** csc-lvl2-1718-57 - [C:\Users\user\Documents\Queens University\Software Engineering\CSC2044-Software Development- Processes and Practice\3. Code\csc-lvl2-1718-57] - [app] - ...app\src\main\java\uk\ac\qub\eeecs\game\cardDemo\CardDemoScreen.java
- Toolbar:** File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
- Project Structure:** Shows the project tree: csc-lvl2-1718-57 > app > src > main > java > uk > ac > qub > eeecs > game > cardDemo > CardDemoScreen
- Code Editor:** The file CardDemoScreen.java is open. The code implements a GameScreen for a card game. It includes imports for GameScreen, GameObject, PushButton, and various utility classes. The code handles card logic, turn records, player scores, and UI elements like buttons and prompts.
- Side Panels:** The left sidebar shows navigation links (1 Project, 2 Structure, Captures, Build variants, Build ping), a Favorites section (2 Favorites), and a Gradle panel. The right sidebar shows Coverage, Gradle Console, and Device File Explorer.
- Bottom Navigation:** Includes tabs for Find, Run, TODO, Android Profiler, Logcat, Version Control, Terminal, and Messages. A status bar at the bottom indicates a successful push to origin.

```
49
50     public class CardDemoScreen extends GameScreen {
51         // Instance Variables
52         private ScreenViewport mScreenViewport;
53         private LayerViewport mLAYERViewport;
54         Utilities helperUtilities = new Utilities(gameScreen: this, mGame);
55         private ArrayList<Card> activeCards = new ArrayList<>();
56         private ArrayList<Card> compSciCards = new ArrayList<>(); // Create Computer Science Array List of Cards
57         private Deck userDeck = new Deck(deckID: 1, deckName: "User Deck", compSciCards, gameScreen: this); // Create User Deck
58         private Hand userHand;
59         HashMap<Integer, String> userTurnRecord = new HashMap<>();
60         HashMap<Integer, String> aiTurnRecord = new HashMap<>(); // default turn record
61         public Turn turnRecord = new Turn(userTurnRecord, aiTurnRecord);
62         private PlayerScore userScore = new PlayerScore(score: 0, Hand.UserType.USER);
63         private PlayerScore defaultScore = new PlayerScore(score: 0, Hand.UserType.AI); // default player score
64         private Round round = new Round(name: "Round1", turnRecord, userScore, defaultScore);
65         private GameScreen aiDemoScreen = new AiDemoScreen(mGame, name: "aiDemoScreen", round);
66         private GameScreen roundEndScreen = new RoundEndScreen(mGame, name: "roundEndScreen", round);
67
68         private String player1DeckSize;
69         private String player1Score;
70         private int graveyardNumber = 0;
71         int turnRecordPosition = 0;
72
73         private GameObject mBoardBG;
74         private GameObject passCheckPrompt;
75         private GameObject userTurnPrompt;
76         private GameObject exitPrompt;
77         private GameObject cardPlacePrompt;
78
79         private PushButton passButton;
80         private PushButton yesButton;
81         private PushButton noButton;
82         private PushButton exitButton;
83
84         boolean passPrompt = false;
85         boolean aiTurnPrompt = false;
86         boolean exitToMainPrompt = false;
87         boolean cardPlace = false;
```

Overworld Screen

The screenshot shows the Android Studio interface with the project 'csc-lvl2-1718-57' open. The code editor displays the 'OverworldScreen.java' file under the 'src/main/java/uk/ac/qub/eeecs/game' package. The code implements a GameScreen with properties for level width and height, and methods for creating a simple menu screen with three location buttons. The code uses Java 8-style syntax like `private final float` and `else if`.

```
public class OverworldScreen extends GameScreen {

    // Properties
    private final float LEVEL_WIDTH = 480.0f;
    private final float LEVEL_HEIGHT = 270.0f;

    /**
     * Define the buttons for playing the 'games'
     */

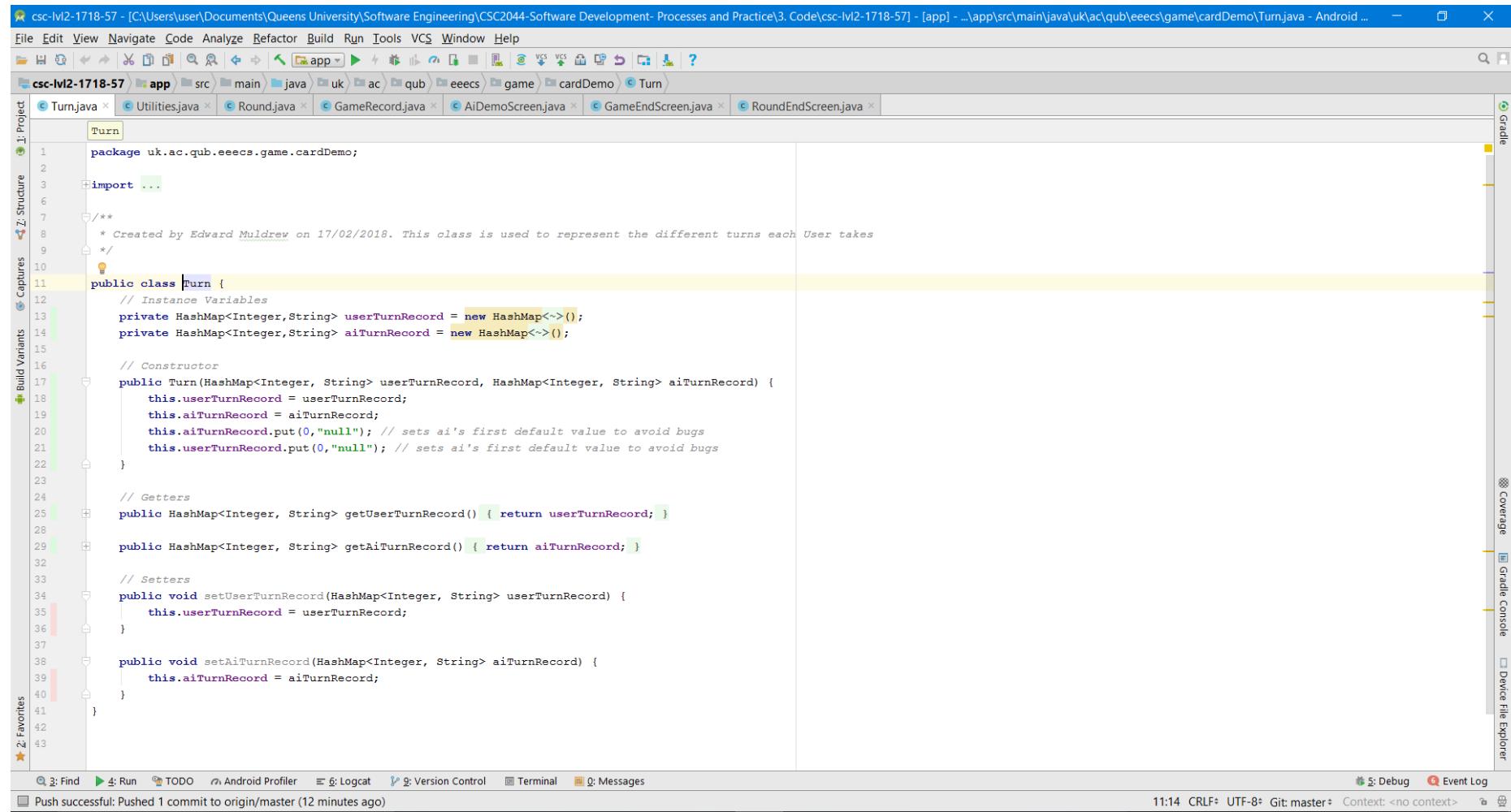
    private LayerViewport mLayerViewport;
    private ScreenViewport mScreenViewport;
    private PushButton mLocationButton;
    private PushButton mLocationButton1;
    private PushButton mLocationButton2;
    private PushButton mLocationButton3;
    private GameObject mOverworldMap;

    /**
     * Create a simple menu screen
     *
     * @param game Game to which this screen belongs
     */
    public OverworldScreen(Game game) {
        super(name: "OverworldScreen", game);

        mScreenViewport = new ScreenViewport(left: 0, top: 0, mGame.getScreenWidth(),
                                             mGame.getScreenHeight());
        if (mScreenViewport.width > mScreenViewport.height)
            mLayerViewport = new LayerViewport(x: 240.0f, y: 240.0f
                                               * mScreenViewport.height / mScreenViewport.width, halfWidth: 240,
                                               halfHeight: 240.0f * mScreenViewport.height / mScreenViewport.width);
        else
```

Here is evidence of the next class relationship model which represents the various screens for our actual game. This is a complex model showing the interactions between the game classes.

Turn class



The screenshot shows the Android Studio interface with the project navigation bar at the top. The current file is Turn.java, located in the app/src/main/java/uk/ac/qub/eeecs/game/cardDemo directory. The code defines a class Turn with two private HashMap fields: userTurnRecord and aiTurnRecord. It includes a constructor, two getters, and two setters for these fields. A Javadoc comment at the top describes the class as representing different turns each User takes.

```

1 package uk.ac.qub.eeecs.game.cardDemo;
2
3 import ...
4
5 /**
6  * Created by Edward Muldrew on 17/02/2018. This class is used to represent the different turns each User takes
7 */
8
9
10 public class Turn {
11     // Instance Variables
12     private HashMap<Integer, String> userTurnRecord = new HashMap<~>();
13     private HashMap<Integer, String> aiTurnRecord = new HashMap<~>();
14
15     // Constructor
16     public Turn(HashMap<Integer, String> userTurnRecord, HashMap<Integer, String> aiTurnRecord) {
17         this.userTurnRecord = userTurnRecord;
18         this.aiTurnRecord = aiTurnRecord;
19         this.aiTurnRecord.put(0,"null"); // sets ai's first default value to avoid bugs
20         this.userTurnRecord.put(0,"null"); // sets ai's first default value to avoid bugs
21     }
22
23     // Getters
24     public HashMap<Integer, String> getUserTurnRecord() { return userTurnRecord; }
25
26     public HashMap<Integer, String> getAiTurnRecord() { return aiTurnRecord; }
27
28     // Setters
29     public void setUserTurnRecord(HashMap<Integer, String> userTurnRecord) {
30         this.userTurnRecord = userTurnRecord;
31     }
32
33     public void setAiTurnRecord(HashMap<Integer, String> aiTurnRecord) {
34         this.aiTurnRecord = aiTurnRecord;
35     }
36
37 }
38
39
40 }
41
42
43

```

The bottom status bar shows "Push successful: Pushed 1 commit to origin/master (12 minutes ago)" and the time "11:14".

Utilities Class

csc-lvl2-1718-57 - [C:\Users\user\Documents\Queens University\Software Engineering\CSC2044-Software Development- Processes and Practice\3. Code\csc-lvl2-1718-57] - [app] - ...app\src\main\java\uk\ac\qub\eeecs\game\cardDemo\Utilities.java - Andro...

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Project Structure Captures Build Variants Z-Structure 2 Favorites

Gradle Coverage Grade Console Device Explorer File Explorer

app src main java uk ac qub eeecs game cardDemo Utilities

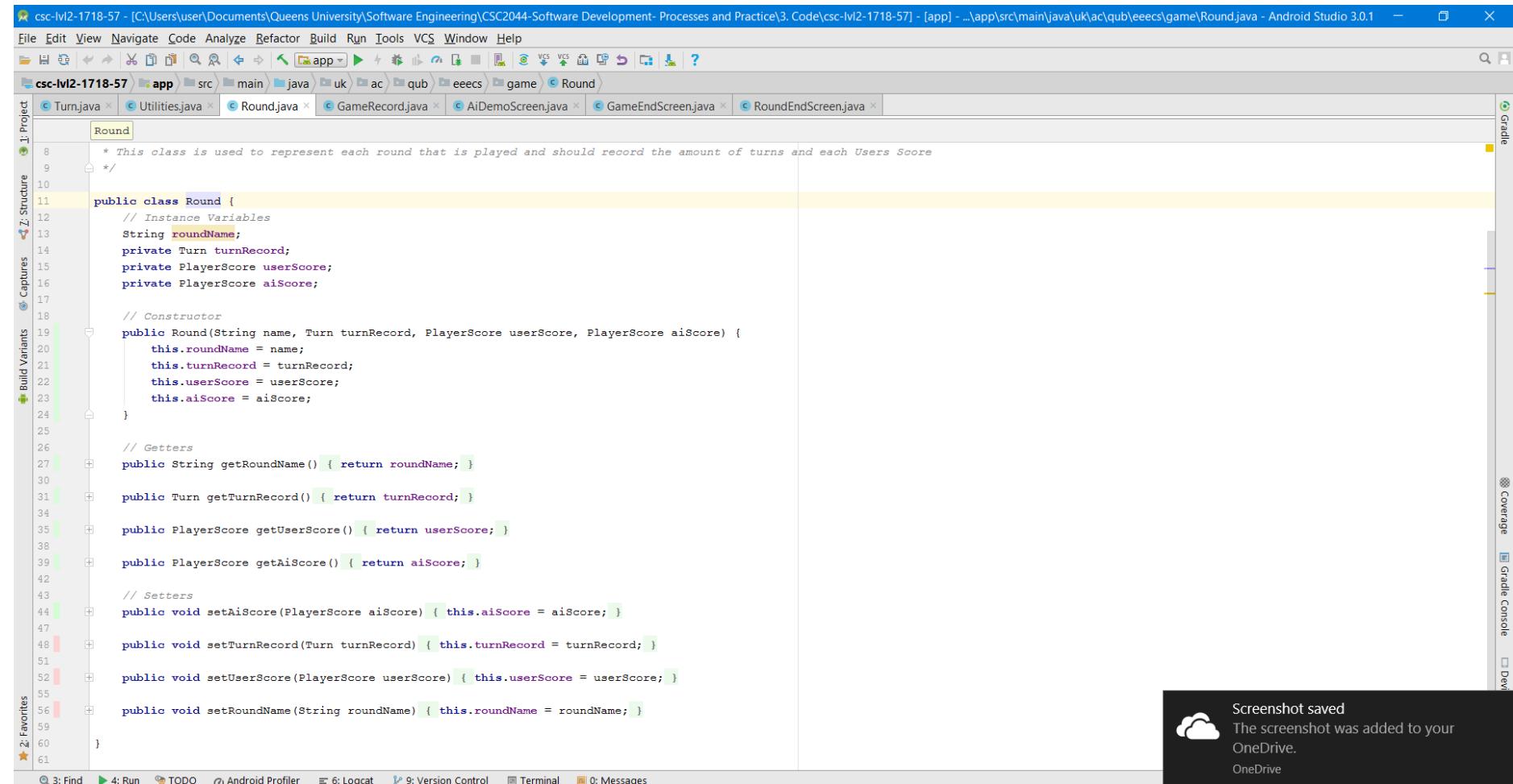
Turn.java Utilities.java Round.java GameRecord.java AI_demoScreen.java GameEndScreen.java RoundEndScreen.java

Utilities setCardXPosition()

```
1 * This class is used to help reduce code on the Card Demo Screen and make an easier place to instantiate and create variables
2 */
3
4 public class Utilities {
5     // DEFAULT X & Y VALUES
6
7     private ArrayList<Card> defaultArrayOfCards = new ArrayList<>(); // Create Computer Science Array List of Cards
8     private Card compSciMinionSpellCards[] = new Card[6];
9     private HeroCard compSciHero;
10    private HeroCard medicineHero;
11    private Card medicineMinionSpellCards[] = new Card[6];
12    private final int MAX_SPELL_CARDS = 3;
13    private int numberofInitialHandCards = 5;
14    private int numberofInitialDeckCards = 6;
15    int recordTurns = 0;
16
17    public Utilities(GameScreen gameScreen, Game mGame) {
18        this.loadAndAddAsests(mGame);
19        compSciHero = new HeroCard(cardID: 1, cardName: "Matthew Collins", Card.Subject.COMPUTER_SCIENCE, Card.Rarity.RARE, cardDescription: "I am your hero", bitmapName: "CompSciHero", gameScreen, StrengthCard.Row.All);
20        compSciMinionSpellCards[0] = new SpellCard(cardID: 2, cardName: "Virus", Card.Subject.COMPUTER_SCIENCE, Card.Rarity.COMMON, cardDescription: "Damage enemy", bitmapName: "VirusSpell", gameScreen, SpellCard.Ability.SKIP);
21        // compSciMinionSpellCards[1] = new SpellCard(3, "Bug", Card.Subject.COMPUTER_SCIENCE, Card.Rarity.COMMON, "Skip AI Turn", "BugSpell", gameScreen, SpellCard.Ability.SKIP);
22        compSciMinionSpellCards[1] = new SpellCard(cardID: 4, cardName: "Ram", Card.Subject.COMPUTER_SCIENCE, Card.Rarity.COMMON, cardDescription: "Boost strength", bitmapName: "RamSpell", gameScreen, SpellCard.Ability.SKIP);
23        compSciMinionSpellCards[2] = new SpellCard(cardID: 5, cardName: "Processor", Card.Subject.COMPUTER_SCIENCE, Card.Rarity.RARE, cardDescription: "Double points", bitmapName: "ProcessorSpell", gameScreen);
24        compSciMinionSpellCards[3] = new MinionCard(cardID: 6, cardName: "Keyboard", Card.Subject.COMPUTER_SCIENCE, Card.Rarity.COMMON, cardDescription: "Back Row", bitmapName: "KeyboardMinion", gameScreen);
25        compSciMinionSpellCards[4] = new MinionCard(cardID: 7, cardName: "Printer", Card.Subject.COMPUTER_SCIENCE, Card.Rarity.COMMON, cardDescription: "Middle Row", bitmapName: "PrinterMinion", gameScreen);
26        compSciMinionSpellCards[5] = new MinionCard(cardID: 8, cardName: "Mouse", Card.Subject.COMPUTER_SCIENCE, Card.Rarity.UNCOMMON, cardDescription: "Front Row", bitmapName: "MouseMinion", gameScreen, StrengthCard.Row.All);
27
28        medicineHero = new HeroCard(cardID: 12, cardName: "Dr Phil", Card.Subject.MEDICINE, Card.Rarity.RARE, cardDescription: "I am your hero", bitmapName: "MedicineHero", gameScreen, StrengthCard.Row.All);
29        medicineMinionSpellCards[0] = new SpellCard(cardID: 13, cardName: "Transplant", Card.Subject.MEDICINE, Card.Rarity.COMMON, cardDescription: "Double points", bitmapName: "TransplantSpell", gameScreen, SpellCard.Ability.SKIP);
30        medicineMinionSpellCards[1] = new SpellCard(cardID: 14, cardName: "Plaster", Card.Subject.MEDICINE, Card.Rarity.UNCOMMON, cardDescription: "Boost strength", bitmapName: "PlasterSpell", gameScreen, SpellCard.Ability.SKIP);
31        medicineMinionSpellCards[2] = new SpellCard(cardID: 15, cardName: "MedKit", Card.Subject.MEDICINE, Card.Rarity.COMMON, cardDescription: "Damage enemy", bitmapName: "MedKitSpell", gameScreen, SpellCard.Ability.SKIP);
32        medicineMinionSpellCards[3] = new MinionCard(cardID: 16, cardName: "Syringe", Card.Subject.MEDICINE, Card.Rarity.RARE, cardDescription: "Back Row", bitmapName: "SyringeMinion", gameScreen, StrengthCard.Row.All);
33        medicineMinionSpellCards[4] = new MinionCard(cardID: 17, cardName: "Stethoscope", Card.Subject.MEDICINE, Card.Rarity.COMMON, cardDescription: "Front Row", bitmapName: "StethoscopeMinion", gameScreen);
34        medicineMinionSpellCards[5] = new MinionCard(cardID: 18, cardName: "Antibiotic", Card.Subject.MEDICINE, Card.Rarity.UNCOMMON, cardDescription: "Middle Row", bitmapName: "AntibioticMinion", gameScreen, StrengthCard.Row.All);
35
36    }
37
38    public Hand generateCompSciHand(GameScreen gameScreen, Hand compSciHand) {
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58 }
```

3: Find 4: Run 5: Debug TODO 6: Android Profiler 7: Logcat 8: Version Control 9: Terminal 0: Messages

Round Class



The screenshot shows the Android Studio interface with the project 'csc-lvl2-1718-57' open. The 'Round.java' file is selected in the project tree. The code defines a class 'Round' with instance variables for round name, turn record, and two player scores. It includes a constructor, four getters, and four setters. A Javadoc comment at the top describes the class's purpose. A 'Screenshot saved' notification is visible in the bottom right corner.

```
* This class is used to represent each round that is played and should record the amount of turns and each Users Score
*/
public class Round {
    // Instance Variables
    String roundName;
    private Turn turnRecord;
    private PlayerScore userScore;
    private PlayerScore aiScore;

    // Constructor
    public Round(String name, Turn turnRecord, PlayerScore userScore, PlayerScore aiScore) {
        this.roundName = name;
        this.turnRecord = turnRecord;
        this.userScore = userScore;
        this.aiScore = aiScore;
    }

    // Getters
    public String getRoundName() { return roundName; }

    public Turn getTurnRecord() { return turnRecord; }

    public PlayerScore getUserScore() { return userScore; }

    public PlayerScore getAiScore() { return aiScore; }

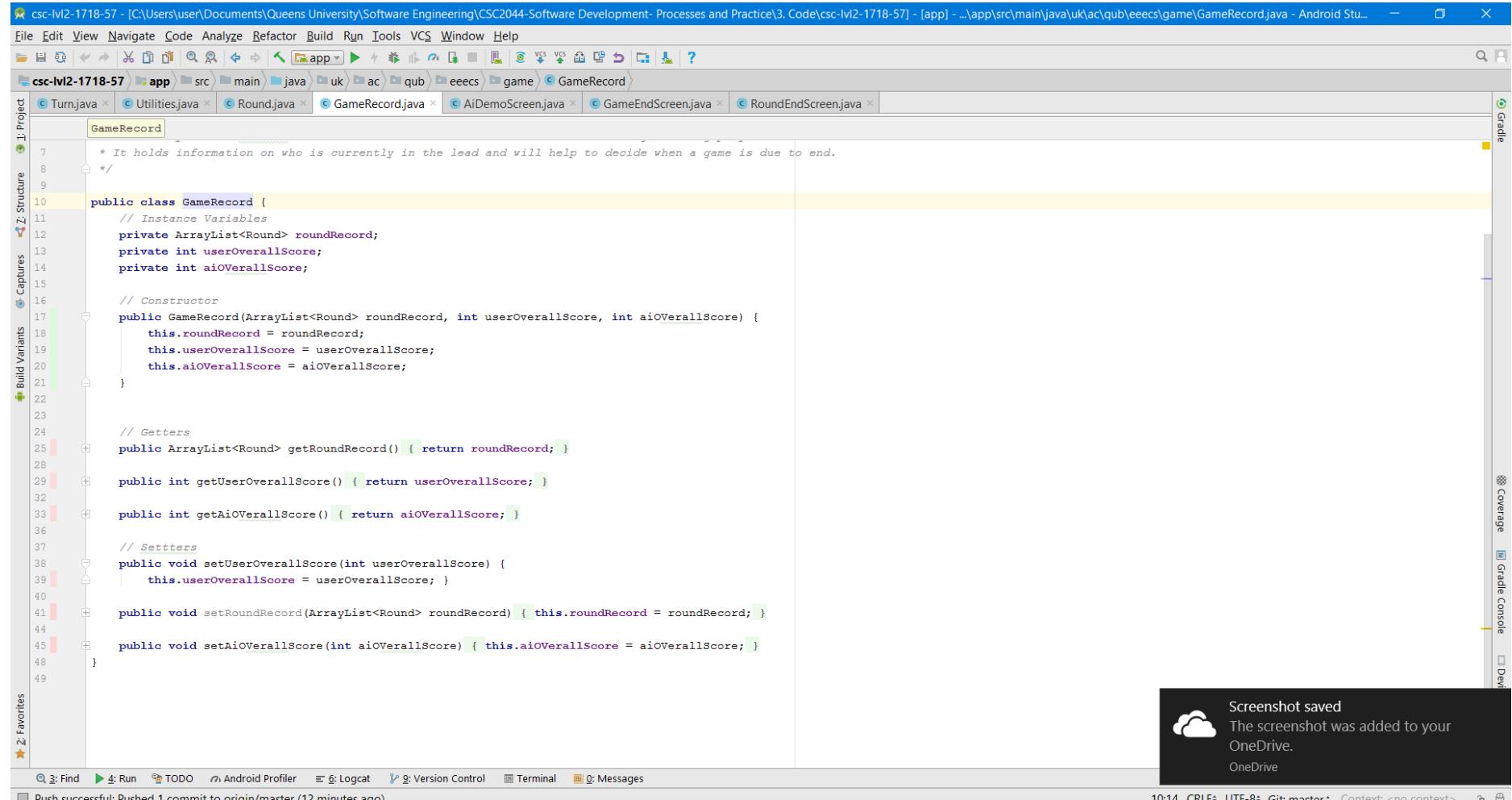
    // Setters
    public void setAiScore(PlayerScore aiScore) { this.aiScore = aiScore; }

    public void setTurnRecord(Turn turnRecord) { this.turnRecord = turnRecord; }

    public void setUserScore(PlayerScore userScore) { this.userScore = userScore; }

    public void setRoundName(String roundName) { this.roundName = roundName; }
}
```

Game Record Class



The screenshot shows the Android Studio interface with the project 'csc-lvl2-1718-57' open. The 'GameRecord.java' file is selected in the navigation bar. The code defines a class 'GameRecord' with fields for round records, user overall score, and AI overall score, along with constructor and getter/setter methods.

```
7 * It holds information on who is currently in the lead and will help to decide when a game is due to end.
8 */
9
10 public class GameRecord {
11     // Instance Variables
12     private ArrayList<Round> roundRecord;
13     private int userOverallScore;
14     private int aioOverallScore;
15
16     // Constructor
17     public GameRecord(ArrayList<Round> roundRecord, int userOverallScore, int aioOverallScore) {
18         this.roundRecord = roundRecord;
19         this.userOverallScore = userOverallScore;
20         this.aioOverallScore = aioOverallScore;
21     }
22
23
24     // Getters
25     public ArrayList<Round> getRoundRecord() { return roundRecord; }
26
27     public int getUserOverallScore() { return userOverallScore; }
28
29     public int getAioOverallScore() { return aioOverallScore; }
30
31     // Setters
32     public void setUserOverallScore(int userOverallScore) {
33         this.userOverallScore = userOverallScore;
34     }
35
36     public void setRoundRecord(ArrayList<Round> roundRecord) { this.roundRecord = roundRecord; }
37
38     public void setAioOverallScore(int aioOverallScore) { this.aioOverallScore = aioOverallScore; }
39
40
41
42
43
44
45
46
47
48
49 }
```

A notification at the bottom right indicates that the screenshot was saved to OneDrive.

AI Demo Screen Class

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** csc-lvl2-1718-57 - [C:\Users\user\Documents\Queens University\Software Engineering\CSC2044-Software Development- Processes and Practice\3. Code\csc-lvl2-1718-57] - [app] - ...\\app\\src\\main\\java\\uk\\ac\\qub\\eeecs\\game\\cardDemo\\AiDemoScreen.java
- Toolbar:** File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
- Side Panels:** Project (AiDemoScreen), Structure, Captures, Build Variants, Favorites.
- Code Editor:** The main editor window displays the Java code for `AiDemoScreen`. The code initializes various game components like `mScreenViewport`, `mLayerViewport`, and `aiDeck`, sets up player scores, and handles card selection logic. It also defines constants for screen dimensions and player rectangles.
- Bottom Navigation:** Find, Run, TODO, Android Profiler, Logcat, Version Control, Terminal, Messages.
- Bottom Status:** Push successful: Pushed 1 commit to origin/master (13 minutes ago)
- Right Sidebar:** Coverage, Gradle Console, Device File Explorer.

```
* Created by Edward Muldrew on the 12/01/2018.
* This class is used tp represent the AI's screen in which the AI will choose to place or pass their respective turn.
*
* @version 1.0
*/
public class AiDemoScreen extends GameScreen {
    // Instance Variables
    private ScreenViewport mScreenViewport;
    private LayerViewport mLayerViewport;
    Utilities helperUtilities = new Utilities( gameScreen: this, mGame);

    private GameObject mBoardBG;
    private ArrayList<Card> activeCards = new ArrayList<>(); // active ai cards placed on the screen
    private ArrayList<Card> medicineCards = new ArrayList<>(); // Create Medicine Array List of Cards
    private Deck aiDeck = new Deck( deckID: 2, deckName: "AI Deck", medicineCards, gameScreen: this); // Create AI Deck
    private Hand aiHand;
    private PlayerScore aiScore = new PlayerScore( score: 0, Hand.UserType.AI);
    HashMap<Integer, String> aiTurnRecord = new HashMap<>(); // keeps record the ai;s turns
    Round round;
    Turn turnRecord;
    int turnRecordPosition = 0;
    int graveyardNumber = 0;
    int positionInHand;

    boolean cardSelected = false; // Boolean value to change once a card has been selected.
    boolean updateThis = true;
    private PushButton passButton;

    private final float LEVEL_WIDTH = 480.0f;
    private final float LEVEL_HEIGHT = 270.0f;
    private final int CONST_Y = 20;
    private double playerStartTime;

    private HandContainer playerHand;
    private Rect playerRect;

    // Constructors
```

Game End Screen Class

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** csc-lvl2-1718-57 - [C:\Users\user\Documents\Queens University\Software Engineering\CSC2044-Software Development- Processes and Practice\3. Code\csc-lvl2-1718-57] - [app] - ...\\app\\src\\main\\java\\uk\\ac\\qub\\eeecs\\game\\cardDemo\\GameEndScreen.java..
- Toolbar:** File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help.
- Project Structure:** Shows the project tree: csc-lvl2-1718-57 > app > src > main > java > uk > ac > qub > eeecs > game > cardDemo > GameEndScreen.
- Code Editor:** The file GameEndScreen.java is open. The code implements a GameEndScreen class that extends GameScreen. It includes fields for ScreenViewports, LayerViewports, Utilities, GameRecord, and PushButtons. The constructor initializes these fields based on the game's screen width and height. The update() method is annotated with a Javadoc comment describing its purpose.
- Side Panels:** The left sidebar shows Project, Structure, Captures, Build Variants, Favorites, and Gradle. The right sidebar shows Coverage, Gradle Console, and Device File Explorer.
- Bottom Bar:** Includes icons for Find, Run, TODO, Android Profiler, Logcat, Version Control, Terminal, and Messages.

Round End Screen

The screenshot shows the Android Studio interface with the project navigation bar at the top. The main editor window displays the `RoundEndScreen.java` file under the package `csc-lvl2-1718-57/app/src/main/java/uk/ac/qub/eeecs/game/cardDemo`. The code implements a game screen for displaying round results and managing new rounds.

```
* Created by Edward Muldrew on the 19/02/2018. This class is used to show the end score of each round between the user and they AI.  
* It will also be used to make decisions on if a new round needs to be created or someone has won the game.  
*  
* @version 1.0  
*/  
  
public class RoundEndScreen extends GameScreen {  
    // Instance Variables  
    private ScreenViewport mScreenViewport;  
    private LayerViewport mLayerViewport;  
    Utilities helperUtilities = new Utilities( gameScreen: this, mGame );  
    private GameScreen gameEndScreen;  
    private GameObject mBoardBG;  
    Round round;  
    int userScore;  
    int aiScore;  
    int userOverallScore = 0;  
    int aiOverallScore = 0;  
    String result = "";  
    String roundName = "";  
    GameRecord gameRecord;  
    ArrayList<Round> roundRecord = new ArrayList<>();  
  
    private final float LEVEL_WIDTH = 480.0f;  
    private final float LEVEL_HEIGHT = 270.0f;  
    private final int CONST_Y = 20;  
    int spacingX;  
    int spacingY;  
    private PushButton newRoundButton;  
  
    // Constructors  
    /**  
     * Create the Round End screen  
     *  
     * @param game Game to which this screen belongs  
     */  
    public RoundEndScreen(Game game, String name, Round round) {
```

Team Minutes

The following team members were present:

- Edward Muldrew
 - Grace Turkington
 - Brandon Smylie
 - Conor Clarke
 - Jack McNaughton

Name: Edward Muldrew

Signature:

Name: Grace Turkington

Signature:

Name: Brandon Smylie

Signature:

Name: Conor Clarke

Signature:

Name: Jack McNaughton

Signature:

Meeting 1: Sprint 4, Week 1

Week commencing: 08/01/2018

Date of this minute: 08/01/2018

Task Reporting

First week of term- did not make progress in the last week

Actions Planned

Edward Muldrew & Role: Programmer

- Complete User Stories for Card Class
- Submit User Stories document
- Continue to do some tests for card class
- Start on User Stories
- Meet up with group on Wednesday to discuss User Stories

Grace Turkington & Role (2):

- Meet with team on Wednesday to discuss feedback from sprint 3 and assign user stories for sprint 4
- Add a to-do list to the project page of my plan for an assigned user story
- Begin working on my assigned user stories
- Commit my finished changes to Git
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Meet with team to discuss feedback and Sprint 4.
- Add my user stories to the document.
- Begin working on those user stories.
- Submit user story document to Phillip.

Conor Clarke & Role (4):

- Attend group meeting to discuss Sprint 3 feedback and Sprint 4 plans
- Add user stories to Sprint 4 document
- Work on user stories from older sprints

Jack McNaughton & Role (5):

- Work on any remaining user stories from previous sprint.
- Meet with team to discuss user stories for next sprint.
- Add new user stories to sprint document
- Start on new user stories
- Watch this YouTube videos

Meeting 2: Sprint 4, Week 2

Week commencing: 15/01/2018

Date of this minute: 15/01/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Met with team on Wednesday to discuss the User Story document
- Started work on User story assigning random cards to each deck
- Initialized and declared medicine deck
- Removed old code from project file
- Added unit tests to test Card Class code

Grace Turkington & Role (2):

- Met with team to assign sprint 4 user stories
- Changed appearance of main menu screen, removing unused code and buttons
- Began user story 9
- Watched this weeks youtube videos

Brandon Smylie & Role (3):

- Met with team to discuss user stories.
- Re-evaluated user story 5 into a grand user story.
- Implemented a container superclass.
- Tested some superclass methods.

Conor Clarke & Role (4):

- Met with team to discuss user stories
- Work on Overworld user stories
- Edit sprite sheets

Jack McNaughton & Role (5):

- Met with team to discuss user stories.
- Wrote user stories to document.
- Attempted user story 17.
- Watched the weeks YouTube videos.

Actions Planned

Edward Muldrew & Role: Programmer

- Continue work on my User Stories
- Add unit tests for the new User Stories created
- Meet up with the team on Tuesday to discuss user story progress and any problems they have thus far
- Commit all finished changes to Git
- Watch supplied YouTube videos from Philip

Grace Turkington & Role (2):

- Add a to-do list to the project page of my plan for an assigned user story
- Keep working on sprint 4 user stories and write some tests for these
- Commit my finished changes to Git
- Team meeting on Tuesday to discuss sprint 4 progress so far
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Re-write my grand user story into 4 smaller, more manageable user stories.
- Commit my finished changes to Git.
- Meet with team on Tuesday to discuss progress.

Conor Clarke & Role (4):

- Meet with team to discuss progress and problems
- Watch helpful youtube videos

Jack McNaughton & Role (5):

- Change user story 17 so that the goal is to draw the text using draw methods instead of reading from a text document
- Start work on user story 18.
- Meet with team to discuss user story progression.
- Watch YouTube videos for the week.

Meeting 3: Sprint 4, Week 3

Week commencing: 22/01/2018

Date of this minute: 22/01/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Team meeting to assess User Story progress with group. Discusses how our game is going to work and play out
- Added random deck generation for both decks
- Started work on Deck and Hand class
- Added copy constructors so duplicate objects of the same type can be created

Grace Turkington & Role (2):

- Met with team to discuss progress with sprint 4 user stories
- Added menu screen tests
- Created options screen (user story 10)
- Added options screen tests
- Began user story 11
- Watched this weeks youtube videos

Brandon Smylie & Role (3):

- Team meeting to assess user story progress.
- Reevaluated my user stories.
- Continued work on user story 5.

Conor Clarke & Role (4):

- Met with team to discuss progress and problems
- Continued to work on overworld, tried to string together a map from a tilesheet
- Listened to the youtube videos

Jack McNaughton & Role (5):

- Met with team to discuss user story progression.
- Watched YouTube videos
- Converted text document into draw text methods.
- Adjusted placement and sizes of text and images used so that they don't block each other out.

Actions Planned

Edward Muldrew & Role: Programmer

- Meet with team to discuss User Story progress
- Add Unit tests to test new code that has been added
- Continue work on Hand and Deck class
- Aim to try get a selected mechanism when User Clicks on card
- Add Amount of cards in deck class on Card Demo Screen

Grace Turkington & Role (2):

- Commit tests I've done to Git
- Keep working on user story 11
- Team meeting to complete sprint 4 submission document on Friday
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Finish grand user story 5.
- Meet with group to submit the Sprint 4 report.
- Submit Sprint 4 report.

Conor Clarke & Role (4):

- Meet with group to add my contribution to sprint 4 report
- Continue to work on what I couldn't finish in the sprint

Jack McNaughton & Role (5):

- Meet with team for user story progression and end of sprint document
- Research how to make an android screen scroll in android studio for user story 19.
- Upload completed code to git.
- Watch weeks YouTube video.

Meeting 4: Sprint 5, Week 1

Week commencing: 29/01/2018

Date of this minute: 29/01/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Met with team to create Sprint 4 document and discussed progress thus far
- Add Unit tests to test recently added code
- Add Hand & Deck class
- Added AI cards set to be blank

Grace Turkington & Role (2):

- Met with team to discuss finalise sprint 4 document
- Completed and committed 3 user stories
- Watched this weeks youtube videos

Brandon Smylie & Role (3):

- Team meeting to discuss, create and finalise sprint 4 document
- Completed and committed 1 grand user story
- Tested basic functionality of container superclass

Conor Clarke & Role (4):

- Met with team to discuss finalise sprint 4 document
- Did sprites and some of the map, no commit

Jack McNaughton & Role (5):

- Met with team to discuss Sprint 4 document and sprint progression.
- Uploaded stories 17 and 18 to git.
- Watched this week's YouTube videos.
- Checked for videos on how to scroll screen using touch.

Actions Planned

Edward Muldrew & Role: Programmer

- Meet up with the team this week to discuss Sprint 4 progress & assign new user stories
- Start work on User Stories
- Commit changes to Git

Grace Turkington & Role (2):

- Meet with team to assign sprint 5 user stories
- Add a to-do list to the project page of my plan for an assigned user story
- Begin working on my assigned user stories
- Commit my finished changes to Git
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Have team meeting to assign sprint 5 user stories
- Begin working on the user stories assigned to myself
- Commit any finished changes to Git
- Email user story document to Phillip

Conor Clarke & Role (4):

- Have team meeting to assign sprint 5 user stories
- Begin working on the user stories assigned to myself
- Put my user stories in the sprint 5 document

Jack McNaughton & Role (5):

- Meet with team to discuss Sprint 5 user stories
- Watch this week's YouTube videos
- Start on user stories.
- Input user stories into user stories document

Meeting 5: Sprint 5, Week 2

Week commencing: 05/02/2018

Date of this minute: 05/02/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Met with team to discuss User Stories
- Started work on Score class
- Completed User Story document

Grace Turkington & Role (2):

- Met with team to discuss create and assign sprint 5 user stories
- Added splash, menu and options screen tests
- Created to do lists for all my user stories (11-13)
- Completed and committed user story 12
- Watched this weeks youtube videos

Brandon Smylie & Role (3):

- Met with team to discuss, create and assign sprint 5 user stories
- Refactored some container code
- Added some more tests

Conor Clarke & Role (4):

- Met with team to discuss, create and assign sprint 5 user stories
- Made a start on the walk method for the player sprite

Jack McNaughton& Role (5):

- Watched this week's YouTube videos
- Met with team to discuss user stories.
- Determined that making the screen scroll in response to touch will take too long
- Broke large segment of text segments into smaller pages.

Actions Planned

Edward Muldrew & Role: Programmer

- Meet with team to discuss User Story progress
- Finish work on Score class
- Add working show size of deck on screen
- Add User Stories to Test Score class

Grace Turkington & Role (2):

- Meet with team to discuss progress on sprint 5 user stories
- Continue working on user stories
- Commit my finished changes to Git
- Continue writing unit tests
- Watch this weeks youtube video

Brandon Smylie & Role (3):

- Meet with team to discuss progress on sprint 5 user stories
- Continue working on refactoring the Container hierarchy
- Commit my finished changes to Git
- Test any additional container methods

Conor Clarke & Role (4):

- Meet with team to discuss progress and problems
- Watch the youtube videos
- Continue work on overworld, try to finish walk method

Jack McNaughton & Role (5):

- Watch this weeks YouTube videos
- Meet with team to discuss user story progress
- Make the screen transition to a different draw method via a button click
- Upload code to git.

Meeting 6: Sprint 5, Week 3

Week commencing: 12/02/2018

Date of this minute: 12/02/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Met with team on Friday to discuss sprint progress. Also discussed future of project and what needs completed in the upcoming sprints
- Added working select and place card
- Added working update score
- Added working show number of cards in deck on screen
- Added working switch between AI and User screens
- Added Utilities class so code can be maintained more easily and code is more readable

Grace Turkington & Role (2):

- Met with team to discuss sprint 5 progress
- Worked on user stories 11 & 13
- Watched this weeks youtube videos

Brandon Smylie & Role (3):

- Met with team on Friday to discuss sprint 5
- Finished refactoring container hierarchy

Conor Clarke & Role (4):

- Did not finish walk method
- Met with team on Friday to discuss sprint 5
- Listened to youtube videos

Jack McNaughton& Role (5):

- Watched this weeks YouTube videos
- Adjusted positioning of text and pictures for draw methods.
- Uploaded code to git
- Met with team to discuss sprint 5

Actions Planned

Edward Muldrew & Role: Programmer

- Meet with team to complete User Story document
- Tidy up code which has been added
- Ensure code works the way it should and removing any bugs
- Add Unit tests to test the newly added classes and code
- Met with team to complete sprint 5 document

Grace Turkington & Role (2):

- Continue working on user stories
- Commit my finished changes to Git
- Continue writing unit tests
- Watch this weeks youtube videos
- Met with team to complete sprint 5 document

Brandon Smylie & Role (3):

- Finish off containers for good
- Commit any finished changes to Git
- Continue writing unit tests
- Meet with team to complete sprint 5 document
- Submit said group document to Phil

Conor Clarke & Role (4):

- Meet with team to complete sprint 5 document
- Continue work on overworld

Jack McNaughton & Role (5):

- Watch this week's YouTube videos
- Finish tutorial.
- Meet team for sprint 5 document
- Add new user stories

Meeting 7: Sprint 6, Week 1

Week commencing: 19/02/2018

Date of this minute: 19/02/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Met up with team on Friday to finalize sprint 5 report and discuss User Story progress
- Add Unit tests to test the various new code that has been implemented
- Added in working pop up, pop down and place cards method
- Added User Prompt to check User is sure they wish to pass their go
- Added text to allow the User to understand the difference between the User and the AI screen

Grace Turkington & Role (2):

- Met up with team on Friday to finalize sprint 5 report and discuss User Story progress
- Worked on user stories 11 & 13 (Did not get these finished)
- Watched this weeks youtube videos
- Added & committed unit tests to the various new code that has been implemented

Brandon Smylie & Role (3):

- Met with team to create Sprint 5 report
- Containers were finished
- Report was emailed to Phil

Conor Clarke & Role (4):

- Met with team to create Sprint 5 report
- Try to implement an interaction method between player and AI sprites

Jack McNaughton & Role (5):

- Met with team for sprint 5 document.
- Watched Youtube videos for the week
- Made the tutorial transition between draw methods using a button.

Actions Planned

Edward Muldrew & Role: Programmer

- Meet with team to discuss User Stories and create User Story document
- Start work on User Stories
- Add turn based mechanism & Round ending screen

Grace Turkington & Role (2):

- Meet with team to assign sprint 6 user stories
- Add a to-do list to the project page of my plan for an assigned user story
- Begin working on my assigned user stories
- Commit my finished changes to Git
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Meet with team to create and assign sprint 6 user stories
- Begin working on my assigned user stories
- Commit my finished changes to Git
- Submit user stories to Phil

Conor Clarke & Role (4):

- Meet with team to create and assign sprint 6 user stories
- Begin working on my assigned user stories

Jack McNaughton & Role (5):

- Meet with team to create user stories for sprint 6.
- Begin work on user stories.
- Watch YouTube videos of the week.
- Commit any significant changes to git.

Meeting 8: Sprint 6, Week 2

Week commencing: 26/02/2018

Date of this minute: 26/02/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Met up with team on Friday to discuss User Story progress and completed User Story document
- Completed Turn Based mechanism
- Completed creating a round ending screen
- Started work on round based mechanism
- Cleaned up code
- Committed code to git

Grace Turkington & Role (2):

- Meet with team to assign sprint 6 user stories
- Add a to-do list to the project page of my plan for an assigned user story
- Begin working on my assigned user stories
- Commit my finished changes to Git
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Met with team to assign sprint 6 user stories
- Began working on my assigned user stories
- Commit any finished changes to Git
- Submitted document to Phil

Conor Clarke & Role (4):

- Met with team to assign sprint 6 user stories
- Began working on my assigned user stories

Jack McNaughton & Role (5):

- Watched week's YouTube videos
- Met with team to discuss User stories
- Began working on assigned user stories.

Actions Planned

Edward Muldrew & Role: Programmer

- Meet up with team during the week to discuss User Story progress
- Start working on some bugs in the code (i.e when user plays all their cards, only allow the ai to play)
- Start work on Spell cards
- Finish round-based mechanism and final screen to show who won the game.
- Commit code to GIT

Grace Turkington & Role (2):

- Meet with team to discuss progress on sprint 6 user stories
- Continue working on user stories
- Commit some changes to Git
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Meet with team to discuss progress
- Continue working my assigned user stories
- Commit some changes to Git
- Start researching AI.

Conor Clarke & Role (4):

- Meet with team to discuss progress
- Continue working my assigned user stories
- Look into AI development

Jack McNaughton & Role (5):

- Meet with team to discuss progress on sprint 6 user stories
- Continue working on user stories
- Watch this week's YouTube videos
- Attempt to get screenshots of actual game to improve tutorial

Meeting 9: Sprint 6, Week 3

Week commencing: 05/03/2018

Date of this minute: 05/03/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Completed round based and game based mechanism
- Game ending screen will appear once a winner has been achieved
- Committed code to git

Grace Turkington & Role (2):

- Met with team to discuss progress on sprint 6 user stories
- Continued working on user stories
- Committed some changes to Git
- Watched this weeks youtube videos

Brandon Smylie & Role (3):

- Met with team to discuss progress on sprint 6 user stories
- Watched multiple videos on AI development

Conor Clarke & Role (4):

- Met with team to discuss progress on sprint 6 user stories
- Went over lecture slides to get an idea of the right way to make an AI

Jack McNaughton & Role (5):

- Met with team to discuss progress on sprint 6 user stories
- Continued working on user stories
- Watched this week's YouTube videos
- Tested game to see if there were any bugs.

Actions Planned

Edward Muldrew & Role: Programmer

- Meet up with the team during the week to complete the sprint submission document
- Work on testing the added code
- Ensure game ending and round ending code work efficiently
- Start working on Spell Cards
- Commit code to GIT

Grace Turkington & Role (2):

- Meet with team to discuss complete sprint 6 submission document
- Continue working on user stories
- Commit some changes to Git
- Write unit tests
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Meet with team to discuss and fill out sprint 6 report
- Refactor some code
- Commit some changes to Git
- Write unit tests for existing classes

Conor Clarke & Role (4):

- Meet with team to discuss and fill out sprint 6 report
- Put finishing touches on overworld screen

Jack McNaughton & Role (5):

- Meet with team to discuss complete sprint 6 submission document
- Continue working on user stories
- Watch this week's YouTube videos
- Attempted to get some screen shots from the game.

Meeting 9: Sprint 7, Week 1

Week commencing: 12/03/2018

Date of this minute: 12/03/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Met up with team on Friday to discuss Sprint Progress and software engineering assignment
- Cleaned up some code
- Added tests and new test classes to test the newly added code
- Committed code to GIT

Grace Turkington & Role (2):

- Met with team to discuss progress on sprint 6 user stories
- Continued working on user stories
- Committed some changes to Git
- Watched this weeks youtube videos

Brandon Smylie & Role (3):

- Met with team to discuss progress on sprint 6 user stories and software engineering document
- Committed some changes to Git

Conor Clarke & Role (4):

- Due to the random nature of the hands and decks it is very difficult to make smart AI
- Met with team to discuss progress on sprint 6 user stories and software engineering document

Jack McNaughton& Role (5):

- Met with team to discuss complete sprint 6 submission document
- Continued working on user stories
- Watched this week's YouTube videos
- Rewrite tutorial to account for changes in game.

Actions Planned

Edward Muldrew & Role: Programmer

- Meet up with team on Tuesday to discuss the final core pieces of functionality that need uploaded
- Work on spell cards
- Tidy up code and refactor
- Improve on existing code & methods
- Continue work on Software Engineering assignment

Grace Turkington & Role (2):

- Meet with team to discuss complete sprint 6 submission document
- Continue working on user stories
- Commit some changes to Git
- Write unit tests
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

- Meet with team to discuss final functionality
- Work on AI script wrapper
- Commit some changes to Git

Conor Clarke & Role (4):

- Create simple AI that plays cards
- Meet with team to discuss final functionality

Jack McNaughton& Role (5):

- Meet with team to discuss complete sprint 6 submission document
- Continue working on user stories
- Watch this week's YouTube videos
- Committed changes to git

Meeting 10: Sprint 7, Week 2

Week commencing: 12/03/2018

Date of this minute: 12/03/2018

Task Reporting

Edward Muldrew & Role: Programmer

- Added working spell cards
- Added exit button
- Added tweaks to placement of cards and collision detection
- Added bug fixes
- Met up with the team on Tuesday to discuss progress

Grace Turkington & Role (2):

- Met with team to discuss progress and work on software engineering assignment
- Refactored code
- Edited unit tests
- Watch this weeks youtube videos

Brandon Smylie & Role (3):

Conor Clarke & Role (4):

- Met with team to discuss progress and work on software engineering assignment
- Assign roles for the assignment

Jack McNaughton& Role (5):

Actions Planned

Edward Muldrew & Role: Programmer

- Complete any last bug fixes/tweaks to code
- Meet with team to complete project submission document
- Continue work on Software Engineering assignment
- Add relevant tests to newly added code
- Clean up code

Grace Turkington & Role (2):

- Meet with team to finalise software engineering assignment
- Keep refactoring code & testing for bugs
- Write more unit tests

Brandon Smylie & Role (3):

Conor Clarke & Role (4):

- Meet with team to finalise software engineering assignment
- Add collision detection to the player sprites

Jack McNaughton& Role (5):

Peer Assessment

Evaluation Group Number: 57 Group Name: Black Box Studios				
Name	Contribution of time and effort ¹	Contribution to team-working and motivation ¹	Contributions to this deliverable ^{1,2}	Peer Score (Range 85 – 115)
Conor Clarke	3	3	3	100
Jack McNaughton	2	3	2	95
Edward Muldrew	5	5	5	115
Brandon Smylie	5	4	4	105
Grace Turkington	4	5	4	105

¹Values: 1 = Less than average; 2 = Slightly less than average; 3 = Average; 4 = Slightly more than average; 5 = More than average

²This value should consider contributions in the round – direct contributions to required deliverables, and contributions that have made the deliverables possible.

Declaration		
Name	Date	Confirmation (<i>use the words shown in the example below!</i>)
Conor Clarke	23/03/18	I agree to the terms of the declaration
Jack McNaughton	23/03/18	I agree to the terms of the declaration
Edward Muldrew	23/03/18	I agree to the terms of the declaration
Brandon Smylie	23/03/18	I agree to the terms of the declaration
Grace Turkington	23/03/18	I agree to the terms of the declaration