Programming Using an Object-Oriented Language (CS12320)

Main Assignment: Patience: a card game

Name: Lauren Davis
Email: lad48@aber.ac.uk
Student ID: 210109168

Table of Contents

1 Introduction

1.1 Project Description

This project is based on a game called Patience; it is a simple card game which is played by a single person. The objective of the game is to end up with a single stack of cards, this is achieved by doing one of two things, placing on card on top of each other if they have the same number or suit and are next to each other or if they are two piles apart. The scoring system is 1 point per pile left on the table after there are no more cards in the face down deck, the main goal is to get as few points as possible.

This project was built on the template provided, it runs from the command line and has a basic graphical interface (WAS IT USED?). It works using a command line menu which allows the user to select an option for how they go about playing the game, once they have emptied the face down deck, they have the option to make a few final moves before ending the game if it is possible to do so. Once they choose to quit, they can choose to try and save their score if it is one that can be entered into the leader board. If the score is not good enough to do so, it will quit the game without saving their score.

1.2 What Has Been Achieved

This assignment had a set of 10 functional requirements and 3 non-functional requirements as set out below in two tables. I have crossed off each one that has been done.

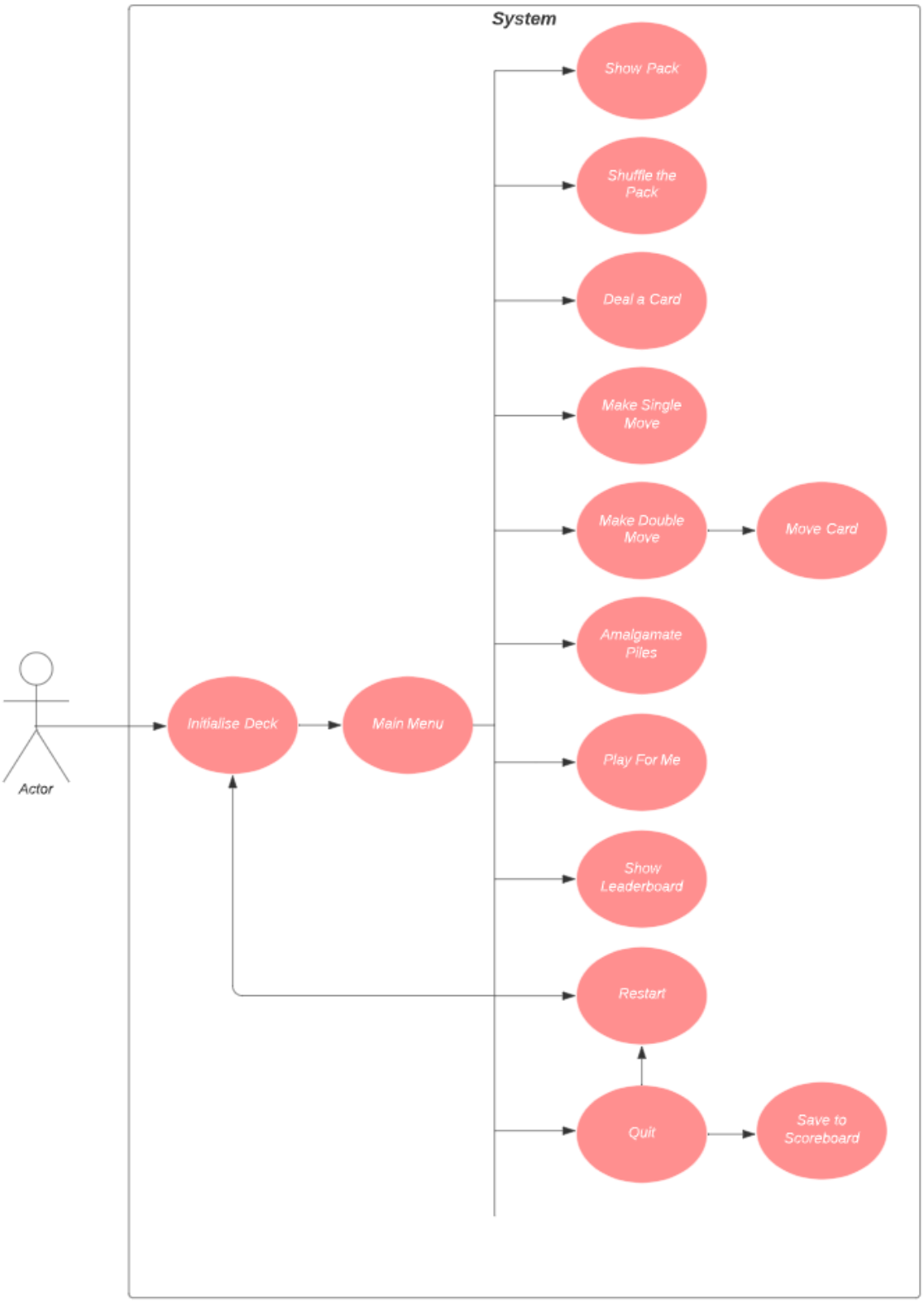
1.2.1 Non-Functional Requirements:

Requirement	**Done?**
Show command line menu	X
Showing cards in graphical interface	
Storing scores to display sorted top 10 scores	X

1.2.2 Functional Requirements:

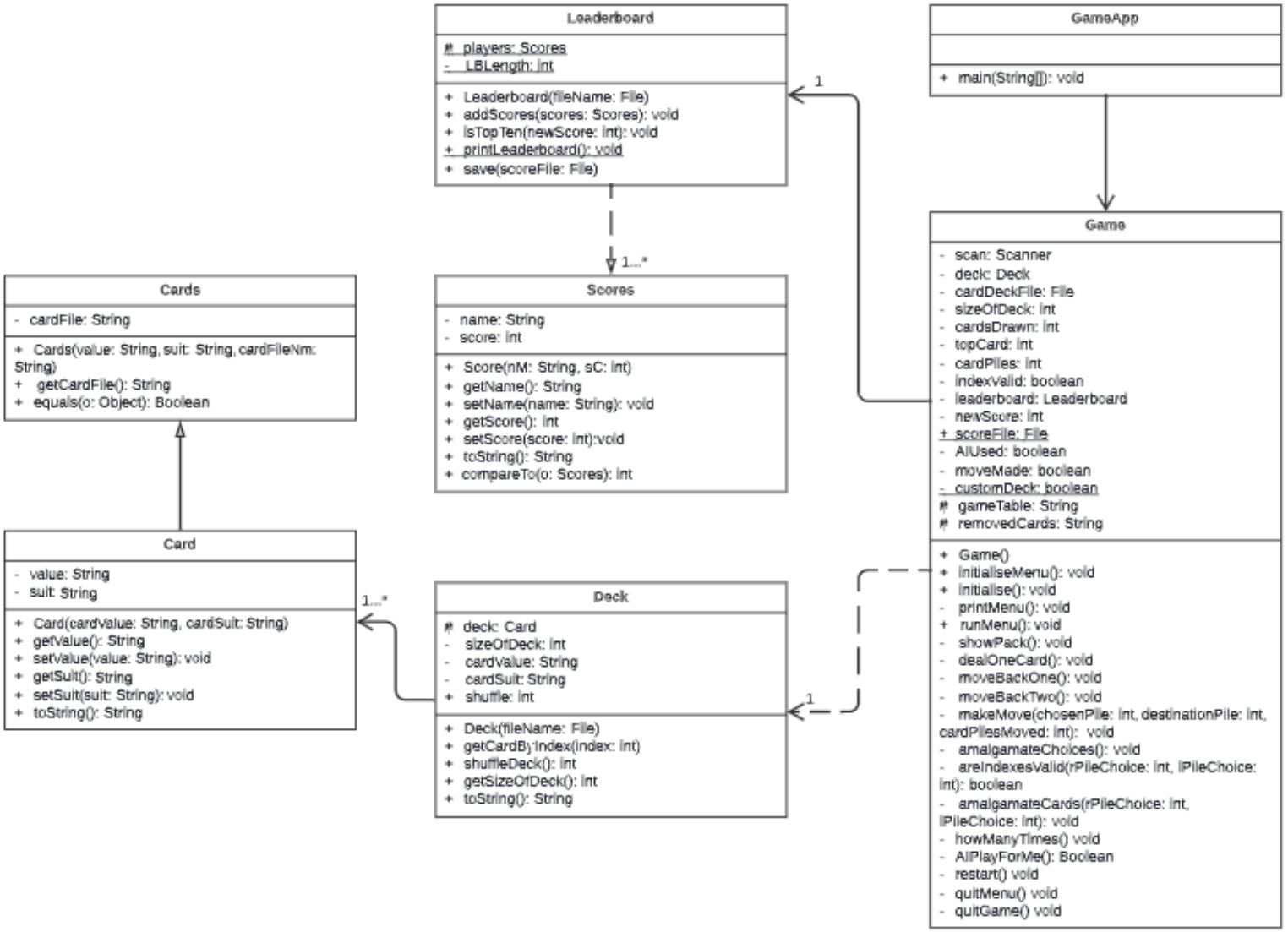
Requirement	**Done?**
Show the pack	X
Shuffle cards	X
Deal a card	X
Move last pile one	X
Move last pile over two	X
Amalgamate by position	X
Show all displayed cards in text form	X
Play for me once	X
Show top 10 results, sorted	X
Quit	X

1.3 Use Case Diagram:



2 Design

2.1 UML Class Diagram:



2.2 Class Descriptions:

GameApp is where the program creates instances of **Game**, it calls functions from **Game** to display the menus starting with the one which asks users what sort of deck they would like to use. Once the user has chosen, it runs the main menu for the user to pick what they would like to do next.

Game is the class which holds the majority of the functional requirements, it calls functions and variables from both **Leaderboard**, **Scores** and **Card**. This class implements various methods of error checking and validation to ensure that the user is unable to input something that would disrupt the flow of the program.

Card is the class which is where a single playing card is represented. Each single card has a value and suit, and an image which is implemented by the **Cards** class.

Cards is a class which extends the Card class, it holds the information required to call **Card** with its graphical format for the GUI implementation.

Deck is the class where the deck of cards is represented. Each deck is complied from a list of cards which is read into the program from a file, it uses **Card** to build each single card and adds them to the list which makes up the deck.

Scores is a class which holds information about the player, including their name and their score. It is referenced by **Leaderboard**.

Leaderboard is a class which holds information about the top 10 scores achieved by players. Player information is referenced from **Score**s, taking their name and score, and organizing it into a list in descending order, lowest to highest.

2.3 Pseudocode for amalgamate:

```
FUNCTION makeMove(INT chosenPile, INT destinationPile, INT cardPilesMoved) {
    STRING chosenCard = GET FROM gameTable using INDEX(chosenPile);
    STRING destinationCard = GET FROM gameTable using INDEX(destinationPile);
    CHARACTER cardValue1 = CHARACTER(0) FROM chosenCard
    CHARACTER cardSuit1 = CHARACTER(1) FROM chosenCard
    CHARACTER cardValue2 = CHARACTER(0) FROM destinationCard
    CHARACTER cardSuit2 = CHARACTER(1) FROM destinationCard
    IF (cardValue1 == cardValue2 OR cardSuit1 == cardSuit2) {
      IF (cardPilesMoved == 1) {
            REMOVE destinationPile FROM gameTable;
            DEINCREMENT cardPiles by 1;
            PRINT(CURRENT STATE OF gameTable);
```

```
    } ELSE IF (cardPilesMoved == 3) {
                REMOVE destinationPile FROM gameTable;
                ADD AT destinationPile THE chosenCard;
                REMOVE chosenPile FROM gameTable;
                DEINCREMENT cardPiles by 1;
               PRINT(CURRENT STATE OF gameTable);
    }
}
```

3 Testing

ID	Req.	Description	Input/Expected outcome	Pass/Fail	Comments
1	N/A	Command line menu that gives users the option to choose which deck configuration they want.	Input of 1 should load a standard deck.	P	See SS1.
			Input of 2 should ask the user what deck they would like to load.	P	See SS1.2.
			Input of Q should quit.	P	See SS1.3.
			Input of a character, symbol or number not listed should ask the user to try again.	P	See SS1.4.
			Input of a file name that does not exist should restart the program.	P	See SS1.5
2	NFR1	Command line menu that presents the options under FR objectives.	The menu should load once a deck has been chosen.	P	See SS2.1
			Inputs that meet one of the cases should call the required functions.	P	See SS2.1
			Inputs that are not one of the cases should tell the user to “Try again”.	P	See SS2.2
			Inputs of characters and symbols should not be allowed.	P	
3	FR1	Showing the pack based on what has been loaded into the program.	Input of a standard deck should show 52 cards listed in order.	P	See SS3.1
			After shuffling, the user should not be able to show the pack	P	
			Input of the custom double deck should show 104 cards.	P	See SS3.2
			Input of the custom half deck should show 26 cards.	P	See SS3.3
4	FR2	Shuffling the deck.	The deck should be shuffled once the option is picked.	P	See SS4.1
			Shuffling should only be allowed once.	P	See SS4.2
			Shuffling should be automatic if the user tries to make any other moves such as dealing a card.	P	See SS4.3

5	FR3	Deal one card from the top of the deck.	Selecting this option in the menu should deal a card, it should be able to be called multiple times.	P	See SS5.1
			Should tell the user when there are no more cards to deal.	P	See SS5.2
6	FR4	Move card onto previous pile.	Should move the card onto the previous pile if the suits or values match and give the current playing field.	P	See SS6.1
			Should tell the user if the move cannot be made.	P	See SS6.2
7	FR5	Move card onto a pile after skipping over two piles.	Should move the card onto pile after skipping two piles if the suits or values match and give the current playing field.	P	See SS7.1
			Should tell the user if the move cannot be made.	P	See SS7.2
8	FR6	Amalgamate piles given their indexes	Input of invalid indexes should ask the user to re-enter them, this includes symbols and letters.	P	See SS
			Input of valid indexes but invalid move should tell the user the move cannot be made.	P	See SS
			Input of valid indexes and valid moves of that in FR4 should be made.	P	See SS8.3
			Input of valid indexes and valid moves of that in FR5 should be made.	P	
9	FR7	Piles on field shown in text form.	Should show user current state of table.		See SS4.3, 6.1
10	FR8	Auto play a move, making the furthest move available	If there are no valid moves, it should draw a card.	P	See SS9.1
			Should run as many times as the user specifies.	P	
			Should be able to move a pile onto previous.	P	
			If it is able to do both move a pile onto previous and skip 2 piles, it should choose to skip 2 piles.	P	See SS9.2

			Should be able to move a pile after skipping 2.	P	
11	FR9	Show sorted Top 10 results	Should sort the scores from lowest to highest and print them.	P	See SS10.1, SS10.2
12	FR10	Quit the program	Should present the options to quit to the user.	P	See SS11.1
			Should save the score if it is low enough to go onto the leaderboard, if not, it should quit.	P	See SS11.2
			Should return to menu.	P	See SS11.3
			Should restart	P	See SS11.4
13	NFR2	Showing cards in GUI	Should be able to play the game through on a GUI	F	Focused on command line for too long and ran out of time.
14	NFR3	Storing saved scores in top 10	Should replace the bottom score if the user has managed to get a score low enough to place on the leaderboard.	P	See SS12.1
			Should not allow the user to save a score if they used AI or a custom deck or have not made a single move.	P	See SS12.2

3.1 Screenshots:

SS1.1

```
Please choose a deck to load:
1 -  Standard deck configuration
2 -  Custom deck configuration
Q -  Quit
1

1 -  Show the pack
2 -  Shuffle the cards
```

SS1.2

```
Please choose a deck to load:
1 -  Standard deck configuration
2 -  Custom deck configuration
Q -  Quit
2
WARNING: For a custom deck, you will not receive a score!
Enter the name of the deck file you would like to use, include .txt at the end:
```

SS1.3

```
Please choose a deck to load:
1 -  Standard deck configuration
2 -  Custom deck configuration
Q -  Quit
Q

Process finished with exit code 0
```

SS1.4

```
Please choose a deck to load:
1 -  Standard deck configuration
2 -  Custom deck configuration
Q -  Quit
£
Input not valid! Try again!
A
Input not valid! Try again!
6
Input not valid! Try again!
```

SS2.1

```
1 -  Show the pack
2 -  Shuffle the cards
3 -  Deal a card
4 -  Make a move: move last card onto previous pile
5 -  Make a move: move last card onto the pile skipping over two piles
6 -  Amalgamate all piles in the middle
7 -  Play for me
8 -  Show top 10 results
9 -  Restart with a new deck
Q -  Quit
What would you like to do:
1
ah 2h 3h 4h 5h 6h 7h 8h 9h th jh qh kh ad 2d 3d 4d 5d 6d 7d 8d 9d td jd qd kd ac 2c
What would you like to do:
2
The deck has been shuffled!
What would you like to do:
3
The drawn card is: 2c
The current playing field: 2c
What would you like to do:
4
Move cannot be made! Try again
What would you like to do:
5
Move cannot be made! Try again
What would you like to do:
6
Enter the position of the card moving:
```

```
What would you like to do:
7
How many times would you like me to make a move for you?
1
Moves cannot be made, so I have drawn a card!
The drawn card is: js
The current playing field: 2c, js
What would you like to do:
8
Top 10 Scores:
Lexi 1
Alice 1
Daniel 2
Lauren 2
Matt 2
Tiff 3
Lee 5
Joshua 6
Alex 6
James 8
What would you like to do:
9
Please choose a deck to load:
```

```
What would you like to do:
Q
Your score is: 52
```

SS2.2

```
What would you like to do:
10
Try again!
What would you like to do:
a
Try again!
What would you like to do:
%
Try again!
What would you like to do:
```

SS3.1

```
1
1 -  Show the pack
2 -  Shuffle the cards
3 -  Deal a card
4 -  Make a move: move last card onto previous pile
5 -  Make a move: move last card onto the pile skipping over two piles
6 -  Amalgamate all piles in the middle
7 -  Play for me
8 -  Show top 10 results
9 -  Restart with a new deck
Q -  Quit
What would you like to do:
1
ah 2h 3h 4h 5h 6h 7h 8h 9h th jh qh kh ad 2d 3d 4d 5d 6d 7d 8d 9d td jd qd kd ac 2c 3c 4c 5c 6c 7c 8c 9c tc jc qc kc as 2s 3s 4s 5s 6s 7s 8s 9s ts js qs ks
What would you like to do:
2
The deck has been shuffled!
What would you like to do:
1
You cannot view the deck once it has been shuffled!
```

SS3.2

```
doubledeck.txt
1 -  Show the pack
2 -  Shuffle the cards
3 -  Deal a card
4 -  Make a move: move last card onto previous pile
5 -  Make a move: move last card onto the pile skipping over two piles
6 -  Amalgamate all piles in the middle
7 -  Play for me
8 -  Show top 10 results
9 -  Restart with a new deck
Q -  Quit
What would you like to do:
1
ah 2h 3h 4h 5h 6h 7h 8h 9h th jh qh kh ad 2d 3d 4d 5d 6d 7d 8d 9d td jd qd kd ac 2c 3c 4c 5c 6c 7c 8c 9c tc jc qc kc as 2s 3s 4s 5s 6s 7s 8s 9s ts js qs ks ah 2h 3h 4h 5h 6h 7h 8h 9h th
What would you like to do:
```

SS3.3

```
halfdeck.txt
 1 -  Show the pack
 2 -  Shuffle the cards
 3 -  Deal a card
 4 -  Make a move: move last card onto previous pile
 5 -  Make a move: move last card onto the pile skipping over two piles
 6 -  Amalgamate all piles in the middle
 7 -  Play for me
 8 -  Show top 10 results
 9 -  Restart with a new deck
 Q -  Quit
 What would you like to do:
 1
 ah 2h 3h 4h 5h 6h 7h 8h 9h th jh qh kh ad 2d 3d 4d 5d 6d 7d 8d 9d td jd qd kd
 What would you like to do:
```

SS4.1

```
What would you like to do:
1
52
ah 2h 3h 4h 5h 6h 7h 8h 9h th jh qh kh ad 2d 3d 4d 5d 6d 7d 8d 9d td jd qd kd ac 2c 3c 4c 5c 6c 7c 8c 9c tc jc qc kc as 2s 3s 4s 5s 6s 7s 8s 9s ts js qs ks
2
The deck has been shuffled!
1
52
8s js 7h 5s 8d 6h tc ad 7s 3h kd 4s qc qs qh as 8c 5c 2d jh 5h 8h 6s 4h 3c 9c 2h 9d ah 2s 6d 9h jc ts 7d kc td 3s jd 6c ac ks qd 9s 4c 2c 3d kh 7c th 5d 4d
```

SS4.2

```
2
Deck can only be shuffled once per game!
What would you like to do:
```

SS4.3

```
What would you like to do:
3
The deck has been shuffled!
The drawn card is: jd
```

SS5.1

```
What would you like to do:
3
The deck has been shuffled!
The drawn card is: 2h
The current playing field: 2h
What would you like to do:
3
The drawn card is: 4h
The current playing field: 2h, 4h
What would you like to do:
3
The drawn card is: 6h
The current playing field: 2h, 4h, 6h
```

SS5.2

```
The drawn card is: kd
The current playing field: 2h, 4h, 6h, kh, qd, 9h, 8d, jd, td,
What would you like to do:
3
There are no more cards in this deck! Choose another option.
What would you like to do:
```

SS6.1

```
The drawn card is: 4s
The current playing field: 3c, 2s, jh, 5s, 9c, qs, 4s
What would you like to do:
4
The move has been made and the current playing field is: 3c, 2s, jh, 5s, 9c, 4s
```

SS6.2

```
Q -  Quit
What would you like to do:
4
Move cannot be made! Try again
What would you like to do:
3
The deck has been shuffled!
The drawn card is: 6s
The current playing field: 6s
What would you like to do:
3
The drawn card is: js
The current playing field: 6s, js
What would you like to do:
3
The drawn card is: kc
The current playing field: 6s, js, kc
What would you like to do:
4
Move cannot be made! Try again!
```

SS7.1

```
The current playing field: kh, 3d, 7c, as, qc, 6c
What would you like to do:
5
The move has been made and the current playing field is: kh, 3d, 6c, as, qc
```

SS7.2

```
The current playing field: kh, 3d, 7c, as
What would you like to do:
5
Move cannot be made! Try again!
```

SS8.1

```
What would you like to do:
6
Enter the position of the card moving:
1
The input you have given is not an integer! Try again!
Enter the position of the card moving:
&
The input you have given is not an integer! Try again!
Enter the position of the card moving:
4
Enter the position of where the card is to be moved to:
8
Indexes are not valid, try again!
```

SS8.2

```
The move has been made and the current playing field is: 2s, ts, qc, 8s, jd
What would you like to do:
6
Enter the position of the card moving:
5
Enter the position of where the card is to be moved to:
4
Move cannot be made! Try again!
```

SS8.3

```
The current playing field: kh, 2s, ts, ks, qc, 8s, jd
What would you like to do:
6
Enter the position of the card moving:
4
Enter the position of where the card is to be moved to:
1
The move has been made and the current playing field is: ks, 2s, ts, qc, 8s, jd
What would you like to do:
6
Enter the position of the card moving:
2
Enter the position of where the card is to be moved to:
1
The move has been made and the current playing field is: 2s, ts, qc, 8s, jd
```

SS9.1

```
What would you like to do:
7
How many times would you like me to make a move for you?
3
The deck has to be shuffled before you start, this has been done automatically
The deck has been shuffled!
Moves cannot be made, so I have drawn a card!
The drawn card is: ks
The current playing field: ks
Moves cannot be made, so I have drawn a card!
The drawn card is: 9s
The current playing field: ks, 9s
The move has been made and the current playing field is: 9s
What would you like to do:
```

SS9.2

```
The drawn card is: jc
The current playing field: 9s, jh, 5s, ac, jc
What would you like to do:
7
How many times would you like me to make a move for you?
1
The move has been made and the current playing field is: 9s, jc, 5s, ac
What would you like to do:
```

SS10.1

```
What would you like to do:
8
Top 10 Scores:
Lexi 1
Alice 1
Daniel 2
Lauren 2
Matt 2
Tiff 3
Lee 5
Alex 6
Joshua 6
James 8
```

SS10.2

```
Lexi 1
James 8
Alice 1
Daniel 2
Alex 6
Lauren 2
Matt 2
Joshua 6
Tiff 3
Lee 5
```

SS11.1

```
Q
Your score is: 52
Please choose an option:
1 -  Save and quit
2 -  Return to menu
3 -  Play again with a new deck
```

SS11.2

```
Your score is: 52
Please choose an option:
1 -  Save and quit
2 -  Return to menu
3 -  Play again with a new deck

1
Top 10 Scores:
Lexi 1
Alice 1
Daniel 2
Lauren 2
Matt 2
Tiff 3
Lee 5
Alex 6
Joshua 6
James 8
Thank you for playing!
```

SS11.3

```
2
1 -  Show the pack
2 -  Shuffle the cards
3 -  Deal a card
```

SS11.4

```
3
Please choose a deck to load:
1 -  Standard deck configuration
2 -  Custom deck configuration
Q -  Quit
```

SS12.1

```
Your score is: 17
Please choose an option:
1 -  Save and quit
2 -  Return to menu
3 -  Play again with a new deck

1
You are in the top 10! Would you like to save your score? Y/N
y
Please enter your name:
lauren
Top 10 Scores:
Lexi 1
Alice 1
Lauren 3
Matt 3
Tiff 5
Daniel 7
James 8
Joshua 10
lauren 17
Lee 51
Thank you for playing!
```

SS12.2

```
q
Your score is: 52
Please choose an option:
1 -  Save and quit
2 -  Return to menu
3 -  Play again with a new deck

1
You cannot save your score!
Thank you for playing!
```

4 Evaluation

4.1 How I went about solving the assignment:

I started by designing the application first, outlining everything I will need for each class, starting with cards, then moving onto deck, and so on. After doing this, I build up their classes – adding in the getters and setters for each class if it required it before moving onto the main class Game. When it came to working on the Game class, it was a lot harder, there were many requirements that had to be met so I decided that the first thing I should do is make the command line menu which would allow me to test each individual requirement separately. I referenced my mini assignment for the best way to do specific functions or how to implement some basic inheritance features.

4.2 What was difficult:

I think the most difficult thing was getting started, I struggled to do much at all in the beginning as I did originally try to do it without a plan and instantly got overwhelmed. Having to break down the code into smaller classes and methods without a real starting point was difficult. Also, remembering to fill in the testing table, it is not as in depth as it should be. I kept forgetting to fill it in with some major functions so there are sometimes only 1 or 2 tests when I did maybe 7 or 8.

4.3 What remains to be done:

The whole GUI aspect was left untouched because it took me so long to work on the command line aspect, I just did not have the time to work on it. I also believe there is more to be done such as adding an undo option, or difficulty levels. I believe that I could have done more work on the inheritance aspect of it all too as I feel it is very basic inheritance.

4.4 What I learnt:

I learnt that it is a good idea to always have a plan and write out what you want to do first, rather than going straight into coding – it is very easy to get overwhelmed and make mistakes, especially when you do not take a moment to think about what you are doing. I learnt that going forward that if you break down the tasks into smaller chunks that can be tested easily, and adding in debug statements, it can solve a lot of problems and makes it easier to find where the issues are. For example, in my AIPlayForMe () function, I was finding that there were multiple problems with it and adding in debug statements allowed me to narrow it down to a logic error not a syntax error. Finally, always take a copy of your work, it helps to work out what happened when it worked before, and then suddenly does not.

4.5 What mark and why:

I would be happy to achieve 70% I believe that I did everything I could, even adding in extra functionality such as custom decks, and functions to force players to make moves. I believe I lost marks for the GUI and the inheritance aspect but covered everything else required.