Title of the assignment

Student Name

student@aber.ac.uk

Submission Date: XX/XX/XXXX

Table of contents

Introduction

For our main assignment in CS12320 we were tasked with creating a card game called Patience (Solitaire), a card game where you pile drawn cards on top of each other if they meet specific conditions, the objective is to have as little piles of cards as possible for the highest score. This text documents my progression and final project.

We were given a project template which contained two classes: Game, which was the main class and CardTable, which had the code for a GUI table where the cards were displayed. Game consisted of base code and required additional code to make the game functional. Initially, I added a superclass named Deck and a subclass named Card. For creativity and innovation, I added three additional classes to implement an accessibility mode, enabling a screen reader for visually impaired players.

Use Case Diagram

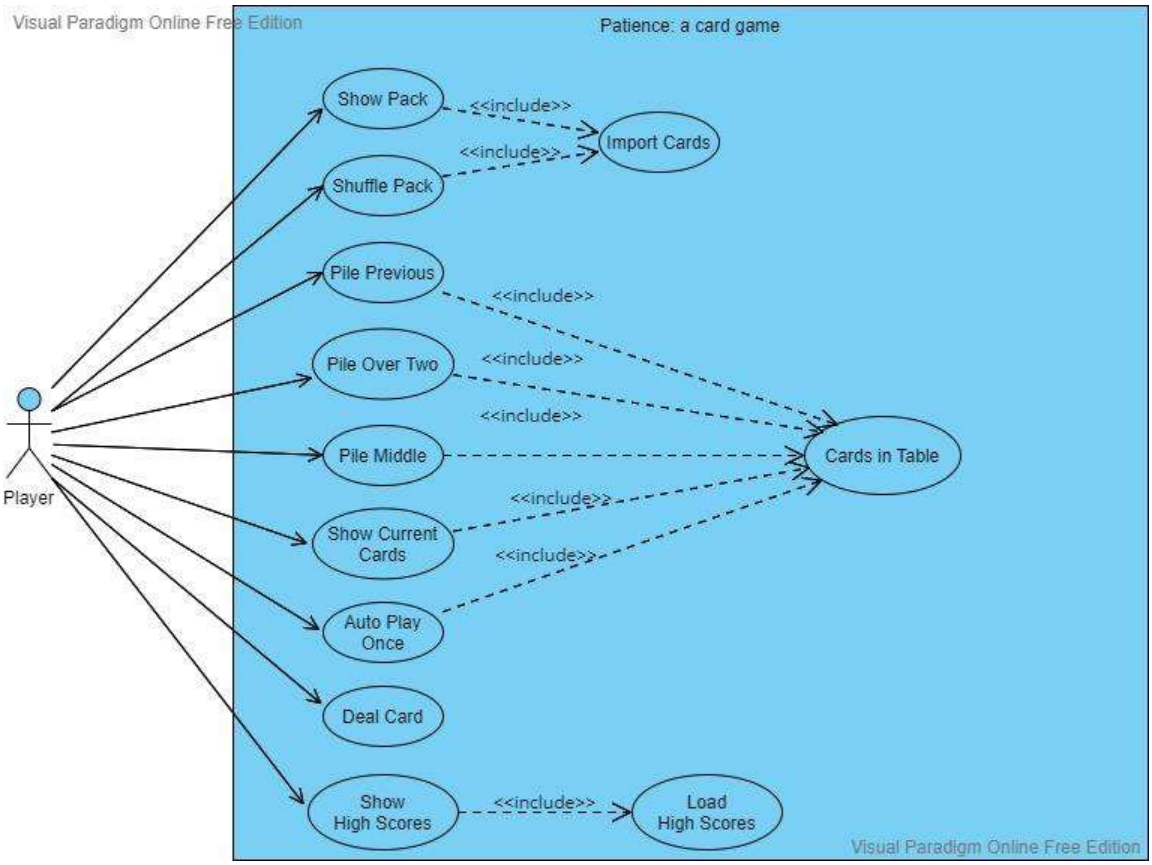


Figure 1 - Use case diagram for Patience: a card game.

Design

Class Diagram

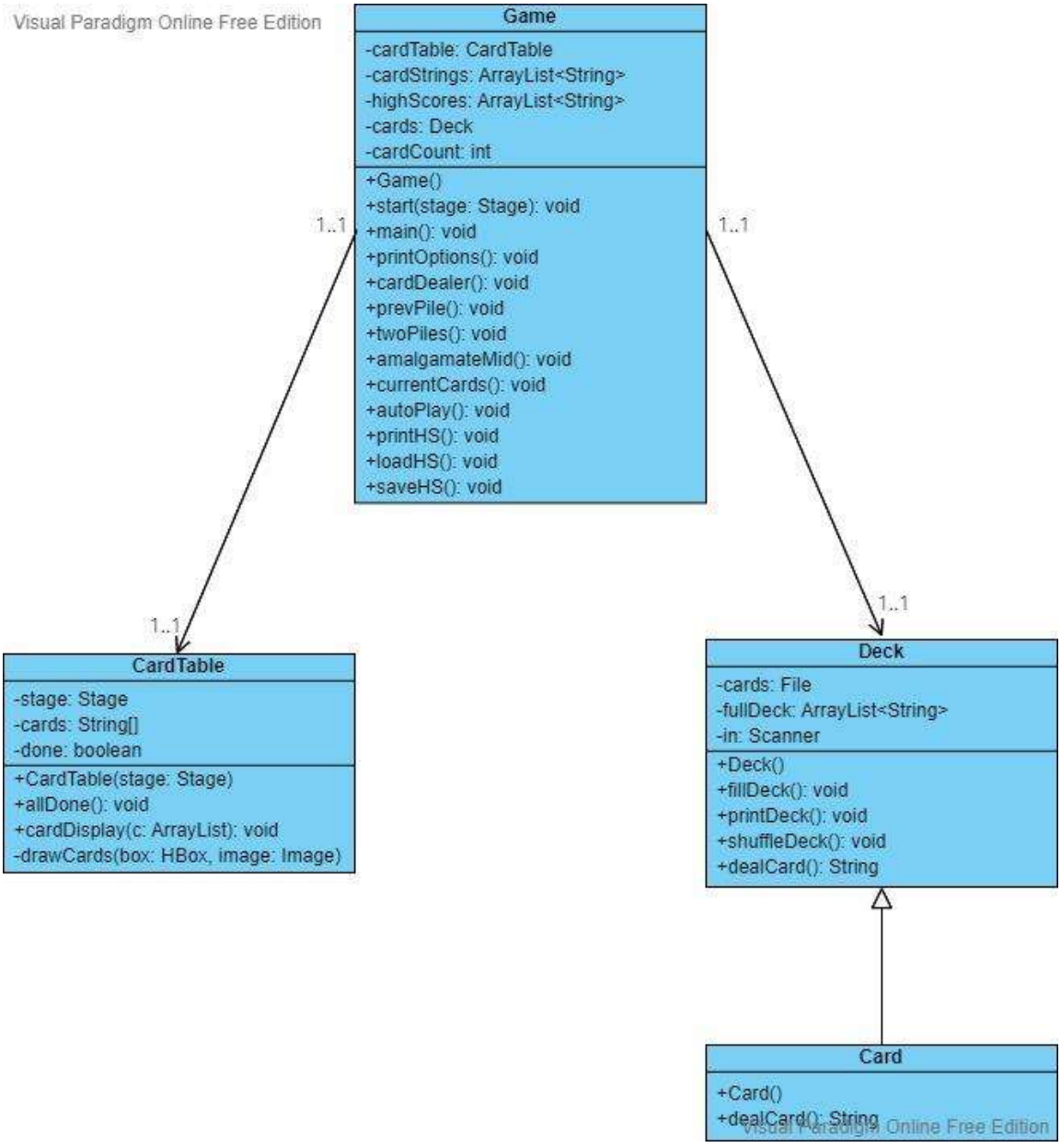


Figure 2 - My initial class diagram, made as the plan for my code.

Figure 2 shows my initial class diagram, I created this before I started my code. As a result of this, a few of the method and attribute names are slightly different. Also, my creativity and innovation implementation was not part of my plan at this stage. An updated class diagram is shown below.

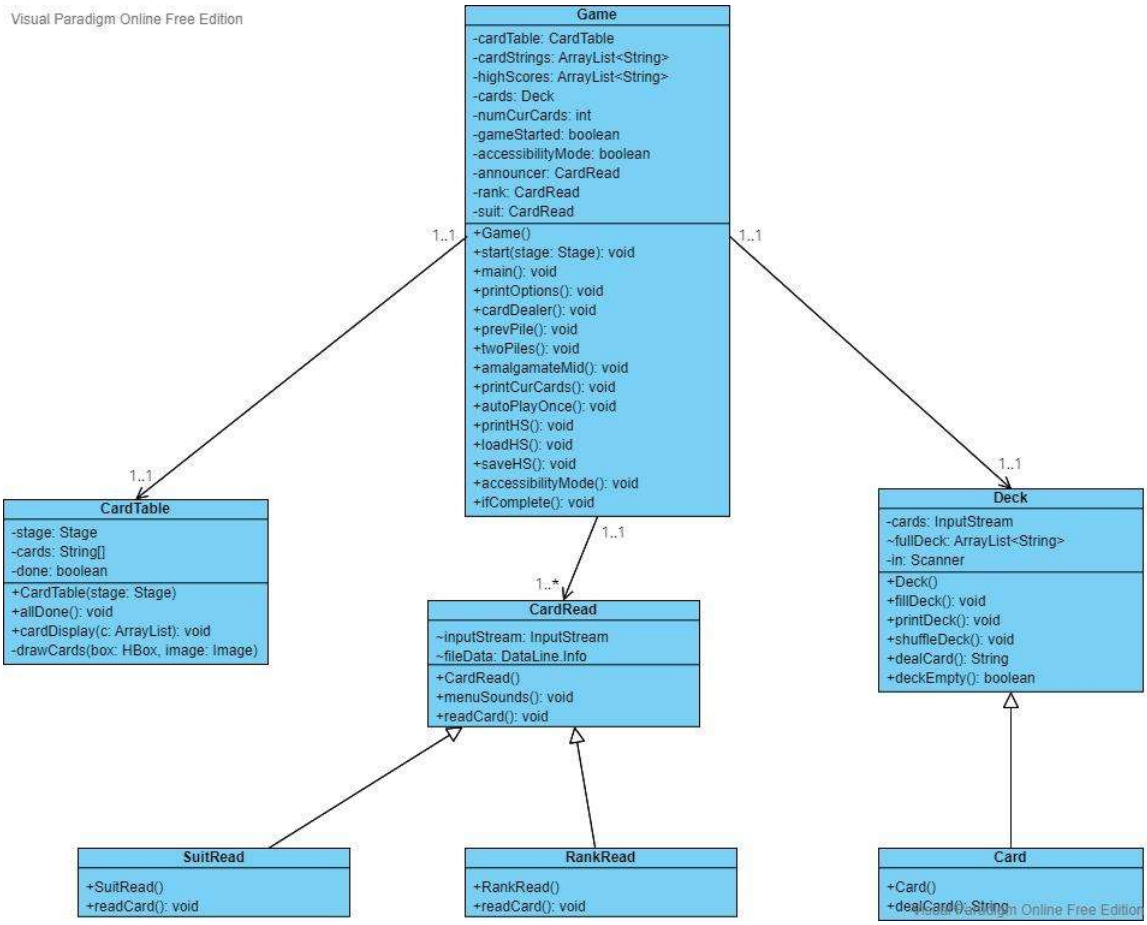


Figure 3 - Updated final class diagram, showing my final attribute/method names and new implementation.

Class Descriptions

Game

This is the main class. When the program is executed, the menu is called from here, which allows the player to start the game. Most of the options in the menu call methods inside of this class. The methods work with the other classes to process each function. Game has a one-to-one relationship with both CardTable and Deck because the game only has one deck and one card table. Game also has a one-to-many relationship with CardRead because the game has many cards to read.

1. Game() – Constructor.
2. start() – The main method of the program; most of the code is called from here. I have implemented a Switch statement which takes user input, it loops on a do…while loop, this can be used to select any option the player needs. The high scores also load from here and save from here when the switch statement is exited.
3. main() – Where the program starts from.
4. accessibilityMode() – An addition setting I have added for the creativity and innovation marks. When the program is started, this method is called. It asks the player visually and audibly if they would like to enable accessibility mode, if the player enters yes, it will be turned on allowing visually impaired players to play the game. It functions from three

classes, enabling audio for every card that is drawn and every option that the player will need to hear. This has only been implemented for the core functionality of the program, the functions that are not necessary, such as printing the pack and high scores, are not included.

5. printOptions() – Displays the menu options on the command-line.
6. cardDealer() – Called when a card is drawn. It checks if the deck is empty, if it is not, it uses the Card class to retrieve a card from the Deck class. This String is then used to add the card to the GUI card table.
7. prevPile() – Called when player wants to move a card to the previous pile. It checks if there are enough cards in the table, if there are it will pile the card as requested.
8. twoPiles() – Called when player wants to move a card back over two piles. Again, it checks if there are enough cards in the table, if there are it will pile the card as requested.
9. amalgamateMid() – Called when a player wants to amalgamate cards in the middle of the table. This method uses Scanner to take user input for what card is moving and where the card is moving to.
10. printCurCards() – Prints the current cards in the table to the command-line in ascending order.
11. autoPlayOnce() – This is the "play for me once" function. It uses an algorithm to check for a far match, if there is not a match, it will check for a close match. It utilises the previously mentioned methods prevPiles() and twoPiles() to execute the move once a match is found. The algorithm for this is shown in the pseudo-code example in this document.
12. printHS() – Prints out the high scores, which are loaded from a file into an ArrayList at the start of the game.
13. loadHS() – Uses FileReader to load the high scores text file and populates the ArrayList highScores with the data.
14. saveHS() – Uses FileWriter to save the highScores ArrayList to a text file.
15. ifComplete() – This is a method that is called when a player completes a game. It will ask for their name, their score will be recorded and added to the highScores Arraylist. A sort will then be executed, if the new score beats any of the old scores it will now be in the ArrayList, otherwise it will be in position 10 which is removed after sorting. After highScores has been updated, saveHS() is called.

CardTable

This class was the provided class, I have not made any edits to the code in this class. CardTable has a one-to-one relationship to Game because the card table is only associated with this game.

Deck

Deck is used to load the "cards.txt" file. From there it processes all functionalities that involve the deck. Deck has a one-to-one relationship with Game because the deck of cards is only used in this game.

1. Deck() – Constructor.
2. fillDeck() – This method is called when the program starts. It populates the ArrayList fullDeck.
3. printDeck() – Called when the first option of the menu is requested. It displays every card left in the deck, starting a new line every quarter for elegant formatting in the command-line.

4. shuffleDeck() – Shuffles the deck using “Collections.shuffle()”.
5. dealCard() – Parent to child class in Card.
6. deckEmpty() – Checks if the deck is empty and returns a Boolean with the output.

Card

The subclass of superclass Deck. This class is where single card functions are executed.

1. Card() – Constructor.
2. dealCard() – This is the method called by cardDealer() in main. It takes the next card in fullDeck (ArrayList in Deck), removes it from the ArrayList and returns the String.

CardRead

This is the superclass for my creativity and innovation feature. It contains methods that utilise AudioInputStream and Clip to source then play each audio file. CardRead has a one-to-one relationship with game as it belongs to only one game.

CardRead() – Constructor.

menuSounds() – Takes a statement as a String parameter, then plays the corresponding file.

readCard() – A method which is used in this class, as well as two subclasses, called by cardDealer() in Game to announce the card that has been dealt.

RankRead

Subclass of CardRead used to read the rank of the card that has been drawn.

RankRead() – Constructor.

readCard() – Override of the Superclass, takes the first character of the card as a parameter to read the rank of the card.

SuitRead

Subclass of CardRead used to read the suit of the card that has been drawn.

SuitRead() – Constructor.

readCard() – Override of the Superclass, takes the second character of the card as a parameter to read the Suit of the card.

Pseudo-Code Example

```
if there are four or more cards in the table
    if cards two spaces away match
        amalgamate the two
    else if cards next to eachother match
        amalgamate the two
    else
        error
else if there are 2 or more cards in the table
    if cards next to eachother match
        amalgamate the two
    else
        error
else
    print message no match possible
```

Figure 4 - Pseudo-code for my autoPlayOnce() method.

Testing

Test Table

ID	Requirement	Description	Input	Expected Output	Pass/ Fail	Comments
A1.1	FR1	Show the pack.	Entered “1”	Screenshot A1.1	Pass	
A2.1	FR2	Shuffle the cards.	Entered “2”	Screenshot A2.1	Pass	Message saying it is complete.
A2.2			Entered “1”	Screenshot A2.2	Pass	FR1 again to show that the shuffle worked.
A3.1	FR3	Deal a card.	Entered “3”	Screenshot A3.1	Pass	Card dealt.
A3.2				Screenshot A3.2 (Error check)	Pass	Showing the message displayed when there are no cards left to deal.
A4.1	FR4	Move the last card onto	Entered “4”	Screenshot A4.1	Pass	Shows the deck before (A4.1) and after (A4.2) the input.
A4.2				Screenshot A4.2		

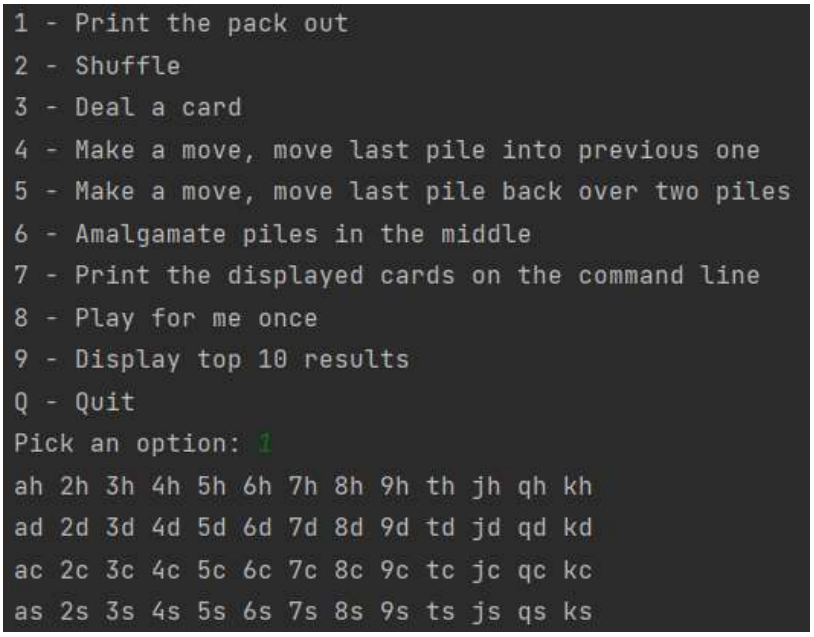
A4.3		previous pile.		Screenshot A4.3 (Error check)	Pass	If there are not enough cards to use this function, a message will tell the player.
A5.1	FR5	Move the last card onto the pile skipping over two piles.	Entered “5”	Screenshot A5.1	Pass	Shows the deck before (A5.1) and after (A5.2) the input.
A5.2				Screenshot A5.2		
A5.3				Screenshot A5.3 (Error check)	Pass	If there are not enough cards to use this function, a message will tell the player.
A6.1	FR6	Amalgamate piles in the middle by giving their numbers.	Entered “6”	Screenshot A6.1	Pass	Shows the deck before (A6.1) and after (A6.2) the input.
A6.2				Screenshot A6.2		
A6.3				Screenshot A6.3 (Error check)	Pass	If there are not enough cards for this function to be useful, a message will tell the player.
A6.4			Entered “6” Entered “#”	Screenshot A6.4 (Error check)	Pass	Checks the input is an integer.
A6.5			Entered “6” Entered “5”	Screenshot A6.5 (Error check)	Pass	If the player tried to input a value that is out of range, the function will tell the player.
A7.1	FR7	Show the displayed cards in text form on the command-line.	Entered “7”	Screenshot A7.1	Pass	Outputs the cards in the table to the command-line.
A8.1	FR8	Play for me once. If there is more than one possible move, then the algorithm will choose the	Entered “8”	Screenshot A8.1	Pass	Shows the deck before (A8.1) and after (A8.2) the input, furthest move priority example.
A8.2				Screenshot A8.2		
A8.3				Screenshot A8.3	Pass	Shows the deck before (A8.3) and after (A8.4) the input, for a close move.
A8.4				Screenshot A8.4		

A8.5		'furthest' one.		Screenshot A8.5	Pass	When there are no moves possible, it will present a message.
A8.6				Screenshot A8.6	Pass	When there are not enough cards to use the function, it will present a message.
A9.1	FR9	Show top ten results of all games so far, sorted with best score at the top.	Entered "9"	Screenshot A9.1	Pass	Displays high scores to command-line.
A10.1	FR10	Quit.	Entered "Q" Entered "Name"	Screenshot A10.1	Pass	When game has been completed, quits game and updates high scores.
A10.2			Entered "Q"	Screenshot A10.2	Pass	When game has been quit early, it quits the game and does not log player's score.
A11.1	NFR1	Command-line menu.	Entered "no"	Screenshot A14.1	Pass	Shown once the accessibility question is answered.
A11.2			Entered "no" Entered "#"	Screenshot A11.2	Pass	If a menu input does not correspond with any of the options, an error message will be returned asking to check input.
A12.1	NFR2	Card changes shown in a basic graphical interface.	Entered "3"	Screenshot A4.1 Screenshot A4.2	Pass	
A13.1	NFR3	Save scores and ask for players name when game ends.	Entered "Q"	Screenshot A10.1	Pass	Saves score if the game has been completed, otherwise the

						score is discarded.
A14.1	Command-line	Show the program running in command-line.	Screenshot A14.1	Screenshot A14.1	Pass	Change command line directory to patience-template folder and use same command as Screenshot A14.1.

Screenshots From Testing

A1.1



A2.1

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 2
Shuffle complete!
```

A2.2

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 1
qc 6c as 2d ad 6s 4c 2h kd jh ac 3h 4d
7h qh 3s ah 7d jd jc 2c qd 9h 8s ts 3d
4s 6d 5d 8d 9s 8h 4h td kh 5h 2s js 9d
kc tc 6h 5s ks th 3c 7c 9c 5c qs 7s 8c
```

A3.1

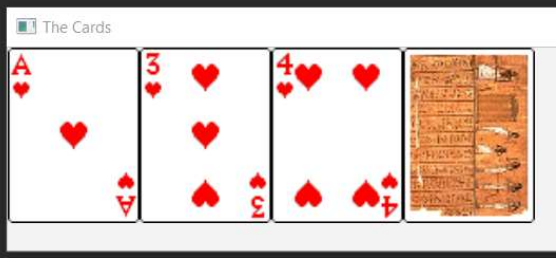
```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 3
Card dealt!
```

A3.2

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 3
No cards left! Make your last possible moves then press Q to quit.
```

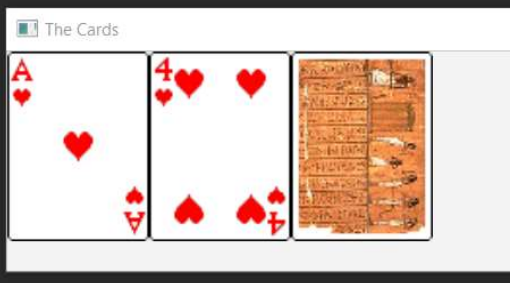
A4.1

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option:
```



A4.2

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 4
```

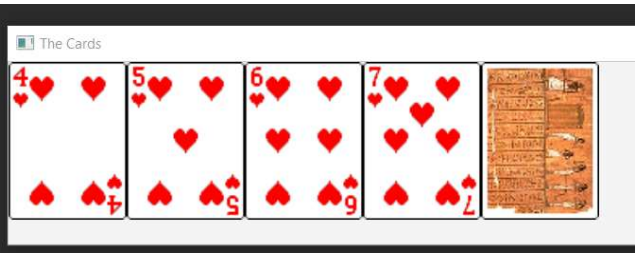


A4.3

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 4
Not enough cards to amalgamate!
Please deal another card and try again.
```

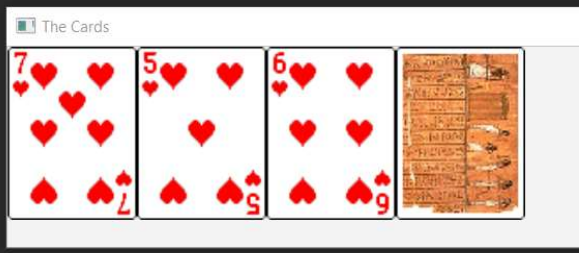
A5.1

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option:
```



A5.2

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 5
```

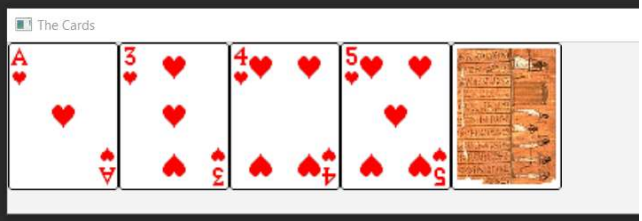


A5.3

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 5
Not enough cards to amalgamate!
Please deal another card and try again.
```

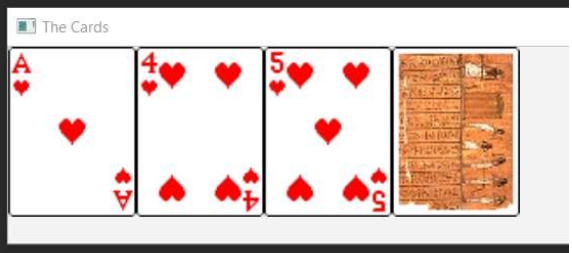
A6.1

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 6
Which pile is moving?
3
Where is it moving to?
```



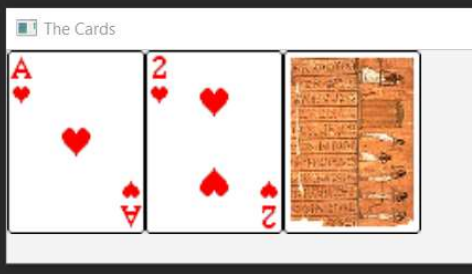
A6.2

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 6
Which pile is moving?
3
Where is it moving to?
2
```



A6.3

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 6
Not enough cards for this function to be needed!
Please deal another card and try again.
```

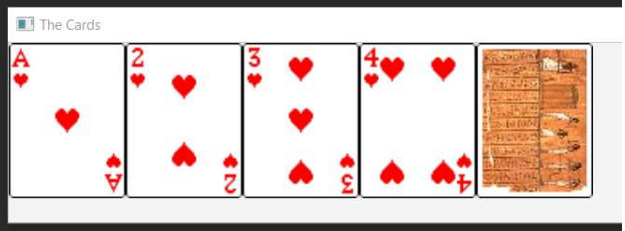


A6.4

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 6
Which pile is moving?
#
Please enter a number.
```

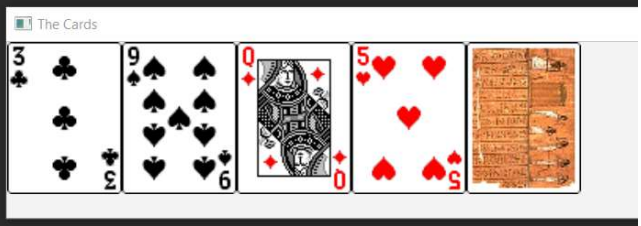
A6.5

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 6
Which pile is moving?
5
Input out of range, please check cards in table
```



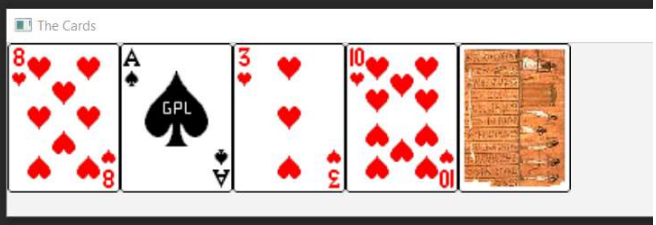
A7.1

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 7
Cards shown in table:
3c 9s qd 5h
```



A8.1

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option:
```

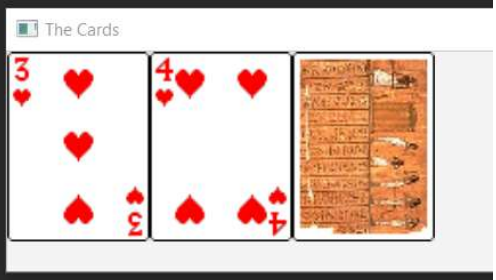


A8.2

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 8
```

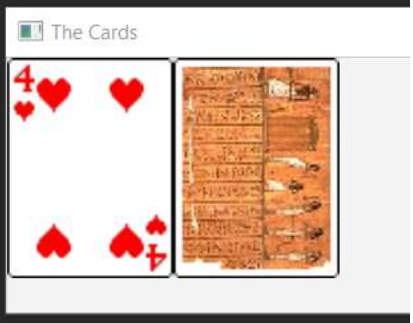
A8.3

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option:
```



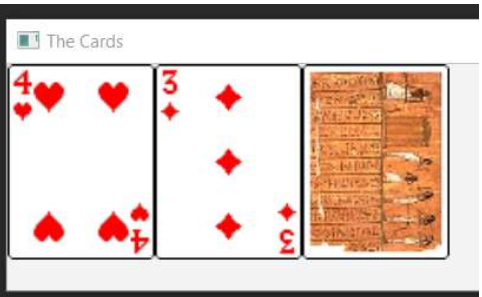
A8.4

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 8
```



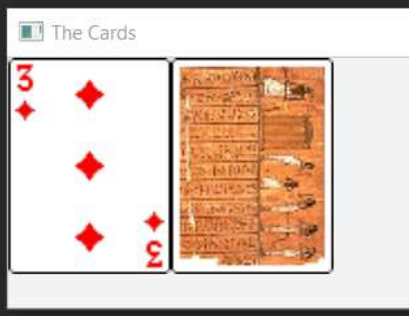
A8.5

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 8
Nothing to play!
```



A8.6

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 8
Not enough cards to make a play!
Please deal another card.
```



A9.1

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: 9
Top ten scores:
1: 42 by Matt
2: 48 by Matt
3: 50 by Justin
4: 51 by John
5: 51 by Katrina
6: 51 by Matt
7: 52 by John
8: 52 by John
9: 52 by John
10: 52 by John
```

A10.1

```
No cards left! Make your last possible moves then press Q to quit.
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: q
Game Complete - Number of piles is 43.
Enter your name:
Name
Close the window to fully stop playing.
```

A10.2

```
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: q
GAME OVER - Number of piles is 2.
But you bailed out! so your score will be erased.
Close the window to fully stop playing.
```

A11.2

```
Would you like to enable accessibility mode? yes or no
no
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option: #
Error, please check your input.
```

A14.1

```
C:\Users\temp>cd desktop/oo main assignment/patience-template

C:\Users\temp\Desktop\OO Main assignment\patience-template>java -cp out\production\patience-template --module-path "C:\U
sers\temp\Documents\JavaFX\javafx-sdk-18.0.1\lib" --add-modules javafx.controls,javafx.fxml uk.ac.aber.dcs.cs12320.Game
Would you like to enable accessibility mode? yes or no
no
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile into previous one
5 - Make a move, move last pile back over two piles
6 - Amalgamate piles in the middle
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Display top 10 results
Q - Quit
Pick an option:
```

Testing Discussion

I have shown every required function, the test table includes comments talking about the outcome of every test. In my testing I encountered no errors. I am confident the vast range of tests I have conducted demonstrates the stability of the program.

Evaluation

How I Solved the Assignment

I made sure to create a reliable plan. For example, I wrote pseudo-code for many of the complex functions. I still encountered some issues during the coding, but I was able to solve them. After I finished the code, I added the accessibility mode for creativity and innovation, it is essentially a built-in screen reader.

What I Found Difficult

Getting used to inheritance and the planning process, working out the class table for it to be as useful in implementation as possible. As shown in this document, my initial class diagram was relatively accurate, but I updated it with the screen-reader classes.

What Remains to Be Done

I was able to complete every function required, resulting in a complete program.

What I Have Learnt

I believe that in this assignment I was able to create a more robust and efficient program in contrast with my mini-assignment outcome. Throughout the coding process, I felt more confident with Java.

What Mark I Think I Should Be Awarded

I think that my code is written to a good standard, and I have spent more hours on this assignment than any before. I am happy with the outcome of the program and think that because of the quality of it, combined with this document, my mark should be close to 75/100.