CS12320 MAIN ASSIGNMENT

-ALAN ANTONY-
ala96@aber.ac.uk
words:1915

CONTENTS

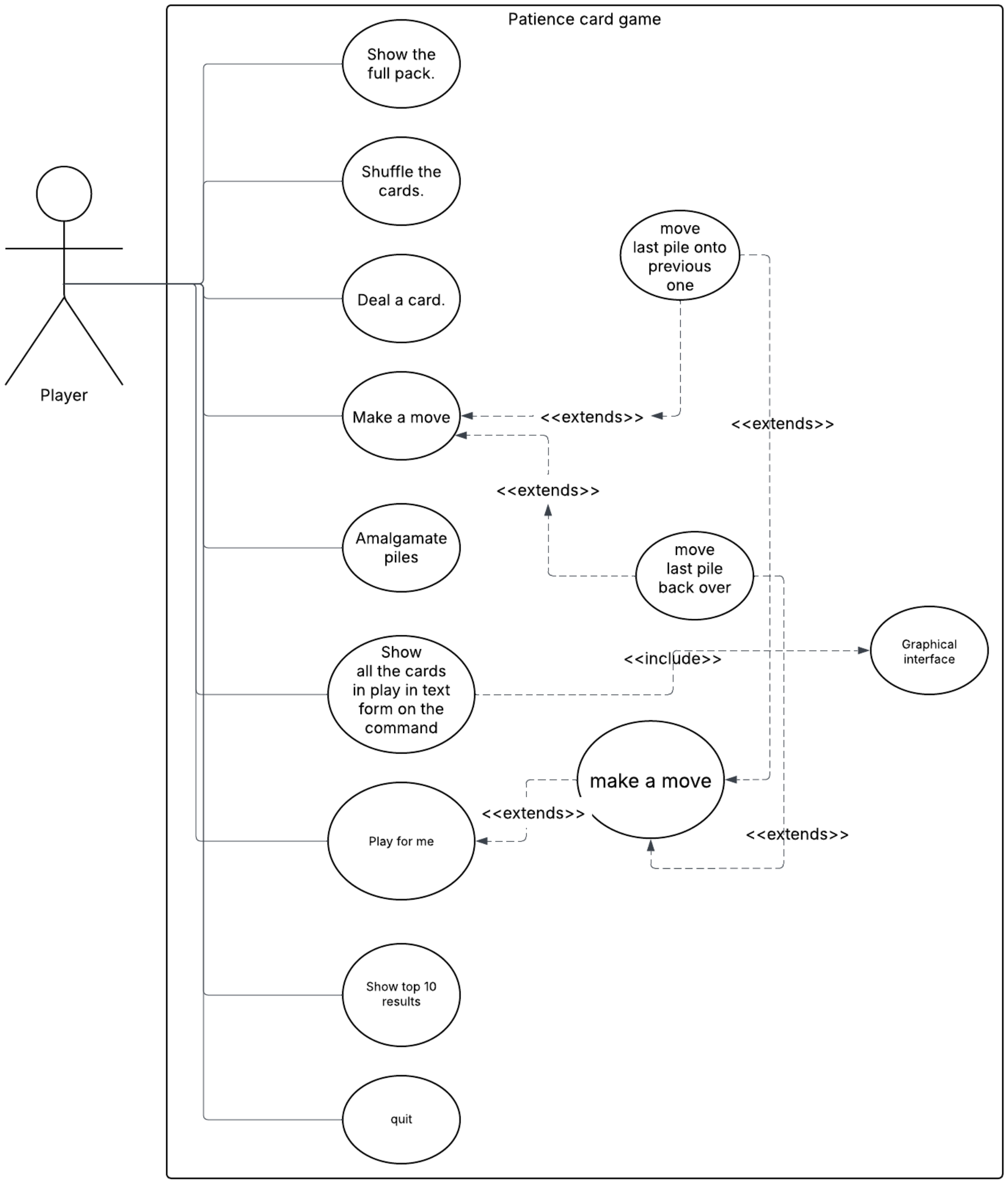
Introduction:

This report documents the implementation of a solitaire variant card game named "Patience" with various different rules. It follows these set of instructions: Shuffle a deck of cards then start dealing into separate piles; combine the cards based on 2 factors: if adjacent cards are the same suit or number or if cards 2 piles away are the same suit of number. My version of the patience game achieves most of the requirements highlighted in the assignment brief, including all required functional requirements and 2 out of the three non-functional requirements. Key features of my game include a command line interface, dealing and shuffling ability, a satisfactory saving score method and object orientated design using encapsulation and inheritance. The following page of this report covers a use-case and UML diagram highlighting how the game is designed, testing of code and an evaluation.

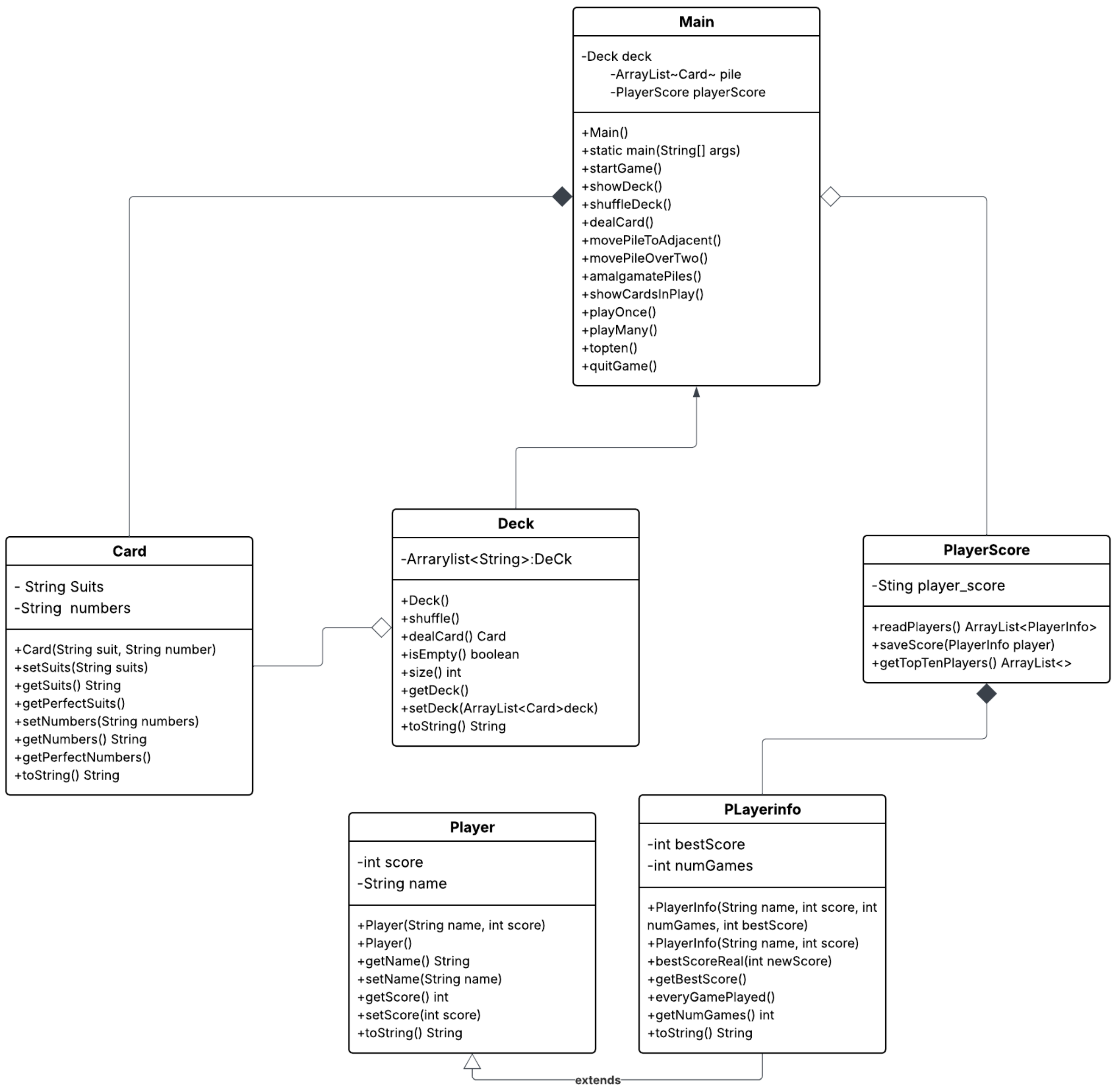
The implementation of this game began with the smaller classes which I found were the foundation for the game: Card and Deck classes. By producing these classes first, I was able to build a solid base before moving on to the more complex parts of code the came later. I found that using array lists were the best way to handle development of cards and to store all information for the game. This allowed a lot of information to be sorted in a way which allowed me to produce the most efficient code for the project. The text based display provides a seamless interaction, allowing users to access all parts of the game making the whole process of playing easy and engaging.

All in all, this report underlines how OOP principles were optimized to produce the most efficient code which checks most requirements assigned. This report will analyze the strengths and weaknesses of the project I am handing carefully evaluating and proving why I deserve the final mark I give myself.

UML Case Diagram:



UML:



Main
-Deck deck
-ArrayList~Card~ pile
-PlayerScore playerScore
+Main()
+static main(String[] args)
+startGame()
+showDeck()
+shuffleDeck()
+dealCard()
+movePileToAdjacent()
+movePileOverTwo()
+amalgamatePiles()
+showCardsInPlay()
+playOnce()
+playMany()
+topten()
+quitGame()
Card
- String Suits
-String numbers
+Card(String suit, String number)
+setSuits(String suits)
+getSuits() String
+getPerfectSuits()
+setNumbers(String numbers)
+getNumbers() String
+getPerfectNumbers()
+toString() String
Deck
-Arrarylist<String>:DeCk
+Deck()
+shuffle()
+dealCard() Card
+isEmpty() boolean
+size() int
+getDeck()
+setDeck(ArrayList<Card>deck)
+toString() String
PlayerScore
-Sting player_score
+readPlayers() ArrayList<PlayerInfo>
+saveScore(PlayerInfo player)
+getTopTenPlayers() ArrayList<>
Player
-int score
-String name
+Player(String name, int score)
+Player()
+getName() String
+setName(String name)
+getScore() int
+setScore(int score)
+toString() String
PLayerinfo
-int bestScore
-int numGames
+PlayerInfo(String name, int score, int numGames, int bestScore)
+PlayerInfo(String name, int score)
+bestScoreReal(int newScore)
+getBestScore()
+everyGamePlayed()
+getNumGames() int
+toString() String
extends

Class description:

- **The main class:**
  This is a class designed to call all the methods that can be used within the game. It takes inputs from the user to then call different methods, which tackle all functional requirements of the game. The startGame method is used to display what the user can do; the switch cases within the code are used to call the different FR methods based on the user's input. The show deck and shuffle deck use the Deck class to print cards and shuffle cards. Shuffling is done using the Fisher-Yates method. This method then prints a confirmation statement showing that the cards have been shuffled. I then tackle the next FR, which is to deal a card using the dealCard method. It first checks if there are any cards left to deal, and if there aren't, it will output a message saying so. However, if there are cards that can be dealt, then it deals a card from - deck.dealCard(). MovePileAdjacent verifies that one card can move on top of another if they have the same suit or number. MovePileOverTwo allows the user to make a move in which, if a card two piles away has the same suit or number, it can move to the pile. Amalgamate piles allow for the cards missed to be moved by choosing which pile you want to move. The play for me methods do what the methods describe; play for the player where the furthest card is used. I did this by using a for statement to check all the cards side by side and two cards apart to see if moves could be completed. The top ten method uses the player info, player, and player score classes, where it calls the text file to which the previous players are saved to find which players have the top 10 scores. The quit game method is used to check if any cards were played, and if they were, then it prompts the user to give a name, and this is saved to the score.txt file along with the score, best score, and number of times played.
- **The card class:**
  Represents individual cards within the deck. It contains two attributes, including the numbers and suits values of a card. These have setters and getters. I have also included two methods which include getPerfectSuits, which saves the suits to an ArrayList and getPerfectNumbers, this saves the numbers to an ArrayList. There is also a toString which returns the string representation of the cards.
- **The deck class:**
  This is used to represent all cards in the deck. The only attribute I use here is the ArrayList which I use to store the combination of the suits and numbers from the card deck. I include the shuffle method here. My biggest problem was the shuffling

part of the program as I couldn't figure out how to do it. After researching, I came upon the Fisher-Yates shuffle that had a unique approach to shuffling using the size of an ArrayList to do so. I could have also used the Collections shuffle, however, I didn't really understand how that worked so I used the Fisher-Yates shuffle instead. I also include the dealCard which removes cards from the deck ArrayList and adds them to the pile.

- **The player class:**
  This stores player basic information. Name - The player's name
  Score - The player's current score
  Getters and setters for name and score
  toString - Returns a formatted string with the player's name and score
- **PlayerInfo:**
  This class extends Player to include additional statistics about a player's performance over time. Key extensions include:
  bestScore - Tracks the player's best (lowest) score across multiple games
  numGames - Tracks how many games the player has played
  bestScoreReal - Updates bestScore if a new score is lower than the current best
  everyGamePlayed - Increments the games played counter
  Override of toString - Returns games played and best score information
- **PlayerScore:**
  This class handles reading and writing player score information to a text file. Key methods include:
  readPlayers - Reads player data from a text file and returns an ArrayList of PlayerInfo objects
  saveScore - Updates or adds a player's score to the file, ensuring best scores are maintained
  getTopTenPlayers - Returns the top ten players sorted by best score

PseudoCode for the playOnce method in the Main class:

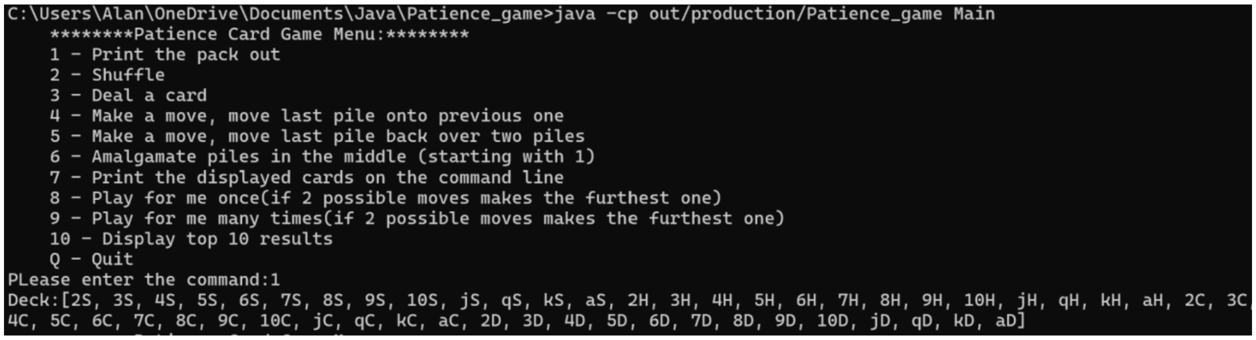
```
Public void method playOnce();

        IF (pile.size() < 2) THEN:
                Print ("Not enough cards to make a move")
                Return
        ENDIF

        moveMade  = false

        IF (pile.size() >= 4) THEN
                FOR (i=0, i < pile.size() -3, i++) THEN
                        Card1 = pile.get(i)
                        Card2 = pile.get(i+3)
                        IF(card.getNumbers().equals(twoCard.getNumbers()) ||
card.getSuits().equals(twoCard.getSuits())
                        movingCard = pile.remove(i+3)
                        pile.set(i,movingCard)
                        PRINT("Make a move jumping over 2 piles")
                        PRINT pile
                        moveMade = true
                        RETURN
                        ENDIF
                ENDFOR
        ENDIF

        IF (moveMade = false) THEN
                FOR (int i = 0; i < pile.size(), i++) THEN
                        Card1 = pile.get(i)
                        Card2 = pile.get(i+1)
                        IF (card.getNumbers().equals(twoCard.getNumbers()) ||
card.getSuits().equals(twoCard.getSuits())
                                movingCard = pile.remove(i+1)
                                pile.set(i, movingCard)
                                PRINT ("Made a move between adjacent piles")
                PRINT pile
                moveMade = true
                RETURN
            ENDIF
        ENDFOR
    ENDIF

    IF(moveMade = false) THEN
        PRINT (“No valid moves available”)
    ENDIF
ENDMETHOD
```

Testing Functional Requirements:

NAME	REQUIRMENTS	EXPECTED OUTCOME	PASS/FAIL
NFR1	Your game must have a command-line menu. The command line menu should have the following functions	Shown in: n/a	F
FR1	Show the full pack	Shown in: 1	Y
FR2	Shuffle the cards	Shown in: 2	Y
FR3	Deal a card	Shown in: 3	Y
FR4	Make a move, move the last pile onto the previous one, if allowed.	Shown in: 4,5	Y
FR5	Make a move, move the last pile back over two piles, if allowed.	Shown in:6,7	Y
FR6	Amalgamate piles in the middle (by giving their numbers).	Shown in:8,9	Y
FR7	Show all the cards in play in text form on the command line.	Shown in:10	Y
FR8	Play for me once	Shown in:11,12	Y
FR9	Play it for me several times	Shown in:13	Y

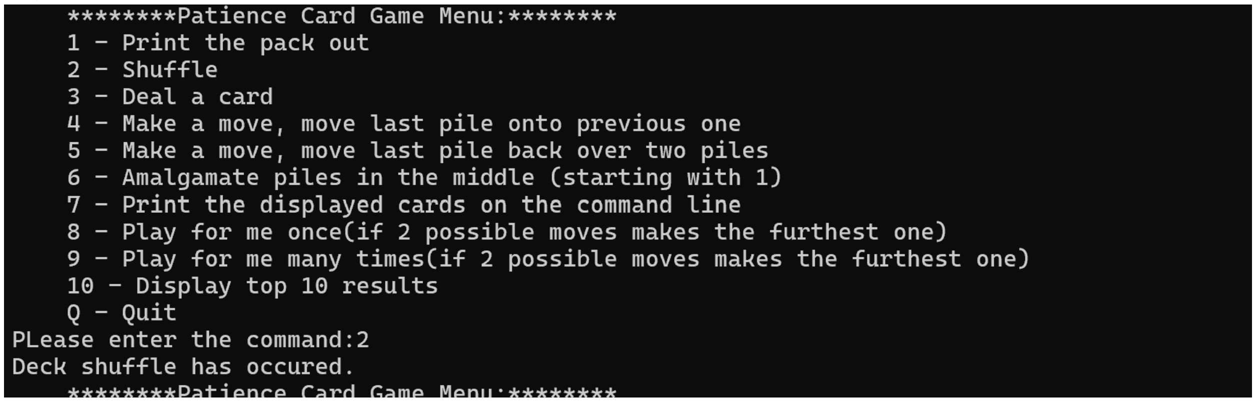
FR10	Show the top ten results	Shown in:14	Y
FR11	Quit the game	Shown in: 15	Y
NFR2	Save score	Shown in: Text file saved in same package as code	Y
NFR3	GUI	Shown in: n/a	N

Figure 1:



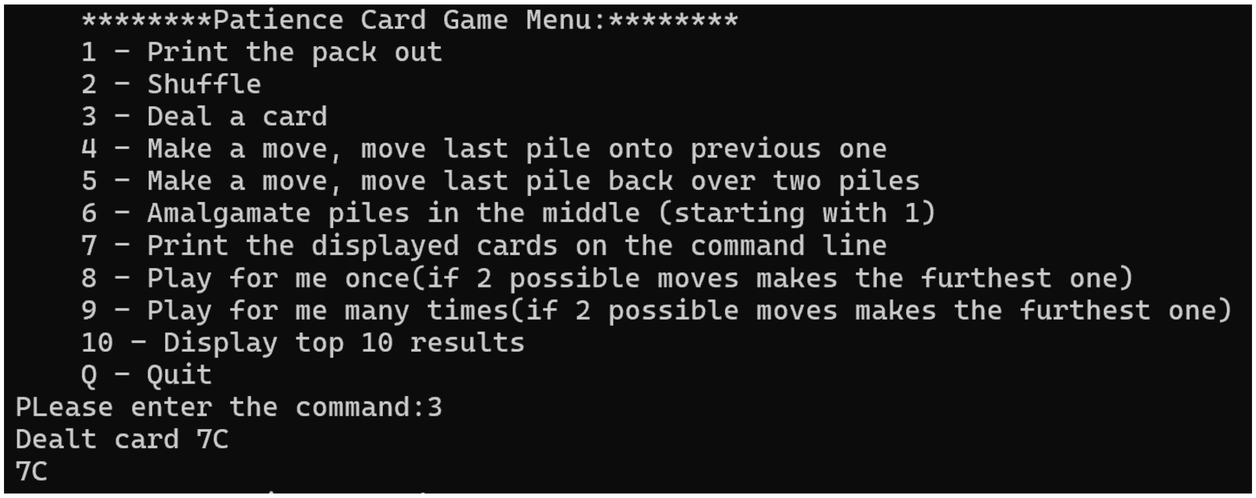
```
C:\Users\Alan\OneDrive\Documents\Java\Patience_game>java -cp out/production/Patience_game Main
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:1
Deck:[2S, 3S, 4S, 5S, 6S, 7S, 8S, 9S, 10S, jS, qS, kS, aS, 2H, 3H, 4H, 5H, 6H, 7H, 8H, 9H, 10H, jH, qH, kH, aH, 2C, 3C
4C, 5C, 6C, 7C, 8C, 9C, 10C, jC, qC, kC, aC, 2D, 3D, 4D, 5D, 6D, 7D, 8D, 9D, 10D, jD, qD, kD, aD]
```

Figure 2:



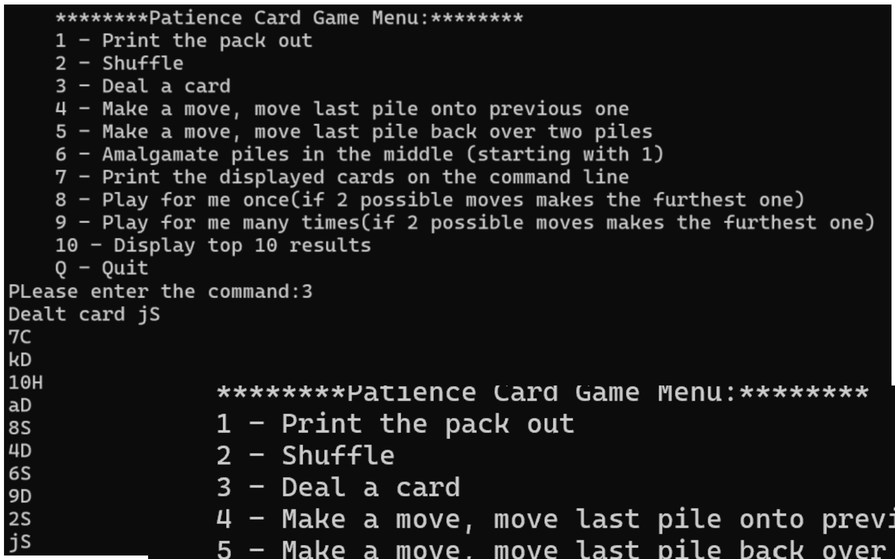
```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:2
Deck shuffle has occured.
    ********Patience Card Game Menu:********
```

Figure 3:

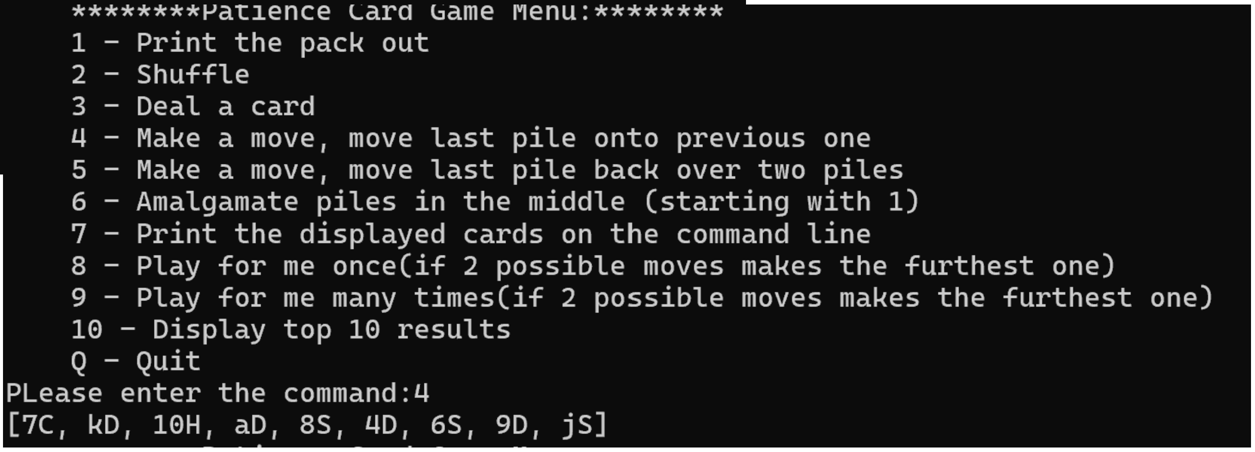


```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:3
Dealt card 7C
7C
```

Figure 4 and Figure 5:

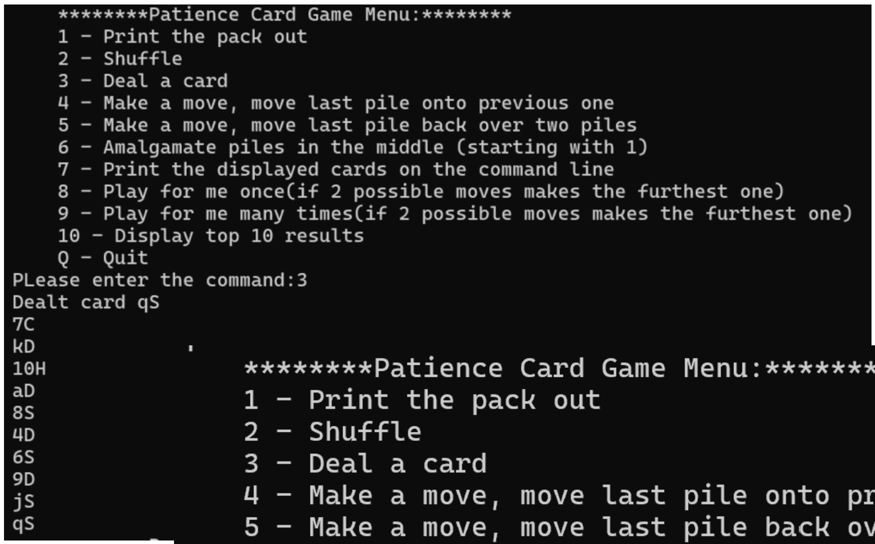


```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:3
Dealt card jS
7C
kD
10H
aD
8S
4D
6S
9D
2S
jS
```

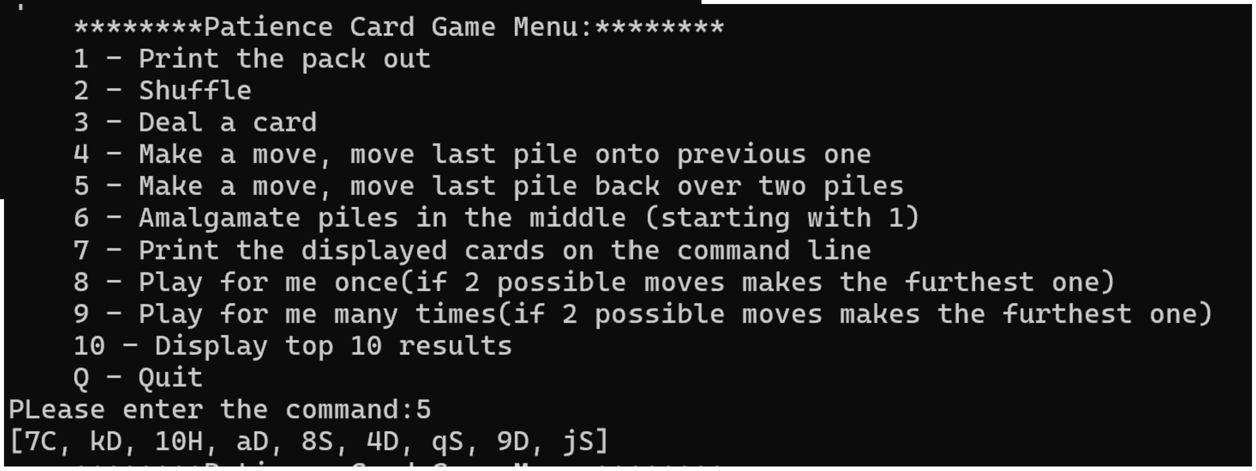


```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:4
[7C, kD, 10H, aD, 8S, 4D, 6S, 9D, jS]
```

Figure 6 and Figure 7:

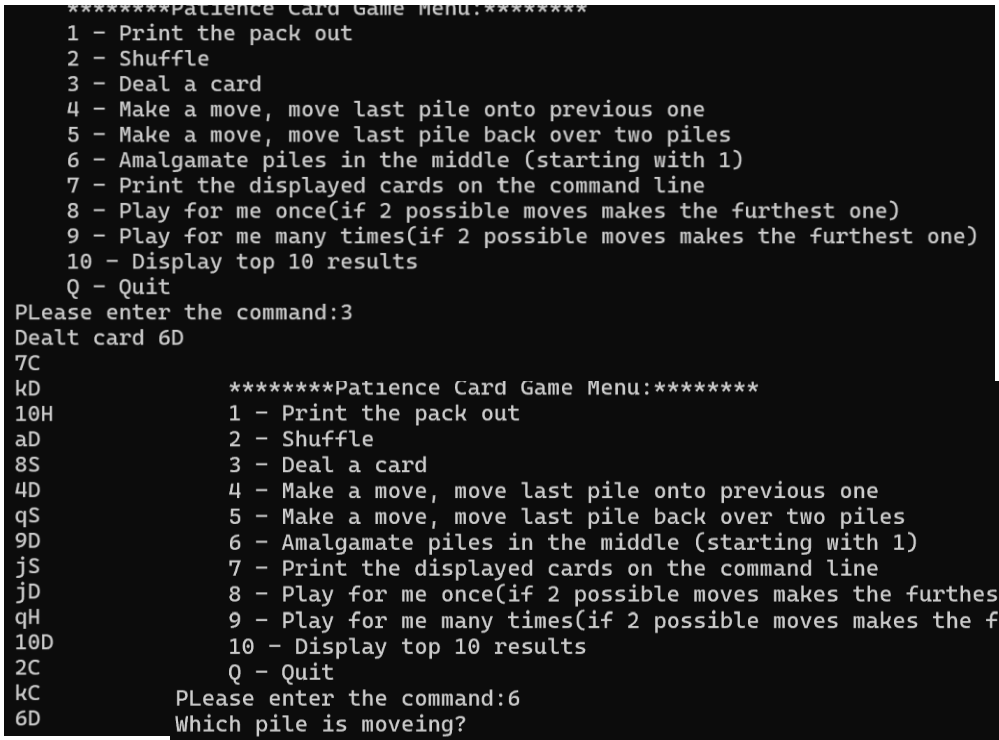


```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:3
Dealt card qS
7C
kD
10H
aD
8S
4D
6S
9D
jS
qS
```

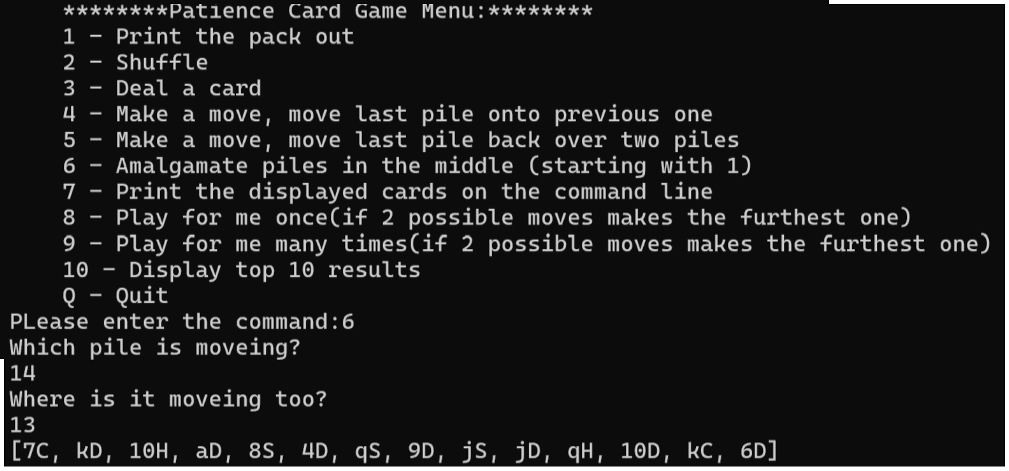


```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:5
[7C, kD, 10H, aD, 8S, 4D, qS, 9D, jS]
```

Figure 8 and Figure 9:

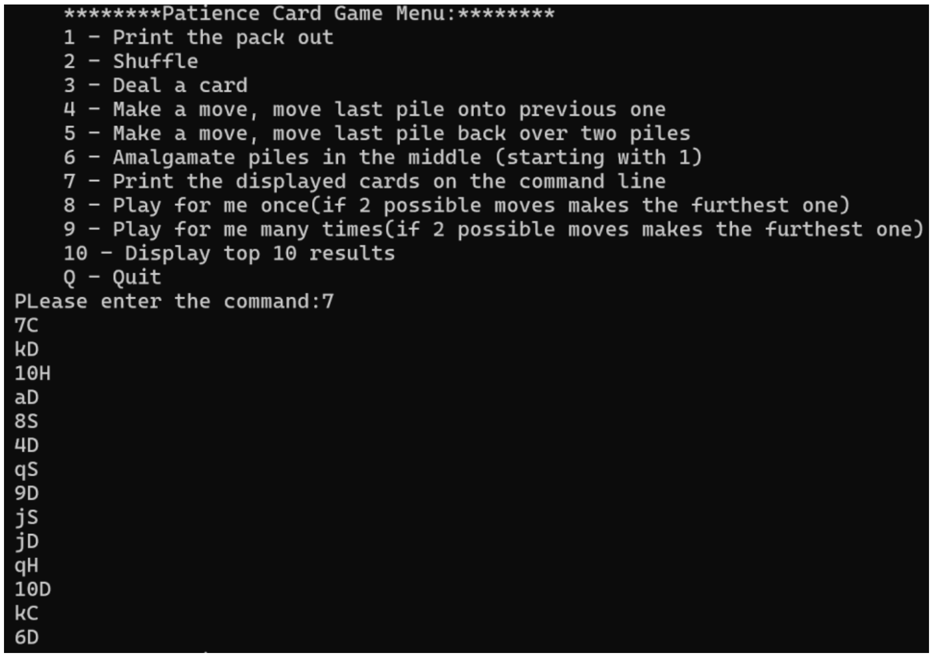


```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:3
Dealt card 6D
7C
kD
10H
aD
8S
4D
qS
9D
jS
jD
qH
10D
2C
kC
6D
```



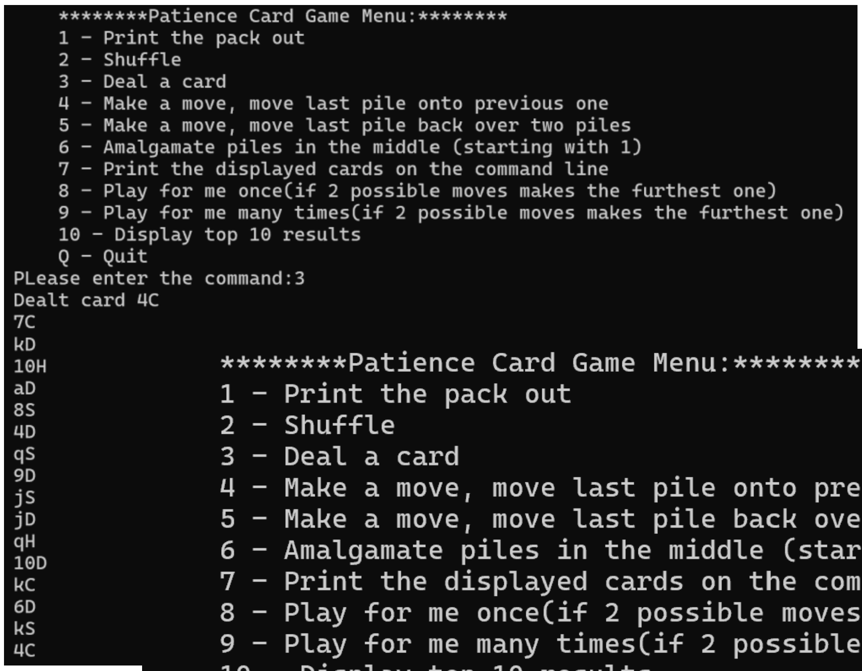
```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:6
Which pile is moveing?
14
Where is it moveing too?
13
[7C, kD, 10H, aD, 8S, 4D, qS, 9D, jS, jD, qH, 10D, kC, 6D]
```

Figure 10:

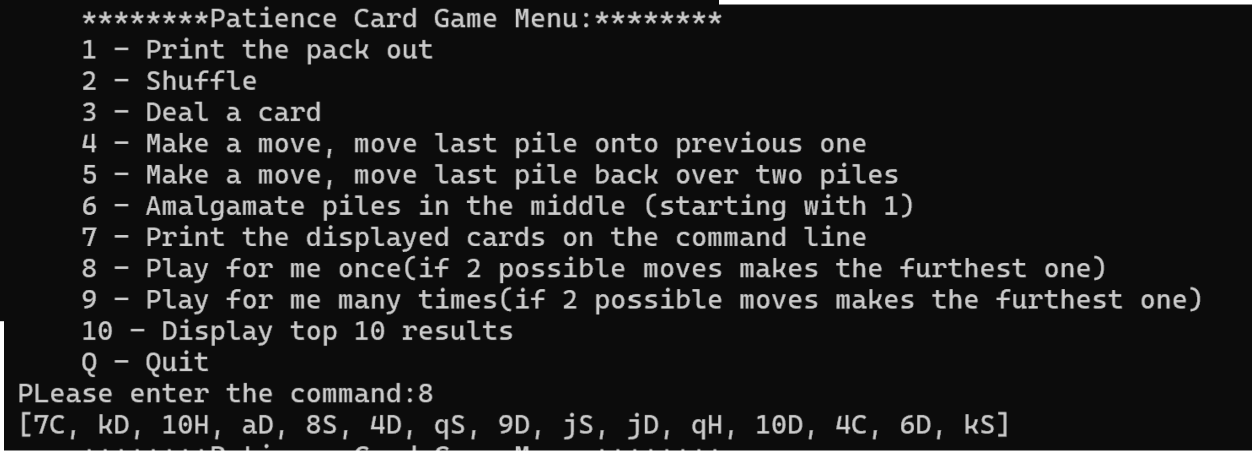


```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:7
7C
kD
10H
aD
8S
4D
qS
9D
jS
jD
qH
10D
kC
6D
```

Figure 11 and Figure 12:

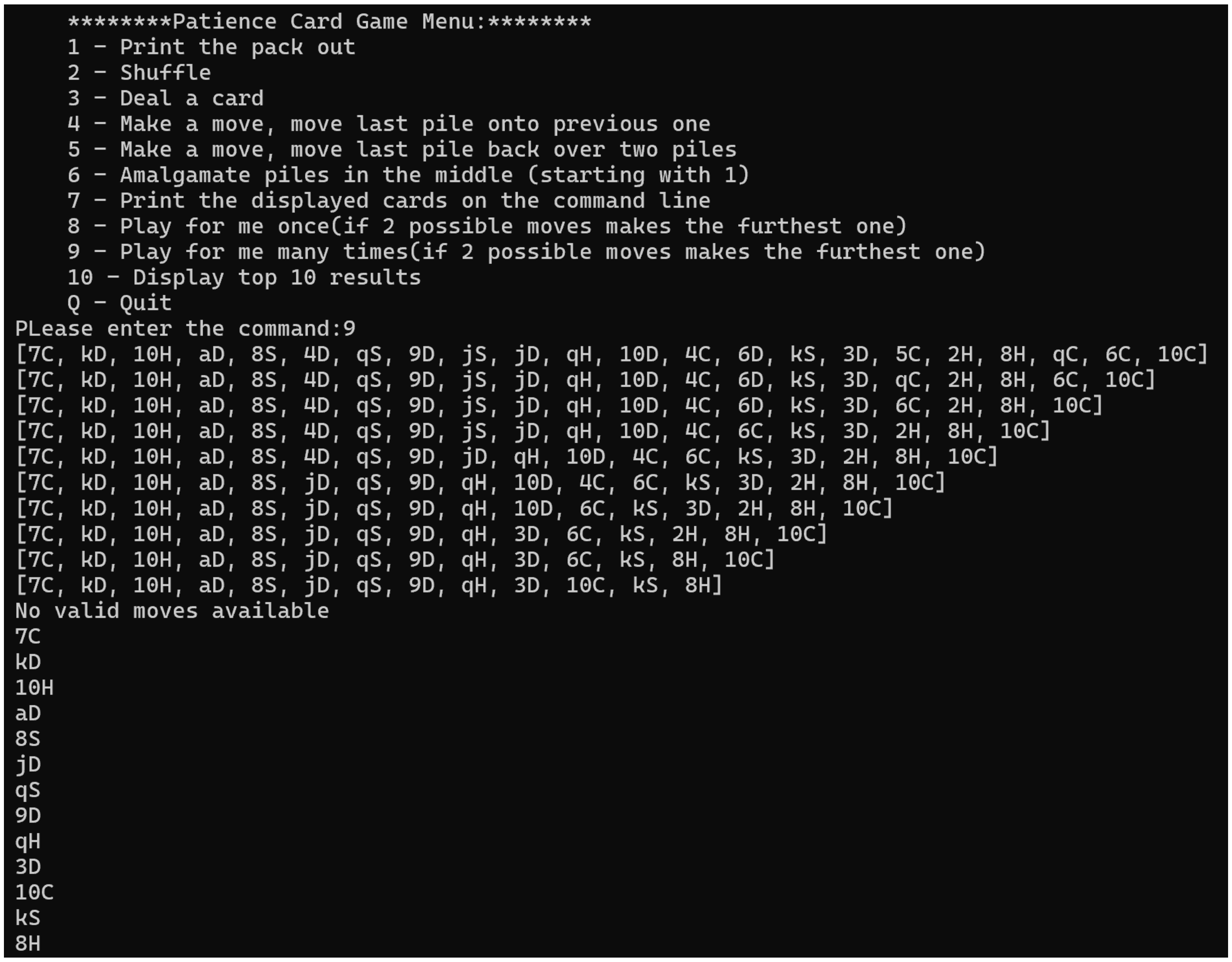


```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:3
Dealt card 4C
7C
kD
10H
aD
8S
4D
qS
9D
jS
jD
qH
10D
kC
6D
kS
4C
```



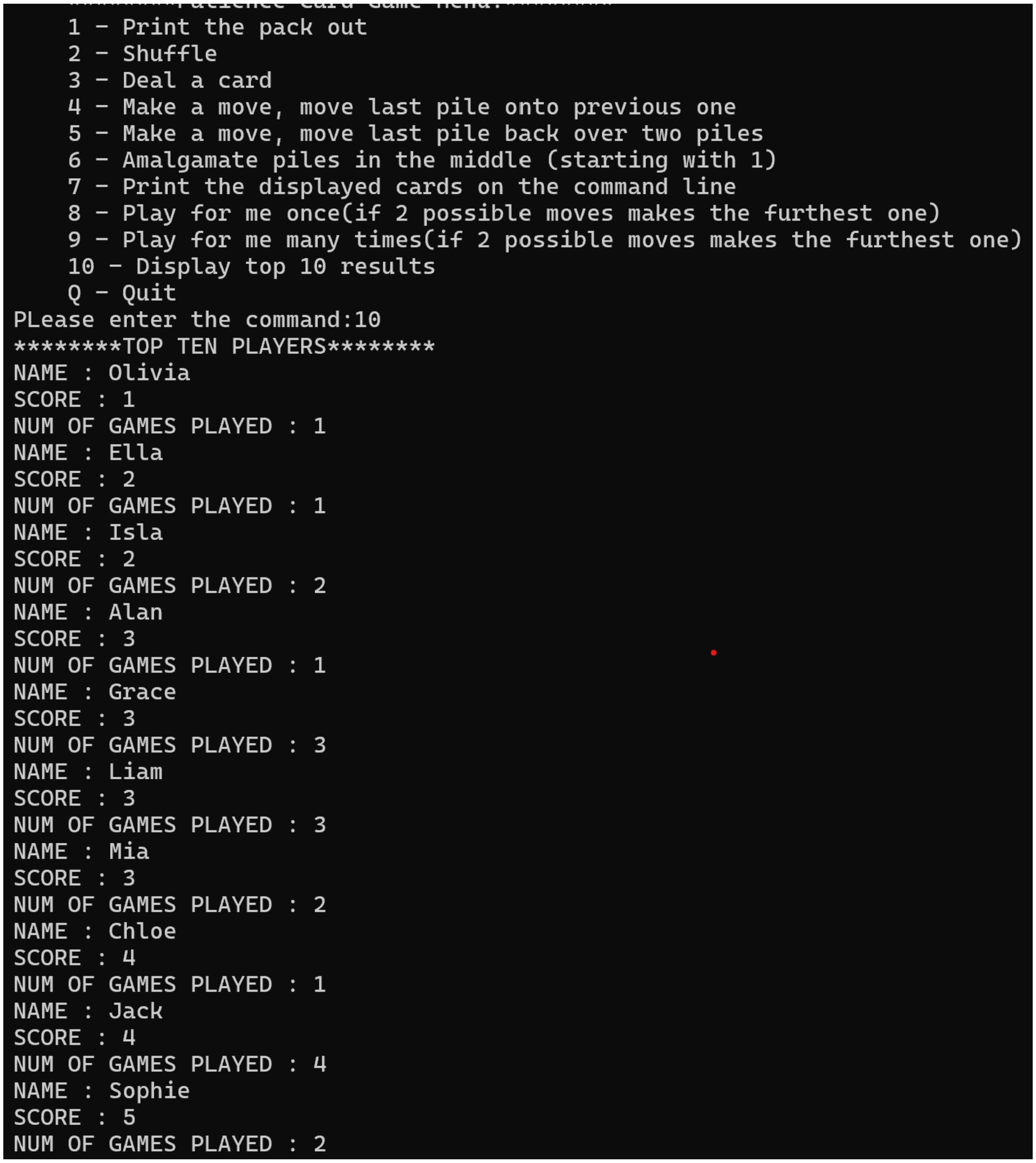
```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:8
[7C, kD, 10H, aD, 8S, 4D, qS, 9D, jS, jD, qH, 10D, 4C, 6D, kS]
```

Figure 13:



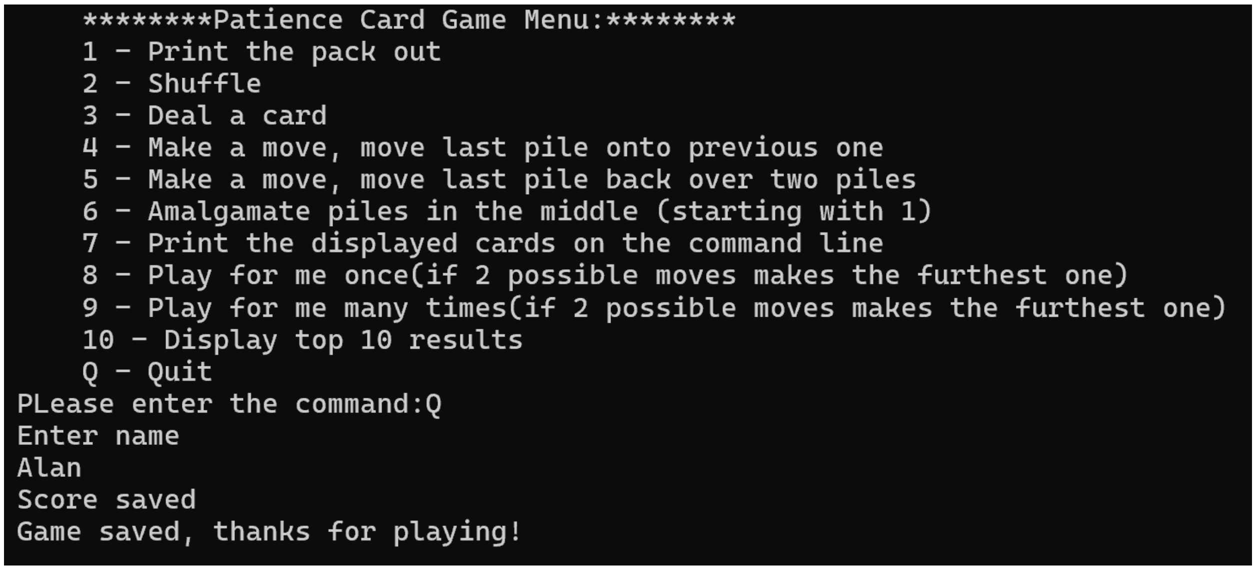
```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:9
[7C, kD, 10H, aD, 8S, 4D, qS, 9D, jS, jD, qH, 10D, 4C, 6D, kS, 3D, 5C, 2H, 8H, qC, 6C, 10C]
[7C, kD, 10H, aD, 8S, 4D, qS, 9D, jS, jD, qH, 10D, 4C, 6D, kS, 3D, qC, 2H, 8H, 6C, 10C]
[7C, kD, 10H, aD, 8S, 4D, qS, 9D, jS, jD, qH, 10D, 4C, 6D, kS, 3D, 6C, 2H, 8H, 10C]
[7C, kD, 10H, aD, 8S, 4D, qS, 9D, jS, jD, qH, 10D, 4C, 6C, kS, 3D, 2H, 8H, 10C]
[7C, kD, 10H, aD, 8S, 4D, qS, 9D, jD, qH, 10D, 4C, 6C, kS, 3D, 2H, 8H, 10C]
[7C, kD, 10H, aD, 8S, jD, qS, 9D, qH, 10D, 4C, 6C, kS, 3D, 2H, 8H, 10C]
[7C, kD, 10H, aD, 8S, jD, qS, 9D, qH, 10D, 6C, kS, 3D, 2H, 8H, 10C]
[7C, kD, 10H, aD, 8S, jD, qS, 9D, qH, 3D, 6C, kS, 2H, 8H, 10C]
[7C, kD, 10H, aD, 8S, jD, qS, 9D, qH, 3D, 6C, kS, 8H, 10C]
[7C, kD, 10H, aD, 8S, jD, qS, 9D, qH, 3D, 10C, kS, 8H]
No valid moves available
7C
kD
10H
aD
8S
jD
qS
9D
qH
3D
10C
kS
8H
```

Figure 14:



```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:10
********TOP TEN PLAYERS********
NAME : Olivia
SCORE : 1
NUM OF GAMES PLAYED : 1
NAME : Ella
SCORE : 2
NUM OF GAMES PLAYED : 1
NAME : Isla
SCORE : 2
NUM OF GAMES PLAYED : 2
NAME : Alan
SCORE : 3
NUM OF GAMES PLAYED : 1
NAME : Grace
SCORE : 3
NUM OF GAMES PLAYED : 3
NAME : Liam
SCORE : 3
NUM OF GAMES PLAYED : 3
NAME : Mia
SCORE : 3
NUM OF GAMES PLAYED : 2
NAME : Chloe
SCORE : 4
NUM OF GAMES PLAYED : 1
NAME : Jack
SCORE : 4
NUM OF GAMES PLAYED : 4
NAME : Sophie
SCORE : 5
NUM OF GAMES PLAYED : 2
```

Figure 15:



```
    ********Patience Card Game Menu:********
    1 - Print the pack out
    2 - Shuffle
    3 - Deal a card
    4 - Make a move, move last pile onto previous one
    5 - Make a move, move last pile back over two piles
    6 - Amalgamate piles in the middle (starting with 1)
    7 - Print the displayed cards on the command line
    8 - Play for me once(if 2 possible moves makes the furthest one)
    9 - Play for me many times(if 2 possible moves makes the furthest one)
    10 - Display top 10 results
    Q - Quit
PLease enter the command:Q
Enter name
Alan
Score saved
Game saved, thanks for playing!
```

Evaluation:

The main challenge for me was saving files to a score text file and reading from that file. After researching bufferReader and bufferWriter, I found this was the easiest method of implementing ways to read the file, allowing me to sort. I also struggled with the amalgamating piles in the middle. I struggled with implementing the OOP principles in amalgamating and decided that it was better to just add it to my main class. I still have the GUI (NFR3) to do as I haven't made a method of displaying cards. The creative part of the project was also a part where I struggled a little bit on as I realized that it was going to be difficult to add things to the game which it didn't already have. However, one implementation is the games played variable that comes up every time the same player with the same name plays. It is simple, however, let's you see what points were gained from practice and which ones were not. My player score class also implements the bestScore attribute specific to a player and not just overall compared to all the players. This means that the players' best scores are counted if the same player plays again. The playonce and playmany methods have also been coded in a way which checks all possible moves so it finds the move that is the most optimised(furthest card) rather than the first move that can be done. This just means that the cards will be played better for a potentially better overall score. I also wanted to add an undo method however after trial and error I couldn't get the code to work so I had to abandon the idea. I believe that based on the on the code that I have provided, and the information laid out through this document I deserve a 85 - 90 % mark for the project as I have completed every requirement that was laid out in the assignment brief. I also used inheritance throughout the player and player info class with well-structured code throughout my project. I also have tested all the functional and non-functional requirements that I have incorporated; along with this I believe that my use of OOP has optimised the code making it worthy of the marks I have stated above.