PATIENCE PROJECT

YASH SINGH

CS12320

yas24@aber.ac.uk

Word Count = 2,060

Contents

1 Introduction

The purpose of the task dealt is to program and code a text-based card game known as “Patience”, where the player keeps looking for card matches, making the base game have no certain time limit.

1.1 GAME OVERVIEW

The game follows a similar combination mechanism to games such as 2048, where each card dealt from a shuffled deck has a n option to match if possible until piles are merged to 1.

1.2 Implementation OVERVIEW

From the project I was tasked with functional requirements needed for the game, FR1-FR11, which took priority and additional Non-functional requirements to help the game look more secure NFR1-3. I managed to apply all the requirements and nonfunctional, with the GUI I only changed card appearances.

2 UML USE CASE DIAGRAM

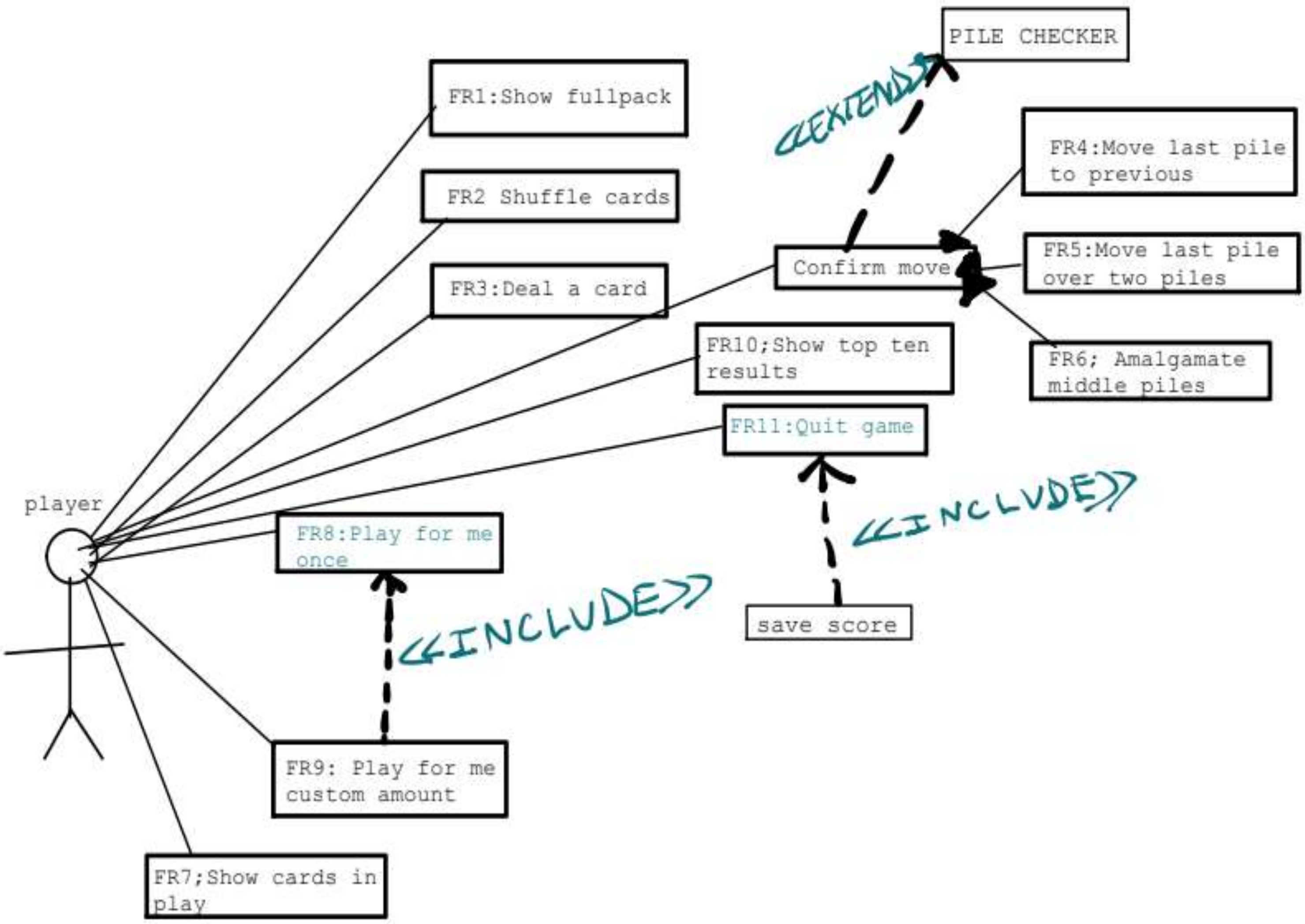


Diagram Breakdown

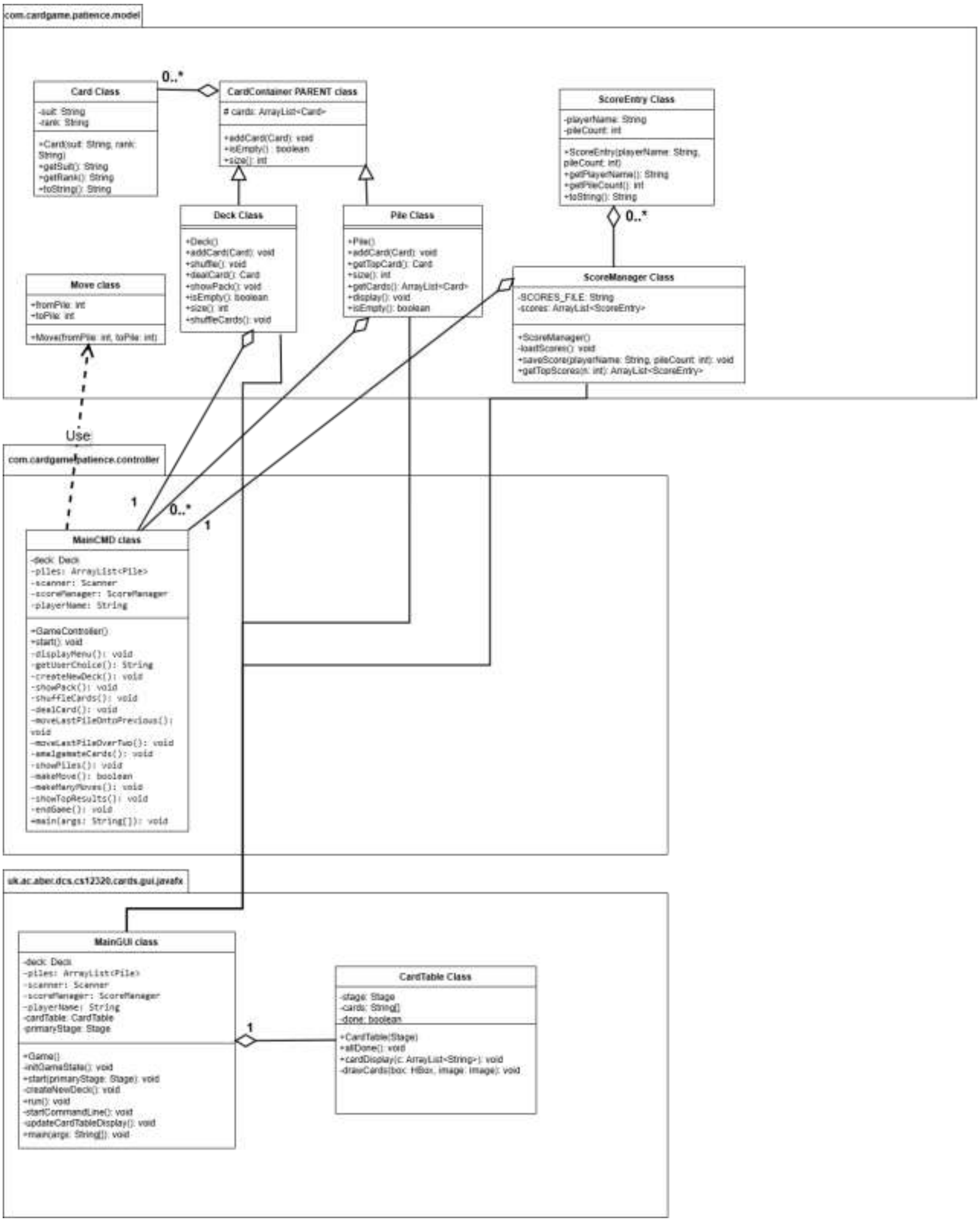
The UML diagram shows the player actions available towards the functional requirements in the project. The key relationships are:

-FR9 including FR8 which shows function reusability-Move operations being generalized by relating manual moves FR4, FR5, FR6

-Save score extends extends after quitting FR11-Pile checker extends confirming move to see if game is won (1 pile left)

3 DESIGN

3.1 CLASS DIAGRAM



1

3.2 CLASS DIAGRAM DESCRIPTION

Card

This is used for representing each single playing card with the suit and rank assigned. It stores the values.

Attributes:

- Suit and Rank – String to display card values'

Methods:

- Card – Constructor method to initialize rank and suit
- GetRank and getSuit – return type

Deck extends CardContainer

Child class which takes care of shuffling and dealing the cards in the game.

Methods:

- Deck – constructor for the class
- shuffle – this method serves as functional requirement as it randomizes the card with the help of the collections.shuffle() function in java, implements FR2
- dealCard – this deals the card always first in the list then used '.remove' to erase card from deck
- showPack – Shows all the cards in the deck, implements FR1

Pile extends CardContainer

Another child class which only shows the top card for each card pile in play.

Methods:

- Pile – creates a new empty pile to hold the cards
- getTopCard - returns the top card of each pile
- Display – This is important for the functional requirement 7, returning topcard

[1] Diagram made using https://app.diagrams.net/

CardContainer

Parent class which implements the ‘is-a’ relationship for Pile and Deck, which shows inheritance by getting common attributes.

Methods:

- addCard(Card) – Builds the deck single card by card
- isEmpty and Size – check deck is empty and its size as condition

Move

Proving important in later stages this was needed for functional requirements 8 and 9 which are the automatic moves.

Attributes:

- fromPile and toPile – used for storing the indices of the piles which are used in the start and end of move.

ScoreEntry

This represents the class for managing a single players score which formats it as their ‘name , piles’.

Methods:

- PlayerName and Playercount - name and piles remaining
- ToString – formatting of how it would look

ScoreManager

This is a vital class for input and output of a text file which manages the scores layout and its order. Used this class to utilise try and catch exceptions.

Methods:

- ScoreManager – constructor for loading previous scores from the file
- loadScores - Implements NFR2 and loads existing scores
- saveScore(playerName, pileCount) – Saves a new score to the file, which also implements NFR2
- getTopScores – retrieves to top n number of scores and orders in terms of lower pile counts being favoured

MainCMD

Main file to run the command line interface and all game logic

Methods:

- MainCMD – constructor which starts the game state and asks for the player name before the game begins
- Start – initializes the main game loop and shows the menu of all the options i.e the FR1-10
- displayMenu and getUserChoice – implements NFR1
- createNewDeck – A helping method to reduce duplication of deck creation logic
- showPack, shuffleCards, dealCard – implements FR1,2,3
- moveLastPileOntoPrevious, moveLastPileOverTwo, amalgamateCards – implements FR4,5,6
- showPiles – implements FR7
- makeMove, makeManyMoves – implements FR8,9
- showTopResults – implements FR10
- endgame – implements FR11
-

MainGUI (GUI class)

Uses javaFX which extends applications to run a graphical user interface. NFR3

Methods:

- initGame() – A method for initiating game interface.
- run – thread for running the game independent of the Command line interface
- updateCardTableDisplay – This is used to update the graphical display, useful for after every move or operation occurring in the game
- startCommandLine – handles command line interface and user inputs

CardTable

Class which displays the images on the JavaFX stage, provided by the supplementary section of the assignment brief as a template.

Relationships and Links:

- ScoreManager contains ScoreEntry
- Deck and Pile extend CardContainer
- Game class requires CardTable for visual representation

3.1 PSEUDO CODE

Play for me once algorithm was the most complex algorithm used in this project as it was checking multiple types of moves and deciding the best move. The below shows how it prioritizes it.

```
// Find possible moves (prioritizing jumps over adjacent)

possibleMoves = []


// Check for jump moves (2 piles between)

FOR i = 0 TO piles.size - 4

    IF piles[i].topCard matches piles[i+3].topCard BY suit or rank
THEN

        ADD Move(i+3, i) to possibleMoves

    END IF

END FOR


// Check for adjacent moves

FOR i = 0 TO piles.size - 2

    IF piles[i].topCard matches piles[i+1].topCard BY suit or rank
THEN

        ADD Move(i+1, i) to possibleMoves
    END IF
END FOR

// Execute best move or return if none available
IF possibleMoves IS EMPTY THEN RETURN false

bestMove = possibleMoves[0]
PRINT "Auto-move: Moving pile " + (bestMove.fromPile + 1) + " to " +
(bestMove.toPile + 1)
MOVE all cards from piles[bestMove.fromPile] to
piles[bestMove.toPile]
REMOVE piles[bestMove.fromPile]
showPiles()
RETURN true
```

4 TESTING SECTION

The following section covers the testing table which includes screenshots showing the output

TR = TestRun

4.1 TEST TABLE

Test id	**Requirement**	**Description**	**input**	**Expected Output**	**Pass/Fail**
TR1	FR1	Show full card pack	ENTER 1	Figure 1	P
TR1.2	FR1	invalid menu handling	Type “create”	Figure 2	P
TR2.1	FR2	Shuffling empty card deck	ENTER 2	**Error! Reference source not found.**	P
TR 2	FR2	Check shuffled deck order	ENTER 1, ENTER 2, ENTER 1	Figure 4	P
TR 3.1	FR3	Deal card from deck to see it removed	ENTER 1, 3 THEN 1	Figure 5	P
TR 3	FR3	Deal from empty deck	Enter 3 straight away	Figure 6	P
TR 4.1	FR4	Mover over last pile	ENTER 4 once same suit or rank	Figure 7 Figure 8	P
TR 4	FR4	Invalid move attempt to cards that can’t be merged	ENTER 4	Figure 9	P

TR 4.3	FR4	move on insufficient piles	ENTER 4 on single pile	Figure 10	P
TR 5	FR5	Move last pile over two	ENTER 5	Figure 11	P
TR 5.2	FR5	Move last pile over two (invalid)	ENTER 5	Figure 12	P
TR 5.3	FR5	Insufficient piles for jump move	ENTER 5	Figure 13	P
TR 6	FR6	Amalgamate piles in middle	ENTER 6	Figure 14 Figure 15	P
TR 6.2	FR6	Amalgamate piles in middle (invalid)	ENTER 6	Figure 16	P

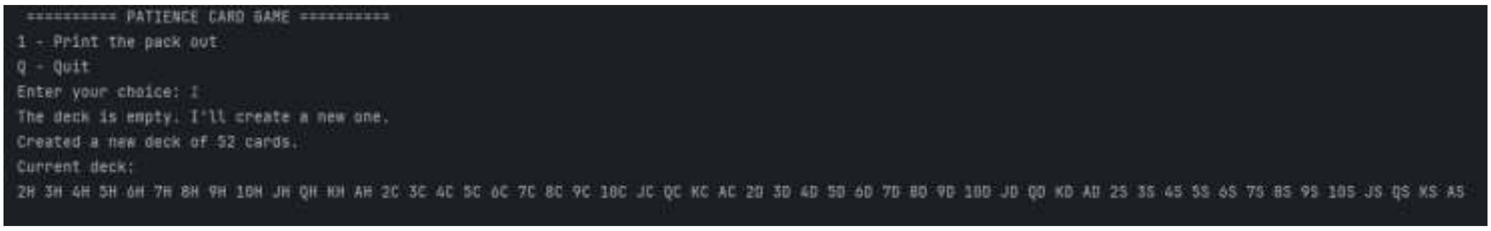
TR 6.3	FR6	Amalgamate piles in middle (invalid - wrong distance)	ENTER 6	Figure 17	P
TR 6.4	FR6	Amalgamate piles in middle invalid input	ENTER 6	Figure 18	F
TR 6.5	FR6	Amalgamate piles using less piles than needed	ENTER 6	Figure 19	P
TR 7	FR7	Show all cards in play	ENTER 7	Figure 20	P
TR 7.2	FR7	Show all cards in play when no cards dealt	ENTER 7	Figure 21	P
TR 8	FR8	Play for me once (jump move available)	ENTER 8	Figure 22	P

TR 8.2	FR8	Play for me once (only adjacent move available)	ENTER 8	Figure 23	P
TR 9	FR9	Play for me many times	ENTER 9	Figure 24	P
TR 9.2	FR9	Play for me many times (limited by available moves)	ENTER 9	Figure 25	P
TR 9.3	FR9	Play for me many times (no valid moves)	ENTER 9	Figure 26	P
TR 10	FR10	Show top results (with existing scores)	ENTER 9	Figure 27	P
TR 10.2	FR10	Show top results (no existing scores)	ENTER	Figure 28Figure 29	P

TR 11	FR11	Quit with score saving	ENTER	Figure 29	P
TR 10.3	FR10	Quit without playing	ENTER	Figure 30	P
TR 12	NFR1	Complete command-line menu	ENTER	Figure 31	P
TR 12.2	NFR1	Invalid menu option – letter or number	ENTER	Figure 32 Figure 33	P
TR 13	NFR2	Score file format	ENTER	Figure 34	P
TR 13.2	NFR2	Score persistence between sessions	ENTER	Figure 35	P
TR 14	NFR3	Graphical interface rendering	ENTER	Figure 36	P
TR 14.2	NFR3	Graphical interface updates	ENTER	Figure 37	P

TR 14.3	NFR3	Graphical interface scrolling	ENTER	Figure 38	P
TR 14.4	NFR3	GUI responsiveness with large pile count	ENTER	Figure 39 Figure 40 Figure 41	P

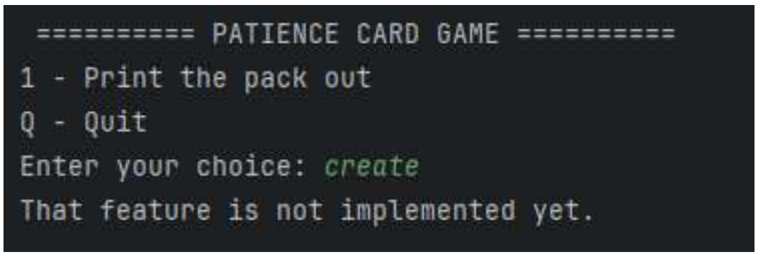
4.2 TESTING SCREENSHOTS



```
 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
Q - Quit
Enter your choice: 1
The deck is empty. I'll create a new one.
Created a new deck of 52 cards.
Current deck:
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 2S 3S 4S 5S 6S 7S 8S 9S 10S JS QS KS AS
```

Figure 1

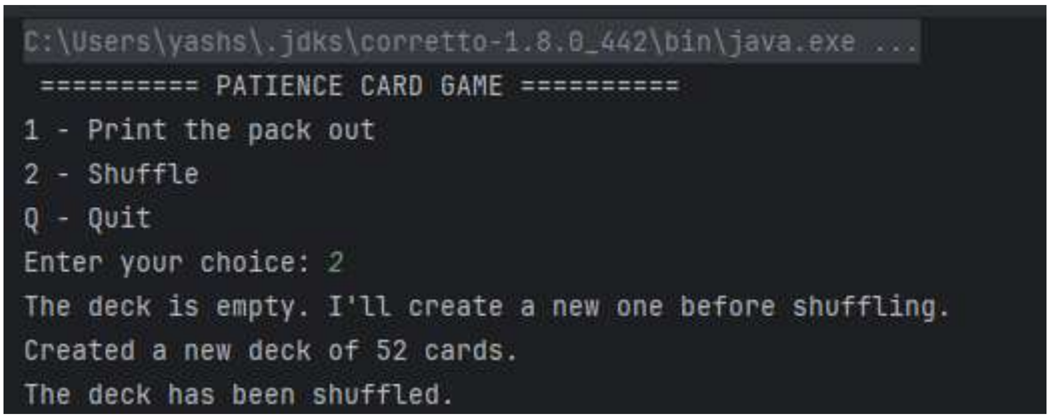
The deck is initially empty, so program checks and creates a new ordered one of 52 cards every time its empty.



```
 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
Q - Quit
Enter your choice: create
That feature is not implemented yet.
```

Figure 2

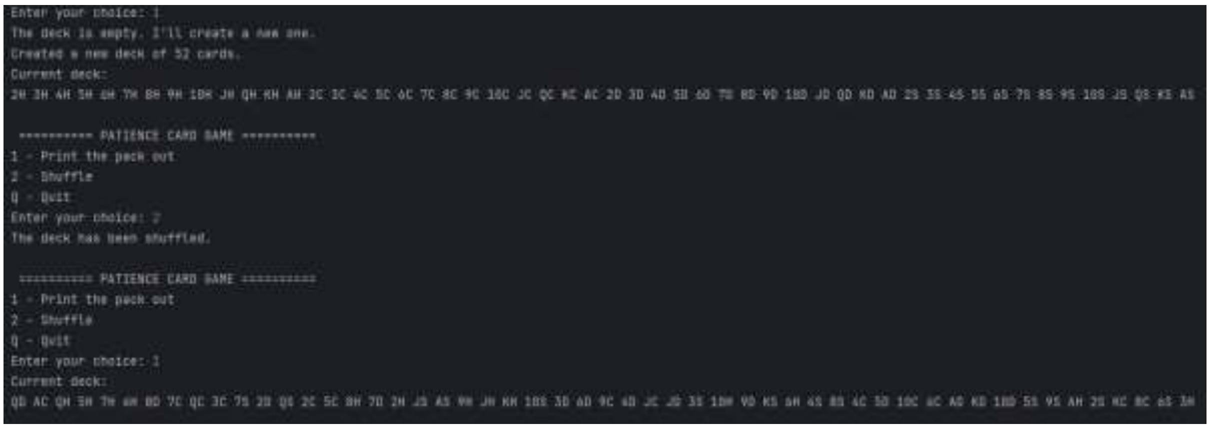
User input other than menu options gets prompted with edge case message



```
C:\Users\yashs\.jdks\corretto-1.8.0_442\bin\java.exe ...
 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
Q - Quit
Enter your choice: 2
The deck is empty. I'll create a new one before shuffling.
Created a new deck of 52 cards.
The deck has been shuffled.
```

Figure 3

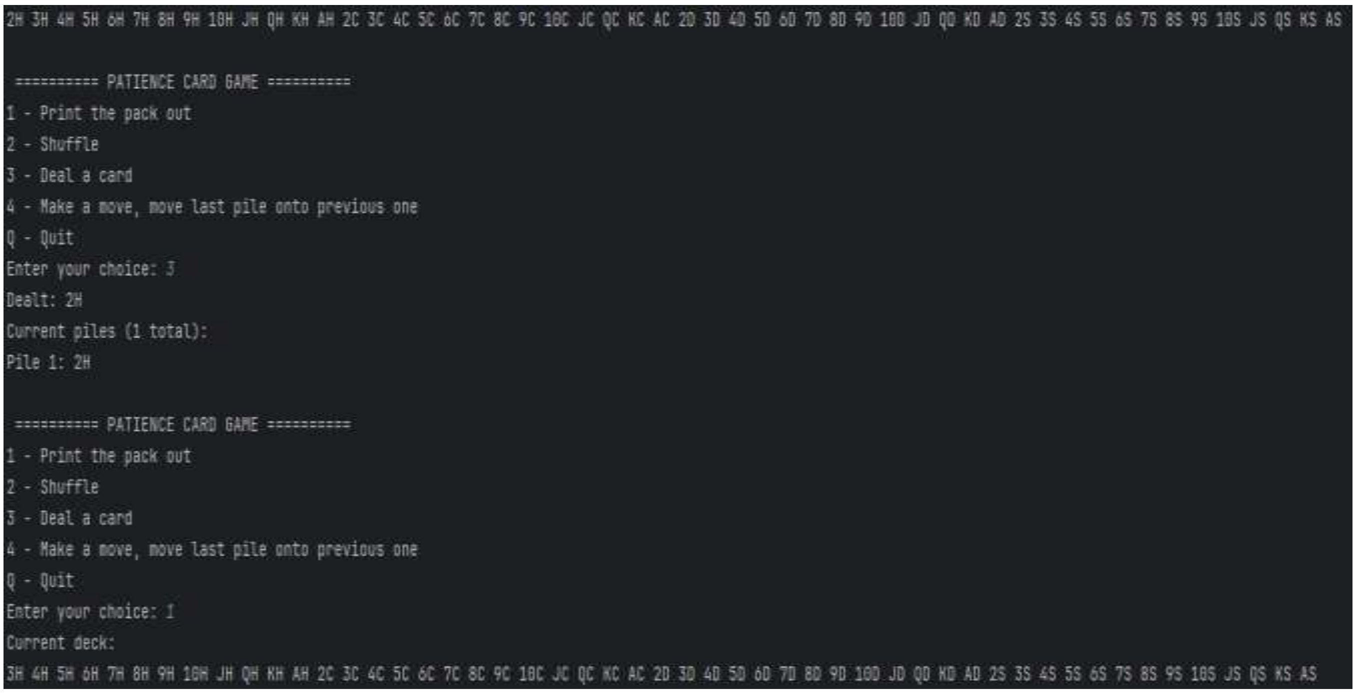
Testing response to shuffling an empty deck



```
Enter your choice: 1
The deck is empty. I'll create a new one.
Created a new deck of 52 cards.
Current deck:
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 2S 3S 4S 5S 6S 7S 8S 9S 10S JS QS KS AS

 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
Q - Quit
Enter your choice: 2
The deck has been shuffled.

 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
Q - Quit
Enter your choice: 1
Current deck:
QD AC QH 5H 7H 4H 8D 7C QC 3C 7S 2D QS 2C 5C 8H 7D 2H JS AS 9H JH KH 10S 3D 6D 9C 4D JC JD 3S 10H 9D KS 6H 4S 8S 4C 5D 10C 6C AD KD 10D 5S 9S AH 2S KC 8C 6S 3H
```

Figure 4

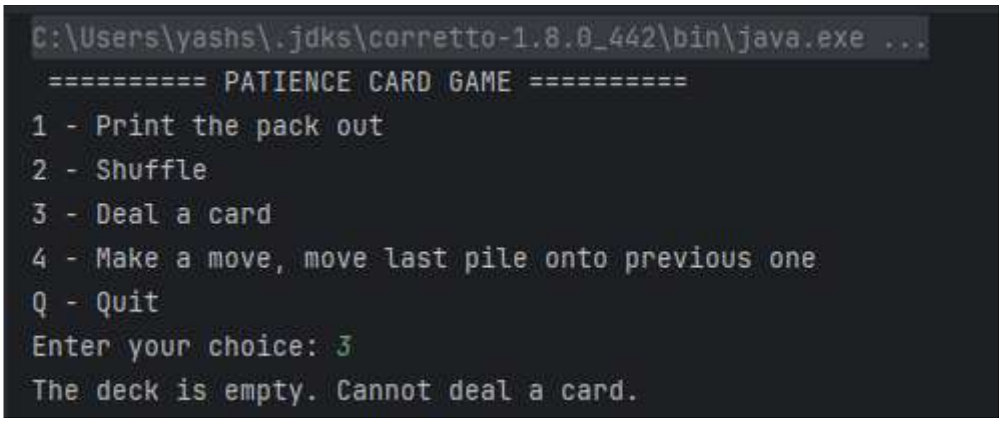
Print out normal deck then user prompts 2 to shuffle deck and then 1 again to reprint shuffled deck.



```
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 2S 3S 4S 5S 6S 7S 8S 9S 10S JS QS KS AS

 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile onto previous one
Q - Quit
Enter your choice: 3
Dealt: 2H
Current piles (1 total):
Pile 1: 2H

 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile onto previous one
Q - Quit
Enter your choice: 1
Current deck:
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 2S 3S 4S 5S 6S 7S 8S 9S 10S JS QS KS AS
```

Figure 5

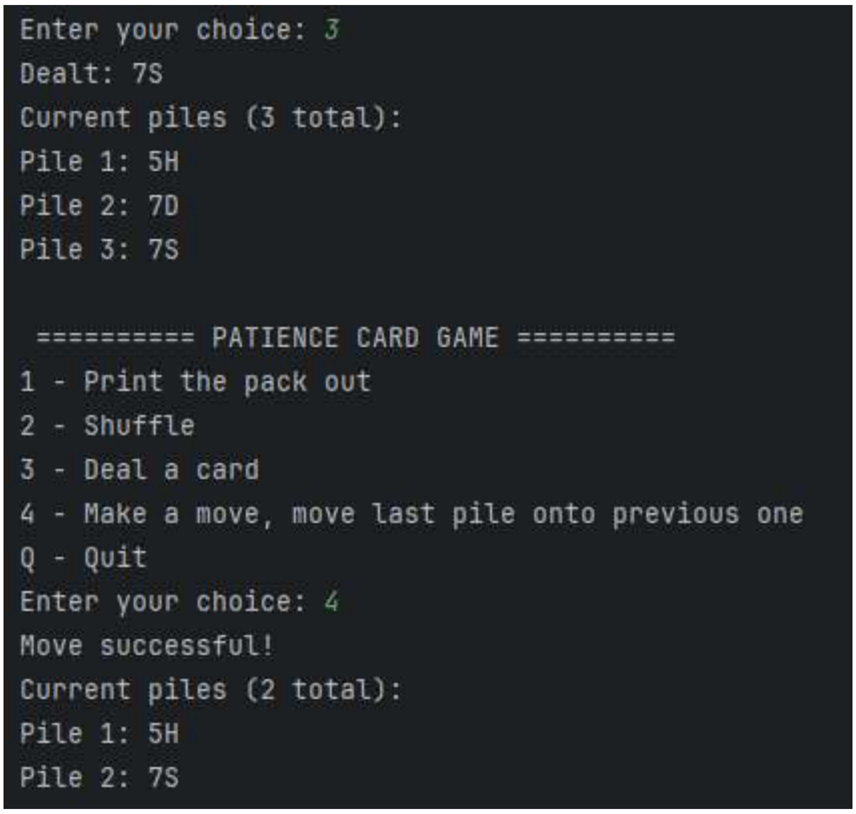
card removed from pack AFTER being dealt into play



```
C:\Users\yashs\.jdks\corretto-1.8.0_442\bin\java.exe ...
 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile onto previous one
Q - Quit
Enter your choice: 3
The deck is empty. Cannot deal a card.
```

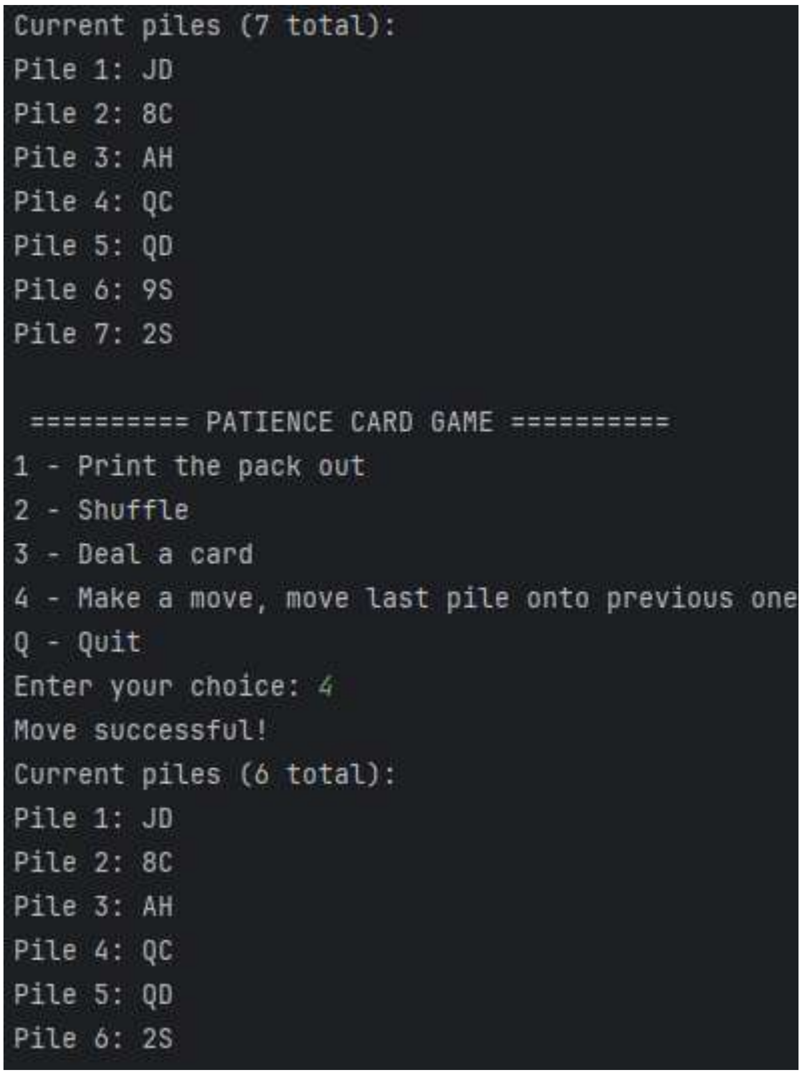
Figure 6

Testing response of being able to deal an empty pack



```
Enter your choice: 3
Dealt: 7S
Current piles (3 total):
Pile 1: 5H
Pile 2: 7D
Pile 3: 7S

 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile onto previous one
Q - Quit
Enter your choice: 4
Move successful!
Current piles (2 total):
Pile 1: 5H
Pile 2: 7S
```

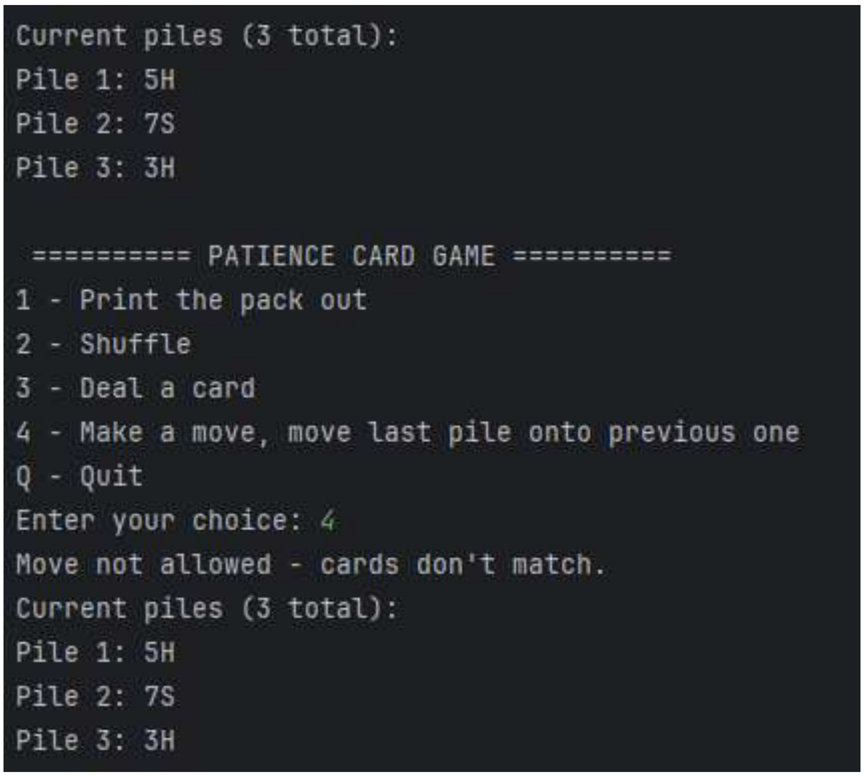
Figure 7
Same rank move



```
Current piles (7 total):
Pile 1: JD
Pile 2: 8C
Pile 3: AH
Pile 4: QC
Pile 5: QD
Pile 6: 9S
Pile 7: 2S

 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile onto previous one
Q - Quit
Enter your choice: 4
Move successful!
Current piles (6 total):
Pile 1: JD
Pile 2: 8C
Pile 3: AH
Pile 4: QC
Pile 5: QD
Pile 6: 2S
```

Figure 8

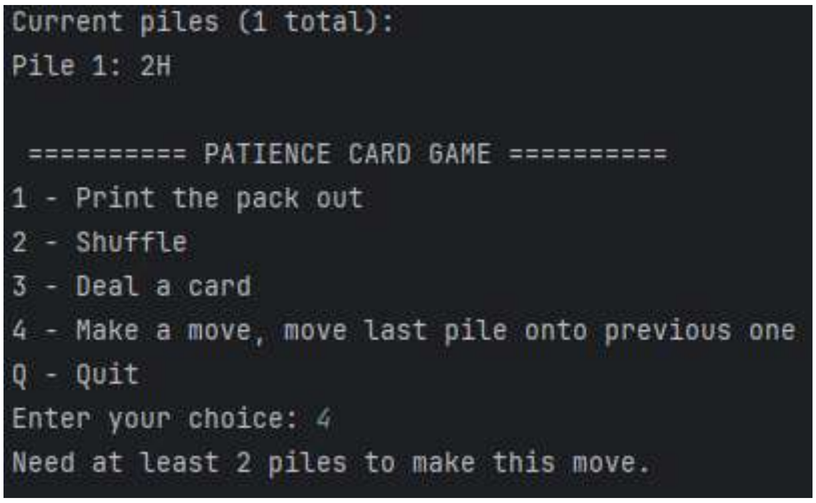
Same suit move



```
Current piles (3 total):
Pile 1: 5H
Pile 2: 7S
Pile 3: 3H

 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile onto previous one
Q - Quit
Enter your choice: 4
Move not allowed - cards don't match.
Current piles (3 total):
Pile 1: 5H
Pile 2: 7S
Pile 3: 3H
```

Figure 9

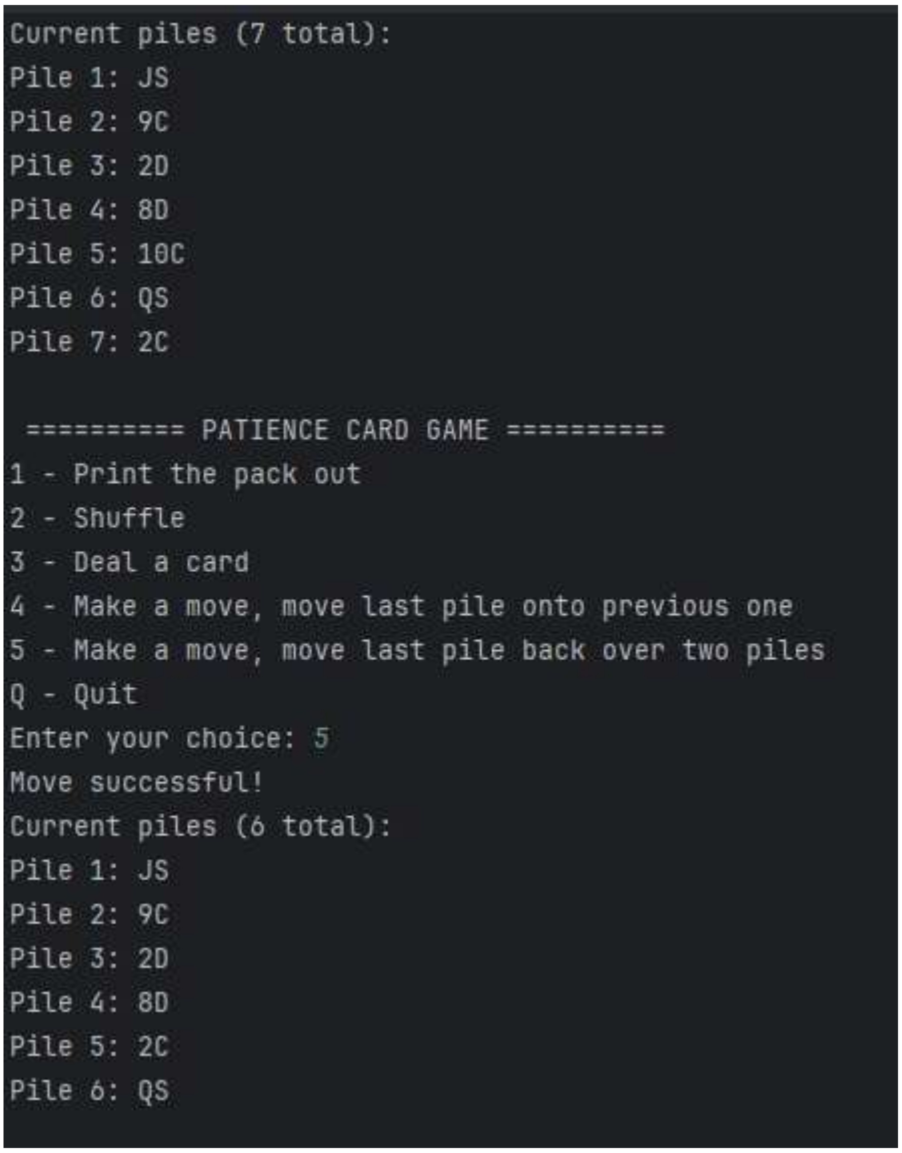
Invalid card check



```
Current piles (1 total):
Pile 1: 2H

 ========== PATIENCE CARD GAME ==========
1 - Print the pack out
2 - Shuffle
3 - Deal a card
4 - Make a move, move last pile onto previous one
Q - Quit
Enter your choice: 4
Need at least 2 piles to make this move.
```

Figure 10

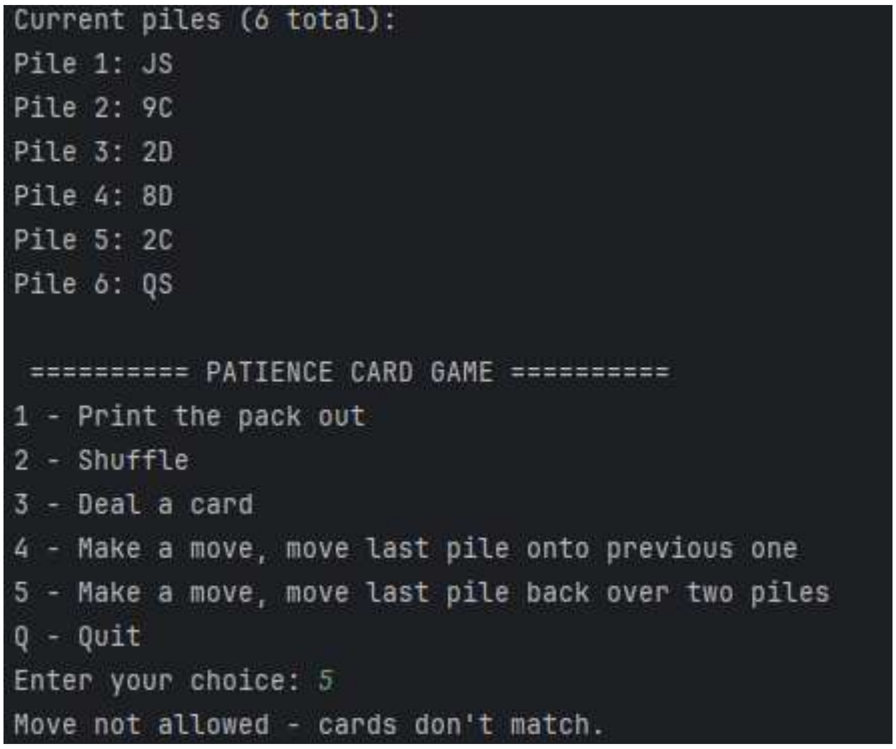
Invalid pile amount check



```
Current piles (7 total):
Pile 1: JS
Pile 2: 9C
Pile 3: 2D
Pile 4: 8D
Pile 5: 10C
Pile 6: QS
Pile 7: 2C

 ========== PATIENCE CARD GAME ==========
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
Q - Quit
Enter your choice: 5
Move successful!
Current piles (6 total):
Pile 1: JS
Pile 2: 9C
Pile 3: 2D
Pile 4: 8D
Pile 5: 2C
Pile 6: QS
```

Figure 11

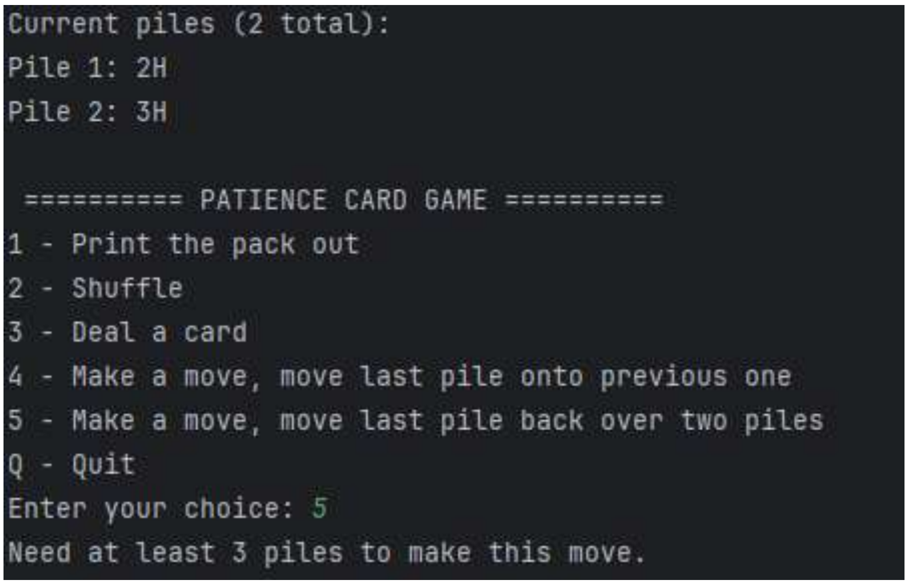
Checking functionality of moving back over 2 piles



```
Current piles (6 total):
Pile 1: JS
Pile 2: 9C
Pile 3: 2D
Pile 4: 8D
Pile 5: 2C
Pile 6: QS

 ========== PATIENCE CARD GAME ==========
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
Q - Quit
Enter your choice: 5
Move not allowed - cards don't match.
```

Figure 12

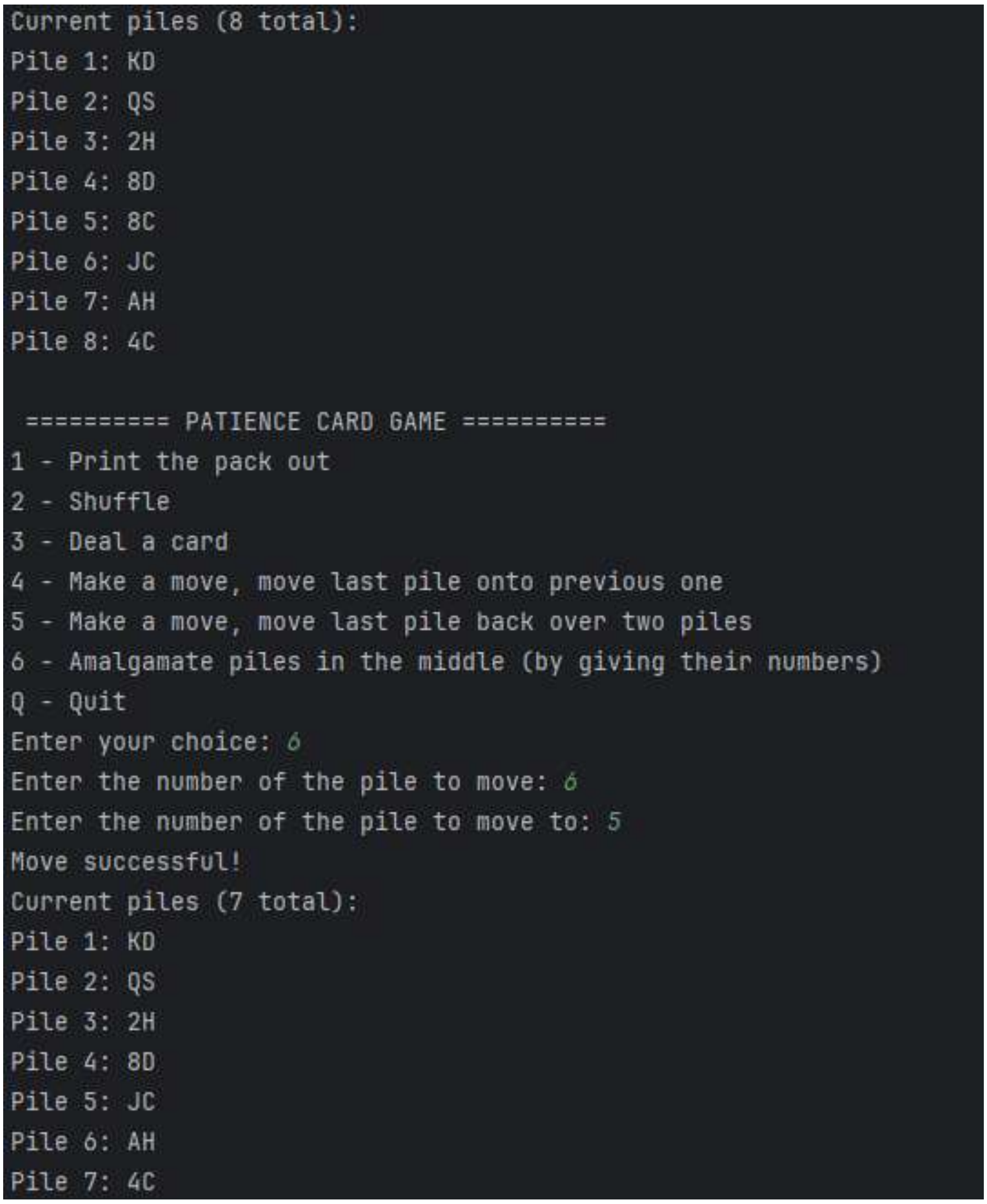
Invalid Card suit/rank checker



```
Current piles (2 total):
Pile 1: 2H
Pile 2: 3H

 ========== PATIENCE CARD GAME ==========
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
Q - Quit
Enter your choice: 5
Need at least 3 piles to make this move.
```

Figure 13

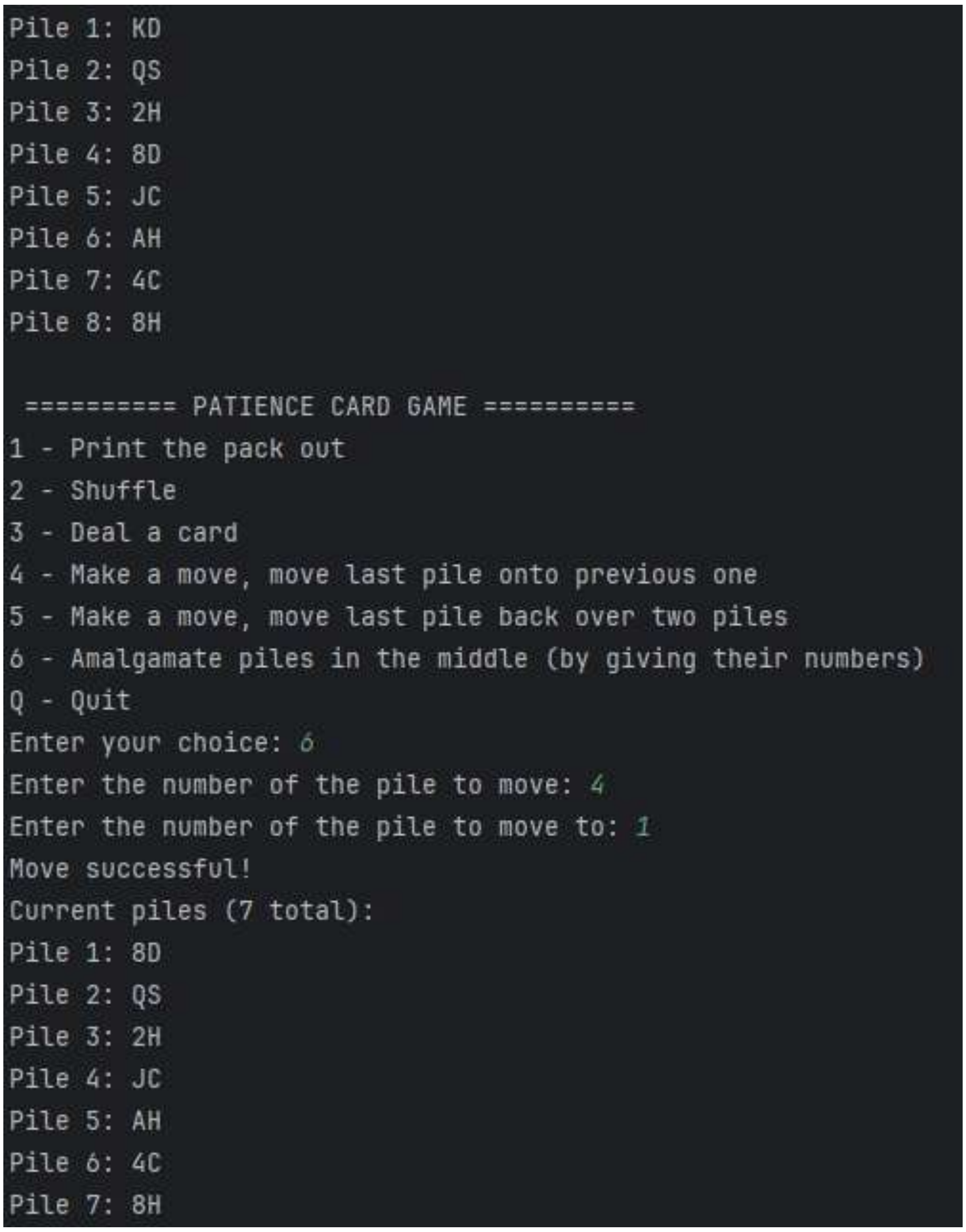
Invalid amount of piles test check



```
Current piles (8 total):
Pile 1: KD
Pile 2: QS
Pile 3: 2H
Pile 4: 8D
Pile 5: 8C
Pile 6: JC
Pile 7: AH
Pile 8: 4C

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
Q - Quit
Enter your choice: 6
Enter the number of the pile to move: 6
Enter the number of the pile to move to: 5
Move successful!
Current piles (7 total):
Pile 1: KD
Pile 2: QS
Pile 3: 2H
Pile 4: 8D
Pile 5: JC
Pile 6: AH
Pile 7: 4C
```

Figure 14

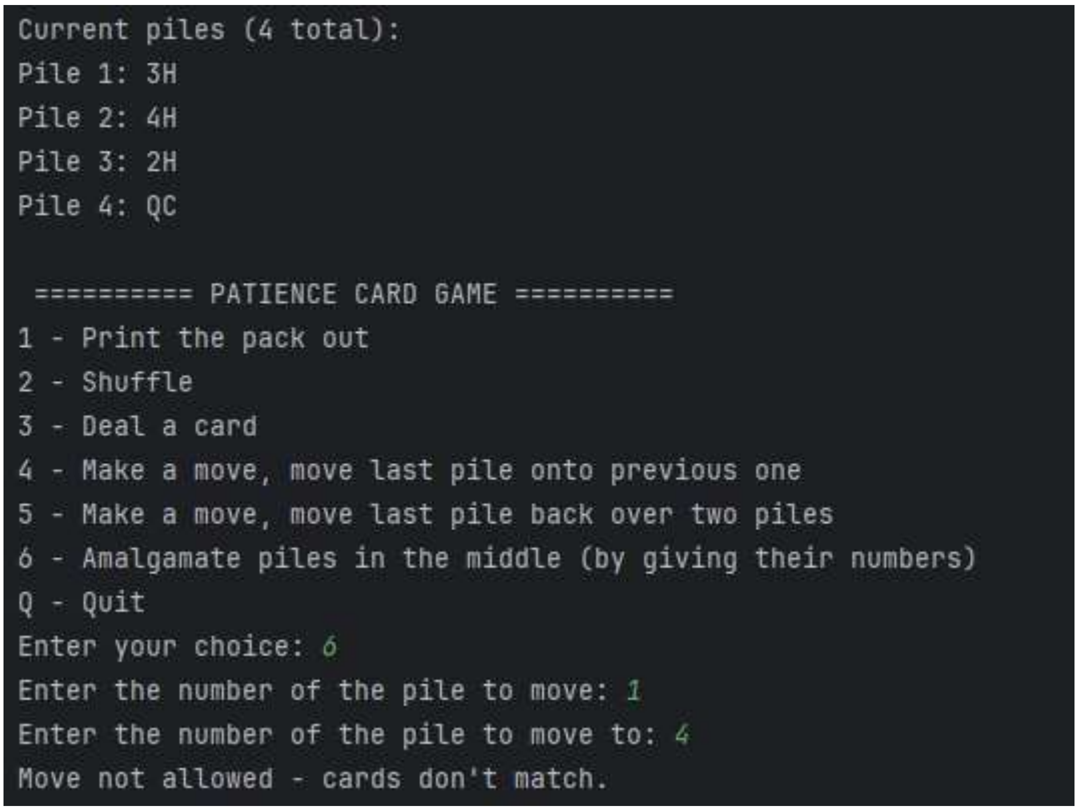
Amalgamate piles test check 1 GAP jump



```
Pile 1: KD
Pile 2: QS
Pile 3: 2H
Pile 4: 8D
Pile 5: JC
Pile 6: AH
Pile 7: 4C
Pile 8: 8H

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
Q - Quit
Enter your choice: 6
Enter the number of the pile to move: 4
Enter the number of the pile to move to: 1
Move successful!
Current piles (7 total):
Pile 1: 8D
Pile 2: QS
Pile 3: 2H
Pile 4: JC
Pile 5: AH
Pile 6: 4C
Pile 7: 8H
```

Figure 15

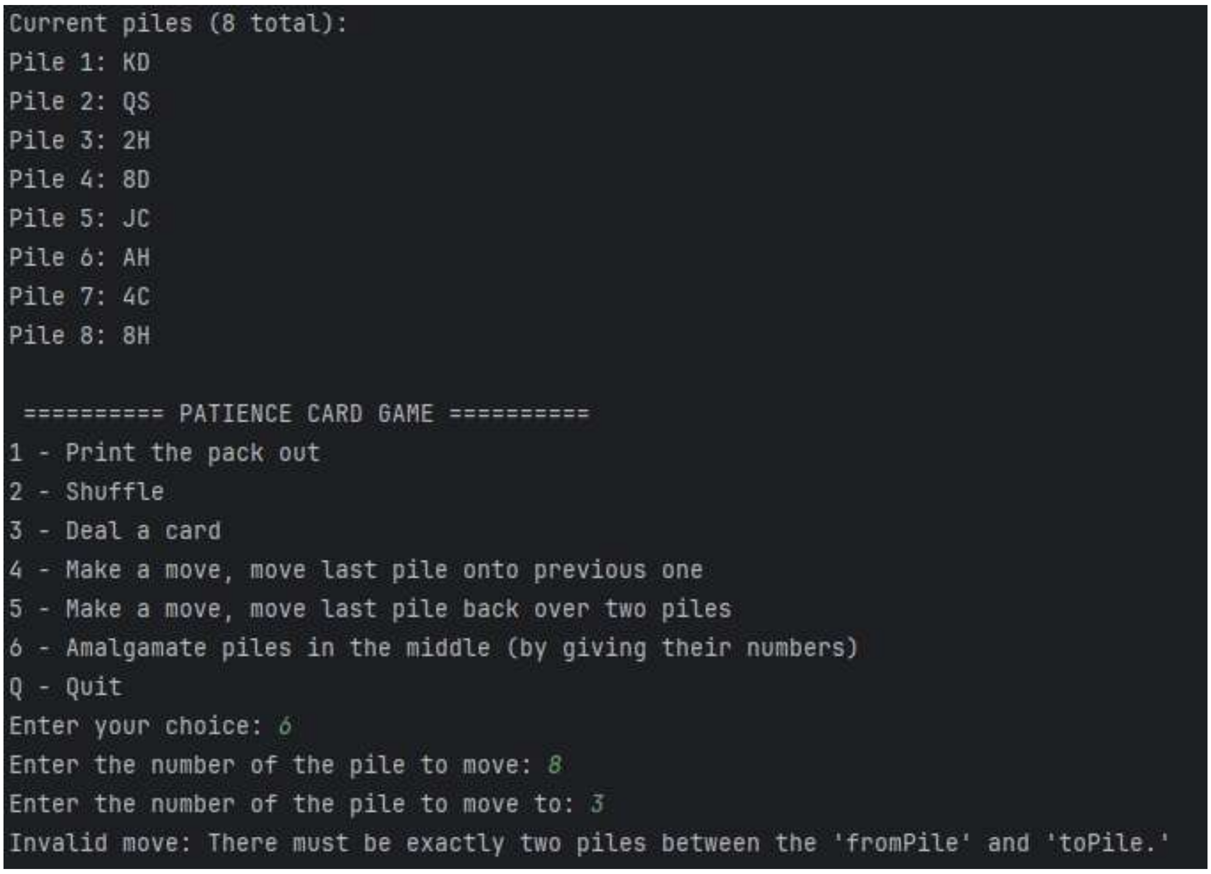
Amalgamate piles test check 2 GAP jump



```
Current piles (4 total):
Pile 1: 3H
Pile 2: 4H
Pile 3: 2H
Pile 4: QC

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
Q - Quit
Enter your choice: 6
Enter the number of the pile to move: 1
Enter the number of the pile to move to: 4
Move not allowed - cards don't match.
```

Figure 16

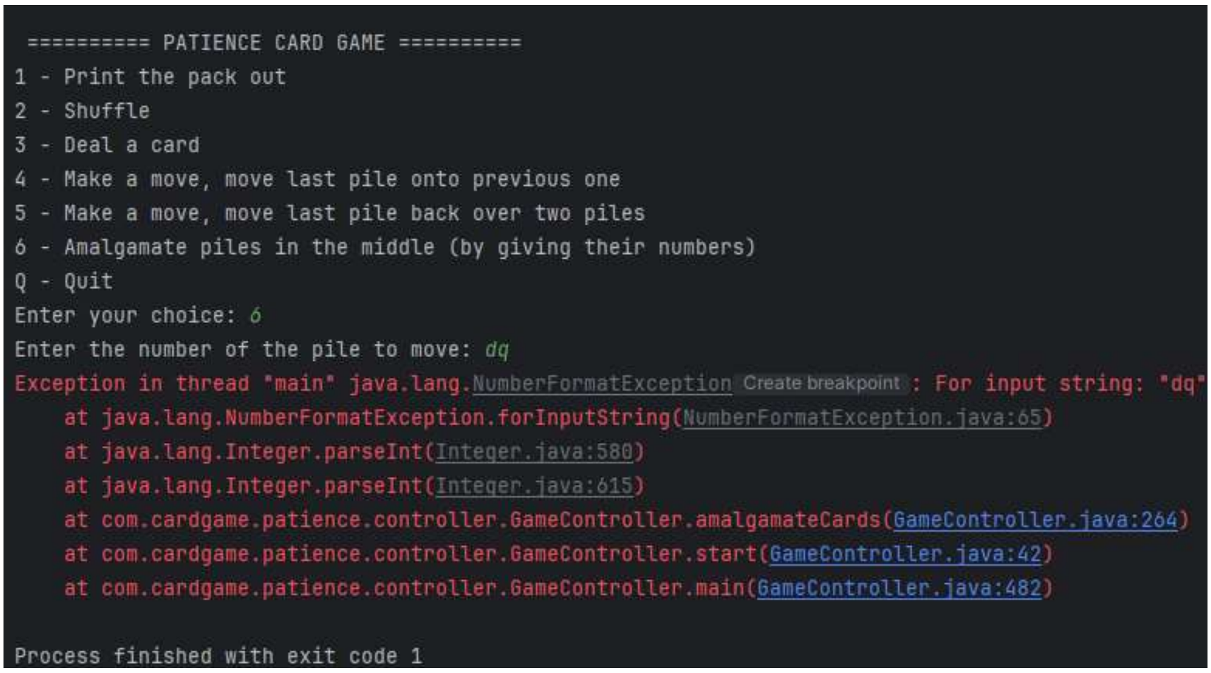
Amalgamate invalid suit/rank test response check



```
Current piles (8 total):
Pile 1: KD
Pile 2: QS
Pile 3: 2H
Pile 4: 8D
Pile 5: JC
Pile 6: AH
Pile 7: 4C
Pile 8: 8H

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
Q - Quit
Enter your choice: 6
Enter the number of the pile to move: 8
Enter the number of the pile to move to: 3
Invalid move: There must be exactly two piles between the 'fromPile' and 'toPile.'
```

Figure 17

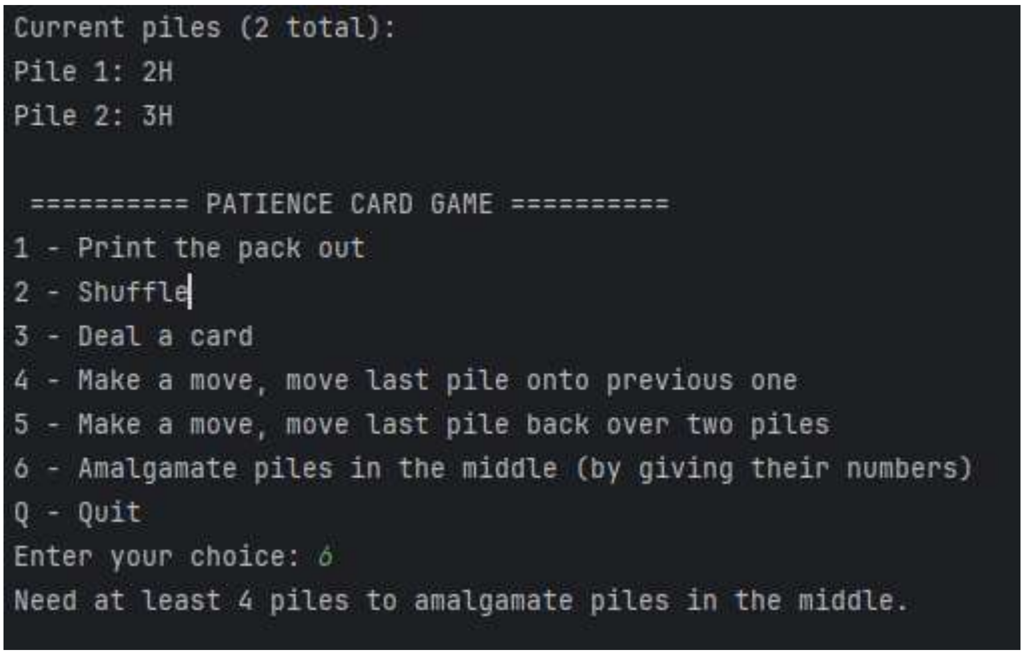
Amalgamate with too big of a gap



```
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
Q - Quit
Enter your choice: 6
Enter the number of the pile to move: dq
Exception in thread "main" java.lang.NumberFormatException Create breakpoint : For input string: "dq"
    at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.lang.Integer.parseInt(Integer.java:580)
    at java.lang.Integer.parseInt(Integer.java:615)
    at com.cardgame.patience.controller.GameController.amalgamateCards(GameController.java:264)
    at com.cardgame.patience.controller.GameController.start(GameController.java:42)
    at com.cardgame.patience.controller.GameController.main(GameController.java:482)

Process finished with exit code 1
```

Figure 18

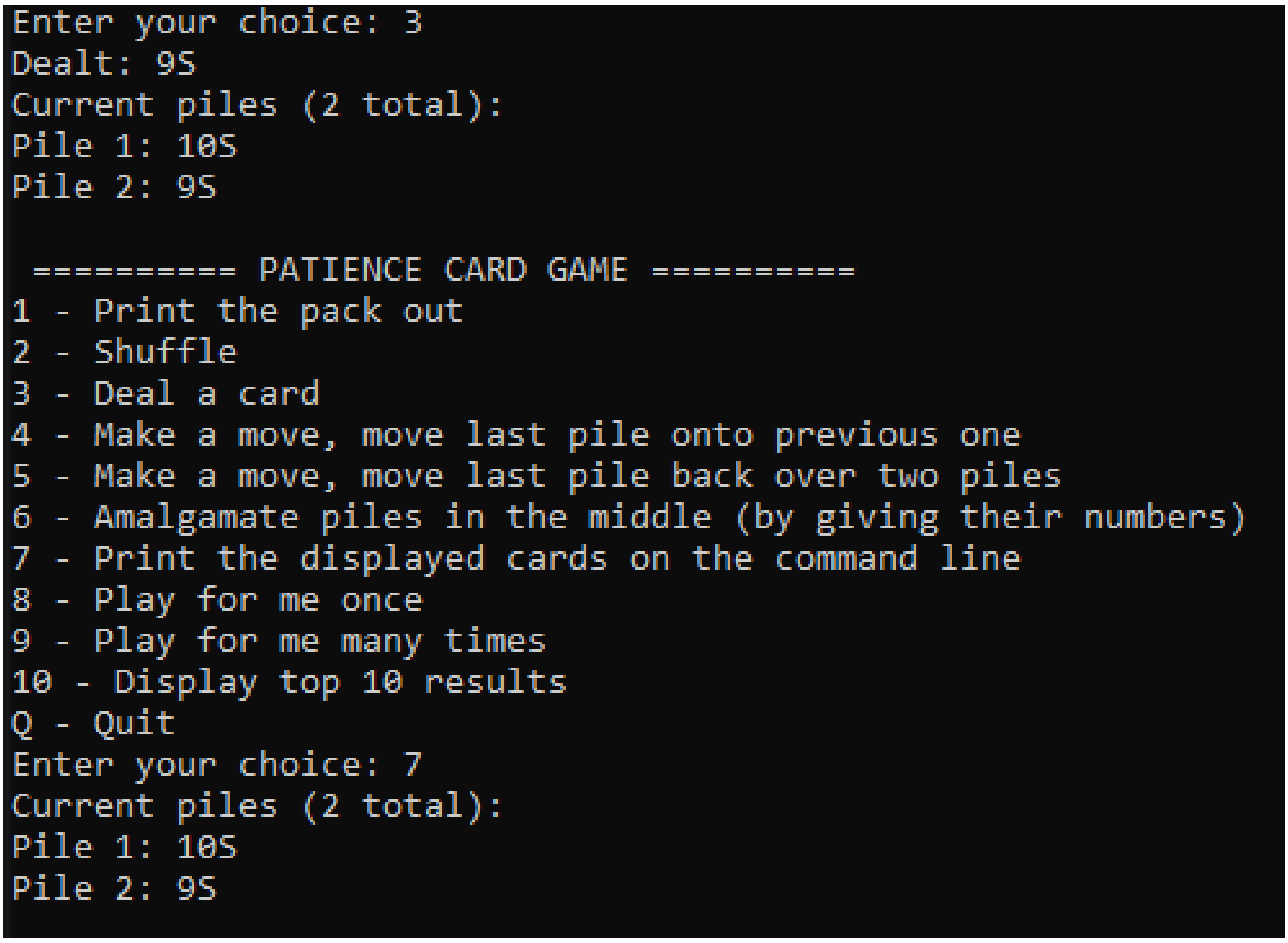
Test for invalid input for piles to move (FAIL to receive other input, NumberFormatException)



```
Current piles (2 total):
Pile 1: 2H
Pile 2: 3H

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
Q - Quit
Enter your choice: 6
Need at least 4 piles to amalgamate piles in the middle.
```

Figure 19

Test for minimal amount of piles for amalgamation option

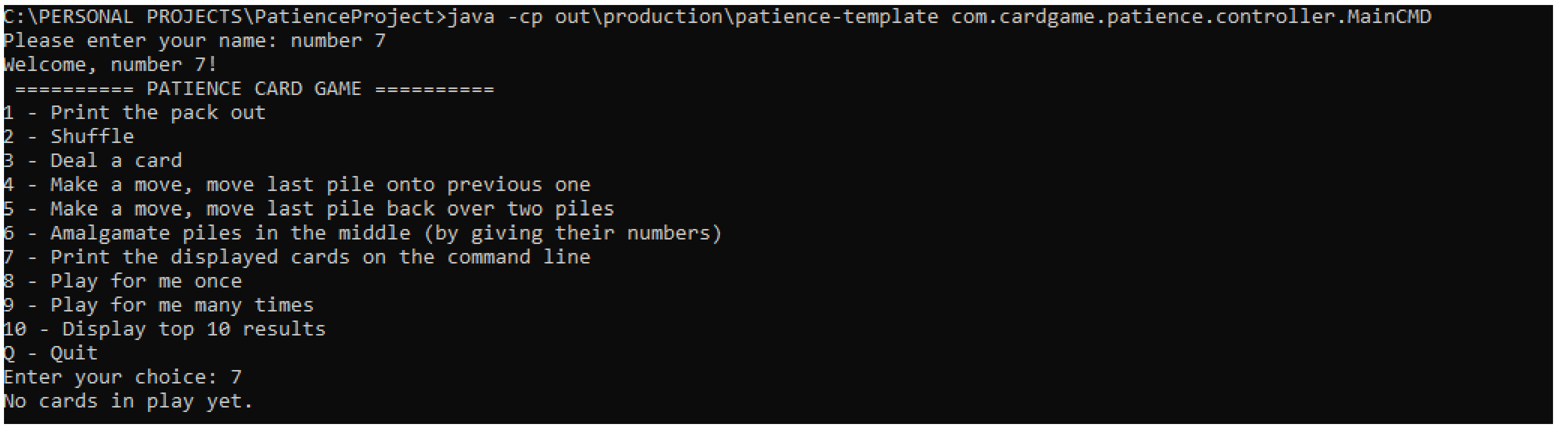


```
Enter your choice: 3
Dealt: 9S
Current piles (2 total):
Pile 1: 10S
Pile 2: 9S

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: 7
Current piles (2 total):
Pile 1: 10S
Pile 2: 9S
```

Figure 20

Test for seeing current piles in play, ALREADY was implemented in early stages to help see changes as code developed in terminal., so show piles is automatic to help game user

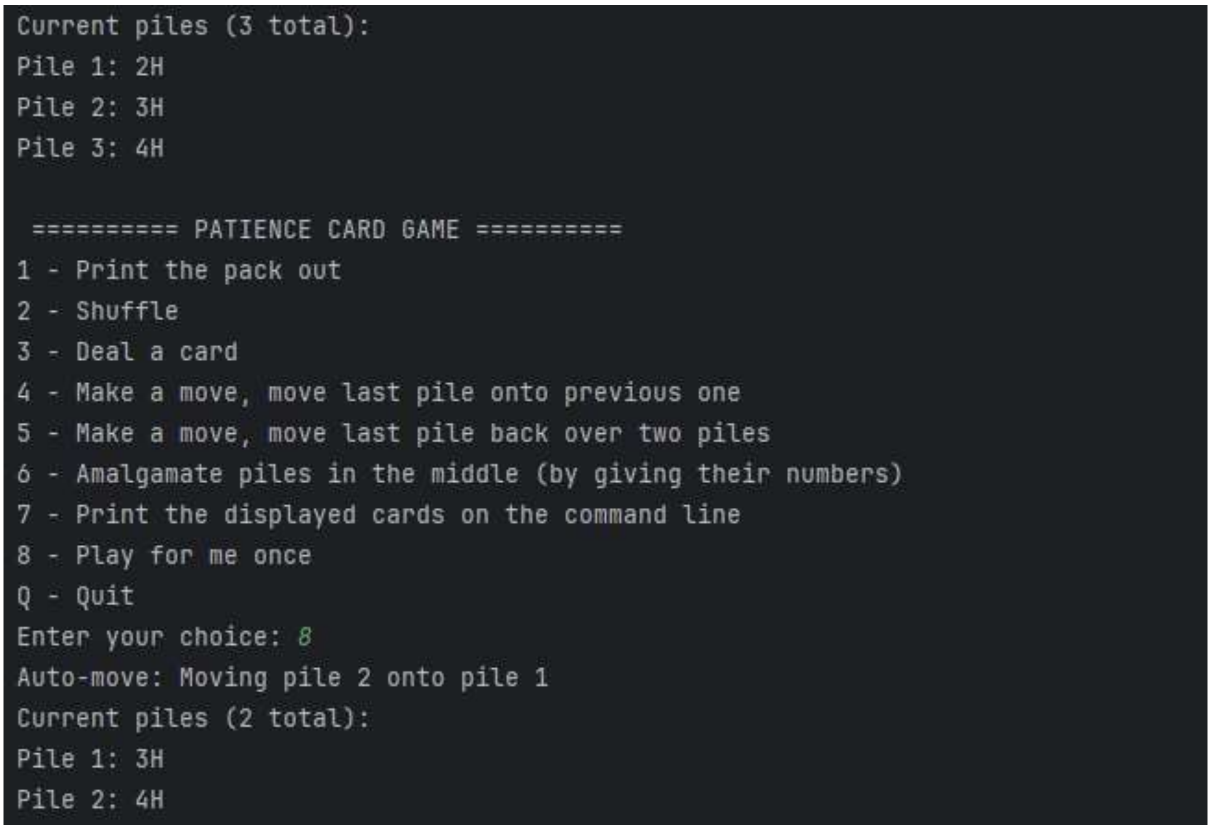
Help of showPiles() helper method



```
C:\PERSONAL PROJECTS\PatienceProject>java -cp out\production\patience-template com.cardgame.patience.controller.MainCMD
Please enter your name: number 7
Welcome, number 7!
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: 7
No cards in play yet.
```

Figure 21

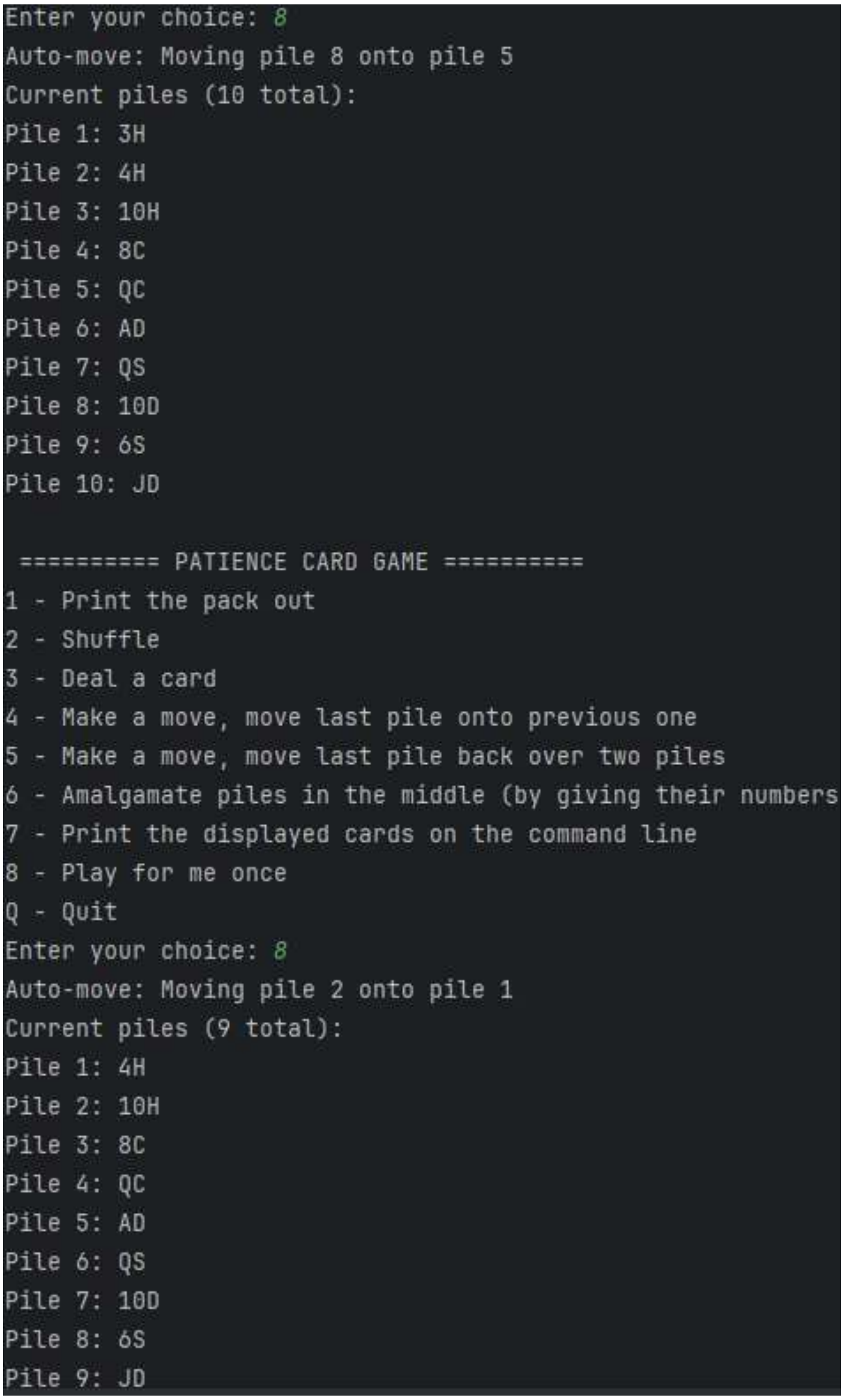
Testing piles in play straight away to handle 0 piles display with it showing no cards in play yet message



```
Current piles (3 total):
Pile 1: 2H
Pile 2: 3H
Pile 3: 4H

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
Q - Quit
Enter your choice: 8
Auto-move: Moving pile 2 onto pile 1
Current piles (2 total):
Pile 1: 3H
Pile 2: 4H
```

Figure 22

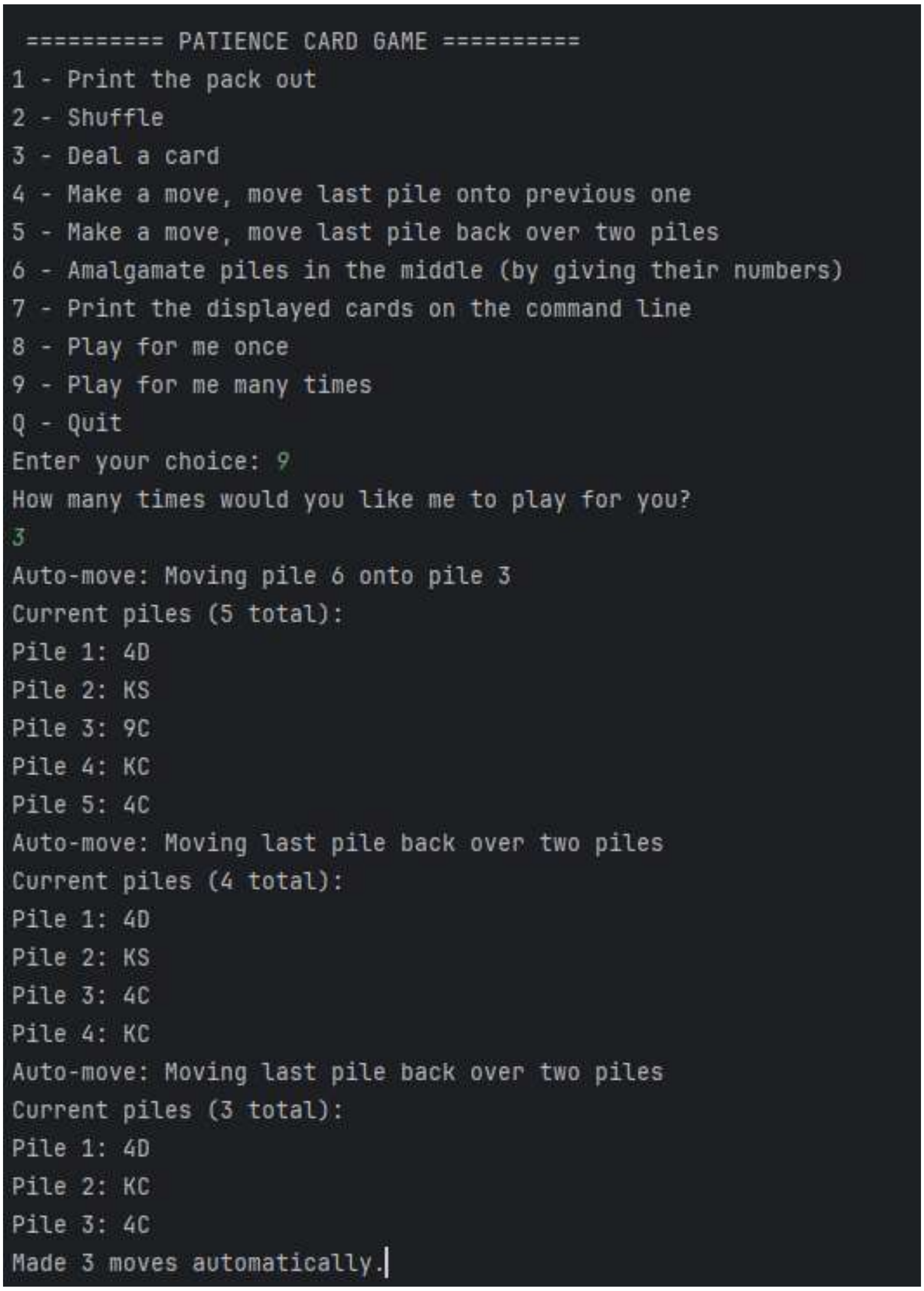
Testing priority of move in auto move feature



```
Enter your choice: 8
Auto-move: Moving pile 8 onto pile 5
Current piles (10 total):
Pile 1: 3H
Pile 2: 4H
Pile 3: 10H
Pile 4: 8C
Pile 5: QC
Pile 6: AD
Pile 7: QS
Pile 8: 10D
Pile 9: 6S
Pile 10: JD

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers
7 - Print the displayed cards on the command line
8 - Play for me once
Q - Quit
Enter your choice: 8
Auto-move: Moving pile 2 onto pile 1
Current piles (9 total):
Pile 1: 4H
Pile 2: 10H
Pile 3: 8C
Pile 4: QC
Pile 5: AD
Pile 6: QS
Pile 7: 10D
Pile 8: 6S
Pile 9: JD
```

Figure 23

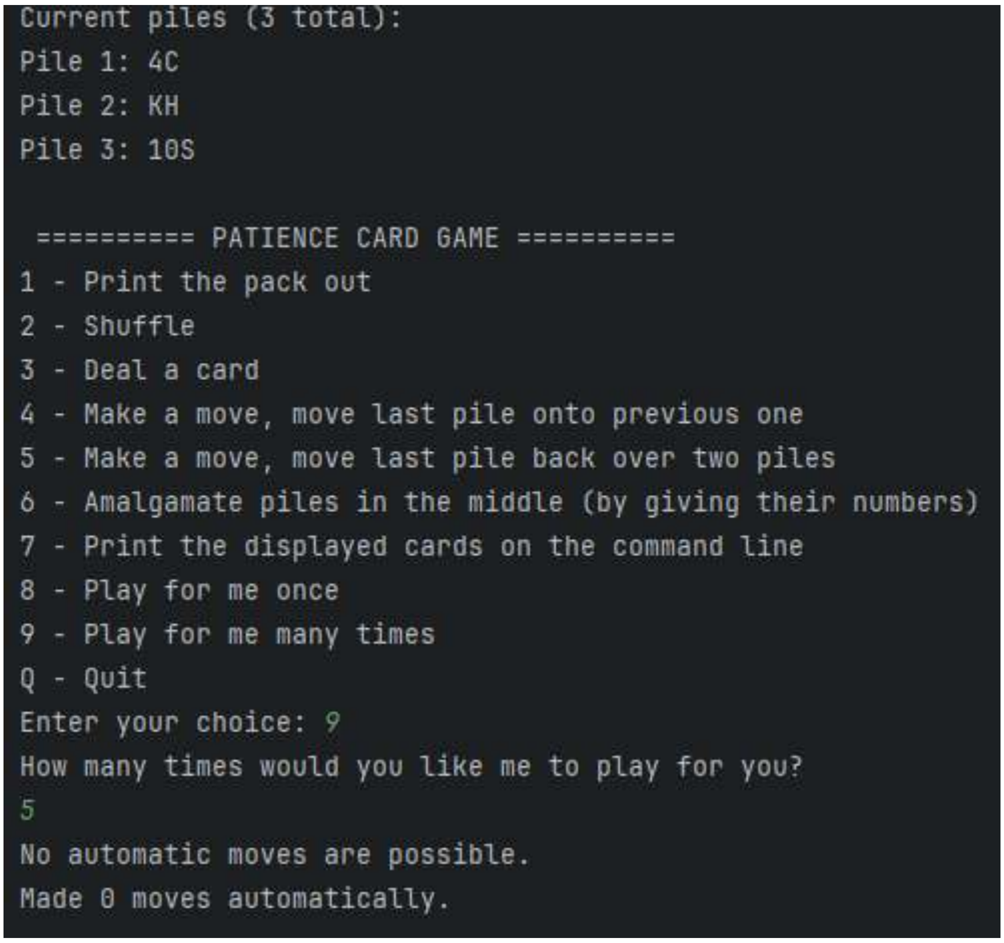
Testing the 2 move sequences of auto move and if farthest jump gets priority



```
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
Q - Quit
Enter your choice: 9
How many times would you like me to play for you?
3
Auto-move: Moving pile 6 onto pile 3
Current piles (5 total):
Pile 1: 4D
Pile 2: KS
Pile 3: 9C
Pile 4: KC
Pile 5: 4C
Auto-move: Moving last pile back over two piles
Current piles (4 total):
Pile 1: 4D
Pile 2: KS
Pile 3: 4C
Pile 4: KC
Auto-move: Moving last pile back over two piles
Current piles (3 total):
Pile 1: 4D
Pile 2: KC
Pile 3: 4C
Made 3 moves automatically.
```

Figure 24

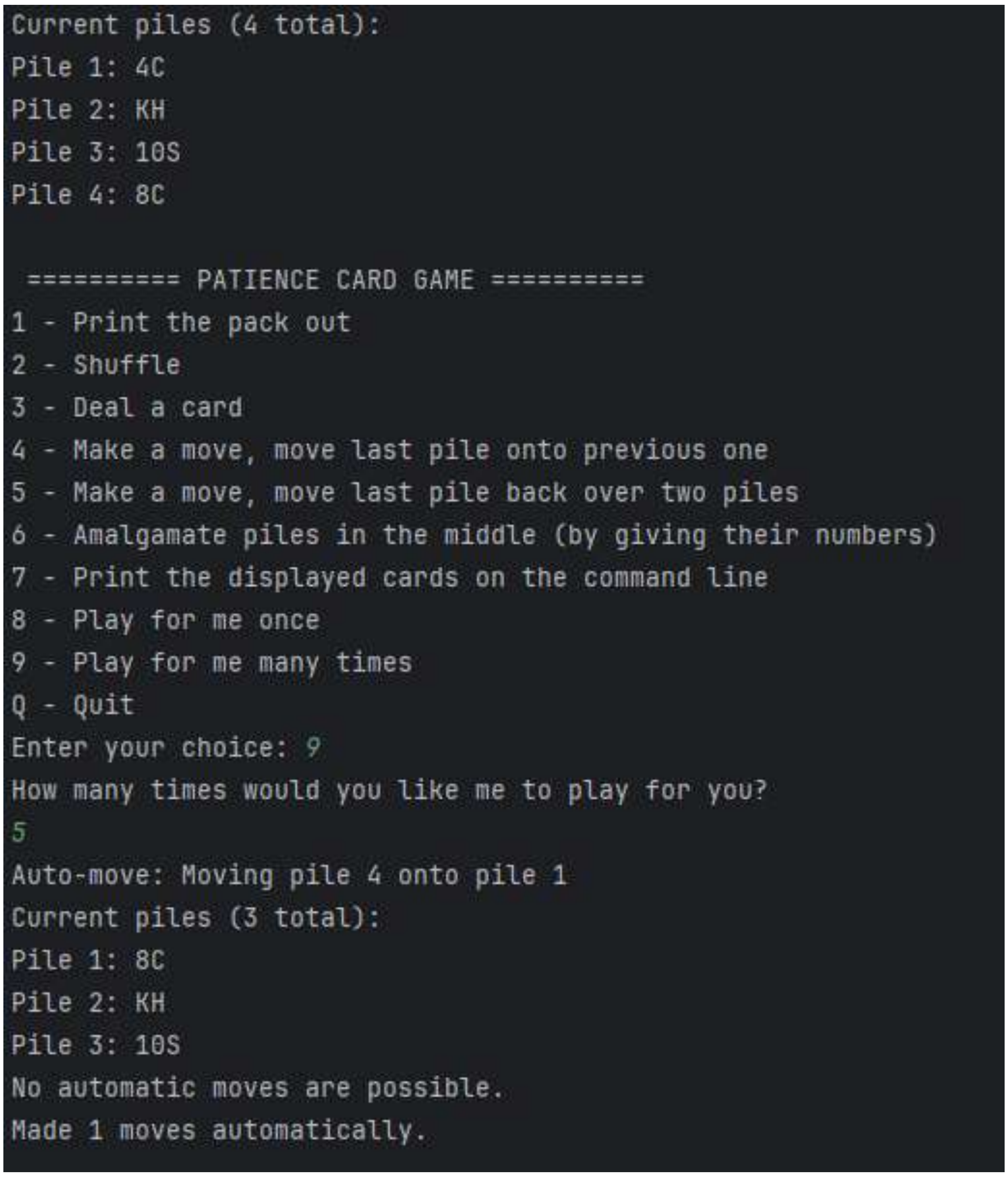
Auto play looped prioritizing big jumps then back over two piles, then adjacent being lowest priority



```
Current piles (3 total):
Pile 1: 4C
Pile 2: KH
Pile 3: 10S

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
Q - Quit
Enter your choice: 9
How many times would you like me to play for you?
5
No automatic moves are possible.
Made 0 moves automatically.
```

Figure 25

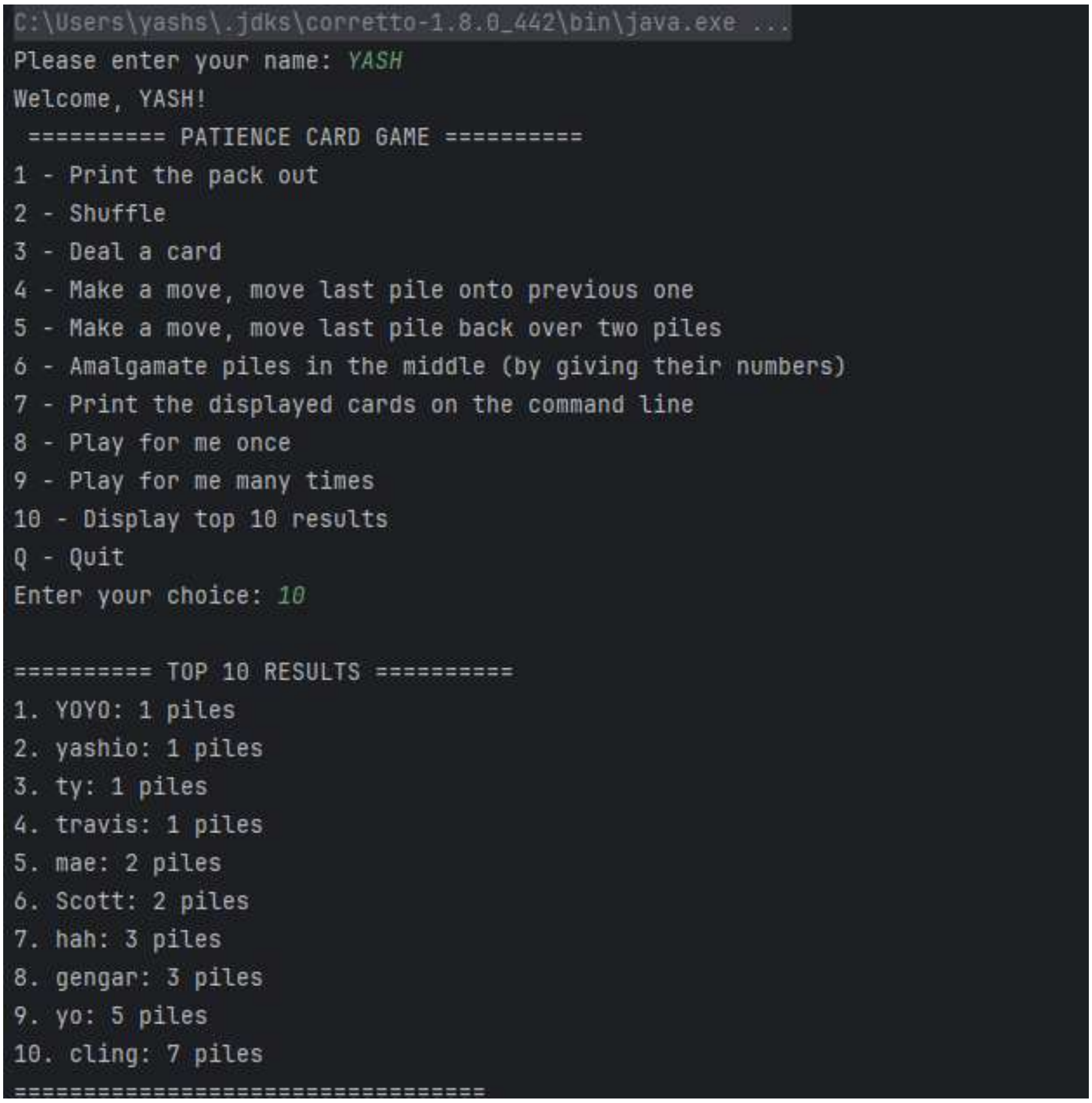
Testing auto moves response to user inputting too many moves to be played past no possible moves being able to be played



```
Current piles (4 total):
Pile 1: 4C
Pile 2: KH
Pile 3: 10S
Pile 4: 8C

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
Q - Quit
Enter your choice: 9
How many times would you like me to play for you?
5
Auto-move: Moving pile 4 onto pile 1
Current piles (3 total):
Pile 1: 8C
Pile 2: KH
Pile 3: 10S
No automatic moves are possible.
Made 1 moves automatically.
```

Figure 26

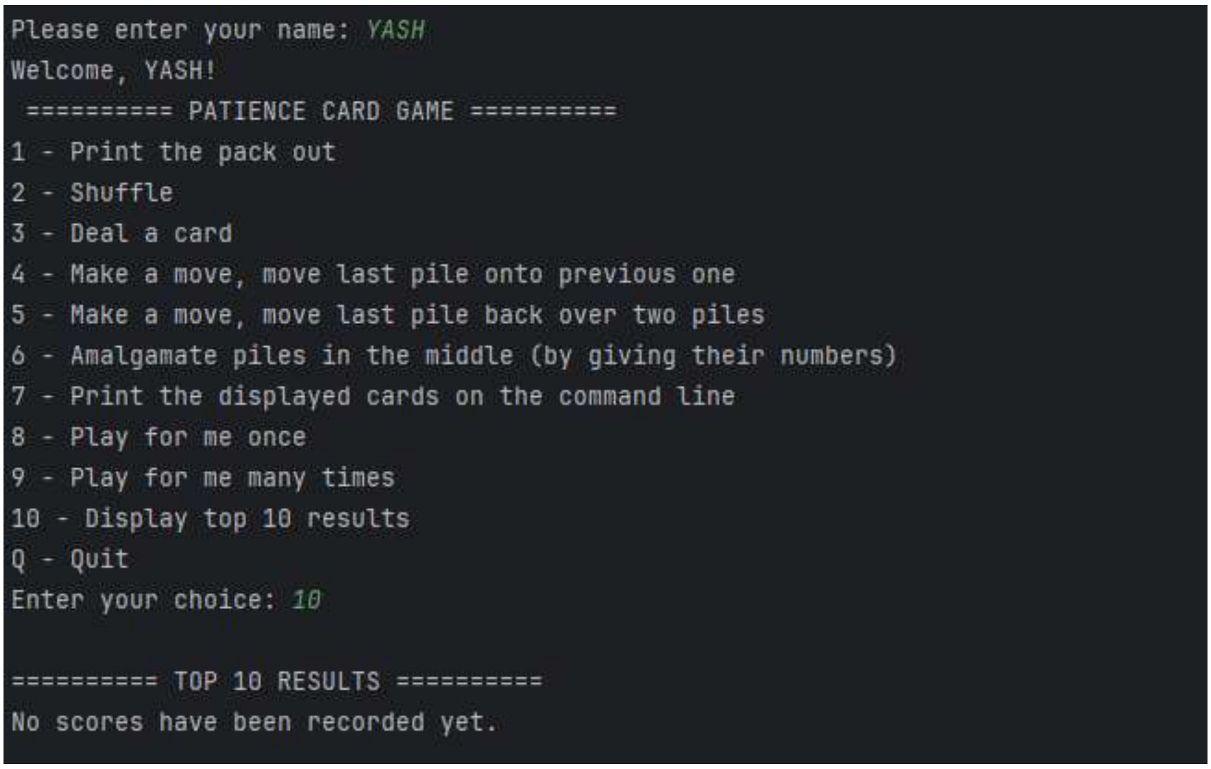
Testing FR9 response when no possible moves are available after finishing moves



```
C:\Users\yashs\.jdks\corretto-1.8.0_442\bin\java.exe ...
Please enter your name: YASH
Welcome, YASH!
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: 10

========== TOP 10 RESULTS ==========
1. YOYO: 1 piles
2. yashio: 1 piles
3. ty: 1 piles
4. travis: 1 piles
5. mae: 2 piles
6. Scott: 2 piles
7. hah: 3 piles
8. gengar: 3 piles
9. yo: 5 piles
10. cling: 7 piles
===================================
```

Figure 27

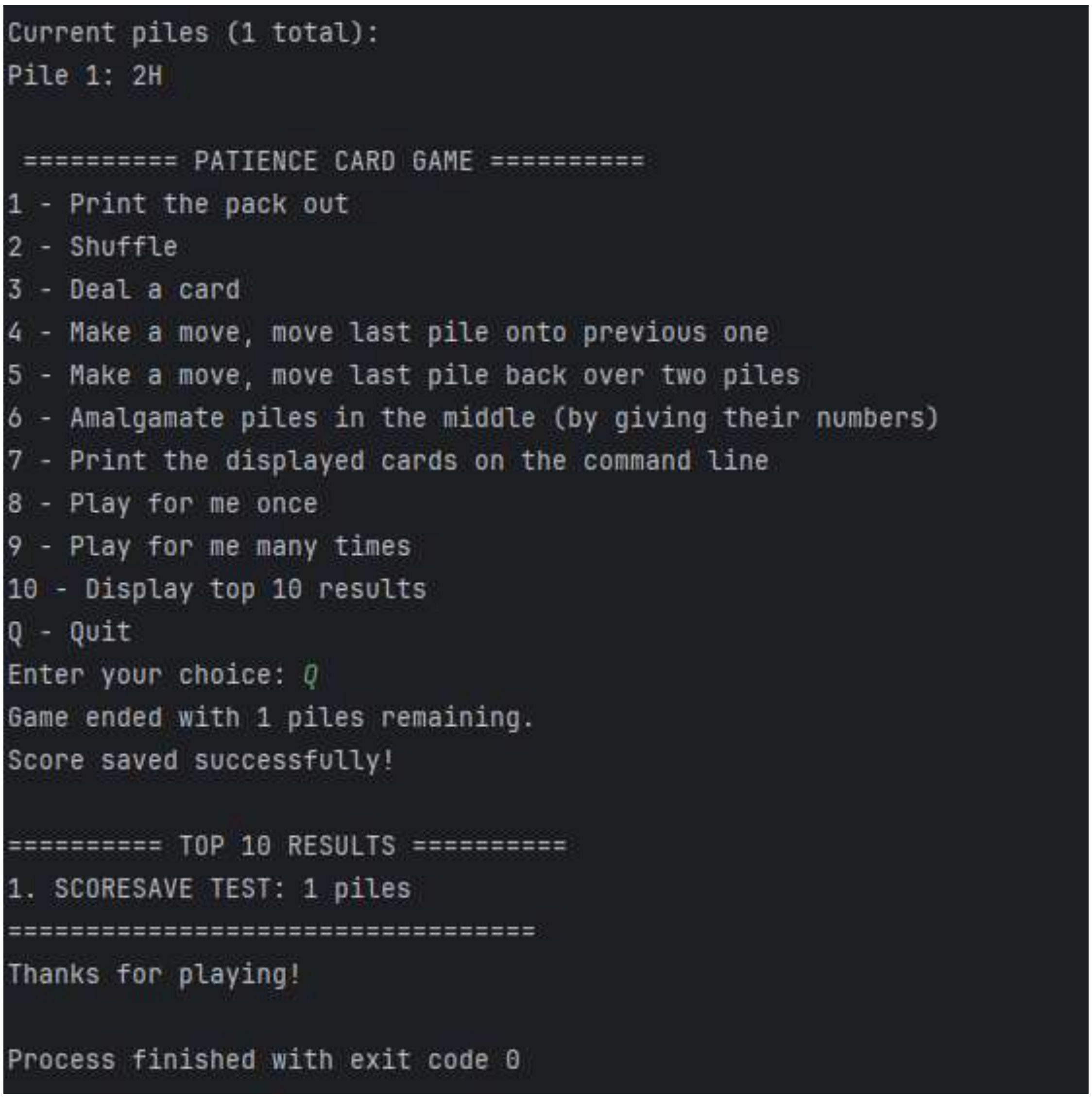
Testing Name input and FR10 , showing previous top 10 results



```
Please enter your name: YASH
Welcome, YASH!
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: 10

========== TOP 10 RESULTS ==========
No scores have been recorded yet.
```

Figure 28

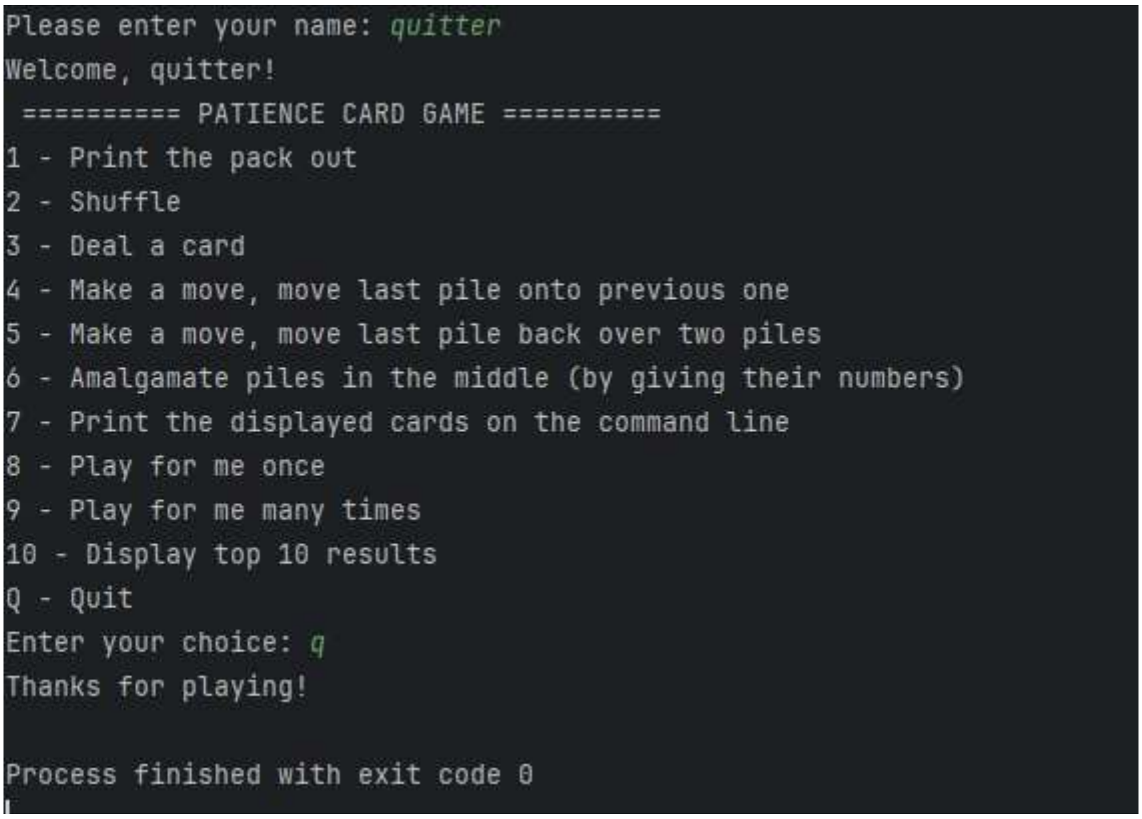
Testing FR10 after deleting patience scores txt file



```
Current piles (1 total):
Pile 1: 2H

 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: Q
Game ended with 1 piles remaining.
Score saved successfully!

========== TOP 10 RESULTS ==========
1. SCORESAVE TEST: 1 piles
====================================
Thanks for playing!

Process finished with exit code 0
```

Figure 29

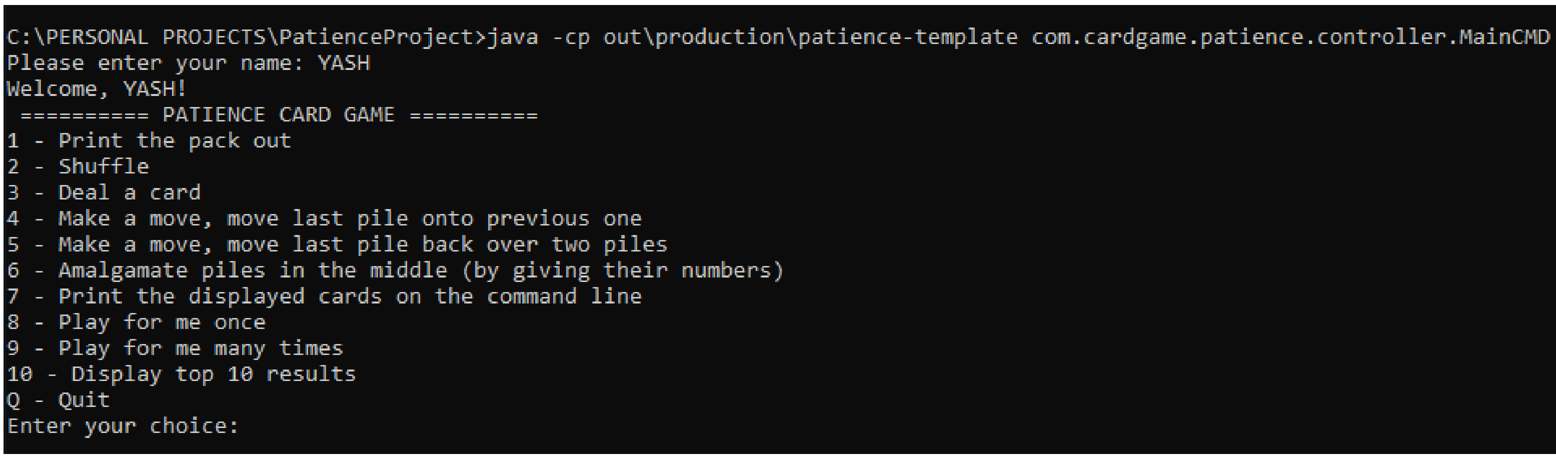
Test for saving user input name and their result alongside and exiting after



```
Please enter your name: quitter
Welcome, quitter!
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: q
Thanks for playing!

Process finished with exit code 0
```

Figure 30

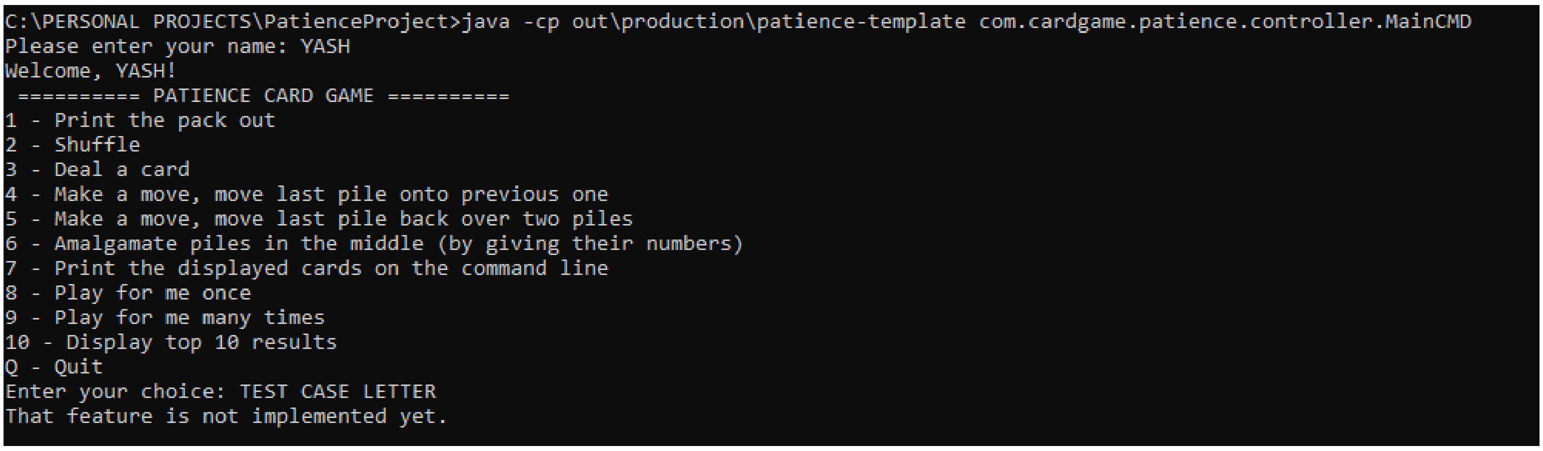
Test for how game responds to void game entry, no piles therefore no score to save



```
C:\PERSONAL PROJECTS\PatienceProject>java -cp out\production\patience-template com.cardgame.patience.controller.MainCMD
Please enter your name: YASH
Welcome, YASH!
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice:
```

Figure 31

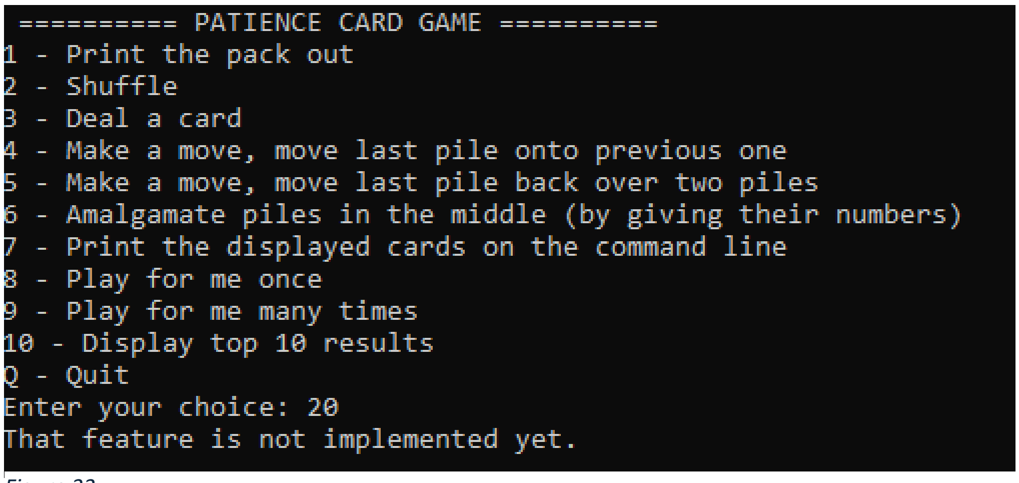
Running project on command line prompt showing full menu



```
C:\PERSONAL PROJECTS\PatienceProject>java -cp out\production\patience-template com.cardgame.patience.controller.MainCMD
Please enter your name: YASH
Welcome, YASH!
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: TEST CASE LETTER
That feature is not implemented yet.
```

Figure 32

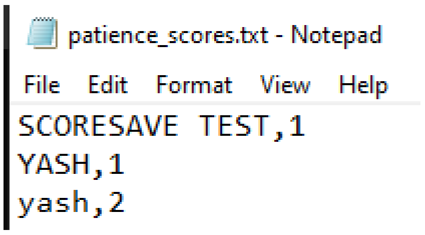
Tests for edge case input not in menu



```
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: 20
That feature is not implemented yet.
```

Figure 33

Tests for edge case input not in menu



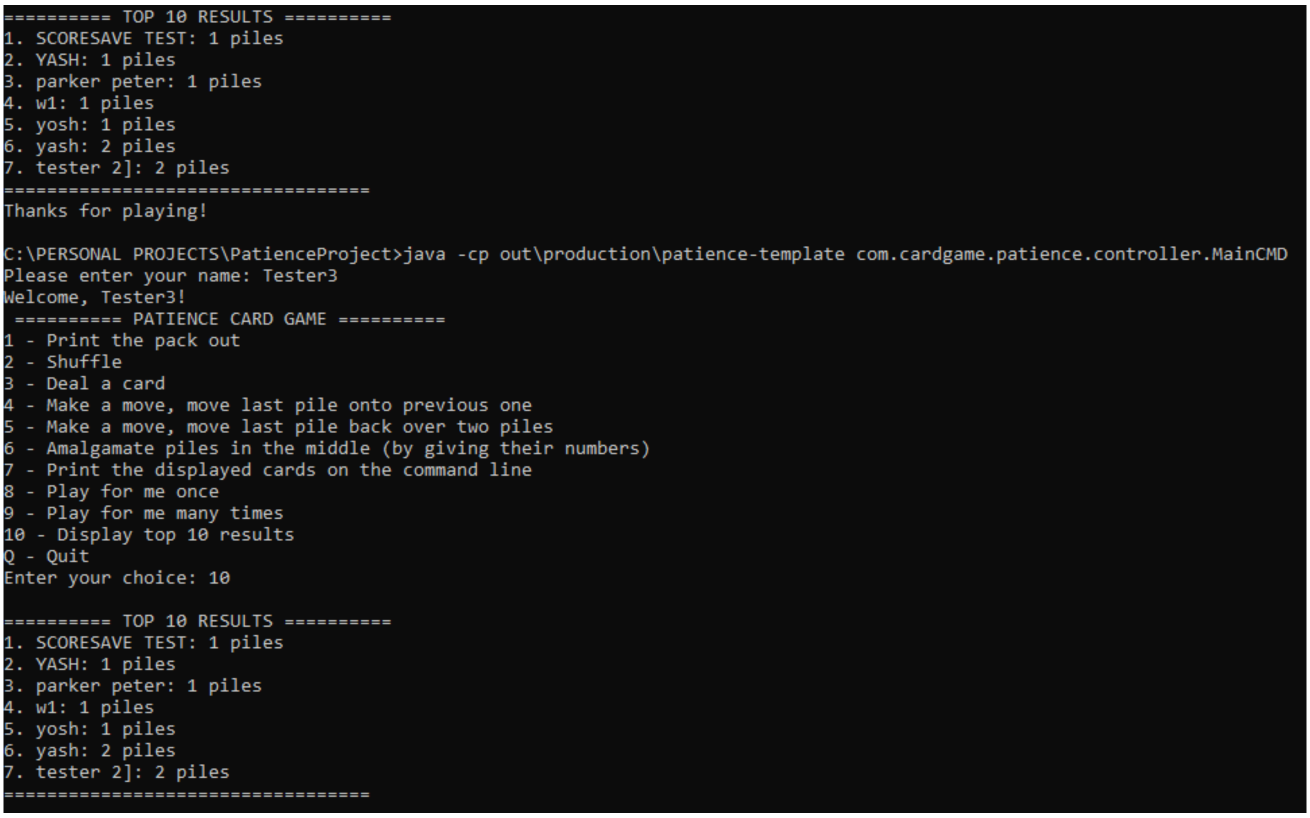
patience_scores.txt - Notepad

File Edit Format View Help

```
SCORESAVE TEST,1
YASH,1
yash,2
```

Figure 34

Test for viewing correct formatting of scores being saved in text file with no corruption



```
========== TOP 10 RESULTS ==========
1. SCORESAVE TEST: 1 piles
2. YASH: 1 piles
3. parker peter: 1 piles
4. w1: 1 piles
5. yosh: 1 piles
6. yash: 2 piles
7. tester 2]: 2 piles
==================================
Thanks for playing!

C:\PERSONAL PROJECTS\PatienceProject>java -cp out\production\patience-template com.cardgame.patience.controller.MainCMD
Please enter your name: Tester3
Welcome, Tester3!
 ========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: 10

========== TOP 10 RESULTS ==========
1. SCORESAVE TEST: 1 piles
2. YASH: 1 piles
3. parker peter: 1 piles
4. w1: 1 piles
5. yosh: 1 piles
6. yash: 2 piles
7. tester 2]: 2 piles
==================================
```

Figure 35

Test for persistence of Top score carrying over in new game

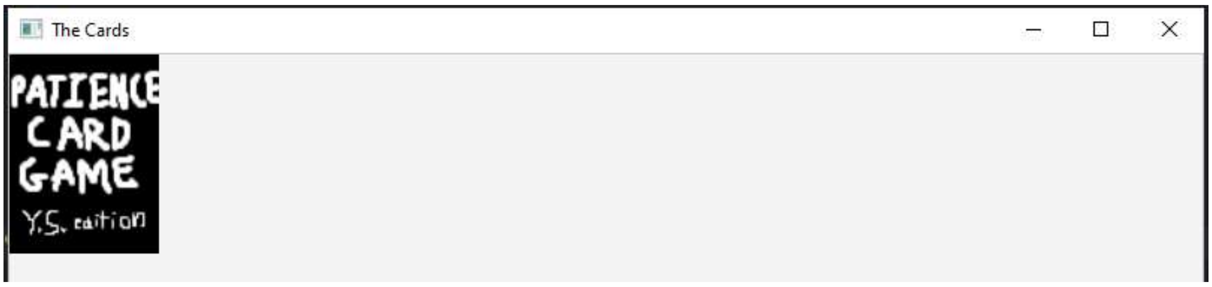
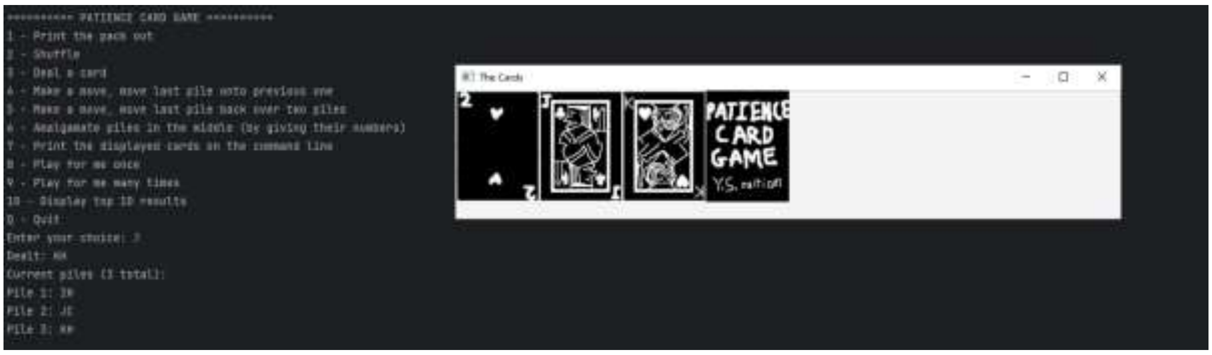


Figure 36

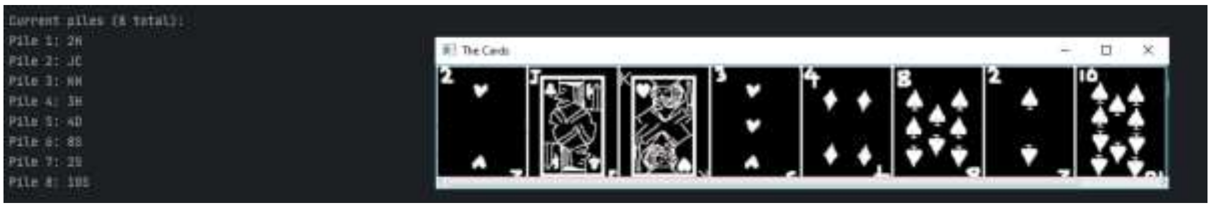
Showing the UI by JAVA fx implemented once game starts



```
========== PATIENCE CARD GAME ==========
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 (by giving their numbers)
7 - Print the displayed cards on the command line
8 - Play for me once
9 - Play for me many times
10 - Display top 10 results
Q - Quit
Enter your choice: 3
Dealt: KH
Current piles (3 total):
Pile 1: 2H
Pile 2: JC
Pile 3: KH
```

Figure 37

As Game progresses UI gets display updated in GUI window



```
Current piles (8 total):
Pile 1: 2H
Pile 2: JC
Pile 3: KH
Pile 4: 3H
Pile 5: 4D
Pile 6: 8S
Pile 7: 2S
Pile 8: 10S
```

Figure 38

Test case for GUI with high amounts of cards, particularly 8 or more the scroll bar appears

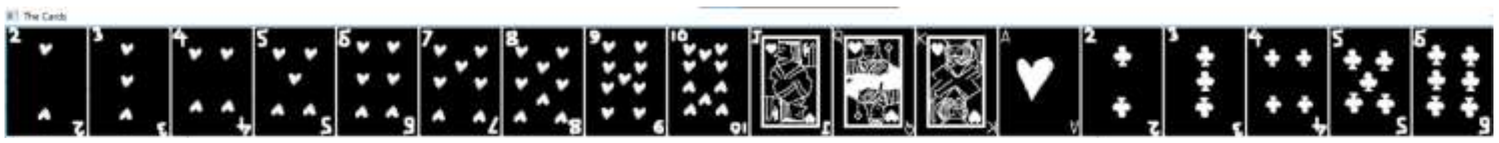


Figure 40

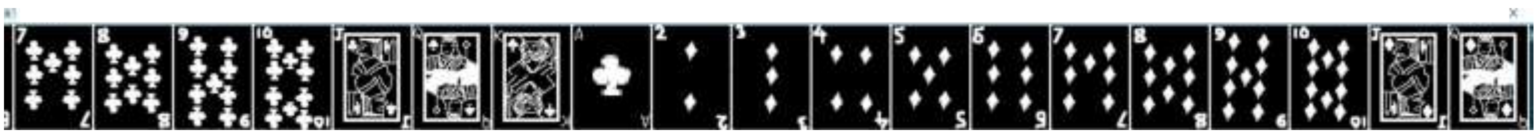


Figure 41

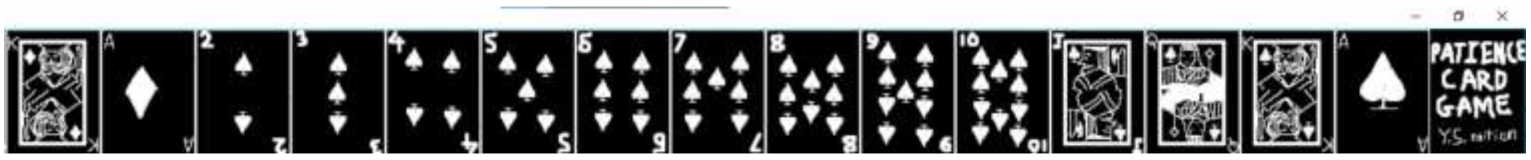


Figure 39

```
^

All Possible cards appear in separate java FX window acting as the
GUI ordered to test how window handles large loads of dealt cards.

Cards all made by me using paint.exe
```

5 EVALUATION

5.1 Process Recap:

I broke it down into key stages and did each of the requirements step by step by breaking down parts on paper with deadlines in between. once I planned and finished the designs of how my code would work it was very useful in terms of development by working in small steps while testing the steps on the way.

5.2 Flair

In terms of flair, I feel there wasn't a huge amount. I contributed by updating the card deck every time a change would happen in terminal, unsure if it was needed it assisted me a lot in later stages when lots of trial runs were happening. Additionally, I wanted to personalize the game, so I altered the existing gif files in paint.exe and applied a dark theme to every card to add some minimalism.

5.3 Self-Reflection

The most important were OOP principles being applied all together such as inheritance, encapsulation, file reading, writing, error case handling and threads. Alongside this, I found that constantly modifying my code to try catch exceptions or many other inputs a tough but rewarding task.

Whilst my project follows and functions according to the game rules, there are areas left to be done such as GUI enhancements such as animations, and more stricter game mechanics such as cards moving from right to left rather than flexibility of both ways. I believe this project earns around 70% based on successful working requirements listed with both a GUI, command line running and a thorough report to show the process of the code developing.