COM498 Algorithms and Data Structures – Testing Document
Niall Morrissey (B00787301)

Introduction

This report outlines the testing carried out on my Elevens Card game application. Through rigorous testing it helps ensure that the application works correctly and to detect and protect against any bugs occurring in the future. Testing was split into two key areas, automated unit testing for each individual class during the development of application and overall usability/validation testing.

Unit Testing

For each individual class I created automated tests for each of the necessary methods to check whether they worked correctly or if any bugs occurred. The test classes and individual tests created are listed below.

Please See ANNEX A for details of testing carried out and the results obtained.

- ListArrayTest Class
    1. Test if methods within AListArray class work correctly by creating a test String ListArray and attempting to manipulate it using all the different methods available.
- QueueTest Class
    1. Test if methods within Queue class work correctly by creating a test String Queue and attempting to manipulate it using all the different methods available.
- CardTest Class
    1. Test if cards objects can be created.
    2. Test if properties and values of cards can be retrieved.
    3. Test if equalsEleven method works correctly.
    4. Test if equalsJQK method works correctly.
- DeckTest Class
    1. Test if a deck of 52 card objects can be created.
    2. Test if all 52 card objects created are unique with no duplicates.
    3. Test if 9 cards can be dealt out shuffled and stored in a test ListArray.
    4. Test if a specific card entry can be retrieved from test ListArray.
    5. Test if all 52 cards can be dealt out to test ListArray and whether there are any. duplicates in test ListArray after all cards being shuffled and dealt out.
- BoardTest Class
    1. Test if 13 cards ranging from Ace to King of Hearts can be created and added to a testBoard and testDeck ListArray.
    2. Test if displayBoard method works correctly in displaying testBoard and testDeck.
    3. Test if checkPossibleMoves and replaceCard methods work correctly by simulating. end of a game using a winning board
    4. Test if Replay queue functionality works correctly recording each step of simulation.
    5. Test if checkPossibleMoves and replaceCard methods work correctly by simulating end of a game using a losing board.
    6. Test if displayBoard method works correctly on current state of cards after simulation.
    7. Test if replaceCards results in error when one of the parameters passed in to be removed is greater than scope of testBoard and if any changes were made.
    8. Test if replaceCards results in error when one of the parameters passed in to be removed is less than scope of testBoard and if any changes were made.

- ElevensGameTest Class
    1. Extensively stress test the program by automatically simulating at first 10 games, then 100 games, 1000 games, 10,000 games and finally 100,000 games.

Usability/Validation Testing

I carried usability test on the main Elevens game application to ensure the game responded as it should from user inputs. These tests also ensured that if a player entered inputs that were out of the range of the game the code would not break and simply ask the player to re-enter a correct input. I set up validation on all key inputs.

Please See ANNEX B for details of testing carried out and the results obtained.

Usability/Validation Testing

List of Tests performed

1. Test if program starts and a game of Elevens can be started.
2. Test when selecting two cards from the board if they are replaced when they equal eleven.
3. Test when selecting two cards from the board if they are replaced when they don't equal eleven.
4. Test if hint functionality works when a letter is entered instead of a slot number
5. Test if card slot that is out of range of the board is accepted using test parameters 0 and 10 which are lower and greater than the range of the board.
6. Test if hint functionality works when a Jack, Queen, King combination is found in the board.
7. Test when selecting three cards from the board if they are replaced when they equal a Jack, Queen King combination.
8. Test when selecting three cards from the board if they are replaced when they don't equal a Jack, Queen King combination.
9. Test if game of Elevens can be lost and ends in stalemate.
10. Test if a new game of Elevens can be played from within the application.
11. Test if game of Elevens can be won.
12. Test if replay functionality works correctly and if user is required to increment through steps.
13. Test if demonstration mode works correctly allowing the user to increment through the steps of a simulated Elevens game.

Overall Results

Based on the individual results from each of the tests run, I adapted the code to ensure that any errors were corrected. The testing regime was used throughout the development of the code to ensure that errors and bugs were corrected as they arose. I have included the results of each of the individual tests carried out in annexes A and B below.

Security

In developing my code, I ensured that security was considered throughout. Using validation on user inputs allowed me to ensure only the correct inputs were being made with the user asked to reinput selections etc if they were outside of the bounds of the game. I made extensive use of validation throughout the code and where an input was requested from the player the validation ensured only inputs within the bounds were allowed.

The validation used in the code is as follows:

- While Loop validation to ensure only integer inputs can be returned in the selection method. Entering letter prompts hint functionality instead of crashing.

- Range validation e.g. only slots 1 to the length of the board were allowed, otherwise user prompted to reenter. This validation was setup dynamically to reflect the number of cards on the Board e.g. if only 4 cards remain on board, slot 5 could not be accepted.
- Try Catch validation used in testing to prevents test crashing when edge case testing with parameters outside the scope of arrays or listArrays.

I developed each class as a separate part of the code which gave me the ability to make individual classes either public, protected, or private depending on the access requirements within the package and enabled me to use data encapsulation which allows for data hiding through the use access modifiers.

The Card Class uses public access modifiers for most methods as access needed from other classes. The Deck Class uses private access modifier for the Deck ListArray and has getter methods in place for access to specific requirements. Most methods in Deck class are set to public. The Board Class uses protected access modifier for the Board ListArray and all the Board class methods as access is required by ElevensGame Class which inherits the ListArray but is closed off against all other classes. The ElevensGame Class is the highest layer of the program and has all methods set to private bar the static main and game methods. Effective use of encapsulation and data hiding is key to securing code from external entities.

Conclusion

As a result of the testing regime and the many tests carried out, I was able to harden my code to ensure that the chances of the code breaking or crashing when playing a game is minimal. I have played the finished game many times and have also asked other family members to play the game with no issues which confirmed the robustness of the code. I also tested the game to ensure that the odds of winning were approx. 10%. You will see in the ElevensGameTest class below I ran the game automatically 100,000 times with a win rate of approx. 10%. This stress tested the code and confirmed win results were as expected and that the code was very robust.

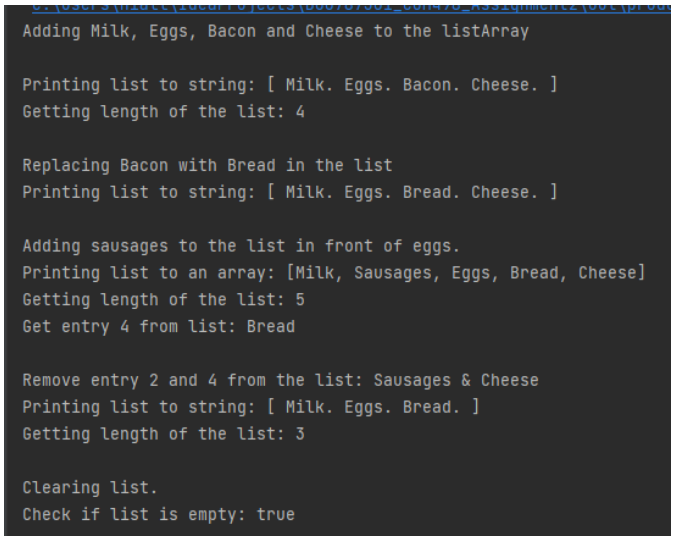
Annex A

AListArrayTest Class

AListArrayTest Class - Test

Test whether all the methods within the AListArray class work as expected including the add, replace, remove, clear, isEmpty, toString, toArray and getLength methods. Use the test String ListArray 'breakfast' to store different food items, testing each of the methods to manipulate the items within the ListArray.

Result

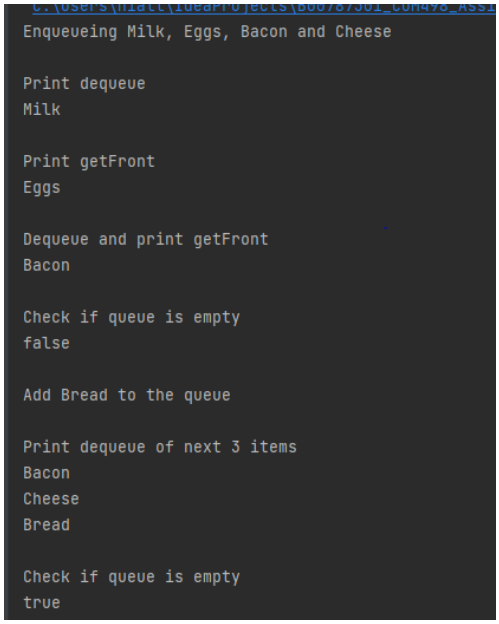


All methods worked correctly on the test String ListArray 'breakfast' as can be seen above. The different food items were manipulated as expected.

QueueTest Class

QueueTest Class - Test

Test whether all the methods within the Queue class work as expected including the enqueue, deqeue, getFront, isEmpty, and clear methods. Use the test String Queue 'breakfast' to store different food items, testing each of the methods to manipulate the items within the Queue.



All methods worked correctly on the test String Queue 'breakfast' as can be seen above.

CardTest Class

CardTest Class - Test One

Test whether eight card objects can be created. Six cards using correct Rank and Suit parameters and two cards using parameters outside of the scope of the Rank and Suit arrays in the card class. Try Catch exception handling to be used for creation of card objects outside of scope.

Card objects that should be created:

- Ace of Hearts
- 4 of Spades
- 10 of Diamonds
- Jack of Clubs
- Queen of Diamonds
- King of Hearts
- (RANKS Array Index ‘15’) of Hearts
- 9 of (SUITS Array Index ‘-1’)

Result

```
ArrayIndexOutOfBoundsException error caught.
Tried to create card with rank index 15 which is out of range.

ArrayIndexOutOfBoundsException error caught.
Tried to create card with suit index -1 which is out of range.
```

All cards with correct parameters created successfully and cards with parameters outside of scope caught and handled appropriately.

Must ensure that when cards are created by the deck class, parameters passed in are within the scope of the RANKS and SUITS arrays to ensure no errors.

CardTest Class - Test Two

Test whether card properties for each created card can be accessed and the rank value for each card retrieved.

Result

```
Properties of card one: Ace of Hearts
Point value of card one: 1

Properties of card two: 4 of Spades
Point value of card two: 4

Properties of card three: 10 of Diamonds
Point value of card three: 10

Properties of card four: Jack of Clubs
Point value of card four: 11

Properties of card five: Queen of Diamonds
Point value of card five: 12

Properties of card five: King of Hearts
Point value of card five: 13
```

All card objects were created with correct parameters and the rank value for each is appropriate.

CardTest Class - Test Three

Test equalEleven method with created test Card objects to ensure only correct combination of cards equal eleven. If card ranks equal eleven should return '1', otherwise '0'.

Result

```
Does Ace of Hearts and 4 of Spades equal eleven: 0
Does Ace of Hearts and 10 of Diamonds equal eleven: 1
Does Jack of Clubs and Queen of Diamonds equal eleven: 0
```

Method appears to be working correctly as only Ace of Hearts and 10 of Diamonds equal eleven.

CardTest Class - Test Four

Test equalJQK method with created test Card objects to ensure only correct combination of cards equal a Jack, Queen, King combination. If cards are JQK should return '1', otherwise '0'.

Result

```
Are the 4 of Spades, Jack of Clubs and King of Hearts a jack, queen, king combination: 0
Are the 10 of Diamonds, King of Hearts and King of Hearts a jack, queen, king combination: 0
Are the King of Hearts, Jack of Clubs and Queen of Diamonds a jack, queen, king combination: 1
```

As this method relies on a combination of cards to equal the rank score of 36 (Jack rank '11', Queen rank '12', and King rank '13') to return true (1). The testing revealed a bug where if a 10, King and King combination which also have a rank score of 36 are passed in, it returned true (1). This has since been fixed to ensure only three cards with unique rank values passed through will return true (1).

DeckTest Class

DeckTest Class - Test One

Create a new instance of the Deck class and test whether the constructor creates a deck of 52 new card objects stored in DECK List Array by retrieving length of the deck and printing each of the newly created cards.

Result

```
Number of cards in deck: 52

New deck of cards created
1 - Ace of Hearts
2 - 2 of Hearts
3 - 3 of Hearts
4 - 4 of Hearts
5 - 5 of Hearts
6 - 6 of Hearts
7 - 7 of Hearts
8 - 8 of Hearts
9 - 9 of Hearts
10 - 10 of Hearts
11 - Jack of Hearts
12 - Queen of Hearts
13 - King of Hearts
14 - Ace of Clubs
15 - 2 of Clubs
16 - 3 of Clubs
17 - 4 of Clubs
18 - 5 of Clubs
19 - 6 of Clubs
20 - 7 of Clubs
21 - 8 of Clubs
22 - 9 of Clubs
23 - 10 of Clubs
24 - Jack of Clubs
25 - Queen of Clubs
26 - King of Clubs
27 - Ace of Spades
28 - 2 of Spades
29 - 3 of Spades
30 - 4 of Spades
31 - 5 of Spades
32 - 6 of Spades
33 - 7 of Spades
34 - 8 of Spades
35 - 9 of Spades
36 - 10 of Spades
37 - Jack of Spades
38 - Queen of Spades
39 - King of Spades
40 - Ace of Diamonds
41 - 2 of Diamonds
42 - 3 of Diamonds
43 - 4 of Diamonds
44 - 5 of Diamonds
45 - 6 of Diamonds
46 - 7 of Diamonds
47 - 8 of Diamonds
48 - 9 of Diamonds
49 - 10 of Diamonds
50 - Jack of Diamonds
51 - Queen of Diamonds
52 - King of Diamonds
```

Length of deck is 52 and all cards have been created correctly.

DeckTest Class - Test Two

Test whether any duplicate card objects have been created within the deck using nested for loops to test through every card for possible matches. If any matches found set Boolean variable 'duplicateFound' to true, otherwise leave false.

Result

```
Duplicates found after creating all 52 cards: false
```

No duplicates were found after searching through for matches which is correct.

DeckTest Class - Test Three

Test whether first 9 cards dealt out are shuffled and can be stored & retrieved from the TEST Card ListArray and test if number of cards in DECK and TEST ListArrays reflect the changes after dealing.

Result

```
Test if first 9 cards are shuffled and can be stored/retrieved in TEST Card ListArray.
1 - King of Spades
2 - 9 of Diamonds
3 - 5 of Diamonds
4 - 8 of Diamonds
5 - Ace of Clubs
6 - 2 of Diamonds
7 - 4 of Hearts
8 - King of Hearts
9 - 10 of Spades
Number of cards in DECK: 43
Number of cards in stored in TEST Card ListArray: 9
```

The first 9 cards dealt out are in completely random order and the number of cards in DECK is 43 and number of cards in TEST is 9 which is correct.

DeckTest Class - Test Four

Test whether a specific card can be retrieved from the TEST ListArray. Attempt to retrieve card in slot 7 from the list as shown in the screenshot above.

Result

```
Test whether card 7 can be retrieved from TEST ListArray above: 4 of Hearts
```

Card entries can be correctly retrieved from a specific slot.

DeckTest Class - Test Five

Test whether all 52 cards can be shuffled and dealt out from DECK ListArray to TEST ListArray. Check whether TEST now contains any card duplicates and whether number of cards in DECK and TEST reflect the changes.

Result

```
Test if all 52 cards can be shuffled and dealt out.

Duplicates found after shuffling and dealing all 52 cards: false

Number of cards in deck: 0
Number of cards in stored in TEST Card ListArray: 52
```

All 52 cards can be shuffled and dealt out. No duplicates are found in the shuffled TEST Card ListArray and the number of cards now in deck is 0 and in TEST is 52 which is correct.

BoardTest Class

BoardTest Class - Test One

Create testBOARD and testDECK ListArrays and between them hold 13 Card objects ranging from Ace to King of Hearts. Test whether length of each ListArray reflects the newly created and added cards.

Result

```
Creating a testBoard and testDeck with only Cards that have suit 'Hearts' ranging from Ace to King.

Length of testBOARD card arrayList: 9
Length of testDECK card arrayList: 4
```

The ListArrays have been created correctly and testBOARD holds 9 cards and the testDECK holds the other 4 cards.

BoardTest Class - Test Two

Test displayBoard method with full testBOARD and remaining testDECK length.

Result

```
Test displayBoard method with full testBoard and remaining testDeck length.

----------Board-------------
Slot 9  -  9 of Hearts
Slot 8  -  8 of Hearts
Slot 7  -  7 of Hearts
Slot 6  -  6 of Hearts
Slot 5  -  5 of Hearts
Slot 4  -  4 of Hearts
Slot 3  -  3 of Hearts
Slot 2  -  2 of Hearts
Slot 1  -  Ace of Hearts
----------------------------
4 cards left in deck.
----------------------------
```

The displayBoard method works correctly highlighting each card and it slot and the remaining length of cards in testDECK.

BoardTest Class - Test Three

Test to check whether checkPossibleMoves method returns any remaining possible moves and replaceCards methods work if any move is found.

This test involves looping through for the length of testBOARD while greater than 0, checking if there are any possible moves remaining, if not return stalemate. If there is a remaining move, then replacing them with new cards if there are any remaining in the deck or just removing the cards if there is not. This loop should continue until a stalemate occurs or the length of testBOARD reaches 0.

This test should result in a win as the 13 cards ranging from Ace to King of Hearts should always be able to win at Elevens.

Result

```
Test checkPossibleMoves method for possible moves and test replaceCards method if move is present
----Test on cards that should Win.----

Returns slots that can be removed as integer ArrayList: [ 2. 9. ]
[ Ace of Hearts. 2 of Hearts. 3 of Hearts. 4 of Hearts. 5 of Hearts. 6 of Hearts. 7 of Hearts. 8 of Hearts. 9 of Hearts. ]

Returns slots that can be removed as integer ArrayList: [ 3. 8. ]
[ Ace of Hearts. Queen of Hearts. 3 of Hearts. 4 of Hearts. 5 of Hearts. 6 of Hearts. 7 of Hearts. 8 of Hearts. King of Hearts. ]

Returns slots that can be removed as integer ArrayList: [ 1. 3. ]
[ Ace of Hearts. Queen of Hearts. 10 of Hearts. 4 of Hearts. 5 of Hearts. 6 of Hearts. 7 of Hearts. Jack of Hearts. King of Hearts. ]

Returns slots that can be removed as integer ArrayList: [ 2. 5. ]
[ Queen of Hearts. 4 of Hearts. 5 of Hearts. 6 of Hearts. 7 of Hearts. Jack of Hearts. King of Hearts. ]

Returns slots that can be removed as integer ArrayList: [ 2. 3. ]
[ Queen of Hearts. 5 of Hearts. 6 of Hearts. Jack of Hearts. King of Hearts. ]

Returns slots that can be removed as integer ArrayList: [ 2. 1. 3. ]
[ Queen of Hearts. Jack of Hearts. King of Hearts. ]

Board after test completed: [ ]
```

The test shows that the checkPossibleMoves and replaceCard methods are working correctly, with the checkPossibleMoves method returning the correct ListArray slots to be removed and the replace cards method replacing/removing the selected cards slots passed through as int parameters. The test continues to step through repeating the steps until the while loop condition is met indicating the game is won.

BoardTest Class - Test Four

Test whether the replay functionality records the board and removed cards at each step of the previous game. The test should have recorded each of the individual steps of the game above.

Result

```
Testing Replay functionality
Board: [ Ace of Hearts. 2 of Hearts. 3 of Hearts. 4 of Hearts. 5 of Hearts. 6 of Hearts. 7 of Hearts. 8 of Hearts. 9 of Hearts. ]
Removed: 9 of Hearts
Removed: 2 of Hearts
Board: [ Ace of Hearts. Queen of Hearts. 3 of Hearts. 4 of Hearts. 5 of Hearts. 6 of Hearts. 7 of Hearts. 8 of Hearts. King of Hearts. ]
Removed: 8 of Hearts
Removed: 3 of Hearts
Board: [ Ace of Hearts. Queen of Hearts. 10 of Hearts. 4 of Hearts. 5 of Hearts. 6 of Hearts. 7 of Hearts. Jack of Hearts. King of Hearts. ]
Removed: 10 of Hearts
Removed: Ace of Hearts
Board: [ Queen of Hearts. 4 of Hearts. 5 of Hearts. 6 of Hearts. 7 of Hearts. Jack of Hearts. King of Hearts. ]
Removed: 7 of Hearts
Removed: 4 of Hearts
Board: [ Queen of Hearts. 5 of Hearts. 6 of Hearts. Jack of Hearts. King of Hearts. ]
Removed: 6 of Hearts
Removed: 5 of Hearts
Board: [ Queen of Hearts. Jack of Hearts. King of Hearts. ]
Removed: King of Hearts
Removed: Jack of Hearts
Removed: Queen of Hearts
```

Test shows that replay functionality is working correctly as every step from Test 3 has been recorded and displayed correctly. During user testing of application, replay functionality should require user to press ‘enter’ to iterate through different moves made.

BoardTest Class - Test Five

Repeat test three but with a preselected shuffled arrangement of cards that is not capable of winning and should result in a stalemate. Will fill testBOARD with 9 new cards and testDECK with 4 new cards. This will test again whether the checkPossibleMoves and replaceCard methods work correctly with different cards.

Result

```
Creating a testBoard and testDeck with shuffled arrangement of cards that shouldn't be able to win

Test checkPossibleMoves method for possible moves and test replaceCards method if move is present
----Test on cards that should Lose.----

Returns slots that can be removed as integer ArrayList: [ 1. 2. ]
[ 5 of Hearts. 6 of Hearts. 3 of Clubs. 7 of Diamonds. 5 of Diamonds. 5 of Clubs. Jack of Hearts. 4 of Diamonds. King of Spades. ]

Returns slots that can be removed as integer ArrayList: [ 2. 8. ]
[ 10 of Spades. 7 of Clubs. 3 of Clubs. 7 of Diamonds. 5 of Diamonds. 5 of Clubs. Jack of Hearts. 4 of Diamonds. King of Spades. ]

Returns slots that can be removed as integer ArrayList: [ 2. 3. ]
[ 10 of Spades. 8 of Diamonds. 3 of Clubs. 7 of Diamonds. 5 of Diamonds. 5 of Clubs. Jack of Hearts. King of Hearts. King of Spades. ]

Returns slots that can be removed as integer ArrayList: [ ]
[ 10 of Spades. 7 of Diamonds. 5 of Diamonds. 5 of Clubs. Jack of Hearts. King of Hearts. King of Spades. ]
STALEMATE
```

Test iterates through each of the possible moves and removes them until it correctly reaches a stalemate as expected.

BoardTest Class - Test Six

Test displayBoard method again but with testBOARD cards in current state as left by test five.

Result

```
Current state of board after test

-----------Board-------------
Slot 7  -  King of Spades
Slot 6  -  King of Hearts
Slot 5  -  Jack of Hearts
Slot 4  -  5 of Clubs
Slot 3  -  5 of Diamonds
Slot 2  -  7 of Diamonds
Slot 1  -  10 of Spades
-----------------------------
0 cards left in deck.
-----------------------------
```

Test board shows correct arrangement of cards as displayed in end of test five with no cards left remaining in deck.

BoardTest Class - Test Seven

Test whether slot 7 and non-existent slot 8 can be removed from the testBOARD as shown in test six. Try Catch exception handling to be used in testing for attempted removal of card objects outside of scope.

Result

```
Test to replace slots 7 & 8 in tstBRD ArrayList.
IndexOutOfBoundsException error caught.
8 is out of range.

Changes made (if any): [ 10 of Spades. 7 of Diamonds. 5 of Diamonds. 5 of Clubs. Jack of Hearts. King of Hearts. King of Spades. ]
```

Test attempted to remove slot entry 8 first as replaceCard method first sorts out entries highest to lowest. 8 is out of scope as only slot 1-7 are available as highlighted in test six and an IndexOutOfBoundsException error was caught stating 8 was out of range. No changes were made to testBOARD as 8 crashed before 7 could be removed.

BoardTest Class - Test Eight

Test whether non-existent slot 0 and slot 1 can be removed from the testBOARD as shown in test six. Try Catch exception handling to be used in testing for attempted removal of card objects outside of scope.

Result

```
Test to replace slots 0 and 1 in tstBRD ArrayList.
IndexOutOfBoundsException error caught.
0 is out of range.

Changes made (if any): [ 7 of Diamonds. 5 of Diamonds. 5 of Clubs. Jack of Hearts. King of Hearts. King of Spades. ]

If changes made, must ensure accidental user selection is limited appropriately to prevent any unwanted crashes or replacements.
```

Test attempted to remove slot entry 1 first as replaceCard method first sorts out entries highest to lowest. Slot entry for 1 is found (10 of Spades) and removed, then slot entry 0 is attempted and Try Catch exception handling is triggered stating ‘IndexOutOfBoundsException error caught’. Changes have been made to testBOARD as slot 1 has been removed. Proper input validation will be necessary to ensure only correct input can be taken.

To prevent slots out of range being passed into the replaceCard method, appropriate user input validation must be setup to limit the range of input to only those current slots displayed to the user on each turn. Any input outside the range of the board length should be rejected and asked for again until the requirement has been satisfied to ensure security of the code and that no errors or bugs can occur.

ElevensGameTest Class

ElevensGameTest Class - Test

Extensively stress test the program by automatically simulating firstly 10 games, then 100 games, 1000 games, 10,000 games and finally 100,000 games. This will ensure that the game is reliable and is robust against potential errors and bugs. For each individual game a new deck is dealt at random and solved by the program.

The test uses a for loop with the range set to the different amounts outlined above. Within the for loop is a while loop that simulates through the process of the game removing cards where possible until it either ends in a Win or a Stalemate allowing the for loop to increment to the next game. The process repeats through the for loop until the range is met.

2 counters are set globally to count the number of Wins and Losses for each game played and are displayed to the user when the games have been completed. The time taken for each range of games in nanoseconds is also recorded and presented.

Results

```
C:\Users\niall\IdeaProjects\B00787301_COM498_Assignment2\ou
Test 10 Elevens games.
Total games played were: 10
Total wins were: 1
Total losses were: 9
Time taken to play games: 2766800 nanoseconds

Test 100 Elevens games.
Total games played were: 100
Total wins were: 12
Total losses were: 88
Time taken to play games: 14824500 nanoseconds

Test 1000 Elevens games.
Total games played were: 1000
Total wins were: 108
Total losses were: 892
Time taken to play games: 64799200 nanoseconds

Test 10000 Elevens games.
Total games played were: 10000
Total wins were: 1136
Total losses were: 8864
Time taken to play games: 349172800 nanoseconds

Test 100000 Elevens games.
Total games played were: 100000
Total wins were: 10752
Total losses were: 89248
Time taken to play games: 1420481900 nanoseconds

Process finished with exit code 0
```

Test results appear promising indicating with the likelihood of winning in most game ranges being a little higher than 1 in 10 odds which appears to be correct as stated in the assignment brief. I repeated the test several times getting results within a similar range indicating that the program is running correctly and is robust against unwanted errors and bugs.

Annex B

Usability/Validation Testing

Test One

Test if program starts and a game of Elevens can be started.

Results

```
-------------------------Welcome to Elevens-------------------------
Rules
Select two cards that add to eleven or a Jack, Queen, King combination.
If you can remove all the cards from the board and the deck you Win!

Enter 'p' to play or 'd' for demonstration mode: p

-----------Board-------------
Slot 9  -  Jack of Spades
Slot 8  -  8 of Clubs
Slot 7  -  10 of Clubs
Slot 6  -  6 of Hearts
Slot 5  -  King of Spades
Slot 4  -  9 of Diamonds
Slot 3  -  9 of Clubs
Slot 2  -  6 of Spades
Slot 1  -  3 of Spades
---------------------------
43 cards left in deck.
---------------------------

Please Enter a Slot Number for any Card (or any letter key for a hint):
```

Program started and required me to enter ‘p’ to start a game which is correct.

Test Two

Test when selecting two cards from the board if they are replaced when they equal eleven.

Results

```
Please Enter a Slot Number for any Card (or any letter key for a hint): 1

Please select a second card to add to Eleven
Please Enter a Slot Number for any Card (or any letter key for a hint): 8

----Removed: 3 of Spades & 8 of Clubs----

-----------Board-------------
Slot 9  -  Jack of Spades
Slot 8  -  Ace of Diamonds
Slot 7  -  10 of Clubs
Slot 6  -  6 of Hearts
Slot 5  -  King of Spades
Slot 4  -  9 of Diamonds
Slot 3  -  9 of Clubs
Slot 2  -  6 of Spades
Slot 1  -  Ace of Spades
---------------------------
41 cards left in deck.
---------------------------

Please Enter a Slot Number for any Card (or any letter key for a hint):
```

Two selected cards (6 of Clubs and 5 of Diamond) as can be seen from the previous test equal eleven and were removed and replaced correctly. Program detected the first card’s rank was lower than a Jack (so the user was attempting a combination of cards that equal Eleven and not a Jack, Queen, King combination) so only asked for one second card which is correct.

Test Three

Test when selecting two cards from the board if they are replaced when they don't equal eleven.

Result

```
Please Enter a Slot Number for any Card (or any letter key for a hint): 1

Please select a second card to add to Eleven
Please Enter a Slot Number for any Card (or any letter key for a hint): 4

Ace of Spades & 9 of Diamonds selected.
----They don't add to Eleven. Please try again----

----------Board------------
Slot 9  -  Jack of Spades
Slot 8  -  Ace of Diamonds
Slot 7  -  10 of Clubs
Slot 6  -  6 of Hearts
Slot 5  -  King of Spades
Slot 4  -  9 of Diamonds
Slot 3  -  9 of Clubs
Slot 2  -  6 of Spades
Slot 1  -  Ace of Spades
---------------------------
41 cards left in deck.
---------------------------
```

Two selected cards (4 of Clubs and 9 of Clubs) don't equal eleven and were not replaced so the board remained the same which is correct.

Test Four

Test if hint functionality works when a letter is entered instead of a slot number

Result

```
Please Enter a Slot Number for any Card (or any letter key for a hint): h
Hint: Ace of Spades & 10 of Clubs

Please Enter a slot number for any card:
```

When letter key 'h' was entered, while loop validation registers that the entry is not an integer and correctly prompts a hint to the user displaying two possible cards that equal eleven that are currently on the board. It then requires the user to enter a slot number again which is correct.

Test Five

Test if card slot that is out of range of the board is accepted using test parameters 0 and 10 which are lower and greater than the range of the board.

Result

```
Please Enter a slot number for any card: 0
Invalid Entry. Not within range of slots in Board.

Please Enter a slot number for any card (or any letter key for a hint): 10
Invalid Entry. Not within range of slots in Board.

Please Enter a slot number for any card (or any letter key for a hint):
```

When slot 0 was entered, the application states it is an invalid entry and not within the range of slots in the board which is correct as it ranges from slots 1-9 as can be seen in the screenshot for test 3. It then requires the user to retry inputting a slot number.

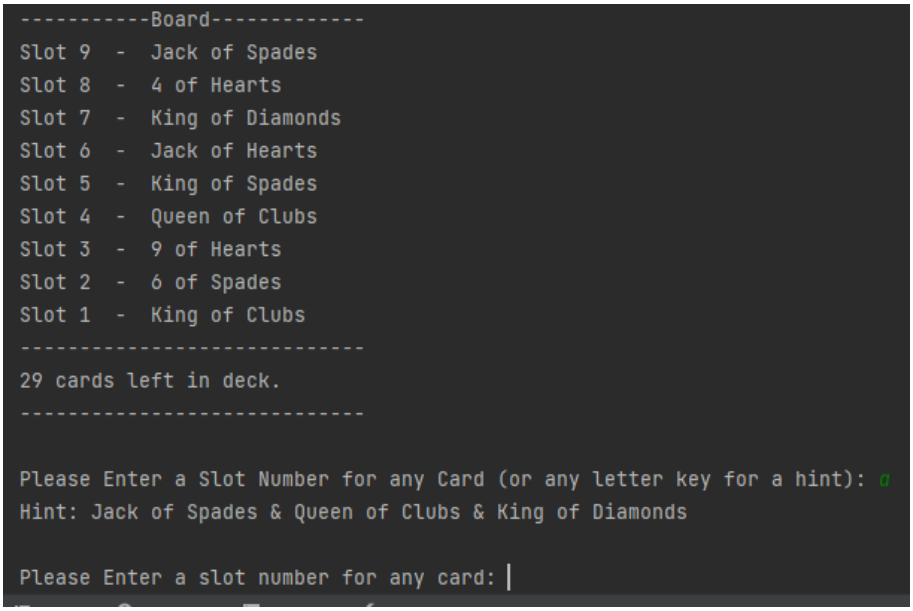
Then when slot 10 was entered, the application again states it is an invalid entry and not within the range of slots in the board. It then requires the user to retry inputting a slot number again. This process is repeated until the user enters a slot from the range of the board.

The results of this test show that the program's input validation is working correctly and that the bug detected from the BoardTest class for the replaceCards method has been removed as an input range outside of the scope of the board will no longer be accepted.

Test Six

Test if hint functionality works when a Jack, Queen, King combination is found in the board.

Result



When letter key 'a' was entered, while loop validation registers that the entry is not an integer and correctly prompts a hint to the user displaying three possible cards that equal a Jack, Queen, King combination that are currently on the board. It then correctly requires the user to enter a slot number again.

Test Seven

Test when selecting three cards from the board if they are replaced when they equal a Jack, Queen King combination.

Result

```
----------Board------------
Slot 9  -  Jack of Spades
Slot 8  -  4 of Hearts
Slot 7  -  King of Diamonds
Slot 6  -  Jack of Hearts
Slot 5  -  King of Spades
Slot 4  -  Queen of Clubs
Slot 3  -  9 of Hearts
Slot 2  -  6 of Spades
Slot 1  -  King of Clubs
---------------------------
29 cards left in deck.
---------------------------

Please Enter a Slot Number for any Card (or any letter key for a hint): 4

Please select two more different picture cards.
Please Enter a Slot Number for any Card (or any letter key for a hint): 5
Please Enter a Slot Number for any Card (or any letter key for a hint): 9

----Removed: Queen of Clubs, King of Spades, Jack of Spades----
```

Three selected cards (Queen of Clubs, King of Spades and Jack of Spades) were removed and replaced correctly. Program detected the first card's rank was a Jack or higher (so the user was attempting a combination of cards to equal a Jack, Queen, King combination and not to eleven) so asked for two more picture cards which is correct.

Test Eight

Test when selecting three cards from the board if they are replaced when they don't equal a Jack, Queen King combination.

Result

```
Please Enter a Slot Number for any Card (or any letter key for a hint): 1

Please select two more different picture cards.
Please Enter a Slot Number for any Card (or any letter key for a hint): 6
Please Enter a Slot Number for any Card (or any letter key for a hint): 3

King of Clubs, Jack of Hearts, 9 of Hearts selected.
----Not a Jack, Queen and King. Please try again----

----------Board------------
Slot 9  -  Jack of Clubs
Slot 8  -  4 of Hearts
Slot 7  -  King of Diamonds
Slot 6  -  Jack of Hearts
Slot 5  -  6 of Clubs
Slot 4  -  2 of Diamonds
Slot 3  -  9 of Hearts
Slot 2  -  6 of Spades
Slot 1  -  King of Clubs
---------------------------
26 cards left in deck.
---------------------------
```

Three selected cards (King of Clubs, Jack of Hearts and 9 of Hearts) don't equal a Jack, Queen, King combination and were not replaced so the board remained the same which is correct.

Test Nine

Test if game of Elevens can be lost and ends in stalemate.

Result

```
Please Enter a slot number for any card: 6

Please select a second card to add to Eleven
Please Enter a Slot Number for any Card (or any letter key for a hint): 9

----Removed: 4 of Clubs & 7 of Spades----

----------Board-------------
Slot 9  -  7 of Clubs
Slot 8  -  Queen of Spades
Slot 7  -  5 of Clubs
Slot 6  -  Queen of Hearts
Slot 5  -  8 of Diamonds
Slot 4  -  Queen of Diamonds
Slot 3  -  Ace of Hearts
Slot 2  -  Jack of Hearts
Slot 1  -  2 of Diamonds
----------------------------
39 cards left in deck.
----------------------------

----STALEMATE----

Enter 'r' to replay your moves, 'p' to play again or any other key to exit:
```

Game correctly ended in stalemate when no more moves were able to be played. User prompted with a winning message and asked to select an option from the menu which is correct.

Test Ten

Test if a new game of Elevens can be played from within the application.

Result

```
----STALEMATE----

Enter 'r' to replay your moves, 'p' to play again or any other key to exit: p

-------------------------Welcome to Elevens-------------------------
Rules
Select two cards that add to eleven or a Jack, Queen, King combination.
If you can remove all the cards from the board and the deck you Win!

Enter 'p' to play or 'd' for demonstration mode: p

----------Board-------------
Slot 9  -  King of Diamonds
Slot 8  -  Ace of Diamonds
Slot 7  -  2 of Spades
Slot 6  -  6 of Diamonds
Slot 5  -  Queen of Hearts
Slot 4  -  7 of Clubs
Slot 3  -  Ace of Clubs
Slot 2  -  Jack of Clubs
Slot 1  -  Jack of Hearts
----------------------------
43 cards left in deck.
----------------------------

Please Enter a Slot Number for any Card (or any letter key for a hint):
```

Game allows user to play again by entering ‘p’ which deals a new deck of cards and starts a new game again which is correct.

Test Eleven

Test if game of Elevens can be won.

Result

```
----------Board-------------
Slot 7  -  8 of Clubs
Slot 6  -  Jack of Clubs
Slot 5  -  King of Hearts
Slot 4  -  Queen of Diamonds
Slot 3  -  2 of Clubs
Slot 2  -  3 of Clubs
Slot 1  -  9 of Spades
---------------------------
0 cards left in deck.
---------------------------

Please Enter a Slot Number for any Card (or any letter key for a hint): 4

Please select two more different picture cards.
Please Enter a Slot Number for any Card (or any letter key for a hint): 5
Please Enter a Slot Number for any Card (or any letter key for a hint): 6

----Removed: Queen of Diamonds, King of Hearts, Jack of Clubs----

----------Board-------------
Slot 4  -  8 of Clubs
Slot 3  -  2 of Clubs
Slot 2  -  3 of Clubs
Slot 1  -  9 of Spades
---------------------------
0 cards left in deck.
---------------------------

Please Enter a Slot Number for any Card (or any letter key for a hint): 2

Please select a second card to add to Eleven
Please Enter a Slot Number for any Card (or any letter key for a hint): 5
Invalid Entry. Not within range of slots in Board.

Please Enter a slot number for any card (or any letter key for a hint): 4

----Removed: 3 of Clubs & 8 of Clubs----

----------Board-------------
Slot 2  -  2 of Clubs
Slot 1  -  9 of Spades
---------------------------
0 cards left in deck.
---------------------------

Please Enter a Slot Number for any Card (or any letter key for a hint): 1

Please select a second card to add to Eleven
Please Enter a Slot Number for any Card (or any letter key for a hint): 2

----Removed: 9 of Spades & 2 of Clubs----

----Congratulations! You've Won!----

Enter 'r' to replay your moves, 'p' to play again or any other key to exit:
```

Game resulted in win when a winning deck is shuffled and dealt out to the user. Attempted to select a card slot (5) outside of the range of current board for the second last move. Prompted that it was an invalid entry as the range of the board were slots 1-4 which is correct. When game was won, user prompted with a winning message and asked to select an option from the menu which is correct.

Test Twelve

Test if replay functionality works correctly and if user is required to increment through steps.

Result

```
Slot 9  -  8 of Diamonds
Slot 8  -  Queen of Spades
Slot 7  -  Ace of Hearts
Slot 6  -  5 of Hearts
Slot 5  -  2 of Spades
Slot 4  -  9 of Diamonds
Slot 3  -  Queen of Clubs
Slot 2  -  Queen of Hearts
Slot 1  -  Jack of Clubs
----------------------------
37 cards left in deck.
----------------------------

Please Enter a Slot Number for any Card (or any letter key for a hint): 5

Please select a second card to add to Eleven
Please Enter a Slot Number for any Card (or any letter key for a hint): 4

----Removed: 2 of Spades & 9 of Diamonds----

----------Board------------
Slot 9  -  8 of Diamonds
Slot 8  -  Queen of Spades
Slot 7  -  Ace of Hearts
Slot 6  -  5 of Hearts
Slot 5  -  Queen of Diamonds
Slot 4  -  Jack of Spades
Slot 3  -  Queen of Clubs
Slot 2  -  Queen of Hearts
Slot 1  -  Jack of Clubs
----------------------------
35 cards left in deck.
----------------------------

----STALEMATE----

Enter 'r' to replay your moves, 'p' to play again or any other key to exit: r
Press 'Enter' key to increment through replay moves.
Board: [ 5 of Clubs. Queen of Hearts. Queen of Clubs. 6 of Hearts. 4 of Diamonds. 5 of Hearts. Ace of Hearts. Queen of Spades. 5 of Spades. ]
Removed: 5 of Spades
Removed: 6 of Hearts

Board: [ 5 of Clubs. Queen of Hearts. Queen of Clubs. 7 of Spades. 4 of Diamonds. 5 of Hearts. Ace of Hearts. Queen of Spades. 8 of Diamonds. ]
Removed: 4 of Diamonds
Removed: 7 of Spades

Board: [ 5 of Clubs. Queen of Hearts. Queen of Clubs. 9 of Diamonds. 6 of Diamonds. 5 of Hearts. Ace of Hearts. Queen of Spades. 8 of Diamonds. ]
Removed: 6 of Diamonds
Removed: 5 of Clubs

Board: [ Jack of Clubs. Queen of Hearts. Queen of Clubs. 9 of Diamonds. 2 of Spades. 5 of Hearts. Ace of Hearts. Queen of Spades. 8 of Diamonds. ]
Removed: 2 of Spades
Removed: 9 of Diamonds

No moves left to replay.
Enter 'r' to replay your moves, 'p' to play again or any other key to exit:
```

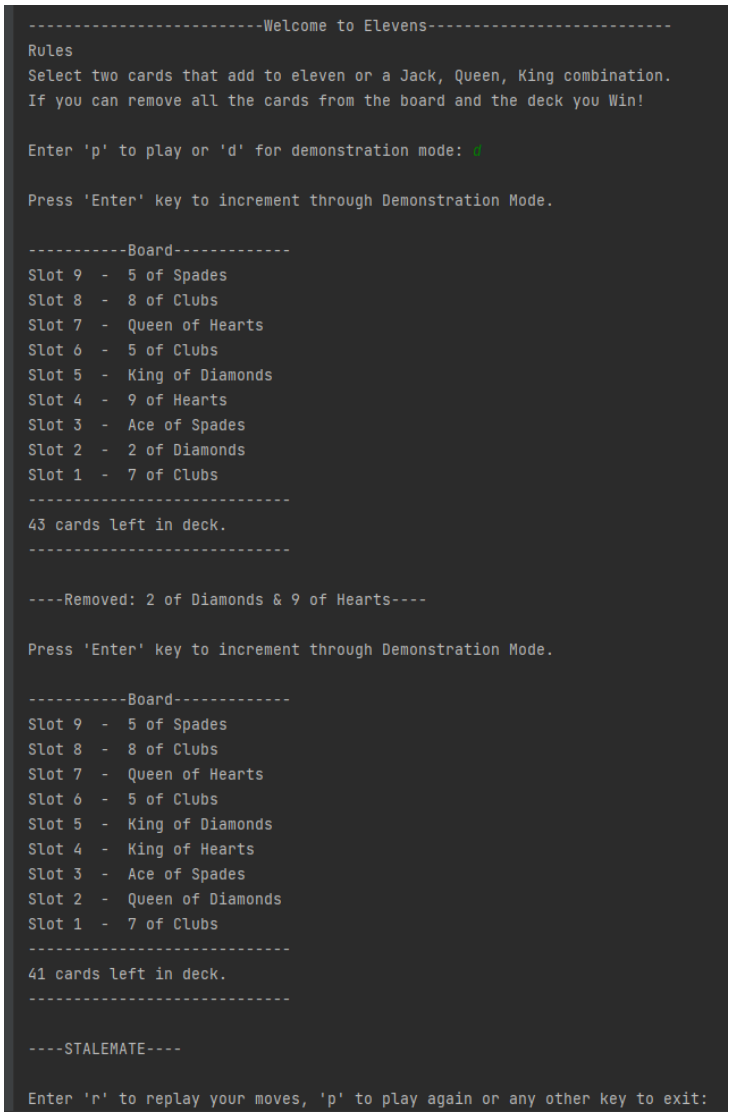
The replay functionality works correctly, recording the board and removed cards at each step of the game and allowing the user to press ‘Enter’ to increment through the replay steps of the game.

Test Thirteen

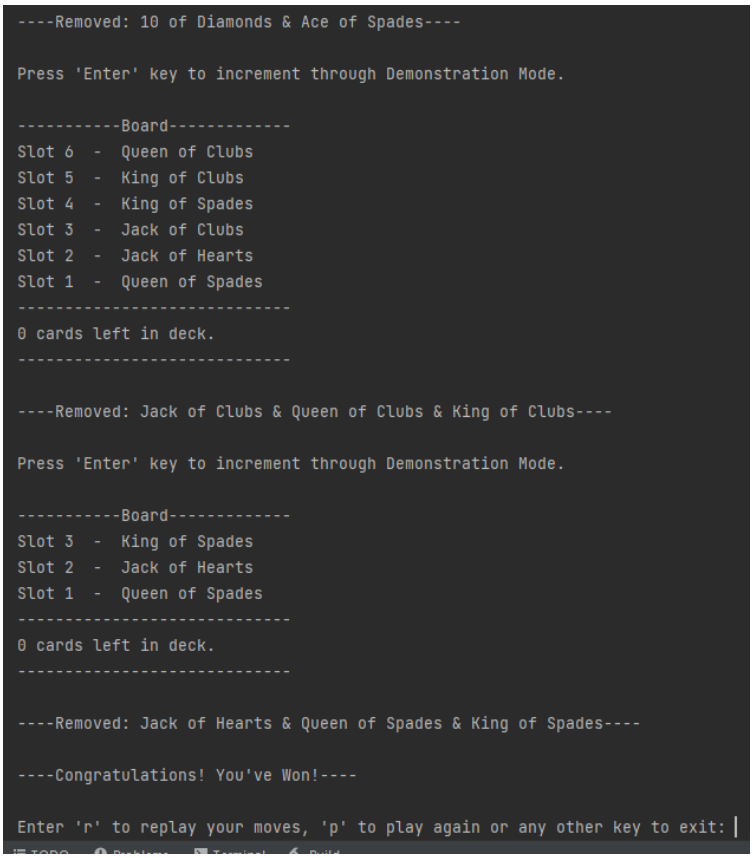
Test if demonstration mode works correctly allowing the user to increment through the steps of a simulated Elevens game.

Result

Demonstration test of Winning Game



Demonstration test of Losing Game



Demonstration mode works correctly, allowing the user by pressing ‘Enter’ to increment through the steps of a simulated Elevens game and each of the steps taken. Above are two screenshots of demonstration mode in the event of a Winning game and a Stalemate. Once the demonstration game had completed, it prompted the user with the menu which is correct.