# Text Description automatically generated with medium confidenceTesting Choices

Tests for all the main classes have been created. These are:

* CardDeckTest.java
* CardTest.java
* PackTest.java
* PlayerTest.java

The tests were written using JUnit version 4.13.1. The tests were made to ensure that each of the classes function correctly.

## CardDeckTest.java

The CardDeck class is used for storing the card objects in a queue data structure, this is used throughout the program and therefore testing of its methods is essential. The main methods that are used in the CardDeck class that need to be tested are the drawing of cards and the placing of cards from and onto a deck.

### void testToString()

This testing method was made to ensure that the ToString override method in the CardDeck class is outputting valid information.

### void testTakeCardFromTop()

This class was made to test that the correct values were being taken from the top of the queue. This was done by:

1. Initialising a new deck
2. Recording the values in the deck
3. Run the method takeCardFromTop
4. Assert the values inside of the deck are now as expected and not the same as the value before the method was executed

### void testPlaceCardOnBottom()

Like the previous method, this test is used to ensure that the cards are properly being places on the deck. This was done by:

1. Initialising a new deck
2. Recording the values in the deck
3. Run the method placeCardOnBottom with a new card as the parameter
4. Assert that the values in the deck are now as expected and not the same as the value before the method was executed

## CardTest.java

The Card class is used to store the value of the card. In this class there is just a simple getter and setter functions.

### void testGetValue()

This method is to test that the getter function inside of the Card class is working as intended

### void testToString()

This method is to test that the value of the ToString override is outputting the correct values

## PackTest.java

This testing class is used to ensure that a text file can be read into the program correctly and that the correct value is output when requested.

### void testToString()

This method is used to test the override ToString method. To initialize the pack object that I used for testing a method called initFile() is used to create a new text file and populate it with numbers, simulating a pack.txt file. This is then read into the program using the constructor in the pack class.

## PlayerTest.java

This class is used to test all the methods to within the Player class, this is very important as these methods are used throughout the program. For this testing class a new player object is initialized at the beginning of each method, this is then used to test the functions with respect to the deck they were given when initialized.

### Player createPlayer()

This is a method that is used in many of the test methods to construct a new player object and then populate the players decks with the required cards for testing

### void testDrawCardFromDeck()

This test is used to ensure that the correct card is being drawn from the players left deck and then placed into their hand. This is done by:

1. Initialising a new player object with the queue of cards
2. Recording the value before any changes have been made
3. Executing the drawCardFromDeck() method
4. Recording the expected and actual output
5. Assertion test to check these outputs

### void testDiscardCardToDeck()

This test is used to ensure that the correct card is discarded from the players deck and is then placed into their right deck, this is done by:

1. Initialising a new player object with the queue of cards
2. Recording the value before any changes have been made
3. Executing the discardCardToDeck() method
4. Recording the expected and actual output
5. Assertion test to check these outputs

### void testPlayGo()

This test is to simulate the players’ hand after they have played a go and ensures that the output is as expected.

1. Initialising a new player object with the queue of cards
2. Simulate a round of play by executing the playGo() method
3. Record the expected output of the deck
4. Assertion test this against the actual output

### void testToString()

This method is used to test the ToString override in the Player class, this is done by:

1. Initialising a new player object with the queue of cards
2. Recording the expected output
3. Using an assertion test to check the expected output to the output of the ToString method

### void testAllSameCards()

This method is used to check the getPlayerWon method in the Player class is working correctly, this is done by:

1. Initialising a new player object with the queue of cards that are all the same
2. Checking the value of the getPlayerWon method which will return true if all the players card values are the same

### void testLogOutput() & void testLoseOutput() & void testWinOutput()

This method is used to check that the correct messages are being outputted to the text file, this is done by

* Initialising a new player object
* Executing the drawCardFromDeck method, which will output to the text file what happens
* Reading the text file and checking that value against what is expected

### void testCurrentHandOutput()

This method is used to output the current hand, this I used throughout the player class when outputting to the log file, this is done by

1. Initialising a new player object
2. Testing the output of the getOrderedCards method with an assertion test

### void testSetPlayerWon()

This method is used to check that the playerWon method within the player class works when the deck satisfies the necessary requirements, this is done by:

1. Creating 3 temp decks that will be used when initialising a new player object
2. The players deck will contain four 1s and one 2, this is to ensure that the player hasn’t won the game on initialization
3. Initialising a new player object with the previously created decks
4. Execute the discardCardToDeck method to discard the single 2 card in the deck preventing it from being a winning hand
5. Execute the drawCardFromDeck method to make the players deck contain one 1s
6. Record if the player has won
7. Execute the setPlayerWon() method
8. Use an assertion test to check that the two values are different