Documentation

**Feedback**:

When looking at both the test and main file, there is a decent amount of code implemented for creating test cases. In the test file, test cases were added with the correct code insertion block. They have clearly defined the project and build on top of the provided source information. One thing to add onto the test file is to insert any helpful comments for each test method. This will make it a bit more clearer as to which test method belongs to the corresponding code block function in the main class. They identified which function I should write the input space testing, and functionality testing. Adding a helper test function at the end would help me try out a coverage to determine strength and performance for a test and see what might need to be improved. Overall, the design of the test and main class is efficient for testing. A total of 7 methods are inserted into the main class, for which each specified function is then tested with the right base coverage. Another object I encountered is some of the given test cases with the comparable methods, it was difficult to write a test case that passes. For example in the input space testing section, an assert null pointer exception has been thrown, or an assert error is encountered.In one of the test cases, an error was encountered, it told me that a file named SavedHand.txt is un located, so I had to modify a few test cases to successfully pass. After looking for possible solutions online as to how to overcome this problem, some of the test cases, which initially had an error, were settled. This might have been a little better to provide some comments for each test case,and provide a more sophisticated understanding of the specified problem.

**Syntax Coverage:**

Terminal Symbol Coverage(TSC), for this using the provided main class, the Terminal Replacement contains each terminal symbol(t) for a specified grammar G.

For Mutation Operator Coverage(MOC): TR contains the requirement to create a mutated string from that production.

|  |
| --- |
| @Test //Mutation Coverage Private **voidmutantValid**(){ //test method mutant validation based on the number of cards obtained in hand. //Int GetInputNumber(); cd.GetInputNumber(); numbersList = cd.DetermineRandomHand(0); Int numbersList = 0; Int expected = 0; assertArrayequals(numbersList, actual: 0); Boolean mutationValid; final Boolean b = true; //This indicates that if the mutation is still active //and corresponds to the #cards in the RandomHand, will return true. assertTrue(condition: true); } |

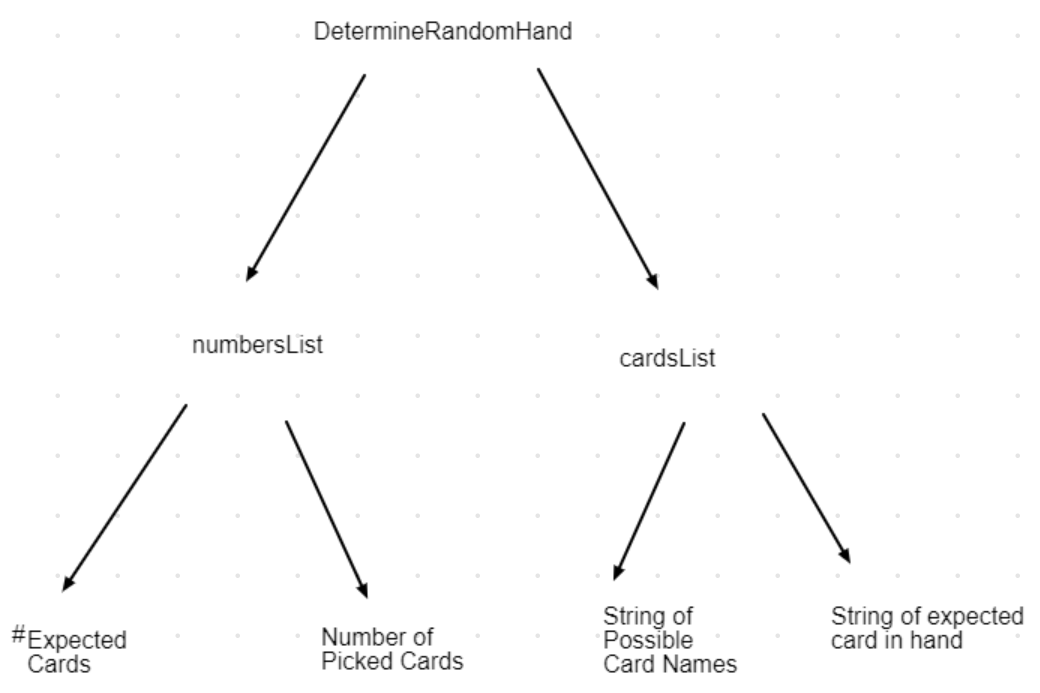
Let’s say that we want to add two cards together, one hand has x # of cards and the other hand contains y # of cards.Using mutation to receive the addition of both hands. After executing each test case, the mutant is killed off to preserve memory.

|  |
| --- |
| @Test //Mutation Operator Coverage Private int ExpectedResult; Private int ActualResult; Private int Util; //Gets the addition of two numbers declared. cd.GetInputNumber(); cd.UseJokers(false); numbersList = cd.DetermineRandomHand(cards: 5, numOfCards:5); //5 cards in each hand. Boolean mutationValid; final boolean b = true; //Arithmetic Operation, where we add the sum of the two numbers 5 and 5, from cards and //numCards. Var i = numberList + numbersList; //the expected addition of one number from one hand with //another number in another hand. Int numbersList = (5); Int numberList = (5); Int result = Util(numberList, numbersList); //Kill of mutation Int expected = 10; assertEquals(expected: 10, actual: 10); assertTrue(condition: true); //Catch unexpected values that might get retrieved from a random hand. Assert.assertNotEquals(unexpected: 10, actual: 5); } |

**Graph Coverage:**

For Graph Coverage, determine the number of nodes, edge coverage, and paths in the random hand of the card deck. Plan out the path for each integer and string combination.

An example of a simple graph for the input number and number of cards. Find a sample test path in this case, determine a possible connection for a random hand with a set number of integer values and character strings.



|  |  |  |
| --- | --- | --- |
| x | numbersList | cardsList |
| Type | Integer values | String lists |
| Occurrences | 52 possible cards out of a deck.  4 suites, 13 cards each.  2 joker cards.  9 number cards from 2-10.  Jack, Queen, King & Ace. | 52 possible string characters, including 2 joker cards. |
| Sample Values, Characters | {13}, {52}, {0}, {2}, {4} | {“Ace”, “Jack”,”Queen”,”King”, “Club”, ‘Diamonds”, “Hearts”,”Spades”, “Joker”} |

Simple Path to test graph coverage

|  |
| --- |
| @Test //Graph Coverage //Testing numbersList integer type cd.GetInputNumber(); cd.UseJokers(false); Int [] numbersList = cd.DetermineRandomHand(cards: 5, numCards: 5); //This covers a possible node, edge for the number of cards in each random hand. Int []expectedValues = {10}; Int actual = 10; assertEquals(expected: 10, actual: 10); } |

Simple Path for graph coverage test

|  |
| --- |
| @Test //Graph Coverage //Testing cardsList StringList cd.GetInputNumber(); cd.DetermineRandomHand(5, 5);//Randomly select a random hand with different or similar card //names. cd.UserJokers(false); String[] cardsList = {"Ace of Clubs", "2 of Diamonds", "3 of Hearts", "Queen of Spades", "King of Spades"}; //Selected random cards from a hand. String[]expectedValues = {"Ace of Clubs", "2 of Diamonds", "3 of Hearts", "Queen of Spades", "King of Spades"}; String[]actual = {"Ace of Clubs", "2 of Diamonds", "3 of Hearts", "Queen of Spades", "King of Spades"}; //Check for incorrect spelling or card not found. Assert.NotEquals(unexpected: "Ace of Jokers", cardsList); outCont = System.out; }//Graph Coverage |

References

“Junit - A Simple Program to Test Addition of Two Number and Project Setup - Unit Test Framework.” *Www.catchmycity.com*, Catchmycity.com, 19 Feb. 2015, www.catchmycity.com/tutorial/unit-test-framework-junit-a-simple-program-to-test-addition-of-two-number-and-project-setup\_80.

“Standard Deck Playing Card Games: Wiki.” *BoardGameGeek*, Geekdo, BoardGameGeek, boardgamegeek.com/wiki/page/standard\_deck\_playing\_card\_games.