|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
| rank must be a integer, either 2,3,4,5,6,7,8,9,10, 11,12,13,14 | 1. rank is either 2,3,4,5,6,7,8,9,10, Jack, Queen, King or Ace | 2. rank is anything other than 2,3,4,5,6,7,8,9,10, 11,12,13,14 |
| suit must be a String either Clubs, Spades, Hearts, or Diamonds | 3. rank is either Clubs, Spades, Hearts, or Diamonds | 4. rank is anything other than Clubs, Spades, Hearts, or Diamonds |
|  |  |  |
|  |  |  |
|  |  |  |

Card

Test Case 1: Testing **instantiation of Card** object with legal values and Illegal values (By extension **testing gets and sets of suit and rank**), and toString()

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
| rank = 10, suit = Clubs | 1,3 |
| rank = 10, suit = Turkey | 1, 4 |
| rank = 0, suit = Diamonds | 2, 3 |
| rank = 0, suit = Turkey | 2, 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Purpose** | **Changed Object State** | **Expected result** |
| Card card1 = new Card(“10”, “Clubs”); | Test proper instantiation of Card, with valid values for suit and rank | rank=”10”  suit=“Clubs” | New Card Created |
| Card card2 = new Card(“10”, “Turkey”) | Test improper instantiation of Card, with valid value for rank, invalid value for Suit | rank=”10”;  suit=””; | IllegalArgument Exception is thrown |
| Card card3 = new Card(“0”, “Diamonds”) | Test improper instantiation of Card, with invalid value for rank, valid value for Suit | rank=””;  suit=”Diamonds”; | IllegalArgument Exception is thrown |
| Card card4 = new Card(“0”, “turkey”) | Test improper instantiation of Card, with invalid value for rank, invalid value for Suit | rank=””;  suit=””; | IllegalArgument Exception is thrown |
| card1.toString() | Test toString() | rank=”10”  suit=“Clubs” | 10 of clubs |

\*Constructors use gets and sets for suit and rank

Deck

Test Case 2: Testing **instantiation of Deck** object, **testing size** and **testing order of the cards**.

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
| The Deck constructor must create the full Deck of Cards | 1.The Deck constructor creates all 52 cards | 2. The Deck creates 51 cards  3.The Deck creates 53 cards |
| The Deck constructor must create 13 cards of each suit | 4.The Deck constructor creates 13 cards of each suit | 5. The Deck constructor creates 14 cards of each suit (Boundary)  6.The Deck creates 12 cards of each suit (Boundary)  7.The Deck does not have all 4 suits: Clubs, Spades, Hearts, or Diamonds (Boundary) |
| The Deck must be created in the correct order, with all cards being created for a single suit before moving to the next suit  (To test that all cards are in the deck, no duplicates ect) | 8. The Deck constructor creates all 52 cards in the order : Creating all cards for single suit, then moving to next suit | 9. The Deck constructor creates the deck in any other order |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
| 52 | 1 |
| 51 | 2 |
| 53 | 3 |
| {Clubs, Spades, Hearts, Diamonds } | 7 |
| 14 | 5 |
| 12 | 6 |
| 13 | 4 |
| <<The entire deck, unshuffled>> | 8, 9 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Purpose** | **Changed Object State** | **Expected result** |
| Deck myDeck = new Deck() | Test instantiation of a new Deck | allCards = <<52 new cards>> | A new Deck is created |
| myDeck.size() | Test the size of the new Deck |  | 52 |
| myDeck.size() == 53 | Boundary test of deck size (upper) |  | False |
| myDeck.size() == 51 | Boundary test of deck size (lower) |  | False |

Test Case 3:

Testing **shuffle()** and **testing order of the cards**.

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
| the deck must not be in the same order as it was unshuffled | 1.Deck is properly shuffled, and cards are not in the same position | 2.cards are in the same position  3. cards are in the same order |
| the deck must not lose or gain any cards | 4. Deck size is still 52 | 5. Deck size is less than 52 (51, lower boundary)  6. Deck size greater than 52 (53, upper boundary ) |
|  |  |  |
|  |  |  |
|  |  |  |

Using Deck myDeck = new Deck();

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
| 52 | 4 |
| 51 | 5 |
| 53 | 6 |
| <<The entire deck, shuffled>> | 1 |
| <<The entire deck, unshuffled>> | 2,3 |

Deck

Test Case 5: Testing **deal()**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
| deal() must return the first card from the queue | 1.The first card is returned from the queue | 2. The last card is returned from the queue  3.Any other card is returned from the queue |
| deal() must return a Card | 4. Returns a Card | 5. Nothing is returned (null)  6. Any other data type is returned (Use Object) |
|  |  |  |

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
| <<first card from the deck>> | 1 |
| <<last card from the deck>> | 2 |
| <<random card that isn’t the first card from the deck>> | 3 |
| null | 5 |
| Object ob | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Purpose** | **Changed Object State** | **Expected result** |
| Deck myDeck = new Deck() | Test instantiation of a new Deck | allCards = <<52 new cards>> | A new Deck is created |
| myDeck.deal() | Test that deal gets and returns the first card of the queue |  | 2 of clubs |
| myDeck.getAllCards.peek() | Test what the first card of the queue is without removing it |  | 2 of clubs |
|  |  |  |  |

War

Test Case 6: Testing start()

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
| start() must deal exactly half the deck to each player (26 cards each) | 1.each hand has 26 cards | 2.A hand has less than 26 cards (25, lower boundary )  3.A hand has more than 26 cards (27, upper boundary) |
|  |  |  |
|  |  |  |

myDeck = new Deck();

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
| 26 | 1 |
| 25 | 2 |
| 27 | 3 |
|  |  |

Test Case 6: Testing play()

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
| Play must take the first card off each hand. | 1.Play takes the first card off each hand and compares them | 2. Play takes the last card from a hand  3. Play takes any other card from a hand |
| If the a hand is empty, the game is over and winner is set | 4. Winner is set to either p1 or p2 | 5.Winner is an empty String  6.Winner is null |
|  |  |  |

Using Deck myDeck = new Deck();

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
| <<first card from each hand>> | 1 |
| <<last card from the deck>> | 2 |
| <<random card that isn’t the first card from the deck>> | 3 |
| <<p1>> or <<p2>> | 4 |
| “” | 5 |
| null | 6 |