**Test Cases for Deck Class**

**Test Case 1: createDeck() : void**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * The method must set the unshuffledDeck variable to a list of 52 cards | 1. 52 | 1. Everything else |
| * The "cards" in the unshuffledDeck queue are instances of the Card class | 1. True | 1. False |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1 | unshuffledDeck.size() = 52 |
| 2 | unshuffledDeck.size() = 51 |
| 3 | unshuffledDeck.peek() instanceof Card = true |
| 4 | unshuffledDeck.peek() instanceof Card = false |

**Test Case 2: shuffle() : void**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * The unshuffled deck must be empty at the end | 1. unshuffledDeck is empty | 1. unshuffledDeck is not empty |
| * Before shuffling the deck should have 52 cards in it | 1. 52 | 1. Numbers other than 52 |
| * Shuffle only once | 1. shuffle once | 1. shuffle more than once |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1,2 | testDeck: [AC, KC, QC, JC, 10C, …] |
| 3 | unshuffled deck with a size of 52 |
| 4 | unshuffled deck with a size of 53 |
| 4 | unshuffled deck with a size of 51 |
| 5,6 | unshuffled deck with a size of 52 |

**Test Case 3: deal(): Card**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * The method must remove the first card from a deck | 1. Removes the first card from a deck | 1. Removes a card other than the first one |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1,2 | deck: AD, 4C, 3H |

**Test Case 4: size(): int**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * Must Return the size of the number of cards that have not been dealt yet | 1. Returns the correct size of shuffledDeck | 1. Returns the wrong size of shuffledDeck |
| * Must Return a value between 0 and 52 (inclusive) | 1. 0 <= size <= 52 | 1. size > 52 |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1,2,3,4 | shuffledDeck: 52 cards |
| 1,2,3,4 | shuffledDeck: 0 cards |
| 1,2,3,4 | shuffledDeck: 51 cards |
| 1,2,3,4 | shuffledDeck: 1 card |

**Test Cases for Card Class**

**Test Case 1: toString(): String**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * Returns a string of the rank and the suit of the current Card object | 1. Returns the correct string | 1. Returns the wrong string |
| * The object must contain the right properties | 1. the object has a rank, suit and value in the right order and with correct type of values | 1. Invalid object |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1,3 | Card: {suit: Spades, rank: 10, value: 9} |
| 2 | Card: {suit: Hearts, rank: 2, value: 1} |
| 4 | Card: {suit: 2, rank: Spades, value: 1} |

**Test cases for Player Class**

**Test Case 1: cardsLeft(): int**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * Must return the number of cards left in the player's hand | 1. Returns the correct number of cards | 1. Returns the wrong number of cards |
| * The returned number must be 0 <= size <= 52 | 1. 0 <= size <= 52 | 1. size > 52 |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1,2,3 | playerHand: [AC, 2D, 3S]  number of cards: 3  cardsLeft () = 3 |
| 1,2,3 | a hand with 52 cards |
| 1,2,4 | a hand with 53 cards |

**Test case 2: getFirstCard() : Card**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * Returns the first card object from the player's hand | 1. returns the first Card | 1. Else |
| * The player hand cannot be empty | 1. has cards in it | 1. is empty |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1,2,3 | playerHand: [Card:2C, Card:JH, Card:5D, Card:7C, Card:10S] |
| 4 | playerHand: [] |

**Test case 3 : addCards(cards: Queue<Card>) : boolean**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * The Queue that is passed to the method cannot be empty | 1. isEmpty() = false | 1. isEmpty() = true |
| * The method must add the cards to the end of the player's hand | 1. The last item added to the hand must be the last in the hand itself | 1. The last item in hand is not the last item of the cards Queue past into the method |
| * The number of cards inside the hand shouldn't exceed 52 after the cards get added | 1. size <= 52 | 1. Other |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1 | cards: [Card:2C, Card:JH, Card:5D, Card:7C, Card:10S] |
| 2 | cards: [] |
| 3 | cards: [Card:2C, Card:JH, Card:5D, Card:7C, Card:10S] |
| 4 | cards: [Card:2C, Card:JH, Card:5D, Card:7C, Card:10S] |
| 5 | cards: [2 cards], playerHand: [50 cards] |
| 6 | cards: [2 cards], playerHand: [51 cards] |

**Test Cases for War Class**

**Test Case 1: start(): boolean**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * To start, the deck must have 52 cards inside | 1. true | 1. false |
| * At the end, each player must have 26 cards in order to start the game | 1. true | 1. false |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1 | deck: [52 card objects] |
| 2 | deck: [26 card objects],  deck: [51 card objects],  deck: [53 cards objects] |
| 3,4 | player 1: 26 cards  player 2: 26 cards |

**Test Case 2: play(): int**

**Rules:**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid**  **Equivalence Classes** | **Invalid**  **Equivalence**  **Classes** |
| * Player 1 wins the round | 1. result = 1 | 1. 2, 0, 11, 22 |
| * Player 2 wins the round | 1. result = 2 | 1. 1, 0, 11, 22 |
| * Tie/War | 1. result = 0 | 1. 1, 2, 11, 22 |
| * Check kitty size and make sure it has a size of 8 | 1. kitty.size() = 8 | 1. Else |
| * Player 1 has no cards left (Player 2 wins) | 1. result = 11 | 1. 1, 2, 0, 22 |
| * Player 2 has no cards left (Player 1 wins) | 1. result = 22 | 1. 1, 2, 0, 11 |

**Mapping:**

|  |  |
| --- | --- |
| **Test equivalence # mapping** | **Test value** |
| 1,2 | play(3 of Spades, 2 of Spades) |
| 3,4 | play(King of Hearts, Ace of Diamonds) |
| 5,6,7,8 | play(Jack of Clubs, Jack of Diamonds) |
| 3,4,7,8 | play(5 of Hearts, 10 of Hearts)  (use the kitty from the previous test scenario) |
| 9,10,11,12 | use cards from the above tests |