**WAR GAME TEST CASES**

**Deck**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
| Create a deck with 52 cards  In the order 2 to Ace for each suit (H, D, S, C) | 1. 2H, 3H, …., KC, AC | 2. null 3. Less than 52 cards  4. more than 52 cards 5. Any other order |
| The deck must be shuffled in a random order | 5. Unshuffled cardsQueue do not equal shuffled cardsQueue | 6. Unshuffled CardsQueue equals shuffled cardsQueue |
| New Deck() -test with unshuffled deck  Must deal the first card on top of the deck | 7. 2H | 8. null  9. card != 2H |

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
| New Deck() | 1, 2, 3, 4 |
| deck.shuffle() | 5, 6 |
| new Deck({2C, 7H, JD, 10S})  deck.deal() | 7, 8, 9 |

**War**

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
| Must determine the highest card out of 2 (suits does not affect the value) | 1. 2H < 3H | 3. 2H >= 3H |
| The player with highest card is the winner of the round(P1 – 3H) > (P2 – 2H) | 4. P1 winner  Value 2 < value 3 | 5. P2 winner |
| The winner of a round should have the 2 played card added to his hand—if P1 wins  P1- 8 cards  P2 – 10 cards | 13. P1 cards + 2 cards = 10 cards &  P2 cards = 10 cards | 14. P1 cards != 10 cards &  P2 cards != 10 cards |
| In the case of a tie, each player must play 3 cards face down – kitty should contain 8 cards total  P1- 9 cards  P2 – 9 cards | 6. Kitty size = 8  8. P1 cards = 6 &  P2 cards = 6 | 7. kitty != 8  16. P1 cards != 6 &  P2 cards != 6 |
| After a tie, the winner should get all cards in the kitty plus the 2 cards from the current round – if P1 wins  P1- 5 cards  P2 – 5 cards | 9. kitty = 0  11. P1 cards + 10 cards = 15 cards &  P2 cards = 5 | 10. kitty != 0  12. P1 cards != 15 &  P2 != 5 |
| After a tie, another tie occurs, the kitty should contain another 8 cards for a total of 16  P1- 5 cards  P2 – 5 cards | 15. kitty = 16  18. P1 cards = 2  P2 cards = 2 | 17. kitty != 16  19. P1 cards != 2  P2 cards != 2 |
| A winner is determined when a player has no more cards left  P1 – 0 cards  P2 – 20 cards | 20. P1 = 0 &  P2 = 20 => winner | 21. P1 != 0 || = 0 and returned as winner |
| A player loses the game if he cannot play during a tie war  P1 – 5 cards  P2 – 2 cards | 22. P1 (wins) | 23. p2 or null |

**Create war object and deck object of 20 cards for all test values**

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
| war.start(); | 1, 3 |
| war.getRndWinner() | 4, 5 |
| war.getPlayer1Cards()  war.getPlayer2Cards() | 13, 14 |
| war.getKitty()  war.getPlayer1Cards()  war.getPlayer2Cards() | 6, 7, 8 , 16, 9, 10, 11, 12, 15, 17, 18, 19 |
| war.getWinner() | 20, 21 |
| War.getWnner() | 22, 23 |

|  |  |  |
| --- | --- | --- |
| **Rules/Constraints** | **Valid Equivalence Classes** | **Invalid Equivalence Classes** |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| **Test value** | **Test equivalence # mapping** |
|  |  |

**Class Card**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Purpose | Object State | Expected Result |
| Card card = new Card(“10”, “D”, 10) | Create a Card object for the 10 of diamonds | Rank = “10”  Suit = “D”  Value = 10 |  |
| card.getValue | Check that the correct value was set | Rank = “10”  Suit = “D”  Value = 10 | 10 |
| card.toString() | Check that the correct rank and suit was set | Rank = “10”  Suit = “D”  Value = 10 | 10D |

**Class Deck**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Purpose | Object State | Expected Result |
| Card deck = new Deck() | Create deck object with the default attributes | cardsQueue = {2H, 3H…AC}  size = 52 |  |
| deck.toString() | Check that all 52 cards have been set in order | cardsQueue = {2H, 3H…AC}  size = 52 | 2H-AH  2D-AD  2S-AS  2C-AC |
| Card deck2 = new Deck() | Create deck object with default value – new unshuffled deck to test the deal method and compare to the shuffled deck | cardsQueue = {2H, 3H…AC}  size = 52 |  |
| deck.shuffle() | Check that the 52 cards get shuffled in a random order | cardsQueue = {random order}  size = 52 | Randomized order |
| deck == deck2 | Compare each card in a loop | cardsQueue = {2H, 3H…AC}  size = 52 | false |
| deck.deal()  deck.deal()  deck.deal()  deck.deal() | Check that it is the first card in the queue that is dealt | cardsQueue = {6H, 7H…AC}  size = 48 | CardObj{2H}  CardObj{3H}  CardObj{4H}  CardObj{5H} |
| deck.size() | Check that the cards are being removed from the deck | cardsQueue = {6H, 7H…AC}  size = 48 | 48 |

**Class War \*** cardQ{3H, 2H, 5H, 5C, 6H, 6D, 7H, 8H, 9H, 10H, AH, AD, JH, KH, 2D, 3D, 4D, 5D, 7D, 8D}

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Purpose | Object State | Expected Result |
| Deck deck = new Deck(cardQ)  War war = new War(“p1”, “p2”, deck); | Create new war object with the default attributes | deck = {3H, 2H, 5H, 5C, 6H, 6D, 7H, 8H, 9H, 10H, AH, AD, JH, KH, 2D, 3D, 4D, 5D, 7D, 8D}  player1Cards = {}  player2Cards = {}  kittyStack = {}  player1 = “p1”  player2 = “p2”  rndWinner = null  tie = false  rndResult = {W, T, E} |  |
| war.getPlayer1() | Verify that the player1 was set correctly | deck = {3H, 2H, 5H, 5C, 6H, 6D, 7H, 8H, 9H, 10H, AH, AD, JH, KH, 2D, 3D, 4D, 5D, 7D, 8D }  player1Cards = {}  player2Cards = {}  kittyStack = {}  player1 = “p1”  player2 = “p2”  rndWinner = null  tie = false  rndResult = {W, T, E} | p1 |
| war.getPlayer2() | Verify that the player2 was set correctly | deck = {3H, 2H, 5H, 5C, 6H, 6D, 7H, 8H, 9H, 10H, AH, AD, JH, KH, 2D, 3D, 4D, 5D, 7D, 8D }  player1Cards = {}  player2Cards = {}  kittyStack = {}  player1 = “p1”  player2 = “p2”  rndWinner = null  tie = false  rndResult = {W, T, E} | p2 |
| war.start() | Start the game, all cards are dealt to the players | deck = {}  player1Cards = {3H, 5H, 6H, 8H, 10H, AD, KH, 3D, 5D, 8D }  player2Cards = {2H, 5C, 6D, 7H, 9H, AH, JH, 2D, 4D, 7D }  kittyStack = {}  player1 = “p1”  player2 = “p2”  rndWinner = null  tie = false  rndResult = {W, T, E} |  |
| war.getPlayer1Cards().size() | Get the size of the player 1 card queue | No change | 10 |
| war.getPlayer2Cards().size() | Get the size of the player 2 card queue | No change | 10 |
| war.getDeck().isEmpty() | Verify that all cards have been delt and the deck is empty | No change | True |
| war.play() | Play the first round, will compare the 2 cards and determine the winner of the round | deck = {}  player1Cards = { 5H, 6H, 8H, 10H, KH, AD, 3D, 5D, 8D 3H, 2H }  player2Cards = { 5C, 6D, 7H, 9H, JH, AH, 2D, 4D, 7D}  kittyStack = {}  player1 = “p1”  player2 = “p2”  rndWinner = “p1”  tie = false  rndResult = {W, T, E} | Enum = W |
| war.getRndWinner() | Verify that the player with the highest card won | No change | p1 |
| war.getPlayer1Cards().size()  war.getPlayer2Cards().size() | Verify that the card count in each player are correct after 1st round | No change | 11  9 |
| war.play() | Play until there is a tie | deck = {}  player1Cards = {AD, KH 3D, 5D, 8D 3H, 2H}  player2Cards = { AH, JH, 2D, 4D, 7D}  kittyStack = {9H, 7H, 10H, 8H, 6D, 6H, 5H, 5C }  player1 = “p1”  player2 = “p2”  rndWinner = null  tie = true  rndResult = {W, T, E} | Enum = T |
| war.getKitty().size() | Check that the kitty contains the 8 cards from the war | No change | 8 |
| war.play() | Verify that if a tie occurs after a tie then the cards from the second war accumulate in the kitty | deck = {}  player1Cards = {8D 3H, 2H }  player2Cards ={7D }  kittyStack = { JH, 2D, 4D, KH, 3D, 5D, AH, AD, JH, 9H, 7H, KH,10H, 8H, 6D, 6H}  player1 = “p1”  player2 = “p2”  rndWinner = null  tie = true  rndResult = {W, T, E} |  |
| war.getKitty().size() | Check that the kitty contains the 16 cards from the war | No change | 16 |
| war.getPlayer1Cards().size()  war.getPlayer2Cards().size() | Verify that the card count in each player are still correct | No change | 3  1 |
| war.play() | Play another round | deck = {}  player1Cards = { 3H, 2H, JH, 2D, 4D, KH, 3D, 5D, AH, AD, JH, 9H, 7H, KH,10H, 8H, 6D, 6H, 8D, 7D }  player2Cards ={ }  kittyStack = { }  player1 = “p1”  player2 = “p2”  rndWinner = “p1”  tie = false  rndResult = {W, T, E} | Enum = E |
| war.getKitty().isEmpty() | Check that the kitty is empty | No change | true |
| war.getWinner() | Check that the player 1 has won the game | No change | p1 |
| cardQ = {2H, 3H, 4H, 5H, 6H, 7H, 8H, 8C, 9H, 10H}  Deck deck = new Deck(cardQ);  War war = new War(“p1”, “p2”, deck); | Create new deck and war object | deck = {2H, 3H, 4H, 5H, 6H, 7H, 8H, 8C, 9H, 10H }  player1Cards = { }  player2Cards ={ }  kittyStack = { }  player1 = “p1”  player2 = “p2”  rndWinner = null  tie = false  rndResult = {W, T, E} |  |
| war.start() | Deal the cards | deck = {}  player1Cards = {2H, 4H, 6H, 8H, 9H}  player2Cards ={3H, 5H, 7H, 8C, 10H }  kittyStack = { }  player1 = “p1”  player2 = “p2”  rndWinner = null  tie = false  rndResult = {W, T, E} |  |
| war.play()  war.play()  war.play() |  | deck = {}  player1Cards = { 8H, 9H }  player2Cards = { 8C, 3H, 5H, 7H, , 10H, 2H, 4H, 6H}  kittyStack = { }  player1 = “p1”  player2 = “p2”  rndWinner = “p2”  tie = false  rndResult = {W, T, E} | W  W  W |
| war.play | Verify that the game will end when a player cannot play all the cards during a tie war | deck = {}  player1Cards = { }  player2Cards ={10H, 2H, 4H, 6H }  kittyStack = {8C, 8H, 9H, 3H, 5H, 7H}  player1 = “p1”  player2 = “p2”  rndWinner = “p2”  tie = false  rndResult = {W, T, E} | E |
| war.getWinner() | Verify that the correct player is set as winner | deck = {}  player1Cards = { }  player2Cards ={6H, 3H, 5H, 7H, 8C, 10H, 8H, 9H, 2H, 4H }  kittyStack = {}  player1 = “p1”  player2 = “p2”  rndWinner = “p2”  tie = false  rndResult = {W, T, E} |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Purpose | Object State | Expected Result |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |