Group A Testing

Skip-Bo Testing Plan

Austin Ball, Jackson Druhan, Muhammad Saleh, Gurdeep Singh

CardModel Test:

- Test constructor by making a card with a number and test with getNumber
- Test setter by using setNumber and getNumber
- Test getter with previous two tests
- Test isSkipBo by making a card with int = 0 and use isSkipBo

DeckModel Test:

- Test shuffleDeck by making two decks, use shuffleDeck on one and compare them
- Test getTopCard by building a deck and checking if the top card is 1 (can be tested multiple times to make sure deck is built properly)
- Test getDeck by building two decks, then use getDeck on one and compare it to the other
- Test removeCardDup by checking the size of the deck before and after removing a few cards from the deck
- Test getSize by building a deck, then using getSize to make sure it is has 162 cards, then use removeCardDup(i) and make sure getSize returns 162 - i
- Test getFirstCard by building and deck and making a #1 card, then make sure that getFirstCard and the #1 card are the same
- Test addGarbageToDeck by building a deck and removing cards , then create a stack of CardModels and use addBuildToGarbage, then use addGarbageToDeck and make sure getSize sums to the sum of both piles
- Test checkSize by building a deck and removing cards until getSize returns less than 10, then create a stack of CardModels and use addBuildToGarbage, then use addGarbageToDeck and make sure the deck replenishes
- Test addBuildToGarbage by creating a stack of cardModels, then calling addBuildToGarbage using it, then use addGarbageToDeck and make sure getSize sums to the sum of both piles

HandModel Test:

- Test constructor by creating a hand and making sure that the hand is empty, and the card number value is zero
- Test GetHand by using get hand and checking the size of the vector, then adding a card and making sure the size changes accordingly

- Test addCard by creating a hand, then using addCard and the size of the vector changes
- Test removeCard by adding cards to a hand, and removing cards one by one and checking the vector size
- Test getNumberOfCards by adding and removing cards from the hand and checking that the size of the vector is the same as the numberOfCards value
- Test useCard by adding cards to the hand, then checking if the return value of the useCard function is equivalent to the added cards

PileModel Test:

- Test constructor by creating a pile and use getSize to make sure it is empty
- Test getTopCard by adding a card to the pile and making sure the card is the same as the getTopCard return
- Test getTopNum by adding a card to the pile then use get number on the card as well as getTopNum and make sure they are the same
- Test addCard by using addCard to add a card then make sure the size of the stack as well as the numberOfCards value increases
- Test removeCard by adding cards to the stack and then use removeCard and check that the size of the stack as well as the numberOfCards value decreases
- Test getNumberOfCards by using the addCard and removeCard functions and checking numberOfCards between each addition or removal
- Test getSize by using the addCard and removeCard functions and checking the size of the stack between each addition or removal
- Test getPile by creating a stack of CardModel* then using addCard to make the same stack. Then use getPile and make sure they are equivalent
- Test clearPile by adding cards to the pile, then using clearPile and making sure the pile is empty and the numberOfCards value is zero

PlayerModel Test:

- Test constructor by creating a player and use getName to make sure the name is set properly
- Test getName by creating a player and making a string with the same name and comparing them
- Test isComputerPlayer by creating two players, setting one to a computer, then confirm that when isComputerPlayer is called, they return true and false respectively
- Test setComputerPlayer by creating a player, confirm isComputerPlayer returns false, then use setComputerPlayer and confirm isComputerPlayer returns true
- Test addToStock by creating a player, making sure the stock size is 0, then use addToStock and make sure stock size increases
- Test handsize by creating a player and making sure handsize returns 0, then use addCard and make sure handsize increases accordingly
- Test stocksize the same way as handsize but use addToStock
- Test addCard the same way handsize was tested

- Test useCard by creating a player and a card, then add the card to the players hand using addCard, then compare the card with the return of the useCard function (do multiple times with different integer calls and cards)
- Test usediscard by creating a player and a card, make sure use discard returns a card with value 20, then add the card to one of the discard piles with usingDiscard and make sure the usediscard function returns the correct value of the card
- Test useStock by creating a player and a card, adding the card using addToStock, then compare useStock with the card
- Test usingDiscard the same as usediscard
- Test removeCard by creating a player and adding multiple cards to the hand, then use removeCard and handsize repeatedly confirming the hand decreases
- Test deleteDiscardCard by adding a card to one of the discard piles and checking its size, then use deleteDiscardCard and confirm that it decreased
- Test deleteStockCard by creating a player and cards, then use addToStock to add cards to the stock, then use stocksize to confirm the size of the stock, then use deleteStockCard and make sure stocksize decreases
- Test returnHand by creating a player and a hand, make the hand and the player hand identical by adding the same cards to each, then compare them using returnHand and make sure they are the same, then change one and make sure they are different
- Test returnStock the same as returnHand but with a pile instead of a hand
- Test returnArrPile the same as returnHand but with an arrPile instead of a hand

ArrPileModel Test:

- Test constructor by creating an arrPile, then use getPile and getNumberOfCards on each pile to confirm that all piles are empty
- Test getPile by creating an arrPile, adding cards to the various piles using , then using getPile and getNumberOfCards, confirm that the piles can be access with getPile
- Test buildPileArr by creating an arrPile, then creating a vector of piles, then set the array to the vector using buildPileArr and checking that the size of the vector is 4
- Test insertCard by creating an arrPile and various cards, then attempt to add cards to the pile confirming that insertCard only returns true when the card added is a skip-bo card (card value of 0) or when the card is +1 the top of the pile
- Test useDiscard by creating an arrPile and various cards, then confirm that useDiscard returns a card with value 20, then add cards to the discard using putCardInDiscard and confirm useDiscard returns a card value that matches the created card
- Test popTopCard by creating an arrPile and various cards, then add the cards to various piles. Then, using popTopCard, getPile and getNumberOfCards, confirm that popTopCard removes one card each time it is called

- Test putCardInDiscard by creating an arrPile and a card, using putCardInDiscard to add it to one of the discard piles, then confirm that it was added by using getPile and getTopNum
- Test clearPile by creating an arrPile and various cards, adding the cards to a pile using putCardInDiscard and check the size using getNumberOfCard, then use clearPile and confirm the pile is empty

Mock Testing Controller

- Create Mocks for PlayerModel, DeckModel, and ArrPileModel that have:
    - Bool functions that return true
    - Int functions that return valid ints
    - CardModel* returning functions that return a card with a valid value
    - HandModel* returning functions that return valid hands
    - PileModel* returning functions that return valid piles
    - ArrPileModel* returning functions that return valid arrPiles
    - Vector returning functions that return valid vectors
    - Void functions that do nothing
- In gameStart ensure:
    - deckSetup is called first and once
    - playerSetup is called twice
    - help is called once
    - playCards and opponentTurn are called over and over until stocksize is 0
- In deckSetup ensure:
    - buildDeck is called first and once
    - shuffleDeck is called once
- In playerSetup ensure:
    - addToStock is called once
- In displayBoard ensure:
    - displayDiscard is called first and once
    - DisplayarrPileViow is called once
    - displayDiscard is called once more
    - displayStock is called once
    - displayHand is called once and last
- In opponentTurn ensure:
    - checkSize is called first and once
    - dealHand is called once
    - checkBuildSize is called once
    - useStock is called once
    - addingToPiles is called four times
    - deleteStockCard is called 0 up to 4 times
    - useCard is called four times
    - addingToPiles is called four more times
    - removeCard is called 0 up to 4 times
    - useCard is called once more
    - discardPilesPick is called once

    - removeCard is called once more
- In playCards ensure:
    - checkSize is called first and once
    - dealHand is called once
    - a loop of the following functions occurs until gate is false
        - checkBuildSize
        - displayBoard
        - options, handsize
        - discardingCard
        - handplay
        - discardPlay
        - stockPlay
        - help
        - leave
- In dealHand ensure:
    - handsize is called first and once
    - addCard and getTopCard are called up to 5 times
- In handplay ensure:
    - handsize I called first and once
    - useCard is called 0 or 1 time
    - selectBuildPile is called 0 or 1 time
    - addingToPiles is called 0 or 1 time
    - removeCard is called 0 or 1 time
- In discardPlay ensure:
    - usediscard is called first and 0 or 1 time
    - getNumber is called 0 or 1 time
    - selectBuildPile is called 0 or 1 time
    - addingToPiles is called 0 or 1 time
    - deleteDiscardCard is called 0 or 1 time
- In stockPlay ensure:
    - useStock is called 0 or 1 time
    - selectBuildPile is called 0 or 1 time
    - addingToPiles is called 0 or 1 time
    - deleteStockCard is called 0 or 1 time
- In discardingCard ensure:
    - handsize is called first and once

      Or

    - discardChoice is called first and once
    - handsize is called twice
    - useCard is called 0 or 1 time
    - discardPile is called 0 or 1 time
    - discardPilesPick is called once
    - removeCard is called once
- In addingToPiles ensure:

    - insertCard is called first and once or not at all
- In discardPilesPick ensure:
    - usingDiscard is called first and once or not at all
- In checkBuildSize ensure:
    - checkBuildSize is called once
- In leave ensure:
    - exitGame is called once