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