Project 2

Title

War

Course

CIS-17C

Section

47914

Due Date

December 11, 2018

Author

Josh McIntyre

Rules & Overview:

War is a 2 player card game in which a standard card deck is used. The game starts with both player's randomly getting 26 cards, the players leave them face down and do not see what cards they have. Player's then flip the top card of their deck at the same time, and whichever player has the card with the higher value takes both cards and puts them at the bottom of their deck. Card values are ranked from least to greatest in this order: 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, and Ace. If the cards of the same value then the players take their next top card and leaving it face down they set it on top of their first card, they then draw the next card and reveal it. Whichever player has the higher card then takes all the drawn cards. If player's keep drawing cards of the same value the process is repeated until 1 player draws a card of higher value than the other player's. A player wins the game when they have all the cards in their deck.

Pseudocode:

1) Card() inherits from AbstractCard(),which is an asbtract base class, and has member functions for setting and getting the values and suits of card objects as stored in a pair utility, getting the name of the cards, and overloading the operators <, >, and == in regards to the value of the cards
2) Player() and Dealer() both inherit from the abstract base class GameEntitity()
3) Dealer() has member functions for shuffling and dealing the deck
4) Player() has member functions for moving cards to the bottom of their hand, removing cards from their hand, getting the top card in their hand, getting the

number of cards in their hand, and getting and setting the player’s name by utilizing a list container

5) The class Deck() utilizes a map container of Card() objects as well as an integer vector parallel to the map of card object keys and mapped index values that is used to query Card() objects from the map container
6) The main function creates instances of 2 Player() objects, a Deck() objects, and a Dealer() object. The deck creates 52 instances of Card() objects and Player() objects can hold up to a maximum of 52 indexes in their vector containers
7) The Dealer() object then calls shuffle() which randomizes the indeces in the vector container in Deck() and deals 26 indexes even to each Player()
8) Players then proceed to play the game by drawing a card by pressing ENTER on the keyboard
9) The game runs telling players what cards were drawn and which player won the cards that turn
10) When a player wins, the game asks them if they had fun, and the winner's name is recorded in a file. The program then terminates

CHANGES TO PROJECT 2:

I implemented a Binary Search Tree Recursive Sorting algorithm that takes an unsorted array, utilizes tree sorting, and returns sorts the original array. It is obviously not very efficient for these purposes, but it was a requirement so that's the way I did it. The shuffle function inside the Dealer class I also changed in order to be recursive to knock out the last of this project's requirements. I had already implemented a sort of hashing by serializing each Card object and creating a custom Comparator function that tells it what order each Card object should be sorted in when stored in a map of Cards. You can sort of think of the serial numbers on each card as being a hash that associates that object to a certain spot in a Deck.

Repository:

https://github.com/JMAK13/McIntyreJosh_CSC_17C_Fall2018/tree/master/Projects/CSC17C_Project_1