**war.cpp**

//Muneer Tomeh

//CECS 282 - Section 07

//Program #1 - War

// 2-13-17

#include "Card.h"

#include "Deck.h"

#include <iostream>

#include <string>

using namespace std;

int main() {

//a new deck is created automatically once the program starts

Deck myDeck;

//showing the main menu for the War program

int menuSelection=0;

do {

if(menuSelection !=4){

cout << "1) Get a new card deck" << endl;

cout << "2) Show all remaining cards in deck" << endl;

cout << "3) Shuffle the deck" << endl;

cout << "4) Play WAR!" << endl;

cout << "5) Exit program" << endl;

cin >> menuSelection;

}

Card c1;

Card c2;

switch (menuSelection) {

cout << menuSelection << endl;

case 1:

//this option resets the card pile inside the deck

myDeck.resetDeck();

cout << "Deck has been reset" << endl;

break;

//this option allows the user to peek at the deck's remanining cards

case 2:

myDeck.displayDeck();

break;

//this choice shuffles a full deck, but if the deck is partial, shuffling will not occur

case 3:

myDeck.shuffle();

break;

//this option initiates the game of War

case 4:

if (myDeck.cardsLeft() == 0) {

cout << "Cannot play War with nothing inside the deck" << endl;

menuSelection = 0;

break;

}

cout << "GET READY TO PLAY WAR!!!" << endl;

cout << " " << endl;

cout << "Total amount of cards in deck: "<< myDeck.cardsLeft() <<endl;

cout << " ....dealing" << endl;

//dealing a card for the user and displaying it

cout << "One for you" << endl;

c1 = myDeck.deal();

c1.displayCardInfo();

c1.displayCard();

cout << "" << endl;

//dealing a card for the AI and displaying it

cout << "One for me" << endl;

c2 = myDeck.deal();

c2.displayCardInfo();

c2.displayCard();

cout << "" << endl;

//if the player's card value is higher than the AI's card value, then the player wins

if (c1.getValue() > c2.getValue()) {

cout << "You win!" << endl;

cout << " " << endl;

}

//if both card values are the same, then it is automatically a draw

else if (c1.getValue() == c2.getValue()){

cout << "It's a tie folks!" << endl;

cout << " " << endl;

}

//the human player loses if his card value is less than the AI's card value

else {

cout << "You lose!" << endl;

cout << " " << endl;

}

char answer;

//inquiring if user wants to play again

if (myDeck.cardsLeft() == 0) { // however if there aren't any cards left, the user will be directed to the main menu

cout << "Cannot play WAR since there are no cards left, REDIRECTING TO MAIN MENU" << endl;

menuSelection = 0;

break;

}

cout << "Would you like to play again?, Y/y for Yes, N/n for No" << endl;

cin >> answer;

if (answer == 'Y' || answer == 'y') { //if the user selects yes, then he is directed to the war game(selection 4)

menuSelection = 4;

} //else, the user will be directed to the War program main menu

else {

menuSelection = 0;

}

break;

case 5://exits program

break;

}

} while (menuSelection !=5);

return 0;

}

**Card.h**

#ifndef CARD\_H

#define CARD\_H

class Card {

private:

char rank; //placement level from best to worst

char suit; //category of a card; hearts, spades, diamonds, clubs

public:

Card();

Card(char r, char s);

void setCard(char r, char s);

void displayCard();

void displayCardInfo();

int getValue();

};

#endif;

**Card.cpp**

#include <iostream>

#include <string>

#include "Card.h"

using namespace std;

//default constructor

Card::Card() {

rank = 'x';

suit = 'y';

}

//overloaded constructor

Card::Card(char r, char s) {

rank = r;

suit = s;

}

//setter for a card

void Card::setCard(char r, char s) {

rank = r;

suit = s;

}

//this method displays a card visually that has been drawn from the deck

void Card::displayCard() {

switch (suit) {

case 'H': //case for a Heart card

cout << "-------------" << endl;

cout << "[" << rank << " ]" << endl;

cout << "[ \*\* \*\* ]" << endl;

cout << "[ \* \* \* ]" << endl;

cout << "[ \* \* ]" << endl;

cout << "[ \* \* ]" << endl;

cout << "[ \* ]" << endl;

cout << "[ " <<rank<<"]" << endl;

cout << "-------------" << endl;

break;

case 'S': //case for a Spade card

cout << "-------------" << endl;

cout << "[" << rank << " ]" << endl;

cout << "[ \* ]" << endl;

cout << "[ \* \* ]" << endl;

cout << "[ \*\*\*\*\* ]" << endl;

cout << "[ \* \* \* ]" << endl;

cout << "[ \* ]" << endl;

cout << "[ " << rank << "]" << endl;

cout << "-------------" << endl;

break;

case 'D': //case for a Diamond card

cout << "-------------" << endl;

cout << "[" << rank << " ]" << endl;

cout << "[ \* ]" << endl;

cout << "[ \* \* ]" << endl;

cout << "[ \* \* ]" << endl;

cout << "[ \* \* ]" << endl;

cout << "[ \* ]" << endl;

cout << "[ " <<rank<<"]" << endl;

cout << "-------------" << endl;

break;

case 'C': //case for a Club card

cout << "-------------" << endl;

cout << "[" << rank << " ]" << endl;

cout << "[ \* ]" << endl;

cout << "[ \* \* \* ]" << endl;

cout << "[ \* \* \* ]" << endl;

cout << "[ \* ]" << endl;

cout << "[ \* ]" << endl;

cout << "[ " << rank << "]" << endl;

cout << "-------------" << endl;

break;

}

}

//This method displays the information of a card and outputs it to the user

void Card::displayCardInfo() {

//the rank as a character data type is then processed to a string

string daRank;

switch (rank) {

case 'A':

daRank = "Ace";

break;

case 'K':

daRank = "King";

break;

case 'Q':

daRank = "Queen";

break;

case 'J':

daRank = "Jack";

break;

case 'T':

daRank = "Ten";

break;

case '9':

daRank = "Nine";

break;

case '8':

daRank = "Eight";

break;

case '7':

daRank = "Seven";

break;

case '6':

daRank = "Six";

break;

case '5':

daRank = "Five";

break;

case '4':

daRank = "Four";

break;

case '3':

daRank = "Three";

break;

case '2':

daRank = "Two";

break;

}

//the suit as a character data type is then processed to a string

string daSuit;

switch (suit) {

case 'H':

daSuit = "Hearts";

break;

case 'S':

daSuit = "Spades";

break;

case 'D':

daSuit = "Diamonds";

break;

case 'C':

daSuit = "Clubs";

break;

}

cout << daRank + " of " + daSuit <<endl;

}

//This method retrieves the card value of a dealt card which is returned for comparison with another

//dealt card's value

int Card::getValue() {

int cardValue;

switch (rank) {

case 'A':

cardValue = 1;

break;

case 'K':

cardValue = 13;

break;

case 'Q':

cardValue = 12;

break;

case 'J':

cardValue = 11;

break;

case 'T':

cardValue = 10;

break;

case '9':

cardValue = 9;

break;

case '8':

cardValue = 8;

break;

case '7':

cardValue = 7;

break;

case '6':

cardValue = 6;

break;

case '5':

cardValue = 5;

break;

case '4':

cardValue = 4;

break;

case '3':

cardValue = 3;

break;

case '2':

cardValue = 2;

break;

}

return cardValue;

}

**Deck.h**

#include <iostream>

#include <string>

using namespace std;

class Deck {

private:

Card pile[52]; //an array of cards that forms a 52 card deck

int totalCardsLeft; // amount of remaining cards left in the deck

int cardIndex; //an index (location) of a card used in the array of cards, pile[]

public:

Deck();

void resetDeck();

Card deal();

void shuffle();

int cardsLeft();

void displayDeck();

};

**Deck.cpp**

#include "Card.h"

#include "Deck.h"

#include <iostream>

#include <string>

#include <ctime>

using namespace std;

//constructor for a new deck

Deck::Deck() {

totalCardsLeft = 52;

cardIndex = 51;

char ranks[13] = { 'A', '2','3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K' };f

char suits[4] = { 'H', 'S', 'D', 'C' };

int k = 0;

//forming a deck that will contain 52 cards

for (int i = 0; i < 13; i++) {

for (int j = 0; j < 4; j++) {

pile[k] = Card(ranks[i], suits[j]);

k++;

}

}

}

//This method resets a deck by starting with a fresh pile of cards

void Deck::resetDeck() {

totalCardsLeft = 52;

cardIndex = 51;

//clearing the old set of cards and instantiating new cards without a suit or rank

for (int i = 0; i < 52; i++){

pile[i] = Card();

}

char ranks[13] = { 'A', '2','3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K' };

char suits[4] = { 'H', 'S', 'D', 'C' };

//assigning a suit and rank to each card in the new pile

int m = 0;

for (int i = 0; i < 13; i++) {

for (int j = 0; j < 4; j++) {

pile[m].setCard(ranks[i], suits[j]);

m++;

}

}

}

//This method deals a card from a deck and gives it to a player

Card Deck::deal() {

Card temp = pile[cardIndex];

totalCardsLeft--;

cardIndex--;

return temp;

}

//This method randomly shuffles a 52 card deck ONLY, not a partial one

void Deck::shuffle() {

if (totalCardsLeft != 52) {

cout << "Cannot shuffle a partial deck" << endl;

cout << " " << endl;

}

else {

cout << "Cards have been shuffled" << endl;

srand(time(0));

for (int i = 0; i < 52; i++) {

//an arbitrary index of the deck is generated in which the card at that location with be swapped

//with a card at an incrementing index, i

int randyIndex = rand() % 52;

Card auxiliary = pile[i];

pile[i] = pile[randyIndex];

pile[randyIndex] = auxiliary;

}

}

}

//This method returns the amount of cards left in a deck

int Deck::cardsLeft() {

return totalCardsLeft;

}

//This method displays the total amount of cards left in the deck

void Deck::displayDeck() {

//tests if the deck doesn't contain any cards

if (totalCardsLeft == 0) {

cout << "Cannot show any cards in the deck, as the deck is EMPTY" << endl;

}

//else, show all remaining cards left in the deck

else {

for (int i = 0; i < totalCardsLeft; i++) {

pile[i].displayCardInfo();

}

}

cout << "" << endl;

}