עבודה ב **31695** OOPסמסטר א' תש"פ, עבודה 1

תאריך ההגשה *19/12/2019 :*

עידו עזרא, 209453935, idoezra9@gmail.com

אור כהן, 318300373, cohenor1112@gmail.com

ציון:

**תיאור התוכנית**

המשחק שלנו בנוי ע"י 4 מחלקות שונות העובדות בשילוב זו עם זו.

להלן פירוט ארבעת המחלקות והסבר על השדות והמתודות:

**המחלקה Card:**

class Card

{

private:

int BJ\_value; // BJ value

string card\_name; //card name

static int number\_of\_cards\_already\_made; //the number of cards that made till now

static int cards\_hist[14]; //the number of cards from each type that made till now ([0]not ligal!!, [1]CARD A's------> [13]CARD K's)

static int total\_number\_of\_cards\_need; //total number of cards in 1 dequeue (52)

static int number\_of\_same\_card; //the number of cards to create for every type in 1 dequeue (4)

//Extras:

static string cards\_names[]; //string array for the name(symbol) of the cards ("A","1","2"...."K")

public:

Card(int value); //ctor

//using default dtor

//using default copy ctor

static int toss\_val(); //Scores a numerical value to the card according to the rules of 4 cards per card type

friend ostream& operator<<(ostream& os, const Card& c); //printing card

//Extras:

unsigned int give\_BJ\_value(const int card\_number); //give the card valid bj value

unsigned int get\_BJ\_value(); //returns the card valid bj value

const string give\_name\_to\_card(const int card\_number); //using in the constractur to initialize the private member "card\_name"

const string get\_card\_name(); //return the card name(symbol)

};

**המחלקה Player:**

class Player

{

private:

string p\_name; //player name

unsigned int cards\_sum; //player cards sum

unsigned int r\_won; //rounds that the player won

unsigned int r\_lose; //rounds that the player lose

public:

Player(string name); //ctor

//using defult dtor

//using defult cctor

friend ostream& operator<<(ostream& os, const Player& player); // print name and card\_sum

//"Extras":

string get\_name(); //return the player name

unsigned int get\_cards\_sum(); //return the cards sum of the player

void update\_cards\_sum(unsigned int \_cards\_sum); //update the cards sum

unsigned int get\_wins(); //return the number of rounds that the player win

unsigned int get\_loses(); //return the number of rounds that the player win

void update\_wins\_and\_loses(bool win\_flag); //update the number of wins or loses according the current round result

};

**המחלקה Dequeue:**

class Dequeue

{

private:

//1)Number of organs in array

unsigned int m\_size;

//2)Size of array

const int m\_capacity = 52;

//3)Array of pointers with m\_capacity length

Card\*\* m\_queue;

//4)Index of the first organ in array m\_queue

int m\_left;

//Extras:

//Index of the last organ in array m\_queue

int m\_right;

public:

//1)Ctor

//Initializes the members m\_size ,m\_left and m\_right to 0

//creating array m\_queue of pointer (length of the array is m\_capacity (52)) , and initialize every pointer to point to NULL

Dequeue();

//2)Dtor

//Delete all created dynamic allocations (the pointer that are in the array m\_queue and the m\_queue itself

~Dequeue();

//3)

//push to the back of the array m\_queue if it is'nt full ,and updating m\_size

void push\_back(Card\* card);

//4)

//pop the last organ in array m\_queue if it is'nt empty ,and updating m\_size

Card\* pop\_back();

//5)

//push to the front of the array m\_queue if it is'nt full ,and updating m\_size

void push\_front(Card\* card);

//6)

//pop the first organ in array m\_queue if it is'nt empty ,and updating m\_size

Card\* pop\_front();

//7)

//Returns the first organ of m\_queue

const Card& front() const;

//8)

//Returns the last organ of m\_queue

const Card& back() const;

//9)

//Returns the number of organs in m\_queue

unsigned int size() const;

//10)

//Returns 1 if m\_queue is empty and 0 if not

bool is\_empty() const;

//Extras:

//Returns 1 if m\_queue is full and 0 if not

bool is\_full() const;

//Receives a push instruction ("push back" or "push front") and updating the appropriate index according to the instruction . (Auxiliary function for the push functions). It refers to the fact that the array is a cyclical array

void for\_push\_next\_index(const string& push\_back\_or\_push\_front);

////Receives a pop instruction ("pop back" or "pop front") and updating the appropriate index according to the instruction . (Auxiliary function for the pop functions) .It refers to the fact that the array is a cyclical array

void for\_pop\_prev\_index(const string& pop\_back\_or\_pop\_front);

//Receives an array of pointers to cards and fills m\_queue

void first\_fill\_Dequeue(Card\* card\_arr[]);

//Mixes the queue randomly

void mix\_Dequeue();

//Gets two indexes and changes them to be random (in range 0-51)

void random\_2\_indexes\_0\_to\_51(unsigned int& index\_a, unsigned int& index\_b);

//Switches between two pointers

void swap\_cards(Card\*& card\_a\_pointer, Card\*& card\_b\_pointer);

};

**המחלקה BlackJack:**

class BlackJack

{

private:

//player pointer

Player\* player\_ptr;

//array of pointers type Card

Card\*\* card\_arr;

//pack of cards type Dequeue

Dequeue\* card\_pile;

//array of pointers type Card to the player (Stores pointer of cards he took)

Card\*\* player\_cards\_pointers;

//array of string of the Card that the player took

string\* player\_card[11];

//array of pointers type Card to the dealer (Stores pointer of cards he took)

Card\*\* dealer\_cards\_pointers;

//array of string of the Card that the dealer took

string\* dealer\_card[11];

//stores how many round the player played

int rounds;

//Extras:

string player\_name;

Player\* Dealer\_ptr;

//index of player last card given

int p\_card\_indx;

//index of dealer last card given

int d\_card\_indx;

//1 if player chose to quit the game and 0 if doesnt

bool player\_done;

//player sum of hand

unsigned int p\_sum;

//dealer sum of hand

unsigned int d\_sum;

//counting the number of aces the player has in his hand

int p\_ace\_counter;

//counting the number of aces the dealer has in his hand

int d\_ace\_counter;

public:

//Ctor

//Initializes the members player\_ptr , player\_card , dealer\_card , /\*Dealer\_ptr ,\*/ p\_card\_indx, d\_card\_indx, player\_done , rounds, p\_sum, d\_sum, p\_ace\_counter, d\_ace\_counter

//creates dealer(type player) , 52 cards (type Card) ,card pile (type dequeue) ,and dealer and player hands(pointers to Card)

BlackJack();

//Dtor

//Delete all created dynamic allocations.

~BlackJack();

//creates player(type player) and starting the blackjack game

void play();

//prints how much the player won and lost

void end\_print(Player& gamer);

//Extras:

//printing to the player his options and returns 1 if he chose new round or 2 if he chose to quit

unsigned int game\_option();

//takes (and remove) from the card pile a card and moving it to the player/dealer hand

void add\_card\_p(Player& player, Dequeue& card\_pile);

//prints the player/dealer cards

void print\_cards(Player& player);

//prints to the player his options and returns 1 if he chose to take new card or 2 if he chose to stand

unsigned int while\_playing\_choose\_option();

//prints who won (dealer or player)

void who\_won(Player& dealer, Player& player);

//restarts everything (dealer/player hand and their sum of hands, card pile) to a new round/to quit the game

void restart(Dequeue& card\_pile);

};

**הסבר על אלגוריתם/כתיבת התוכנית:**

מהלך התוכנית bj.play() :

בהתחלה התכנית יוצרת שחקן עם שמו של המשתמש ושואלת אותו האם הוא רוצה לשחק.

תחילה החפיסה מתערבבת ,מחלקת לדילר שני קלפים ולשחקן שני קלפים ומדפיסה את המצב הנוכחי (את היד של הדילר וסכומה ואת היד של השחקן וסכומה).

כעת התכנית מחולקת לשני לולאות מרכזיות – לולאה בה סכום היד של הדילר קטן מ17 (הערך בו הוא לא מקבל יותר קלפים) ולולאה בה סכום ידו של הדילר גדול מ17 .

בלולאה הראשונה גם הדילר וגם השחקן מקבלים קלפים עד ש:או שהדילר עולה מעל 17 ויוצאים ללולאה השניה , או במידה ומישהו מהדילר או השחקן מפסיד)

בלולאה השניה הדבר דומה אך שם רק השחקן מקבל את הקלפים ורק הוא במצב של מועמד לפסילה).

כאשר מסיימים את המשחק / השחקן בוחר לצאת מהמשחק נשאל האם הוא רוצה סיבוב נוסף – במידה וכן הידיים והחבילה מתאפסים למצב ההתחלתי וחוזרים ללולאה , ובמידה ולא הידיים והחבילה מתאפסים ,מודפסות תוצאות המשחקים שהיו והמשחק נגמר.

**מקורות:**

1. הרצאות קורס OOP מהמודל.
2. מעבדות קודמות מקורס OOP.

**בעיות ידועות:**

אין בעיות ידועות בפרויקט זה.