 Universitatea Tehnică „Gheorghe Asachi” din Iași 

Facultatea de Automatică și Calculatoare

Calculatoare și Tehnologia Informației

Anul III

**Proiect 2022**

**Five-Card Draw Poker**

**Disciplina**: Ingineria Programarii

**Profesor coordonator**: Florin Leon

**Studenți**:

* Ciopeica Sergiu Andrei
* Tănăsache Cristian

**Grupa**: 1308B

**Cuprins**

**Capitolul 1 Prezentarea generală a ideii de proiect**

* Lucruri generale

**Capitolul 2 Descrierea proiectului**

2.1 Diagrame UML

2.1.1 Cazuri de utilizare

2.1.2 Clase

2.1.3 Activitati

2.1.4 Secvente

2.2 Programul in executie / Capturi ecran

2.2.1 Codul sursa

2.3 Autorii acestui proiect

**Capitolul 3 Cerinte**

3.1 Sablon de proiectare

3.2 Tratarea exceptiilor

3.2.1 Testarea unitatilor

**Capitolul 4 Despre, reguli si strategii**

4.1 Mai multe despre proiect

4.2 Regulile jocului

4.3 Strategii de baza

**Concluzii**

**Capitolul 1 Prezentarea generală a ideii de proiect**

Ideea de proiect pe care noi o propunem este una destul de obişnuită si simpla cunoscuta publicului larg si foarte populara in casinouri si se numeste: poker cu 5 carti

Asa cum mai este obisnuit denumit termenul in engleza “five-card draw” este unul dintre cele mai clasice jocuri de poker care exita. Pana la aparitia Texas Hold’Em, a dominat scena pariurilor. Este asemanator, dar are o turnura foarte diferita.

In acest proiect noi am incercat sa recream acest joc si sa fie pe cat posibil aproape de jocul real.O sa prezentam de modul prin care am decis si de ce am ajuns sa facem acest joc, cum am reusit sa-l facem, elementele de baza si cele mai putin de baza, clasele, legaturile s.a.m.d.

* ***Lucruri generale***

Proiectul noastru poate fi folosit de orice categorie de varsta, avand reguli si functionalitate simpla, astfel incat, chiar si cei fara ani de experienta in arta jocului pot sa se bucure si sa aiba parte de o experienta completa fara probleme. Acest joc ar fi mai important pentru oamenii care doresc sa joace o partida de poker fara frica sau riscul de a pierde o suma foarte mare de bani si tot o data pentru a-si petrece timpul intr-un mod cat mai placut. Deasemenea poate fi de ajutor si celor care vor sa invete jocul mai bine fara a iesi din confortul casei si a se deplasa.

La pornirea jocului, se va deschide o interfata, care initial va porni cu menu-ul jocului.

Acesta va acea urmatoarele : New Game, Rules, Options si Exit.

La apasarea butonului New Game se va permite utilizatorului sa joace o partida de poker cu adversari. Deasemenea poate apasa pe butonul Rules pentru a citi regulile jocului inainte de a incepe sau doar pentru a arunca o scurta privire, iar la butonul Option utilizatorul poate redimensiona fereastra si/sau sa schimbe volumul melodiei.

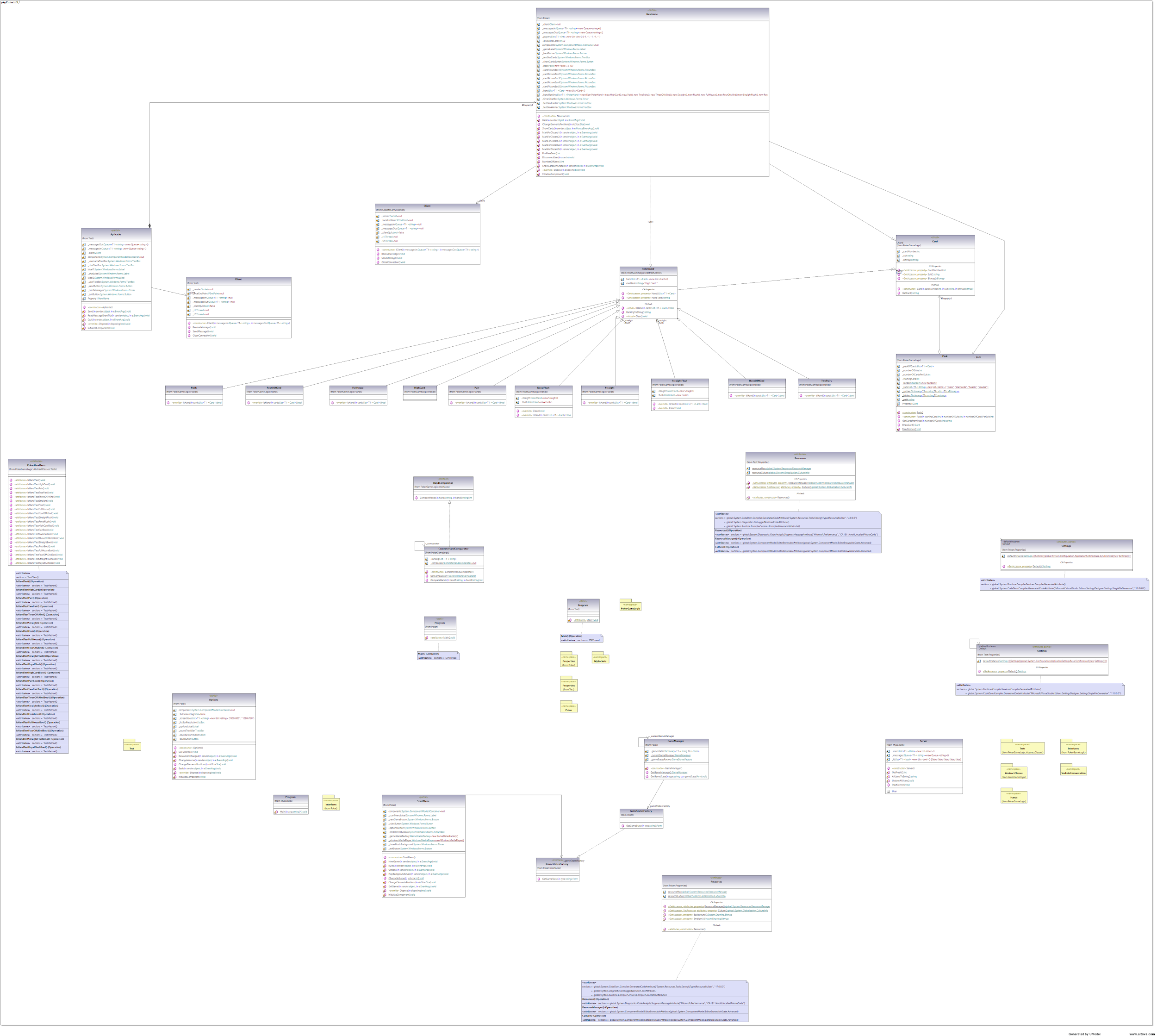
Dupa toate acestea utilizatorul poate experimenta cum e sa joci o partida de poker cu 5 carti.

**Capitolul 2 Descrierea proiectului**

2.1 Diagrame UML

2.1.1 Cazuri de utilizare

2.1.2 Clase



2.1.3 Activitati

2.1.4 Secvente

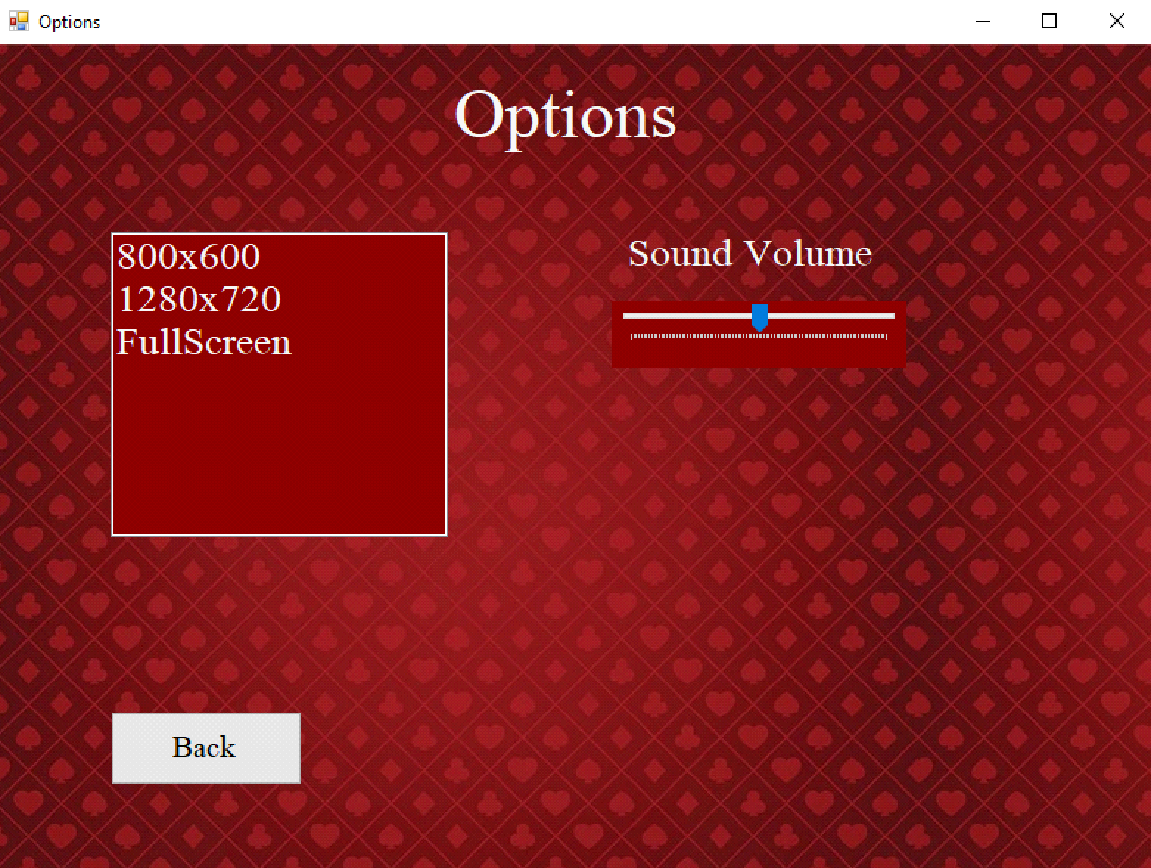
2.2 Programul in executie / Capturi ecran

Capturi de ecarn la functionarea programului:

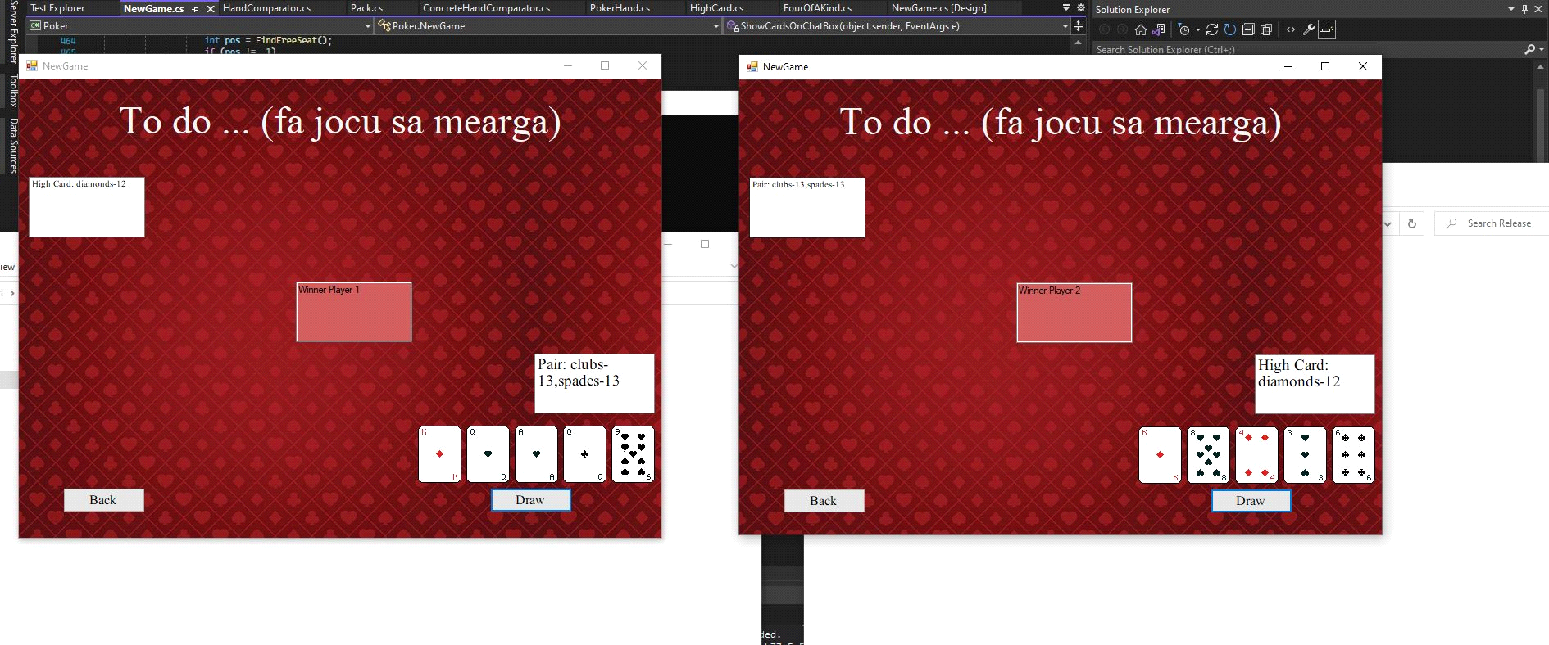
captura de ecran cu primul panou, Start Menu, al interfetei la rularea programului.



captura de ecran cu al doilea panou, Options, al interfetei la rularea programului.



captura de ecran cu al doilea panou, Options, al interfetei la rularea programului.



**2.2.1 Codul sursa**

Această clasă abstractă este utilizată în implementarea tuturor comparatorilor utilizați în determinarea rank-urilor unei mâini de poker.

using PokerGameLogic;

using System.Collections.Generic;

using System.Linq;

namespace PokerGameLogic.AbstractClasses

{

public abstract class PokerHand

{

protected List<Card> hand = new List<Card>();

protected string cardRanks = "High Card: ";

public List<Card> Hand {

get { return hand; }

}

virtual public bool IsHand(List<Card> cards)

{

//aici trebuie puse ceva exceptii

if (cards.Count == 0)

return false;

List<Card> orderedCards = cards.OrderBy(o=>o.CardNumber).ToList();

if (orderedCards[0].CardNumber == 1)

{

hand.Add(orderedCards[0]);

return true;

}

hand.Add(orderedCards[orderedCards.Count-1]);

return true;

}

public string RankingToString()

{

string result = "";

for (int i=0; i<hand.Count-1; i++)

result+=hand[i].GetCard() + ",";

result += hand[hand.Count - 1].GetCard();

return cardRanks+result;

}

virtual public void Clear()

{

hand.Clear();

}

public string HandType {

get

{

return cardRanks;

}

}

}

}

/////////////////////////////////////////////////////////////

Clasa care se ocupă cu compararea a două seturi de cărți. (Șablon de proiectare singleton)

using PokerGameLogic.AbstractClasses;

using PokerGameLogic.Interfaces;

using System.Collections.Generic;

namespace PokerGameLogic

{

public class ConcreteHandComparator : HandComparator

{

private List<string> \_ranking;

static ConcreteHandComparator \_comparator = null;

private ConcreteHandComparator()

{

\_ranking = new List<string> {"High Card", "Pair", "Two Pairs", "Three of a Kind", "Straight", "Flush", "Full House", "FourOfAKind", "Straight flush", "Royal Flush" };

}

public static ConcreteHandComparator GetComparator()

{

if(\_comparator == null)

\_comparator = new ConcreteHandComparator();

return \_comparator;

}

public int CompareHands(string handA, string handB)

{

string []handARanking = handA.Split(':');

string []handBRanking = handB.Split(':');

int indexA = \_ranking.IndexOf(handARanking[0].Replace(" ", string.Empty));

int indexB = \_ranking.IndexOf(handBRanking[0].Replace(" ", string.Empty));

if (indexA > indexB)

return 1;

else if (indexA < indexB)

return -1;

else

{

string[] cardsA = handARanking[1].Split(',');

string[] cardsB = handBRanking[1].Split(',');

for (int i = 0; i < cardsA.Length; i++)

{

string cardNumberA = cardsA[i].Split('-')[1];

string cardNumberB = cardsB[i].Split('-')[1];

if (int.Parse(cardNumberA) > int.Parse(cardNumberB))

return 1;

else if (int.Parse(cardNumberA) < int.Parse(cardNumberB))

return -1;

}

return 0;

}

}

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Serverul utilizat în cadrul acestui joc**

using System;

using System.Collections.Generic;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

namespace MySockets

{

public class Server

{

public class User

{

private Socket \_clientSocket = null;

private Queue<string> \_messages = null;

private Thread \_th = null;

private int \_userId;

public User(Socket socket, Queue<string> messages, int userId)

{

\_userId = userId;

\_th = new Thread(ReceiveMessage);

\_clientSocket = socket;

\_messages = messages;

\_th.Start();

}

public void ReceiveMessage()

{

try

{

while (true)

{ // Data buffer

byte[] bytes = new Byte[1024];

string data = null;

while (true)

{

int numByte = \_clientSocket.Receive(bytes);

data += Encoding.ASCII.GetString(bytes,

0, numByte);

if (data.IndexOf("<EOF>") > -1)

break;

}

string auxMsg = data.Replace("<EOF>", "");

if (auxMsg == "bye")

{

CloseConnection();

return;

}

Console.WriteLine("Received: "+ auxMsg);

\_messages.Enqueue(auxMsg);

}

}

catch (Exception e)

{

\_messages.Enqueue(e.ToString());

}

}

public bool SendMessage(string msg)

{

try

{

byte[] message = Encoding.ASCII.GetBytes(msg);

// Send a message to Client

// using Send() method

\_clientSocket.Send(message);

return true;

}

catch (Exception e)

{

\_messages.Enqueue(e.ToString());

return false;

}

}

public void CloseConnection()

{

try

{

\_messages.Enqueue(\_userId.ToString()+":disconnect");

\_clientSocket.Shutdown(SocketShutdown.Both);

\_clientSocket.Close();

\_th.Abort();

}

catch (Exception e)

{

\_messages.Enqueue(e.ToString());

}

}

}

private List<User> \_users = new List<User>();

private Queue<string> \_messages = new Queue<string>();

private List<bool> \_id = new List<bool>() {false, false, false, false, false};

public Server()

{

Thread th = new Thread(UpdateAllUsers);

th.Start();

StartServer();

}

private int GetFreeId()

{

for (int i = 0; i < \_id.Count; i++)

if (\_id[i] == false)

{

return i;

}

return -1;

}

private string AllUsersToString()

{

string result = "";

for (int i = 0; i < \_id.Count; i++)

if(\_id[i] == true)

result+=","+i.ToString();

return result;

}

void UpdateAllUsers()

{

while (true)

{

while (\_messages.Count != 0)

{

string message = \_messages.Dequeue();

Console.WriteLine(message+"\r\nUsers: "+\_users.Count);

for(int i=0; i<\_users.Count; i++)

{

if (!\_users[i].SendMessage(message))

{

\_users[i].CloseConnection();

\_users.RemoveAt(i);

--i;

\_id[i] = false;

}

}

}

}

}

public void StartServer()

{

// Establish the local endpoint

// for the socket. Dns.GetHostName

// returns the name of the host

// running the application.

IPHostEntry ipHost = Dns.GetHostEntry(Dns.GetHostName());

IPAddress ipAddr = ipHost.AddressList[0];

IPEndPoint localEndPoint = new IPEndPoint(ipAddr, 11111);

// Creation TCP/IP Socket using

// Socket Class Constructor

Socket listener = new Socket(ipAddr.AddressFamily,

SocketType.Stream, ProtocolType.Tcp);

try

{

// Using Bind() method we associate a

// network address to the Server Socket

// All client that will connect to this

// Server Socket must know this network

// Address

listener.Bind(localEndPoint);

// Using Listen() method we create

// the Client list that will want

// to connect to Server

listener.Listen(10);

while (true)

{

// Console.WriteLine("Waiting connection ... ");

// Suspend while waiting for

// incoming connection Using

// Accept() method the server

// will accept connection of client

int pos = GetFreeId();

if (pos != -1)

{

\_users.Add(new User(listener.Accept(), \_messages, pos));

string result = AllUsersToString();

\_users[\_users.Count - 1].SendMessage(pos.ToString() + result);

Thread.Sleep(500);

\_messages.Enqueue(pos+":connect");

\_id[pos] = true;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

}

}

}

}

2.3 Autorii acestui proiect :

Ciopeică Sergiu-Andrei 1308B

Tănăsache Cristian 1308B

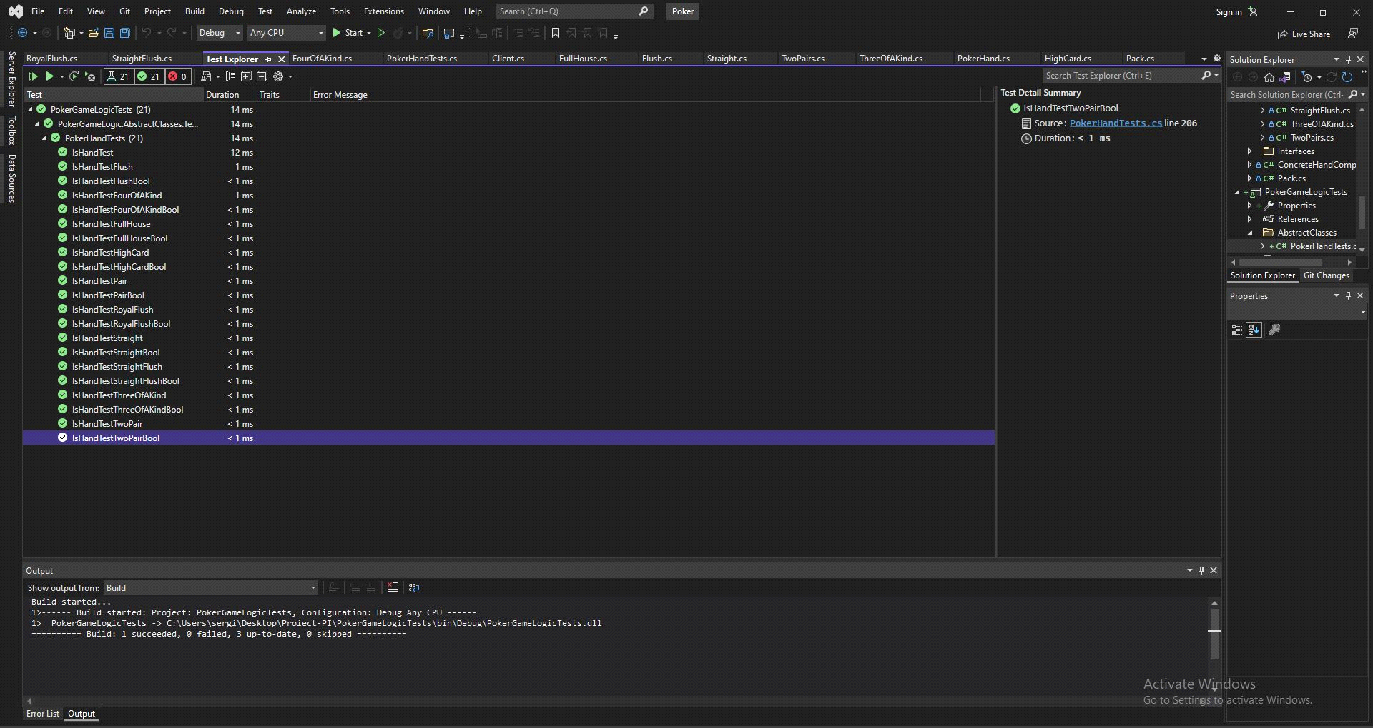
**Capitolul 3 Cerinte**

***3.1 Sablon de proiectare***

Ca sabloane de proiectare am folosit singleton si factory. Sablonul de proiectare singleton creaza premizele ca o anumita clasa sa fie instantiata doar o singura data, permitandu-se un acces global la instanta respectiva. Avantaje sunt ca controleaza accesul la instanta unica, deoarece clasa Singleton incapsuleaza propria instant, conduce la reducerea spatiului de nume, deoarece nu implica folosirea variabilelor globale si asigura o flexibilitate mai mare decat daca s-ar utiliza metode statice. O alternativa la sablonul Singleton ar fi fost ca toate functiunile clasei Singleton sa fie realizate cu ajutorul unor metode statice (practic in acest caz am fi avut de a face cu 0 instante, nu cu una). Factory este sablonul care perminte crearea unei noi ferestre pentru a avea acces la joc. Am folosit aceste sabloane de proiectare pentru a realiza mobilitatea mai usoara pe joc. In acest proiect exista un o clasa numita public class ConcreteHandComparator ce apartine sablonului singleton si o alta clasa numite public class GameStatesFactory ce apartine sablonului factory.

***3.2 Testarea unitatilor***

Se testeaza scenarii pentru user si pentru cazurile diferite de maini de joc pe care le poti avea. De exemplu, daca numele cartile este introdus corect, daca mainile de joc sunt corect de numite si functioneaza. S-au facut 21 de teste in privinta aceasta, si dupa cum se va putea observa in imagina atasata mai jos, toate acestea au fost cu success.



**Capitolul 4 Despre, reguli si strategii**

Dacă ești complet nou pe partea de poker, trebuie să știi cum se clasifică mâinile înainte de orice altceva. Dacă nu o faci, ai putea avea o mână câștigătoare și nici măcar nu-ți dai seama! Așadar, înainte de a aborda ceva specific jocului cu 5 cărți sau sa intram in orice altceva tehnic, haideți să stabilim ierarhia mâinilor, începând cu cea mai mică:

* Carte mare (aka nimic)
* O pereche
* Două perechi
* Trei de un fel
* Chintă
* Culoare
* Full house
* Patru de un fel
* Chintă de culoare
* Chintă roială
* Cinci de un fel (dacă se joacă cu un wild)

**4.1 Regulile jocului**

Pokerul se joacă cu un pachet standard de 52 de cărți. (Unele variante de jocuri folosesc pachete multiple sau adaugă câteva cărți numite jokeri.) Cărțile sunt clasificate (de la mare la mic) As, Rege, Damă, Valet, 10, 9, 8, 7, 6, 5, 4, 3, 2, As. (Asul poate fi mare sau mic, dar de obicei este mare). Există patru culori (pică, inimă, romb și treflă); cu toate acestea, nicio culoare nu este mai mare decât alta. Toate mâinile de poker conțin cinci cărți, iar cea mai mare mână câștigă.

Unele jocuri au cărți wild, care pot lua orice culoare și rang dorește posesorul lor. Uneori, jokerii vor fi folosiți ca jokeri, alteori, jocul va specifica ce cărți sunt wild (duci, valeți cu un singur ochi sau orice altceva).

Deci, cum pariezi? Pokerul este, până la urmă, un joc de noroc. În cele mai multe jocuri, trebuie să "anteziți" ceva (suma variază în funcție de joc, în jocurile noastre este de obicei o monedă de 5 lei), doar pentru a primi cărțile. După aceea, jucătorii pariază în potul din mijloc. La sfârșitul mâinii, mâna cea mai mare (care nu a renunțat) câștigă potul. Practic, atunci când pariurile ajung la tine (de obicei, pariurile se fac în ordinea acelor de ceasornic), ai una dintre cele trei opțiuni:

Să dai call

Când pariați call, pariați suficient pentru a vă potrivi cu ceea ce s-a pariat de la ultima dată când ați pariat (de exemplu, dacă ați pariat 10 lei data trecută, iar altcineva a pariat 25 lei, veți fi dator cu 15 lei).

Raise

Atunci când mizați, mai întâi pariați suficient de mult pentru a vă potrivi cu ceea ce s-a pariat de la ultima dată când ați pariat (ca la call), apoi "măriți" pariul cu încă o sumă (până la dvs., dar de obicei există o limită.) Continuând cu exemplul de mai sus, dacă ați pariat o monedă de 10 lei, iar cealaltă persoană a mărit cu 15 lei (până la 25 lei), ați putea mări cu 25 lei (până la 50 de lei). Deoarece datorați potului 15 lei pentru call și 25 pentru raise, ați pune 40 lei în pot.

Fold

Atunci când renunțați, vă retrageți din mâna curentă (pierzând orice posibilitate de a câștiga potul), dar nu trebuie să puneți bani în pot.

Pariurile continuă până când toată lumea plătește sau se retrage după un raise sau un pariu inițial.

**4.2 Strategii de baza**

* Joacă mai puține mâini și joacă-le agresiv.
* Nu fiți primul jucător care se lasă păgubaș.
* "Semi-Bluff" în mod agresiv cu draw-urile tale.
* Joacă-ți rapid mâinile puternice pentru a construi potul și a face mai mulți bani.
* Apără-ți Blind-ul Mare (cu mâinile potrivite)
* Renunțați atunci când nu sunteți sigur

**Concluzii**