МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ  
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| д.т.н. проф. |  |  |  | Охтилев М.Ю. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ |
| РАЗРАБОТКА Игры «ПОКЕР» |
| по дисциплине: Объектно ориентированное программирование |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4531 |  |  |  | В.А.Муратова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2017

Оглавление

[Постановка задачи, описание правил игры 3](#_Toc501963626)

[Модули игры 5](#_Toc501963627)

[Интерфейсы и паттерны игры 6](#_Toc501963628)

[Описание тестов 7](#_Toc501963629)

[Диаграммы классов 9](#_Toc501963630)

[Руководство пользователя 10](#_Toc501963631)

[Заключение 14](#_Toc501963632)

[Библиографический список 15](#_Toc501963633)

[Приложение А 16](#_Toc501963634)

[Листинги программы 16](#_Toc501963635)

# Постановка задачи, описание правил игры

Целью данного курсового проекта является создание игры «Покер». Для реализации логики игры требуется изучить и грамотно подобрать паттерны проектирования.

Игра создана на объектно-ориентированном языке программирования С#. Представляет собой логическую игру, где основная задача игрока выиграть ставки, собрав как можно более высокую покерную комбинацию, используя 5 карт, или вынудив соперника (бота) прекратить участвовать в игре. Соперник игрока оснащен искусственным интеллектом. Игра идёт с полностью закрытыми картами соперника.

В покер играют колодой —52 карты. Игра состоит из нескольких фаз раундов, каждая из них начинается с раздачи новых карт.

Каждый игрок имеет банк, где хранятся его средства, которые он может преувеличивать(преуменьшать) во время игр. Изначально игрокам дается 5000$. Если игрок проигрывает все средства, ему заново дается 5000$.

Организация игры:

Каждый игрок получает по две карты в закрытую, следует круг торговли.

На стол перед всеми игроками кладутся три карты в открытую (флоп), следует круг торговли.

На стол кладётся четвёртая карта в открытую (тёрн), следует круг торговли.

На стол кладётся пятая карта в открытую (ривер). Таким образом, на столе лежит 5 карт, следует последний круг торговли.

Игроки могут использовать свои (карманные) и общие карты со стола для составления 5-карточных комбинаций.

Комбинации, которые обрабатываются в игре:

Роял-флэш - состоит из 5 старших (туз, король, дама, валет, десять) карт одной масти.

Стрит- флэш - любые пять карт одной масти по порядку.

Каре - четыре карты одного достоинства.

Фулл-хаус - одна тройка и одна пара.

Флэш - пять карт одной масти

Стрит пять карт по порядку любых мастей.

Сет - три карты одного достоинства.

Две пары - две пары карт.

Одна пара - две карты одного достоинства.

Старшая карта - ни одна из вышеописанных комбинаций.

Торговля

В процессе торговли игрок может делать следующие ставки:

Поставить — сделать ставку;

Поставить все деньги — Сделать ставку на все деньги;

Ответить — поставить столько же, сколько поставил соперник — уравнять;

Поднять — увеличить ставку — поставить больше, чем соперники;

Пасовать — отказаться от дальнейшего участия и сбросить карты;

Отметиться — в ситуациях, когда ставки не были сделаны соперниками — не добавлять ставку, оставить «как есть».

Круг торговли заканчивается, когда все соперники сделали равные ставки или сбросили карты.

Сделанные ставки складываются в банк. Если в процессе последнего круга торговли осталось больше одного игрока, то карты открываются и комбинации игроков сравниваются между собой. Если только один игрок остался в игре, то он забирает банк. Если больше одного игрока победили в игре, банк делится поровну между всеми выигравшими.

# Модули игры

Модули игры:

* Окно пользователя – позволяет пользователю задать имя, фото и просмотреть свой счет банка. А также добавить нового пользователя.
* Окно игры (стол) – содержит дилера, информацию о пользователе и роботе – счет банка, имя и игровые кнопки.
* Объекты игры – содержат конструкторы основных элементов игры и представляют интерфейсы.
* Модуль логики – содержит логику игры, подсчет результатов.
* Модуль работы с БД.

# Интерфейсы и паттерны игры

В данном проекте реализованы следующие паттерны проектирования:

1. Singleton (одиночка) - то порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.[5]
2. MVC (Model-View-Controller) – структурный Шаблон проектирования предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо. Модель (Model) предоставляет данные предметной области представлению и реагирует на команды контроллера, изменяя свое состояние. Представление (View) отвечает за отображение данных предметной области (модели) пользователю, реагируя на изменения модели. Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.
3. Strategy — это поведенческий паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс. После чего, алгоритмы можно взаимозаменять прямо во время исполнения программы.[5]

# Описание тестов

Данный набор тестов разработан таким образом, чтобы покрыть весь функционал игры, и выявить возможные несоответствия алгоритма или коллизии.

|  |  |  |
| --- | --- | --- |
| Тест | Описание | Результат |
| Раздача карт | Нажатие на кнопки Flop, Tern, River. | Вывод новых карт на стол |
| Поднять ставку (бот поддержал ставку) | Нажатие на кнопку «Call» | Формирование банка, с учетом поддержки ставки бота |
| Сброс карт пользователем | Нажатие на кнопку “Fold” | Присвоение банка сопернику (боту) и сброс карт |
| Поднять ставку (бот не поддержал ставку) | Нажатие на кнопку «Call» | Сброс карт бота и присвоение банка игроку |
| Сброс карт ботом | Программное «нажатие на кнопку» “Fold” | Присвоение банка игроку и сброс карт |
| Поднятие ставки | Больше, чем ставка бота | Ставка поднимается на максимально возможную ставку бота, остальное возвращается игроку |
| Поднятие ставки | Больше, чем ставка игрока | Ставка поднимается на максимально возможную ставку игрока, остальное возвращается боту |
| Результаты игры | Выигрывает бот | Открываются карты бота, выводится сообщение победителю и банк бота увеличивается. |
| Результаты игры | Выигрывает игрок | Не открываются карты бота, выводится сообщение победителю и банк игрока увеличивается. |
| Результаты игры | Никто не выигрывает | Выводится сообщение о ничьей и банк делиться поровну. |

# Диаграммы классов

Опишем общую структуру диаграммы классов для паттерна Singleton и Strategy

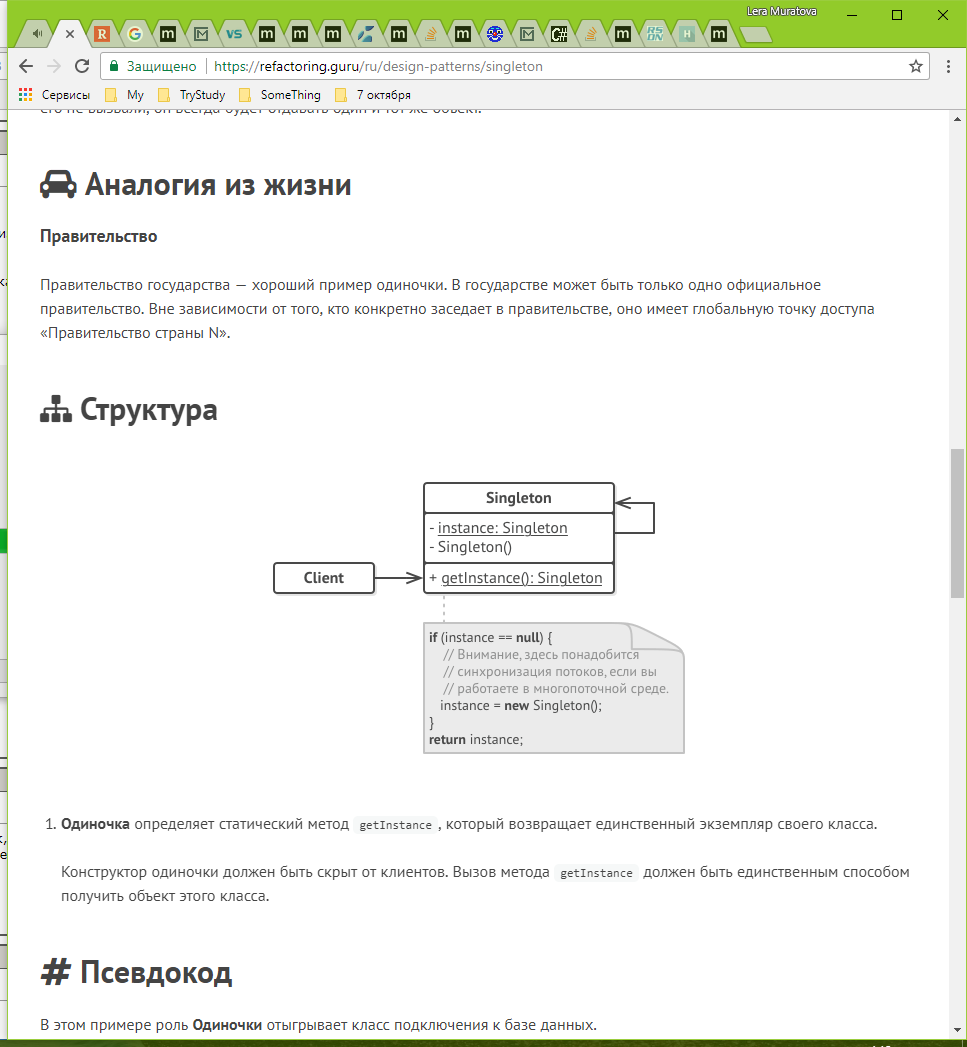


Рисунок 1 - Обобщенная структура паттерна Singleton

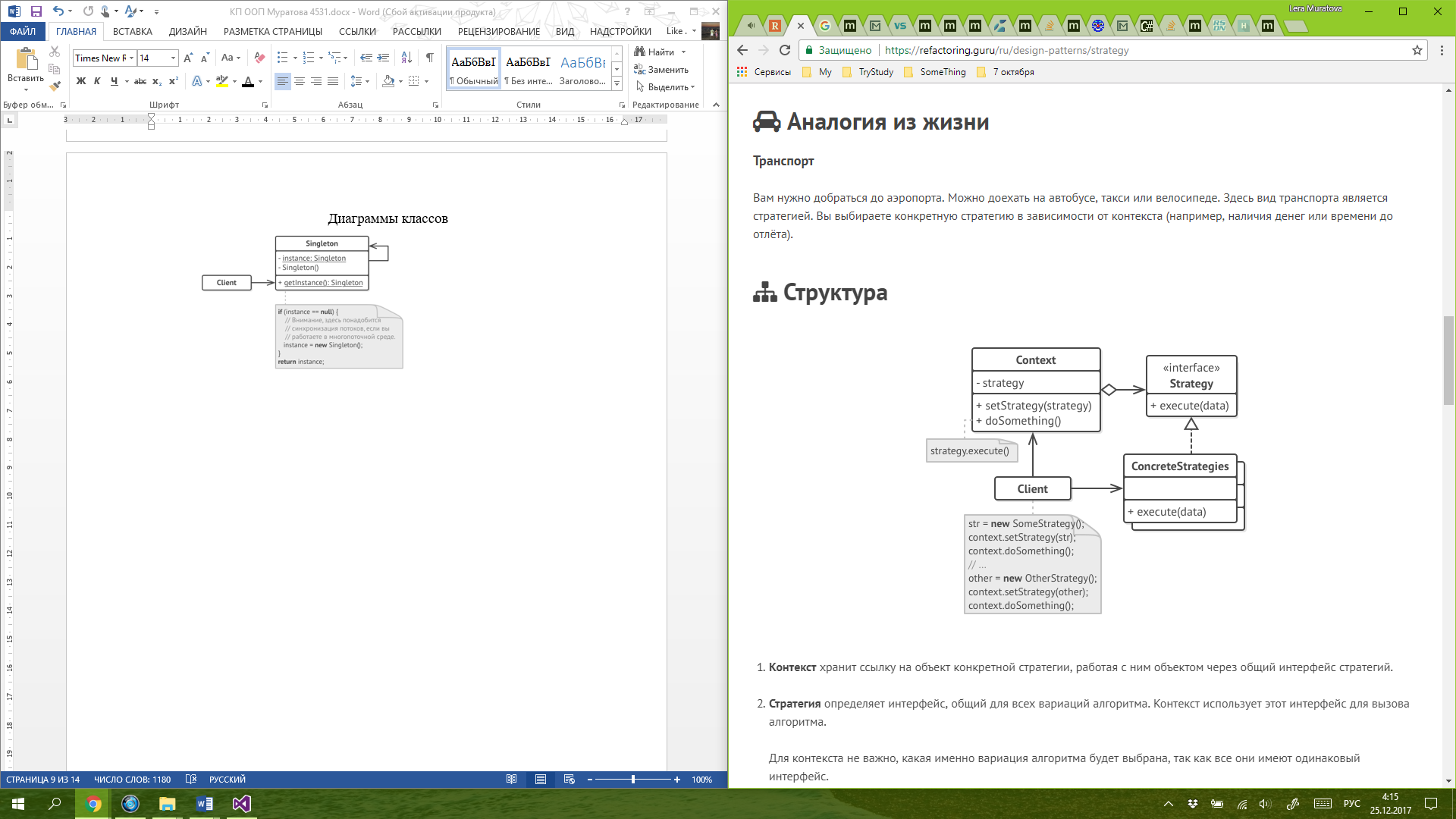


Рисунок 2 - Обобщенная диаграмма классов паттерна Strategy

# Руководство пользователя

Данное приложение легко в использовании. При входе Вам необходимо ввести свое имя и, если вы уже были зарегистрированы, то Ваши данные выведутся на экран, если же нет – предложим пополнить наши ряды. Также вы можете загрузить картинку, правда, пока, это достаточно бесполезная функция, но приятно. Осталось сделать один шаг – нажать кнопку «Поехали».

Дальше вы начинаете игру. За столом Вы и очень умный (но не умнее Вас) робот «NLP».

Начать игру Вам позволит кнопка «Flop», которая раздаст Вам и Роботу по 2 карманных карты и три на стол (флоп). Дальше Робот и Вы делаете ставки, если кто-то из Вас поднял ставки, вы не можете не поддержать ее – только сбросить или поднять. Потом по нажатию на кнопку «Tern», раздается еще одна карта на стол (Терн). Аналогично проводятся торги. И последняя раздача – кнопка “River”, выкладывает еще одну карту и вычисляет победителя.

Чтобы продолжить игру достаточно снова кликнуть по кнопке «Flop».

Если вы или робот сбросили – банк присвоится победителю и надо будет опять нажать кнопочку «Flop.

Если Вы или робот настолько азартны, что проиграли все, не расстраивайтесь, эта игра даст Вам бесконечное множество шансов – Банк заново пополниться до 5000$.

При закрытии формы все данные сохраняться.

Все очень просто и доступно. Желаем Вам Удачи и завидных комбинаций!

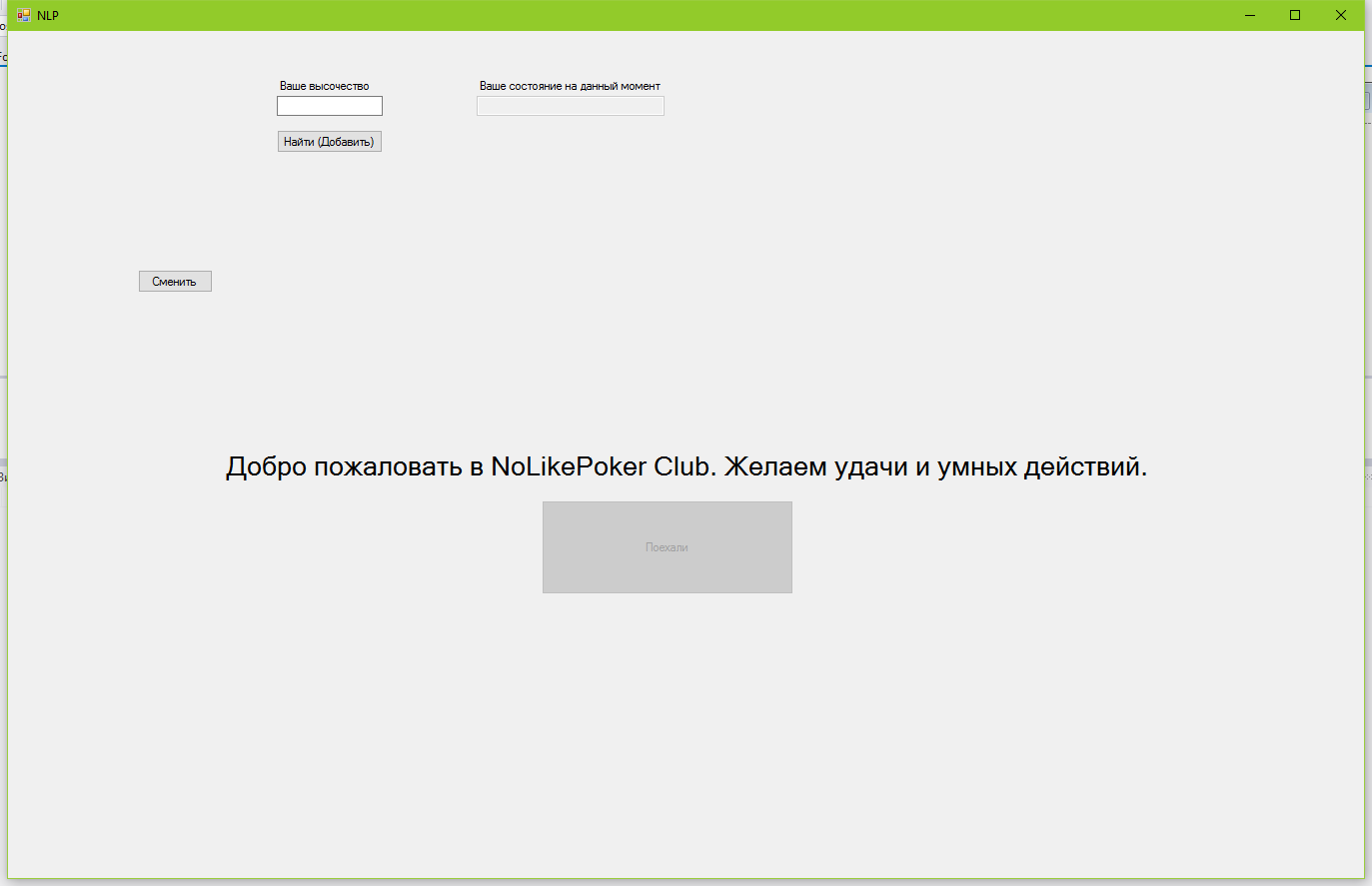


Рисунок 3 - Пример стартовой страницы

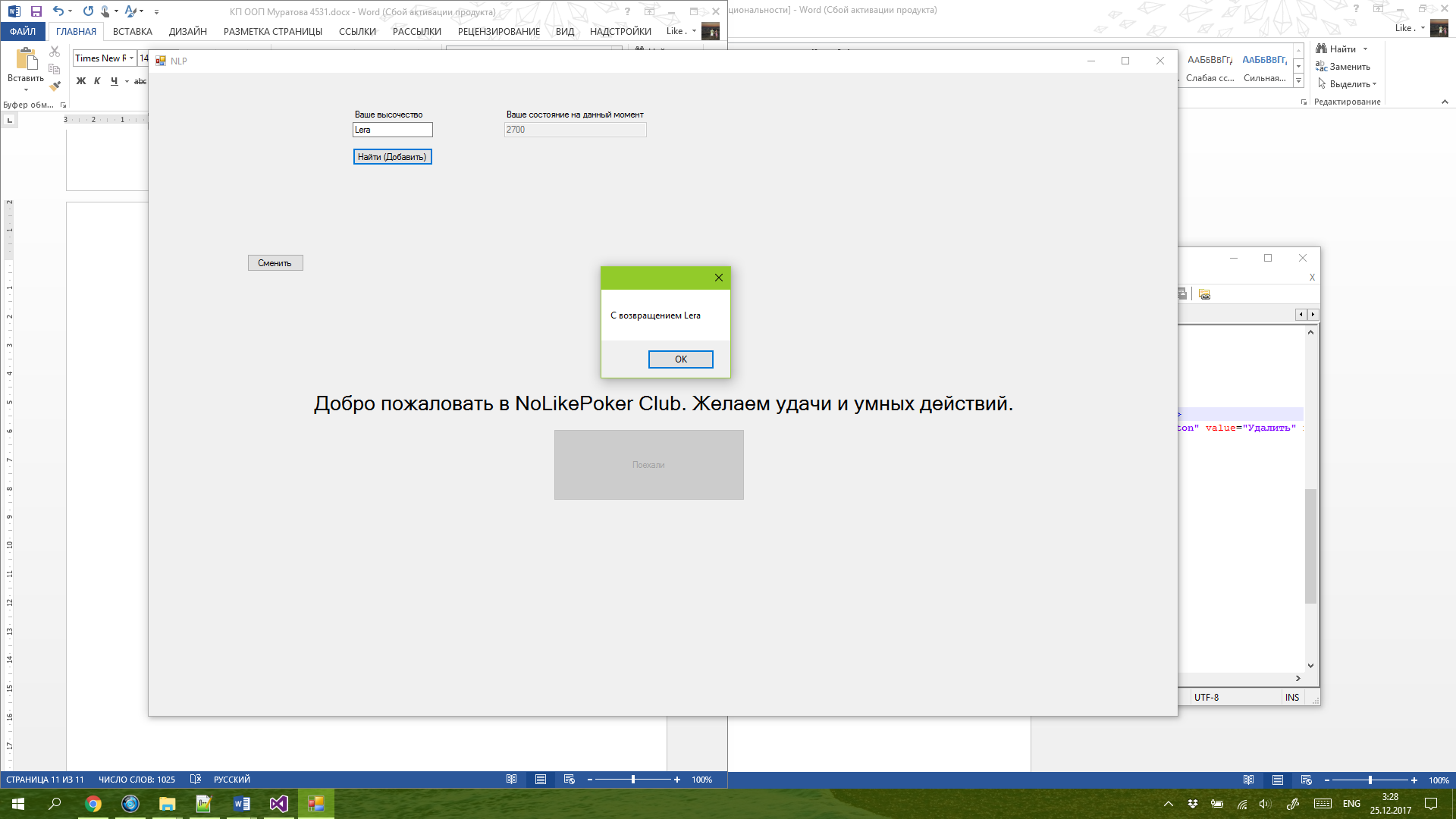


Рисунок 4 - Пример считывания из БД

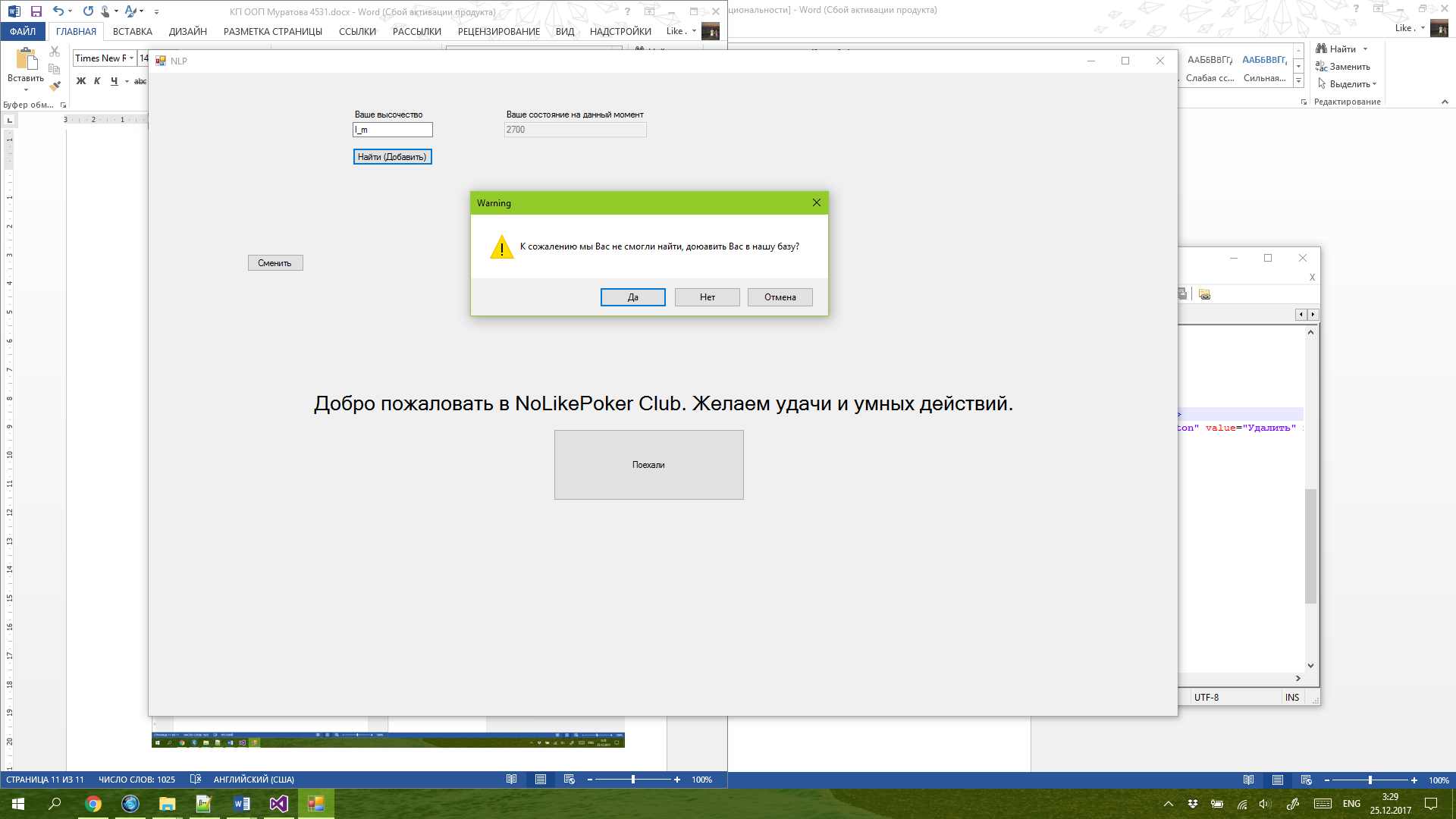


Рисунок 5 - Пример добавления в БД

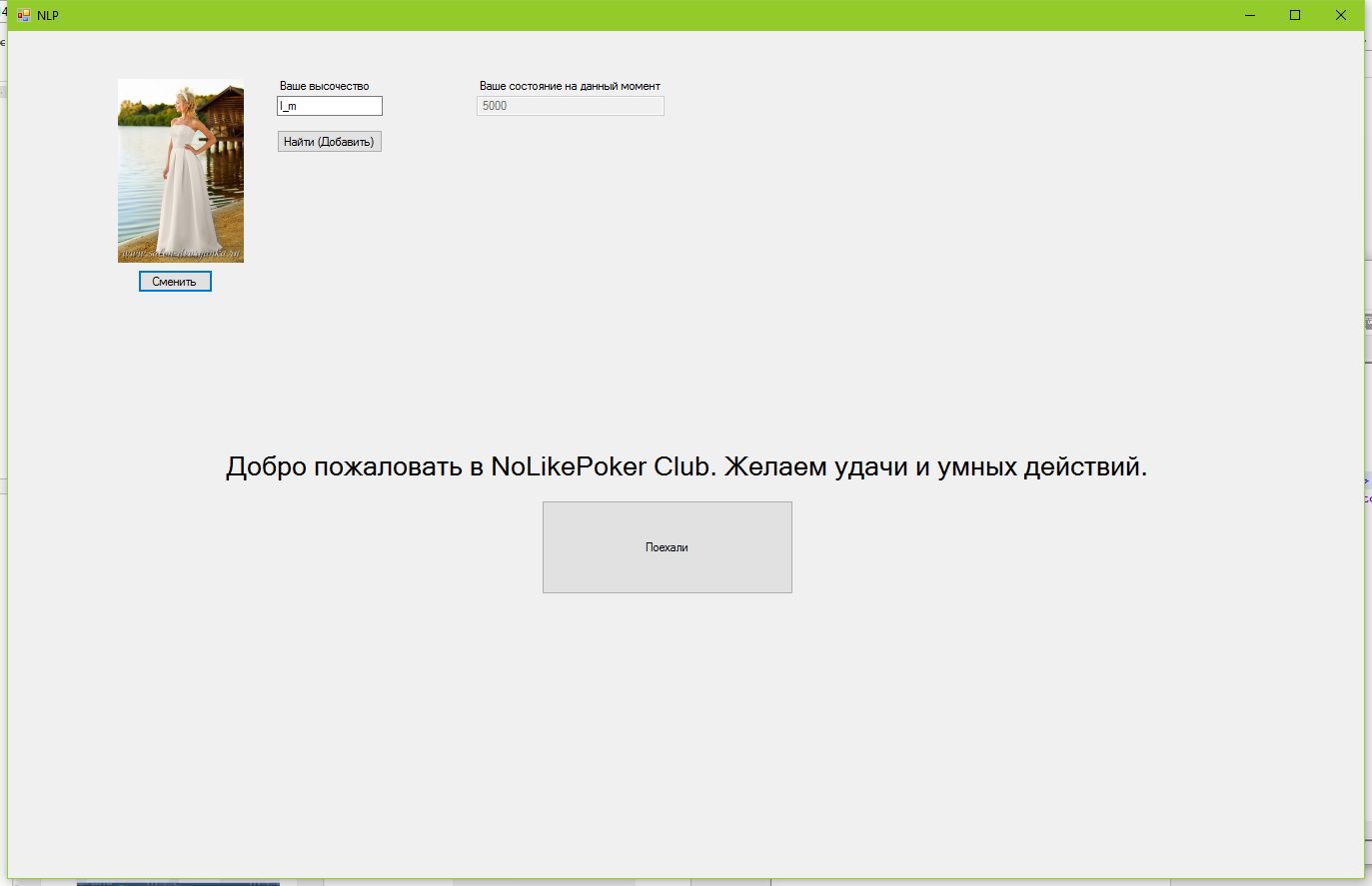


Рисунок 6 - Пример добавления в БД

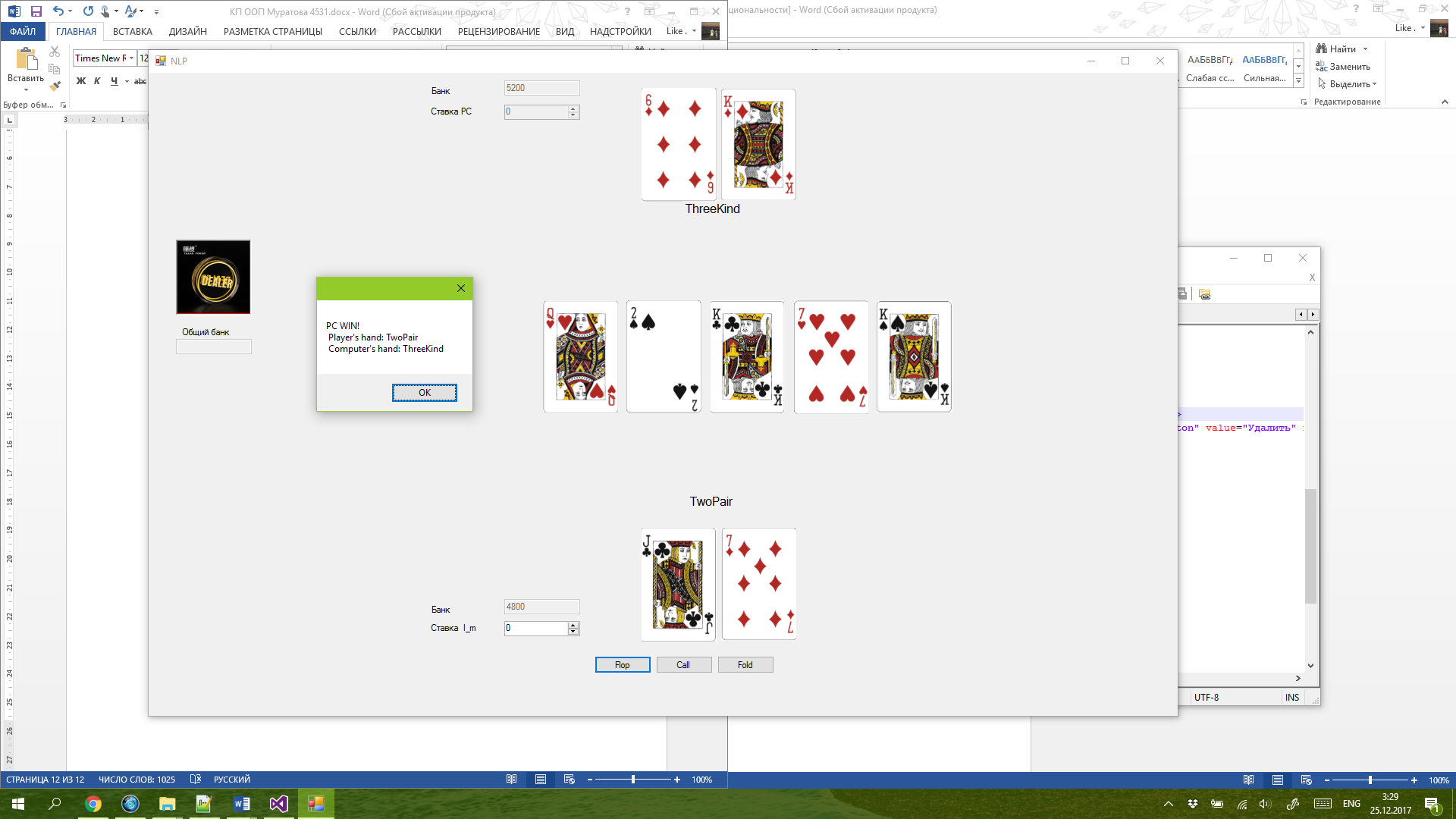


Рисунок 7 - Пример игрового процесса

# Заключение

В ходе выполнения карового проекта были закреплены на практике патеррны проектирования, изученные в ходе курса лекций ООП. Логика игры Покер достаточно непроста в реализации, но паттерны проектирования помогли упростиь некоторые части. Благодаря встроенным возможностям великолепного языка С# - лямбда функциям, алогритм поиска комбинации прилично сократился. Поскольку целью проекта является реализация логики, акцентироаванеи на дизайн не было, но в будущих релизах все можно с легкостью исправить. Игра является легкой в использовании и не занимает много места. В будущем игру можно будет реализовать для сервера, чтобы могли играть много игроков.

С# является объектно-ориентированным языком и позволяет легко и реализовать такие приложеия, как в нашем курсовом проекте, с чем и связан выбор данной среды разработки.

# Библиографический список

1. Э.Грамм, Р.Хелм, Р.Джонсон, Д.Влиссидес. «Приемы объектно-ориентированного проектирования. Паттерны проектирования» - Питер, 2016г. – 366с.
2. Д.Албхари. «C# 5.0. Справочник. Полное описание языка». – Вильямс, 2013г. – 1008с.
3. Рефакторинг.Гуру [Электронный ресурс] – Режим доступа: https://refactoring.guru
4. Сpp-reference [Электронный ресурс] – <http://cpp-reference.ru>
5. Каталог паттернов проектирования [Электронный ресурс] – <https://refactoring.guru/ru/design-patterns/catalog>

# Приложение А

# Листинги программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MyPoker

{

public enum Hand

{

Nothing, OnePair, TwoPair, ThreeKind,

Straight, Flush, FullHouse, FourKind,

FlushStraight, RoyalFlush

}

public struct HandValue

{

public int Total { get; set; }

public int HighCard { get; set; }

public int Aut { get; set; }

}

class HandEvaluator : Card

{

private Card[] cards;

private HandValue handValue;

private Control.ControlCollection Control;

public HandEvaluator(Card[] sortedHand)

{

// int i = Array.FindAll(sortedHand, element => element.count >0).Length;

cards = new Card[sortedHand.Length];

Cards = sortedHand;

handValue = new HandValue();

}

public void setControl(Control.ControlCollection control)

{

Control = control;

}

public HandValue HandValues

{

get { return handValue; }

set { handValue = value; }

}

public Card[] Cards

{

get { return cards; }

set

{

for (int i = 0, j = 0; i < cards.Length; i++)

{

cards[j] = value[i];

j++;

}

}

}

public Hand EvaluateHand()

{

RobotStat();

int k=Straight();

//get number of each suit on hand

// getNumberOfSuit();

if (FourOfKind())

return Hand.FourKind;

else if (FullHouse())

return Hand.FullHouse;

else if (Flush())

return Hand.Flush;

else if (k==3)

return Hand.Straight;

else if (k == 2)

return Hand.FlushStraight;

else if (k == 1)

return Hand.RoyalFlush;

else if (ThreeOfKind())

return Hand.ThreeKind;

else if (TwoPairs())

return Hand.TwoPair;

else if (OnePair())

return Hand.OnePair;

//if nothing

handValue.HighCard = (int)cards[cards.Length-1].MyValue;

return Hand.Nothing;

}

private void RobotStat()

{

handValue.Aut = 0;

int k = Straight();

handValue.Aut += (Array.FindAll(cards, element => element.count == 4).Length \* 8);

handValue.Aut += ((Array.FindAll(cards, element => element.count == 3).Length + Array.FindAll(cards, element => element.count == 2).Length) \* 7);

handValue.Aut += ((Array.FindAll(cards, element => element.MySuit == Card.SUIT.HEARTS).Length + Array.FindAll(cards, element => element.MySuit == Card.SUIT.CLUBS).Length + Array.FindAll(cards, element => element.MySuit == Card.SUIT.DIAMONDS).Length + Array.FindAll(cards, element => element.MySuit == Card.SUIT.SPADES).Length) \* 4);

handValue.Aut += (Array.FindAll(cards, element => element.count == 3).Length \* 4);

handValue.Aut += (Array.FindAll(cards, element => element.count == 2).Length \* 3);

handValue.Aut += (Array.FindAll(cards, element => element.count == 1).Length \* 2);

if (k == 1)

handValue.Aut += (Array.FindAll(cards, element => element.count == 1).Length \* 10);

if (k == 2)

handValue.Aut += (Array.FindAll(cards, element => element.count == 1).Length \* 9);

if (k == 3)

handValue.Aut += (Array.FindAll(cards, element => element.count == 1).Length \* 5);

}

private bool FourOfKind()

{

if (Array.FindAll(cards, element => element.count ==4).Length==1)

{

VALUE total=Array.FindAll(cards, element => element.count == 4)[0].MyValue;

handValue.Total = (int)total \* 4;

handValue.HighCard = (int)Array.FindAll(cards, element => element.MyValue != total)[Array.FindAll(cards, element => element.MyValue != total).Length-1].MyValue;

return true;

}

return false;

}

private bool FullHouse()

{

if ((Array.FindAll(cards, element => element.count == 3).Length == 1) && (Array.FindAll(cards, element => element.count == 2).Length >= 1))

{

handValue.Total = (int)Array.FindAll(cards, element => element.count == 3)[0].MyValue \* 3 + (int)Array.FindAll(cards, element => element.count == 2)[0].MyValue \* 2;

return true;

}

return false;

}

private bool Flush()

{

if (Array.FindAll(cards, element => element.MySuit == Card.SUIT.HEARTS).Length == 5)

{

handValue.Total = (int)Array.FindAll(cards, element => element.MySuit == Card.SUIT.HEARTS)[Array.FindAll(cards, element => element.MySuit == Card.SUIT.HEARTS).Length-1].MyValue;

return true;

}

else

if (Array.FindAll(cards, element => element.MySuit == Card.SUIT.CLUBS).Length == 5)

{

handValue.Total = (int)Array.FindAll(cards, element => element.MySuit == Card.SUIT.CLUBS)[Array.FindAll(cards, element => element.MySuit == Card.SUIT.CLUBS).Length-1].MyValue;

return true;

}

else

if (Array.FindAll(cards, element => element.MySuit == Card.SUIT.DIAMONDS).Length == 5)

{

handValue.Total = (int)Array.FindAll(cards, element => element.MySuit == Card.SUIT.DIAMONDS)[Array.FindAll(cards, element => element.MySuit == Card.SUIT.DIAMONDS).Length-1].MyValue;

return true;

}

else

if (Array.FindAll(cards, element => element.MySuit == Card.SUIT.SPADES).Length == 5)

{

handValue.Total = (int)Array.FindAll(cards, element => element.MySuit == Card.SUIT.SPADES)[Array.FindAll(cards, element => element.MySuit == Card.SUIT.SPADES).Length-1].MyValue;

return true;

}

return false;

}

private int Straight()

{

Card[] total = Array.FindAll(cards, element => element.count == 1);

if (total.Length == 5)

{

int i;

for (i = 1; i < 5; i++)

{

if (total[i - 1].MyValue + 1 != total[i].MyValue)

break;

}

if (i == 5)

{

if (Flush())

{

if (total[4].MyValue == Card.VALUE.ACE)

{

handValue.Total = 8888;

return 1;

}

else

{

handValue.Total += (int)(Array.FindAll(cards, element => element.count == 1))[(Array.FindAll(cards, element => element.count == 1)).Length - 1].MyValue;

return 2;

}

}

handValue.Total = (int)(Array.FindAll(cards, element => element.count == 1))[(Array.FindAll(cards, element => element.count == 1)).Length-1].MyValue;

return 3;

}

}

return -1;

}

private bool ThreeOfKind()

{

if (Array.FindAll(cards, element => element.count == 3).Length == 1)

{

VALUE total = Array.FindAll(cards, element => element.count == 3)[0].MyValue;

handValue.Total = (int)total \* 4;

handValue.HighCard = (int)Array.FindAll(cards, element => element.MyValue != total)[Array.FindAll(cards, element => element.MyValue != total).Length-1].MyValue;

return true;

}

return false;

}

private bool TwoPairs()

{

if (Array.FindAll(cards, element => element.count == 2).Length >= 2)

{

int i = 0;

if (Array.FindAll(cards, element => element.count == 2).Length == 3)

i = 1;

Card[] total = Array.FindAll(cards, element => element.count == 2);

handValue.Total = (int)total[i].MyValue \* 2 + (int)total[i+1].MyValue \* 2;

handValue.HighCard = (int)Array.FindAll(cards, element => ((element.MyValue != total[0].MyValue) && (element.MyValue != total[i+1].MyValue)))[Array.FindAll(cards, element => ((element.MyValue != total[i].MyValue) && (element.MyValue != total[i+1].MyValue))).Length-1].MyValue;

return true;

}

return false;

}

private bool OnePair()

{

if (Array.FindAll(cards, element => element.count == 2).Length == 1)

{

VALUE total = Array.FindAll(cards, element => element.count == 2)[0].MyValue;

handValue.Total = (int)total \*2;

//string str = "Picbox" + Array.FindAll(cards, element => element.count == 2)[0].MyValue + "\_" + Array.FindAll(cards, element => element.count == 2)[0].MySuit;

//PictureBox pic = Control.Find(str, true).FirstOrDefault() as PictureBox;

//pic.Location = new System.Drawing.Point(pic.Location.X, pic.Location.Y - 20);

//str = "Picbox" + Array.FindAll(cards, element => element.count == 2)[1].MyValue + "\_" + Array.FindAll(cards, element => element.count == 2)[1].MySuit;

//pic = Control.Find(str, true).FirstOrDefault() as PictureBox;

//pic.Location = new System.Drawing.Point(pic.Location.X, pic.Location.Y - 20);

handValue.HighCard = (int)Array.FindAll(cards, element => element.MyValue != total)[Array.FindAll(cards, element => element.MyValue != total).Length-1].MyValue;

return true;

}

return false;

}

}}