|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДАЮ |  |  |  | УТВЕРЖДАЮ |
| (должность) |  |  |  | (должность) |
|  |  |  |  |  |
| (ФИО) |  |  |  | (ФИО) |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г.  М.П. |  |  |  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г.  М.П. |

**ПРИЛОЖЕНИЕ-ИМИТАЦИЯ ИГРЫ «КАЗИНО»**

**Руководство программиста**

**ЛИСТ УТВЕРЖДЕНИЯ**

**ХХХ.ХХХХХХХХ.ХХХХХ-01 01 03-ЛУ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| СОГЛАСОВАНО |  | СОГЛАСОВАНО |  | СОГЛАСОВАНО |
| (должность) |  | (должность) |  | (должность) |
|  |  |  |  |  |
| (ФИО) |  | (ФИО) |  | (ФИО) |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г.  М.П. |  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г.  М.П. |  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г.  М.П. |
| СОГЛАСОВАНО |  | СОГЛАСОВАНО |  | СОГЛАСОВАНО |
| (должность) |  | (должность) |  | (должность) |
|  |  |  |  |  |
| (ФИО) |  | (ФИО) |  | (ФИО) |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г.  М.П. |  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г.  М.П. |  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_20\_\_ г.  М.П. |

2020

УТВЕРЖДЕН

ХХХ.ХХХХХХХХ.ХХХХХ-01 01 03-ЛУ

**ПРИЛОЖЕНИЕ-ИМИТАЦИЯ ИГРЫ «КАЗИНО»**

**Руководство программиста**

**ХХХ.ХХХХХХХХ.ХХХХХ-01 01 03**

**Листов 33**

2020

**АННОТАЦИЯ**

В данном программном документе приведено руководство программиста по настройке и использованию программы «Приложение-имитация игры “Казино”».

В разделе «Назначение и условия применения программы» указаны назначение и функции, выполняемые программой, условия, необходимые для выполнения программы (объем оперативной памяти, требования к составу и параметрам периферийных устройств, требования к программному обеспечению и т. п.).

В разделе «Характеристика программы» приведено описание основных характеристик и особенностей программы.

В разделе «Входные и выходные данные» приведено описание организации используемой входной и выходной информации.

В разделе «Сообщения» указаны тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

В Приложении приведены правила игр, реализованных в программе.

**СОДЕРЖАНИЕ**

[1. Назначение и условия применения программы 5](#_Toc56849253)

[1.1. Назначение программы 5](#_Toc56849254)

[1.2. Функции, выполняемые программой 5](#_Toc56849255)

[1.3. Условия, необходимые для выполнения программы 6](#_Toc56849256)

[1.3.1. Объем оперативной памяти 6](#_Toc56849257)

[1.3.2. Требования к составу периферийных устройств 6](#_Toc56849258)

[1.3.3. Требования к параметрам периферийных устройств 6](#_Toc56849259)

[1.3.4. Требования к программному обеспечению 6](#_Toc56849260)

[1.3.5. Требования к персоналу (программисту) 6](#_Toc56849261)

[2. Характеристика программы 7](#_Toc56849262)

[2.1. Описание основных характеристик программы 7](#_Toc56849263)

[2.1.1. Режим работы программы 7](#_Toc56849264)

[2.2. Описание основных особенностей программы 7](#_Toc56849265)

[2.3. Исходный код основных модулей программы 7](#_Toc56849266)

[2.3.1. Исходный код модуля игры «Игровые автоматы» 7](#_Toc56849267)

[2.3.2. Исходный код модуля игры «Русская рулетка» 8](#_Toc56849268)

[2.3.3. Исходный код модуля игры «Барабан» 9](#_Toc56849269)

[2.3.4. Исходный код методов по работе с уведомлениями 13](#_Toc56849270)

[2.3.5. Исходный код методов, связанных с аккаунтом пользователя 13](#_Toc56849271)

[2.3.6. Исходный код модуля регистрации пользователя 16](#_Toc56849272)

[2.3.7. Исходный код модуля изменения данных пользователя 18](#_Toc56849273)

[2.3.8. Исходный код метода изменения баланса 19](#_Toc56849274)

[3. Обращение к программе 21](#_Toc56849275)

[3.1. Запуск программы 21](#_Toc56849276)

[3.2. Информация о клиенте 21](#_Toc56849277)

[3.2.1. Просмотр данных 21](#_Toc56849278)

[3.2.2. Изменение данных 21](#_Toc56849279)

[3.2.3. Пополнение баланса 22](#_Toc56849280)

[3.3. Игровой режим игры «Игровые автоматы» 22](#_Toc56849281)

[3.4. Игровой режим игры «Русская рулетка» 22](#_Toc56849282)

[3.5. Игровой режим игры «Барабан» 23](#_Toc56849283)

[4. Входные и выходные данные 24](#_Toc56849284)

[4.1. Организация используемой входной информации 24](#_Toc56849285)

[4.2. Организация используемой выходной информации 24](#_Toc56849286)

[4.3. Кодирование используемой информации 24](#_Toc56849287)

[5. Сообщения 25](#_Toc56849288)

[5.1. Уведомления об ошибке ввода (заполнения текстовых полей) 25](#_Toc56849289)

[5.2. Уведомление об ошибке авторизации 28](#_Toc56849290)

[5.3. Уведомления об ошибке недостаточности баланса 28](#_Toc56849291)

[5.4. Уведомление об успешной регистрации 29](#_Toc56849292)

[5.5. Уведомление об успешном изменении данных пользователя 30](#_Toc56849293)

[5.6. Уведомление об успешном входе в комнату в игре «Русская рулетка» 30](#_Toc56849294)

[5.7. Уведомления о результате игры 31](#_Toc56849295)

[Приложение А. Правила игр 33](#_Toc56849296)

# **НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ**

* 1. **Назначение программы**

Программа предоставляет возможность играть в такие игры, как «Игровые автоматы», «Русская рулетка», «Барабан» на персональном компьютере. Кроме того, программа дает возможность пользователю зарегистрировать аккаунт для сохранения данных и последующего их использования при эксплуатации программы.

* 1. **Функции, выполняемые программой**

Программа обеспечивает возможность выполнения перечисленных ниже функций:

1. функции авторизации пользователя;
2. функции регистрации пользователя;
3. функции проверки корректности ввода;
4. функции уведомления пользователя в случае ошибок;
5. функции предоставления данных пользователя;
6. функции изменения данных пользователя;
7. функции пополнения баланса аккаунта;
8. функции сохранения данных пользователя после регистрации;
9. функции обновления данных пользователя после их изменения;
10. функции игрового процесса игры «Игровые автоматы» в соответствии с правилами игры (Приложение А);
11. функции игрового процесса игры «Русская рулетка» в соответствии с правилами игры (Приложение А);
12. функции игрового процесса игры «Барабан» в соответствии с правилами игры (Приложение А);
13. функции проверки достаточности баланса для участия в играх;
14. функции уведомления пользователя о результате игры (кроме игры «Русская рулетка»);
15. функции выхода из аккаунта и перехода к авторизации.
    1. **Условия, необходимые для выполнения программы**
       1. **Объем оперативной памяти**

Объем оперативной памяти должен составлять не менее 64 Мбайт. Рекомендуемый объем оперативной памяти 2 Гб или выше.

* + 1. **Требования к составу периферийных устройств**

Требования к составу периферийных устройств не предъявляются.

* + 1. **Требования к параметрам периферийных устройств**

Подключаемые периферийные устройства должны быть IBM-совместимыми.

* + 1. **Требования к программному обеспечению**

Системные программные средства, используемые программой, должны быть представлены версией ОС Windows 7 или выше, macOS Catalina v. 10.15.7 или выше.

* + 1. **Требования к персоналу (программисту)**

Программист должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы, должен быть аттестован минимум на II квалификационную группу по электробезопасности, должен иметь квалификацию «Пользователь ЭВМ».

# **ХАРАКТЕРИСТИКА ПРОГРАММЫ**

* 1. **Описание основных характеристик программы**
     1. **Режим работы программы**

Работа программы может осуществляться только в стандартном режиме работы — оконном режиме работы в операционной системе MacOS или Windows.

* 1. **Описание основных особенностей программы**

Программа предоставляет возможность играть в такие игры, как «Игровые автоматы», «Русская рулетка», «Барабан» на персональном компьютере. Кроме того, программа дает возможность пользователю зарегистрировать аккаунт для сохранения данных и последующего их использования при эксплуатации программы.

* 1. **Исходный код основных модулей программы**

В данном пункте представлены исходные коды программы с комментариями, поясняющими предназначение отдельных фрагментов кода.

* + 1. **Исходный код модуля игры «Игровые автоматы»**

package BackEnd;

public class Bet extends Account {

//Метод проверки цифр от 0 до 8(для игры slot machines)

protected static boolean CheckNum(String string){

try {

int num = Integer.parseInt(string);

return num > 0 && num < 7;

}

catch (Exception ex){

return false;

}

}

//Метод проведения игры(для slot machines)

public boolean Game(String vote, int bet, double coefficient, int percent){

int voteNum = Integer.parseInt(vote);

int answer = 1 + (int)(Math.random() \* 6);

if (voteNum == answer){

MinusBalance(bet);

double WinBet = bet \* coefficient;

double PercentAdmin = ((bet \* coefficient) \* percent)/100;

double FinalWin = WinBet - PercentAdmin;

Admin.PersonalBalance += PercentAdmin;

accounts.get(NumberUser).Balance += Admin.MinusBalance(FinalWin);

return true;

}

else {

Admin.OverallBalance += MinusBalance(bet);

return false;

}

}

}

* + 1. **Исходный код модуля игры «Русская рулетка»**

package BackEnd;

import javafx.scene.control.Alert;

import java.util.ArrayList;

import static BackEnd.OtherInterf.Transition;

public class RRbet extends Bet {

protected static ArrayList<Account> Room0 = new ArrayList<>();

protected static ArrayList<Account> Room1 = new ArrayList<>();

protected static ArrayList<Account> Room2 = new ArrayList<>();

protected static ArrayList<Account> Room3 = new ArrayList<>();

protected static ArrayList<Account> Room4 = new ArrayList<>();

protected static ArrayList<Account> Room5 = new ArrayList<>();

//метод конверсии

protected String Conversion(int Number){ return String.valueOf(Number); }

//метод очистки комнаты(Russian Roulette)

private void ClearRoom(ArrayList<Account> Room){ Room.clear(); }

private boolean CheckOnCoincidence(ArrayList<Account> Room, Account account){

boolean bool = true;

for (Account value : Room){

if (value == account){

bool = false;

break;

}

}

return bool;

}

//метод добавления аккаунта в комнату(Russian Roulette)

protected void AddAccount(ArrayList<Account> Room){

if (Room.size() < 7 && CheckOnCoincidence(Room, accounts.get(NumberUser)))

Room.add(accounts.get(NumberUser));

else

System.out.println("Size > 6");

}

//метод проведения игры(Russian Roulette)

public void RRGame(ArrayList<Account> Room, int bet, double coefficient, int percent){

if (CheckBalance(bet)){

MinusBalance(bet);

AddAccount(Room);

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Notification");

alert.setHeaderText(null);

alert.setContentText("You are in the game");

alert.showAndWait();

if (Room.size() == 6){

int answer = 0 + (int)(Math.random() \* 6);

Account winner = Room.get(answer);

for (Account account : accounts) {

if (account == winner) {

double WinBet = bet \* coefficient;

double PercentAdmin = ((bet \* coefficient) \* percent) / 100;

double FinalWinBet = WinBet - PercentAdmin;

Admin.PersonalBalance += PercentAdmin;

account.Balance += FinalWinBet;

ClearRoom(Room);

break;

} else System.out.println("Not user");

}

}

else System.out.println("Size != 6");

} else Transition("/FileFXML/NotBalance.fxml");

}

}

* + 1. **Исходный код модуля игры «Барабан»**

package BackEnd;

public class DrumBet extends RRbet implements OtherInterf {

protected int DropNum = 0;

//метод подсчета выигрыша

protected void Calculations(int bet, int percent, double coefficient){

MinusBalance(bet);

double WinBet = bet \* coefficient;

double PercentAdmin = ((bet \* coefficient) \* percent)/100;

double FinalWinBet = WinBet - PercentAdmin;

Admin.PersonalBalance += PercentAdmin;

accounts.get(NumberUser).Balance += FinalWinBet;

}

//метод проверки числа на совпадение с элементом массива

private boolean CheckArray(int[] array, int number){

boolean bool = false;

for (int value : array){

if (value == number) {

bool = true;

break;

}

}

return bool;

}

//проверка ставки

private boolean CheckBet(int bet){

return bet > 0 && bet < 1000001;

}

//метод для проведения игры по цвету(red, black) для игры drum

public void VoteColor(String color, int bet, int percent, double coefficient){

int answer = 1 + (int)(Math.random() \* 36);

int distributor;

if (color.equals("Red"))

distributor = 1;

else

distributor = 2;

switch (distributor){

case 1:

if (CheckBalance(bet)){

DropNum = answer;

if (CheckBet(bet)){

if (answer % 2 == 0){

Calculations(bet, percent, coefficient);

OtherInterf.FXNotification("You won!");

}

else {

Admin.OverallBalance += MinusBalance(bet);

OtherInterf.FXNotification("You lost");

}

}

else OtherInterf.FXNotification("Invalid balance entered. Bet should be between 0 and 1000000");

}

else OtherInterf.FXNotification("Insufficient funds");

break;

case 2:

if (CheckBalance(bet)){

DropNum = answer;

if (CheckBet(bet)){

if (answer % 2 != 0 && color.equals("Black")){

Calculations(bet, percent, coefficient);

OtherInterf.FXNotification("You won!");

}

else {

Admin.OverallBalance += MinusBalance(bet);

OtherInterf.FXNotification("You lost");

}

}

else OtherInterf.FXNotification("Invalid balance entered. Bet should be between 0 and 1000000");

}

else OtherInterf.FXNotification("Insufficient funds");

break;

default:

System.err.println("Unexpected error");

}

}

//метод проведения игры по столбцу(drum)

public void VoteColumn(int numberColumn, int bet, int percent, double coefficient){

int answer = 1 + (int)(Math.random() \* 36);

int[] array1 = {1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34};

int[] array2 = {2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35};

int[] array3 = {3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36};

switch (numberColumn){

case 1:

if (CheckBet(bet)){

if (CheckBalance(bet)){

DropNum = answer;

if (CheckArray(array1, answer)){

Calculations(bet, percent, coefficient);

OtherInterf.FXNotification("You won!");

}

else {

Admin.OverallBalance += MinusBalance(bet);

OtherInterf.FXNotification("Lose");

}

}

else OtherInterf.FXNotification("Insufficient funds");

}

else OtherInterf.FXNotification("Invalid balance entered. Bet should be between 0 and 1000000");

break;

case 2:

if (CheckBet(bet)){

if (CheckBalance(bet)){

DropNum = answer;

if (CheckArray(array2, answer)){

Calculations(bet, percent, coefficient);

OtherInterf.FXNotification("You won!");

}

else {

Admin.OverallBalance += MinusBalance(bet);

OtherInterf.FXNotification("Lose");

}

}

else OtherInterf.FXNotification("Insufficient funds");

}

else OtherInterf.FXNotification("Invalid balance entered. Bet should be between 0 and 1000000");

break;

case 3:

if (CheckBet(bet)){

if (CheckBalance(bet)){

DropNum = answer;

if (CheckArray(array3, answer)){

Calculations(bet, percent, coefficient);

OtherInterf.FXNotification("You won!");

}

else {

Admin.OverallBalance += MinusBalance(bet);

OtherInterf.FXNotification("Lose");

}

}

else OtherInterf.FXNotification("Insufficient funds");

}

else OtherInterf.FXNotification("Invalid balance entered. Bet should be between 0 and 1000000");

break;

default:

System.err.println("Unexpected error");

}

}

//метод проведения игры по заданному интервалу чисел(drum)

public void FromTo(int from, int to, int bet, int percent, double coefficient){

int answer = (int) (Math.random() \* 36);

if (CheckBet(bet)){

if (CheckBalance(bet)){

DropNum = answer;

if (answer > from - 1 && answer < to + 1){

Calculations(bet, percent, coefficient);

OtherInterf.FXNotification("You won");

}

else {

Admin.OverallBalance += MinusBalance(bet);

OtherInterf.FXNotification("You lose");

}

}

else OtherInterf.FXNotification("Insufficient funds");

}

else OtherInterf.FXNotification("Invalid balance entered. Bet should be between 0 and 1000000");

}

}

* + 1. **Исходный код методов по работе с уведомлениями**

package BackEnd;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.image.Image;

import javafx.stage.Modality;

import javafx.stage.Stage;

import java.io.IOException;

public interface OtherInterf {

//метод для перехода между окнами приложения

static void Transition(String path){

FXMLLoader loader = new FXMLLoader(OtherInterf.class.getResource(path));

Parent root = null;

try { root = (Parent)loader.load(); }

catch (IOException e) { e.printStackTrace(); }

Stage stage = new Stage();

stage.initModality(Modality.APPLICATION\_MODAL);

stage.setTitle("Casino");

stage.getIcons().add(new Image("file:src/ImgK.png"));

stage.setResizable(false);

stage.setScene(new Scene(root));

stage.show();

}

//метод для вывода уведомлений

static void FXNotification(String text){

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Notification");

alert.setHeaderText(null);

alert.setContentText(text);

alert.showAndWait();

}

}

* + 1. **Исходный код методов, связанных с аккаунтом пользователя**

package BackEnd;

import java.util.ArrayList;

public class Account {

public String Login;

public String Password;

public String CardNumber;

public String FullName;

public String DateExpirationCard;

public String CVV;

public double Balance;

public static int NumberUser;

public static ArrayList<Account> accounts = new ArrayList<>();

public Account(){}

public Account(String Login, String Password, String CardNumber, String FullName, String DateExpirationCard, String CVV){

this.Login = Login;

this.Password = Password;

this.CardNumber = CardNumber;

this.FullName = FullName;

this.DateExpirationCard = DateExpirationCard;

this.CVV = CVV;

}

//Метод преобразует буквы в верхний регистр

public String TransformationFullName(String string){

String str = string.toUpperCase();

char[] arrayChar = str.toCharArray();

return String.valueOf(arrayChar);

}

//Метод проверки аккаунта и выдачи идентификатора пользователю

public Boolean CheckAccount(String str, String str1){

boolean bool = false;

for (int i = 0; i < accounts.size(); i++){

if (accounts.get(i).Login.equals(str) && accounts.get(i).Password.equals(str1)) {

bool = true;

NumberUser = i;

break;

}

else

bool = false;

}

return bool;

}

//Метод проверки аккаунта на дубликаты

public boolean CheckDuplicateAccount(String login, String cardNumber) {

for (Account account : accounts) {

if (login.equalsIgnoreCase(account.Login) || cardNumber.equals(account.CardNumber))

return false;

}

return true;

}

//Метод проверки баланса на достаточность

public boolean CheckBalance(int bet){

return bet == accounts.get(NumberUser).Balance || bet < accounts.get(NumberUser).Balance;

}

protected double MinusBalance(int bet){

accounts.get(NumberUser).Balance -= bet;

return bet;

}

//Метод проверки срока годности карты

protected boolean CheckDate(String expirationDataReg){

try {

char[] array = expirationDataReg.toCharArray();

String str0 = Character.toString(array[0]);

String str1 = Character.toString(array[1]);

String str2 = Character.toString(array[2]);

String str3 = Character.toString(array[3]);

String str4 = Character.toString(array[4]);

String string1 = str0 + str1;

String string2 = str3 + str4;

int num1 = Integer.parseInt(string1);

int num2 = Integer.parseInt(string2);

if (num1 > 0 && num1 < 13){

if (num2 > 19 && num2 < 100){

return str2.equals(".");

}

else return false;

}

else return false;

}

catch (Exception ex){

return false;

}

}

//Методы создания игровых аккаунтов

public void TestCreateAccount(){

accounts.add(new Account("qwe", "123", "1111222233334444", "As Sa", "11.01", "123"));

accounts.add(new Account("Player2", "222", "2222333344445555", "Ed Di", "22.01", "456"));

accounts.add(new Account("Player3", "333", "3333444455556666", "Mi Is", "17.03", "789"));

accounts.add(new Account("Player4", "444", "3333634523556666", "Di Im", "23.03", "321"));

accounts.add(new Account("Player5", "555", "3333444458786666", "Ig Go", "01.03", "654"));

accounts.add(new Account("Player6", "666", "3311134465556666", "Ch He", "04.03", "987"));

}

public void TestCreateAccount5el(){

accounts.add(new Account("qwe", "123", "1111222233334444", "As Sa", "11.01", "123"));

accounts.add(new Account("Player2", "222", "2222333344445555", "Ed Di", "22.01", "456"));

accounts.add(new Account("Player3", "333", "3333444455556666", "Mi Is", "17.03", "789"));

accounts.add(new Account("Player5", "555", "3333444458786666", "Ig Go", "01.03", "654"));

}

}

* + 1. **Исходный код модуля регистрации пользователя**

package Controllerss;

import BackEnd.Account;

import BackEnd.OtherInterf;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import javafx.stage.Stage;

public class RegUserController extends Account implements OtherInterf {

@FXML

private TextField CardNumberRegField;

@FXML

private TextField CardHolderRegField;

@FXML

private TextField ExpirationDataRegField;

@FXML

private TextField CVVGegField;

@FXML

private TextField PasswordRegField;

@FXML

private TextField LoginRegField;

@FXML

private Button RegButton;

@FXML

private Button BackGerField;

@FXML

void initialize() {

//в кнопке прописаны действия для проверки вводимых полей(регистрация пользователя), запись в файл.

RegButton.setOnAction(event -> {

if (LoginRegField.getText().length() > 0 && LoginRegField.getText().length() < 17){

if (PasswordRegField.getText().length() > 0 && PasswordRegField.getText().length() < 17){

if (CardNumberRegField.getText().length() == 16 && CardNumberRegField.getText().matches("\\d+")){

if (CheckDuplicateAccount(LoginRegField.getText(), CardNumberRegField.getText())) {

if (ExpirationDataRegField.getText().length() == 5 && CheckDate(ExpirationDataRegField.getText())) {

if (CVVGegField.getText().length() == 3 && CVVGegField.getText().matches("\\d+")) {

String str = TransformationFullName(CardHolderRegField.getText());

Account account = new Account(LoginRegField.getText(), PasswordRegField.getText(), CardNumberRegField.getText(), str, ExpirationDataRegField.getText(), CVVGegField.getText());

accounts.add(account);

OtherInterf.FXNotification("The account is registered");

Stage stg = (Stage) BackGerField.getScene().getWindow();

stg.close();

OtherInterf.Transition("/FileFXML/sampleee.fxml");

} else OtherInterf.FXNotification("CVV entered incorrectly");

} else OtherInterf.FXNotification("Date entered incorrectly");

} else OtherInterf.FXNotification("There is already an account with this login and/or card number");

} else OtherInterf.FXNotification("Card number entered incorrectly");

} else OtherInterf.FXNotification("Password entered incorrectly");

} else OtherInterf.FXNotification("Login entered incorrectly");

});

BackGerField.setOnAction(event -> {

Stage stg = (Stage)BackGerField.getScene().getWindow();

stg.close();

OtherInterf.Transition("/FileFXML/sampleee.fxml");

});

}

}

* + 1. **Исходный код модуля изменения данных пользователя**

package Controllerss;

import BackEnd.Account;

import BackEnd.OtherInterf;

import javafx.fxml.FXML;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import javafx.stage.Stage;

public class EditingDataController extends Account implements OtherInterf {

@FXML

private TextField NewCVVField;

@FXML

private TextField NewCardNumberField;

@FXML

private TextField NewFullNameField;

@FXML

private TextField NewDataExpirationField;

@FXML

private Button ToChangeButton;

@FXML

private Button NotChangeButton;

@FXML

void initialize() {

//в кнопке прописаны действия для проверки вводимых полей(изменение данных пользователя), запись в файл.

ToChangeButton.setOnAction(event -> {

if (NewCardNumberField.getText().length() == 16 && NewCardNumberField.getText().matches("\\d+")){

if (CheckDuplicateAccount("", NewCardNumberField.getText())) {

if (!NewFullNameField.getText().equals("")) {

if (NewDataExpirationField.getText().length() == 5 && CheckDate(NewDataExpirationField.getText())) {

if (NewCVVField.getText().length() == 3 && NewCVVField.getText().matches("\\d+")) {

accounts.get(NumberUser).CardNumber = NewCardNumberField.getText();

accounts.get(NumberUser).FullName = NewFullNameField.getText();

accounts.get(NumberUser).DateExpirationCard = NewDataExpirationField.getText();

accounts.get(NumberUser).CVV = NewCVVField.getText();

// Скорее всего, удалить имеющегося аккаунта и добавить нового

/\*

\* Создать функцию нахождения аккаунта в классе Account, похожую на Account.CheckAccount,

\* но возвращающую номер аккаунта в списке аккаунтов.

\*/

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Notification");

alert.setHeaderText(null);

alert.setContentText("The account info had been edited");

alert.showAndWait();

Stage stg = (Stage) NotChangeButton.getScene().getWindow();

stg.close();

OtherInterf.Transition("/FileFXML/PayData.fxml");

} else OtherInterf.FXNotification("CVV entered incorrectly");

} else OtherInterf.FXNotification("Date entered incorrectly");

} else OtherInterf.FXNotification("Full name entered incorrectly");

} else OtherInterf.FXNotification("There is already an account with this card number");

} else OtherInterf.FXNotification("Card number entered incorrectly");

});

NotChangeButton.setOnAction(event -> {

Stage stg = (Stage)NotChangeButton.getScene().getWindow();

stg.close();

OtherInterf.Transition("/FileFXML/PayData.fxml");

});

}

}

* + 1. **Исходный код метода изменения баланса**

package BackEnd;

public class Admin {

public static double OverallBalance = 1000000;

public static double PersonalBalance = 0;

//Метод списания денег с баланса

public static double MinusBalance(double WinBet){

OverallBalance -= WinBet;

return WinBet;

}

}

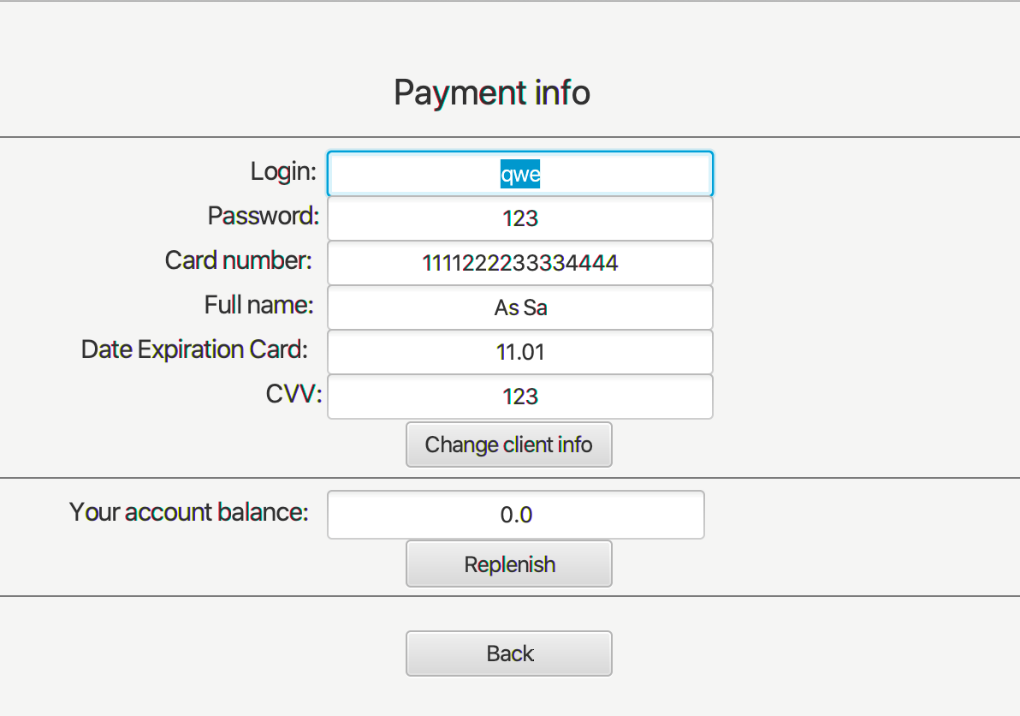
# **ОБРАЩЕНИЕ К ПРОГРАММЕ**

* 1. **Запуск программы**

При запуске программы, появится окно аутентификации пользователя. Необходимо выполнить вход в систему по уже имеющимся данным, или произвести регистрацию в системе для дальнейших действий.

* 1. **Информация о клиенте**
     1. **Просмотр данных**

В программе предусмотрен режим просмотра пользовательских данных. Для просмотра данных необходимо перейти в модуль информации пользователя (пункт меню «Client information» Главного окна программы) (рис. 1).

  
Рис 1. Модуль «Информация пользователя»

* + 1. **Изменение данных**

Для изменения данных необходимо произвести действия ввода данных в текстовые и числовые поля (заполнить), предварительно нажав на кнопку «Change client information», а затем нажать на кнопку «Apply changes».

* + 1. **Пополнение баланса**

Для пополнения баланса необходимо ввести сумму, желаемую к пополнению в поле, которое появляется после нажатия на кнопку «Replenish».

* 1. **Игровой режим игры «Игровые автоматы»**

Данный игровой режим предоставляет возможность игры со ставками (рис. 2).

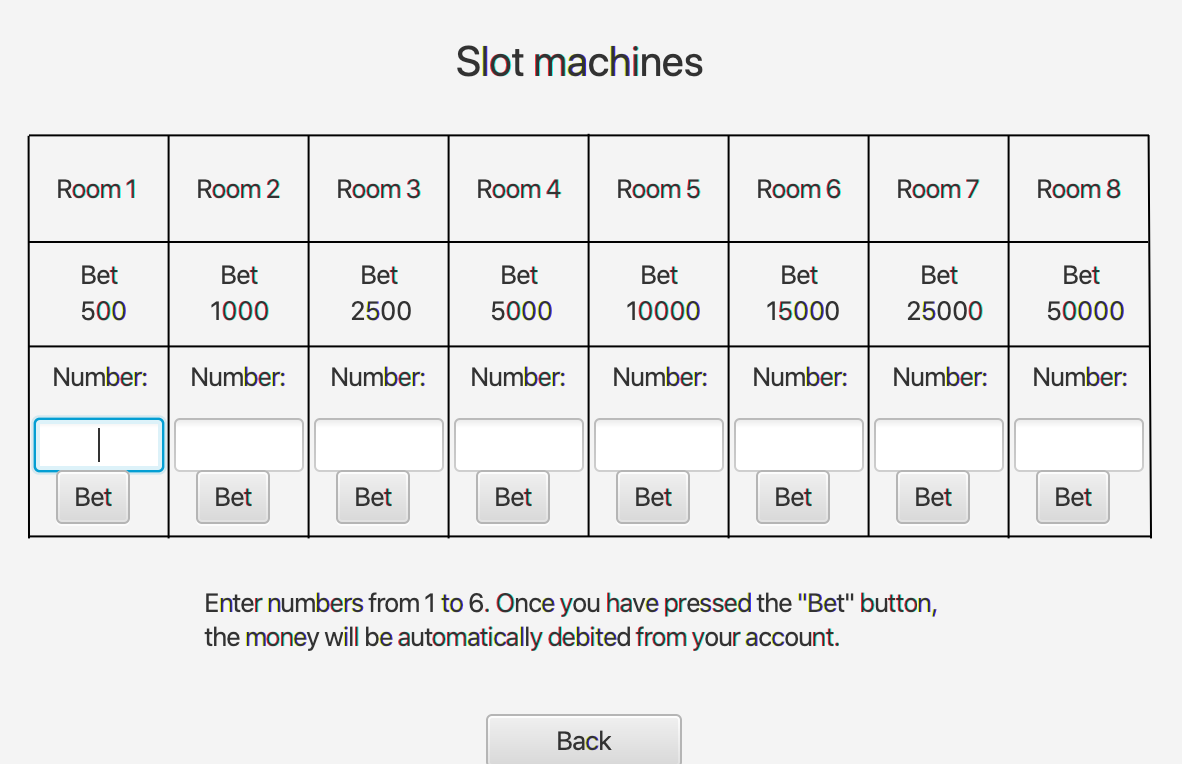


Рис. 2. Комната игрового режима игры «Игровые автоматы»

* 1. **Игровой режим игры «Русская рулетка»**

Данный игровой режим предоставляет возможность игры со ставками (рис. 3).

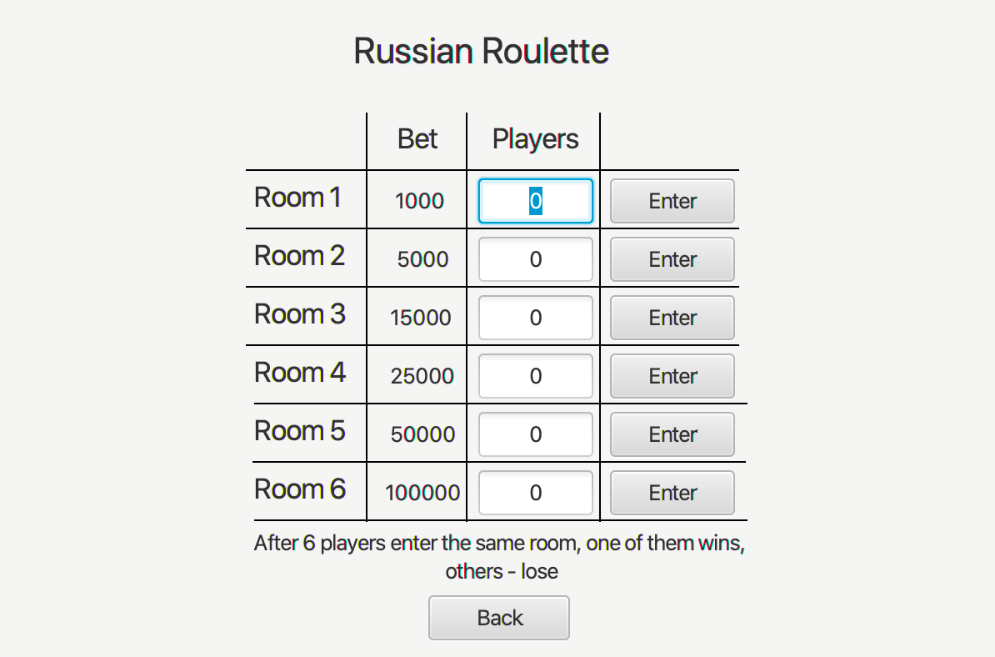


Рис. 3. Комната игрового режима игры «Русская рулетка»

* 1. **Игровой режим игры «Барабан»**

Данный игровой режим предоставляет возможность игры со ставками (рис. 4).

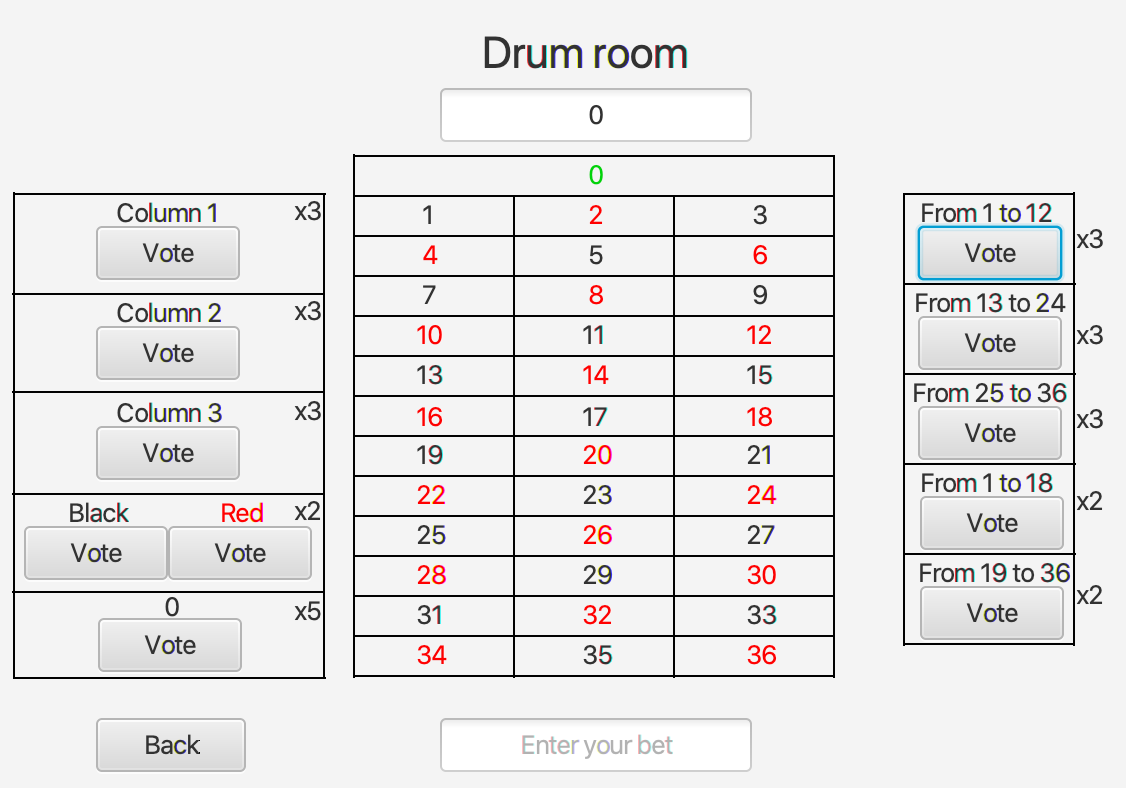


Рис. 4. Комната игрового режима игры «Барабан»

# **ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ**

## **Организация используемой входной информации**

Входная информация в закодированном виде поступает одним блоком из файла «profile.prf».

* 1. **Организация используемой выходной информации**

Выходная информация в закодированном виде по протоколу Protocol Buffers сохраняется в файл «profile.prf».

Информация логирования системы в формате JSON записывается в файл «system.log».

* 1. **Кодирование используемой информации**

Кодирование информации происходит в автоматическом режиме технологией компании Google — Protocol Buffers, с использованием непубличных алгоритмов кодирования и обработки данных.

# **СООБЩЕНИЯ**

Сообщения компетентному специалисту выводятся в виде диалоговых окон с текстом сообщения в случае возникновения ошибок и уведомлений о результате выполнения функций программы.

* 1. **Уведомления об ошибке ввода (заполнения текстовых полей)**

Ошибка ввода (заполнения текстовых полей) возникает в том случае, когда пользователь заполнил текстовые поля некорректно. Нарушение корректности ввода может быть вызвано в следующих случаях:

1. Некорректное заполнение поля «Login» при регистрации пользователя (рис. 5, 8).

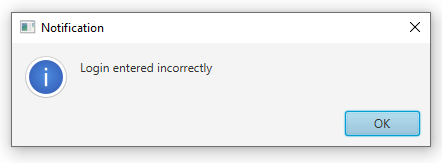


Рис. 5. Окно с уведомлением об ошибке заполнения поля «Login»

1. Некорректное заполнение поля «Password» при регистрации пользователя (рис. 6).

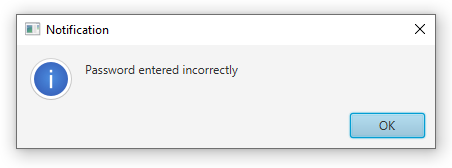


Рис. 6. Окно с уведомлением об ошибке заполнения поля «Password»

1. Некорректное заполнение поля «Card number» при регистрации или изменении данных пользователя (рис. 7–8).

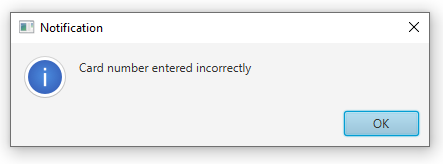


Рис. 7. Окно с уведомлением об ошибке заполнения поля «Card number»

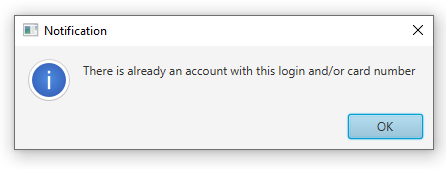


Рис. 8. Окно с уведомлением о некорректном заполнении поля «Login» и (или) поля «Card number»

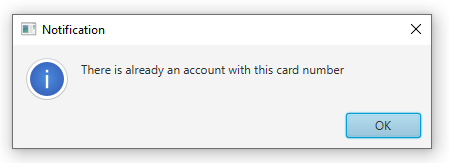


Рис. 9. Окно с сообщением об ошибке в случае изменения номера карты на номер, занятый другим уже зарегистрированным пользователем

1. Некорректное заполнение поля «Card expiration date» при регистрации или изменении данных пользователя (рис. 10).

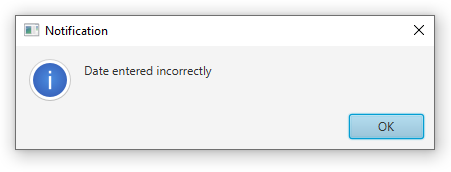


Рис. 10. Окно с уведомлением об ошибке заполнения поля «Card expiration date»

1. Некорректное заполнение поля «CVV» при регистрации или изменении данных пользователя (рис. 11).

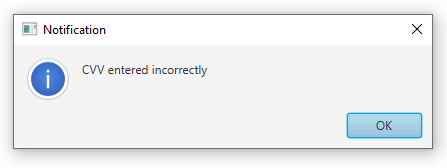


Рис. 11. Окно с уведомлением об ошибке заполнения поля «CVV»

1. Некорректное заполнение текстового поля при пополнении баланса (рис. 12).

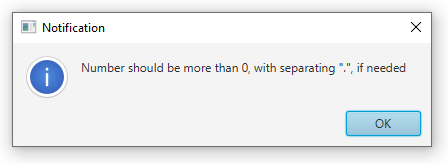


Рис. 12. Окно с уведомлением о некорректном заполнении поля для пополнения баланса

1. Некорректное заполнение текстового поля для ввода числа в игре «Игровые автоматы» (рис. 13).

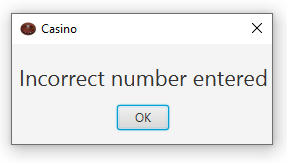


Рис. 13. Окно с уведомлением об ошибке заполнения поля для указания ставки в игре «Игровые автоматы»

1. Некорректное заполнение текстового поля для ввода числа в игре «Барабан» (рис. 14).

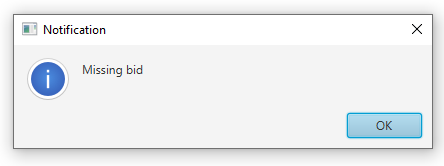


Рис. 14. Окно с уведомлением об ошибке заполнения поля для указания ставки в игре «Барабан»

Пользователь может закрыть Окно с сообщением об ошибке ввода посредством нажатия на крестик или кнопку «OK».

* 1. **Уведомление об ошибке авторизации**

Ошибка авторизации возникает в том случае, когда в процессе авторизации программе не удалось найти среди зарегистрированных пользователей пользователя с введенными в полях «Login» и «Password» данными (рис. 15).

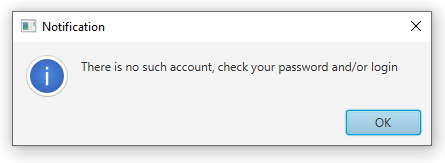


Рис. 15. Окно с уведомлением об ошибке авторизации

Пользователь может закрыть Окно с уведомлением об ошибке авторизации посредством нажатия на крестик или кнопку «OK».

* 1. **Уведомления об ошибке недостаточности баланса**

Ошибка недостаточности баланса возникает в том случае, если при указании ставки для игры (игры «Игровые автоматы», «Барабан») или при входе в комнату в игре «Русская рулетка» у пользователя недостаточно средств на балансе для участия в игре (рис. 16–17).

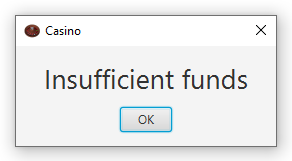


Рис. 16. Окно с уведомлением об ошибке недостаточности баланса (игры «Игровые автоматы», «Русская рулетка»)

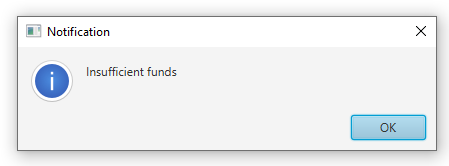


Рис. 17. Окно с уведомлением об ошибке недостаточности баланса (игра «Барабан»)

Пользователь может закрыть Окно с уведомлением об ошибке недостаточности баланса посредством нажатия на крестик или кнопку «OK».

* 1. **Уведомление об успешной регистрации**

В случае успешного завершения функции регистрации пользователь получает Окно с уведомлением (рис. 18). Окно с уведомлением об успешной регистрации может быть закрыто пользователем посредством нажатия на крестик или кнопку «OK».

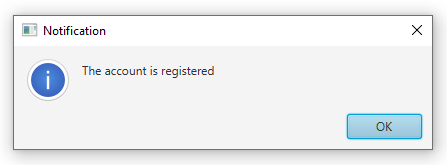


Рис. 18. Окно с уведомлением об успешной регистрации

* 1. **Уведомление об успешном изменении данных пользователя**

В случае успешного изменения данных пользователь получает Окно с уведомлением об успешном изменении данных пользователя (рис. 19). Окно с уведомлением об успешном изменении данных пользователя может быть закрыто пользователем посредством нажатия на крестик или кнопку «OK».

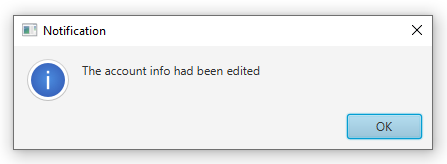


Рис. 19. Окно с уведомлением об успешном изменении данных пользователя

* 1. **Уведомление об успешном входе в комнату в игре «Русская рулетка»**

Во время выполнения игрового процесса игры «Русская рулетка» пользователь входит в комнату в Окне игры «Русская рулетка». В случае успешного входа в комнату пользователь получает Окно с уведомлением об успешном входе в комнату (рис. 20).

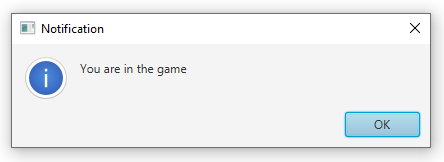


Рис. 20. Уведомление об успешном входе в комнату в игре «Русская рулетка»

Окно может быть закрыто пользователем посредством нажатия на крестик или кнопку «OK».

* 1. **Уведомления о результате игры**

После завершения игрового процесса (игры «Игровые автоматы», «Барабан») пользователь получает Окно с сообщением о результате игры (рис. 21 для игр «Игровые автоматы» и «Русская рулетка», рис. 22 для игры «Барабан»).

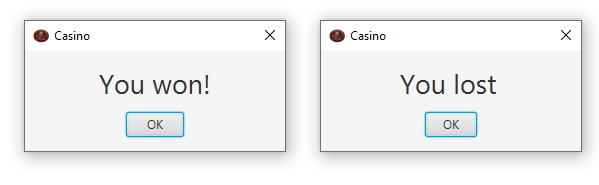


Рис. 21. Окна с сообщением о результате игры (игры «Игровые автоматы», «Русская рулетка»)

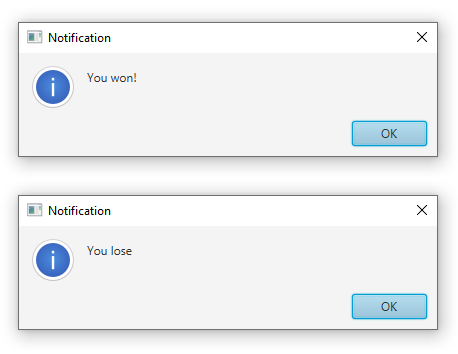


Рис. 22. Окна с сообщением о результате игры (игра «Барабан»)

Окна с сообщением о результате игры могут быть закрыты пользователем посредством нажатия на крестик или кнопку «OK».

# **ПРИЛОЖЕНИЕ А**

**Правила игр**

Правила игр, сценарии действий и реакции на ввод пользователя представлены ниже:

1. Правила игры «Игровые автоматы».

Пользователь выбирает комнату (с первой по восьмую) и выбирает любое число в диапазоне от 1 до 6. В зависимости от комнаты меняется ставка. Если пользователь угадывает число, ставка записывается на его счет. Если пользователь не угадывает число, ставка списывается со счета пользователя.

1. Правила игры «Русская рулетка».

Пользователь выбирает комнату (с первой по шестую). В зависимости от комнаты меняется ставка. Игра происходит, когда в комнате набирается 6 человек. После набора 6 человек по жребию выбирается победитель. На его счет зачисляется 98 % от суммарной ставки всех игроков.

1. Правила игры «Барабан».

Пользователь вводит ставку и выбирает вариант игры: по колонкам, по интервалам чисел, по цветам или выпадение нуля. В каждом варианте игры разный коэффициент выигрыша. Если пользователь выигрывает, то на его счет зачисляется ставка, умноженная на коэффициент. В случае проигрыша ставка списывается со счета пользователя.

Пользователь может участвовать в игре только в том случае, если на его балансе достаточно средств.