Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 16

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Инкапсуляция»

Выполнил: ст. гр. ТУУ-111

Волобуев Е.Д.

Вариант №13

Проверил: к.т.н., доц. Сафронов А.И.

Зачтено

Москва – 2024 г.

## 1. Цель работы

Разработать приложение с инкапсулированным классом.

## 2. Формулировка задачи

**Вариант 13.** Создайте класс «банковский счёт». Все поля класса инкапсулированы в классе. При создании нового банковского счёта вводятся параметры логин (get) и пароль (set), к счёту прикрепляется Ф.И.О. владельца (get; set), уникальный идентификатор счёта, дата рождения владельца (get; set) и его номер телефона (get; set). Команда на создание счёта выдаётся по нажатии на экранную кнопку. При работе программного обеспечения возникает запрет на открытие счёта, если возможный владелец ещё не достиг 16-летнего возраста. Действует правило: один банковский счёт на одного владельца. Уникальный идентификатор демонстрируется в программном обеспечении только один раз при успешном создании счёта в окне побуждающего сообщения (warning) с максимальным привлечением внимания пользователя. Логин для счёта менять запрещено. Смотреть логин (get) можно только в информирующем побуждающем сообщении (information), вызываемом по нажатии на соответствующую экранную кнопку при условии, что в поле логин в этот момент введён уникальный идентификатор, а после, вдобавок, в дочерней экранной форме пользователь корректно ввёл номер телефона владельца, связанный с указанным уникальным идентификатором. Пароль (set) при вводе не демонстрируется в поле (шифруется произвольным символом), но имеется возможность отображения пароля для проверки ввода по нажатии на специально предусмотренную экранную кнопку. При корректном вводе сочетания логин + пароль на экранной форме отображаются ранее скрытые элементы управления, в которых можно посмотреть и при необходимости изменить Ф.И.О. владельца (get; set), дату рождения владельца (get; set), номер телефона владельца (get; set), удалить банковский счёт. Можно изменить пароль (set) на новый при условии, что повторно будет введено сочетание логин + пароль в появившейся дочерней экранной форме. Для остальных параметров в качестве подтверждения дважды должен быть введён пароль, совпадающий с заложенным владельцем ранее (текущим) паролем.

## 3. Составление диаграммы классов, входящих в состав решения

Account

- private void createAccountButton\_Click(object sender, EventArgs e)

- private void showLoginButton\_Click(object sender, EventArgs e)

- private void showPasswordButton\_Click(object sender, EventArgs e)

- private BankAccount FindAccountByID(string id)

- private void deleteAccountButton\_Click(object sender, EventArgs e)

- private void changePasswordButton\_Click(object sender, EventArgs e)

- private void changeInfoButton\_Click(object sender, EventArgs e)

- private void saveEdit\_Click(object sender, EventArgs e)

- private void okButton\_Click(object sender, EventArgs e)

-private void ConfirmButton\_Click(object sender, EventArgs e)

Account

- private void createAccountButton\_Click(object sender, EventArgs e)

- private void showLoginButton\_Click(object sender, EventArgs e)

- private void showPasswordButton\_Click(object sender, EventArgs e)

- private BankAccount FindAccountByID(string id)

- private void deleteAccountButton\_Click(object sender, EventArgs e)

- private void changePasswordButton\_Click(object sender, EventArgs e)

- private void changeInfoButton\_Click(object sender, EventArgs e)

- private void saveEdit\_Click(object sender, EventArgs e)

## 4. Сети Петри запрограммированного технологического процесса

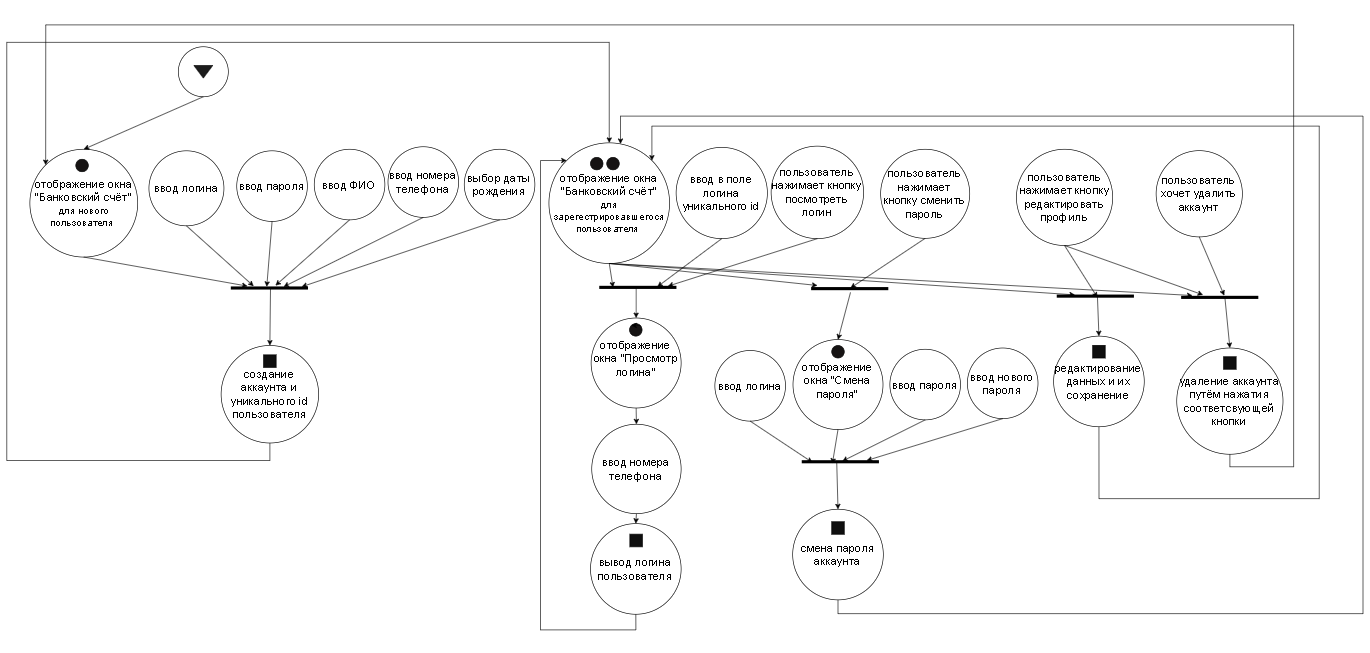


Рисунок 1 - Сеть Петри запрограммированного технологического процесса

## 5. Схемы алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка )

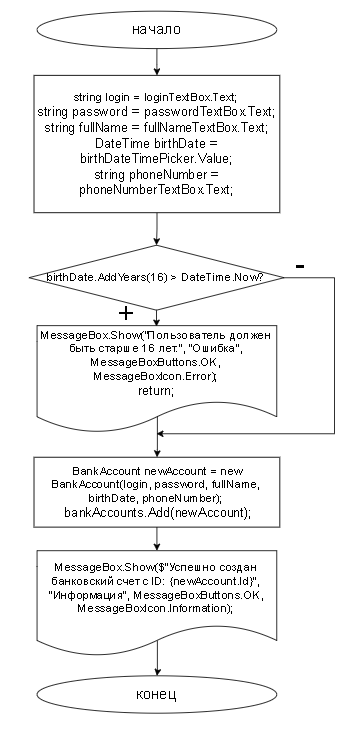


Рисунок 2 – Схема алгоритма эффекта «Создание аккаунта и уникального ID

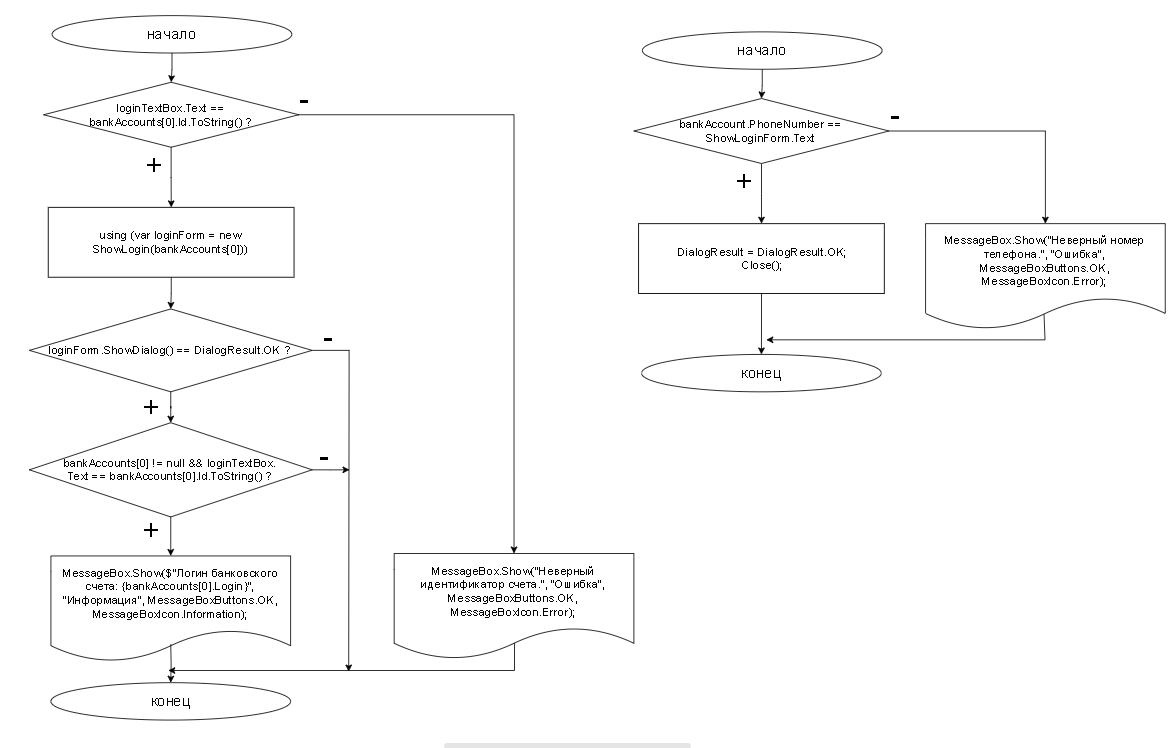


Рисунок 3 – Схемы алгоритмов эффекта «Вывод логина пользователя»

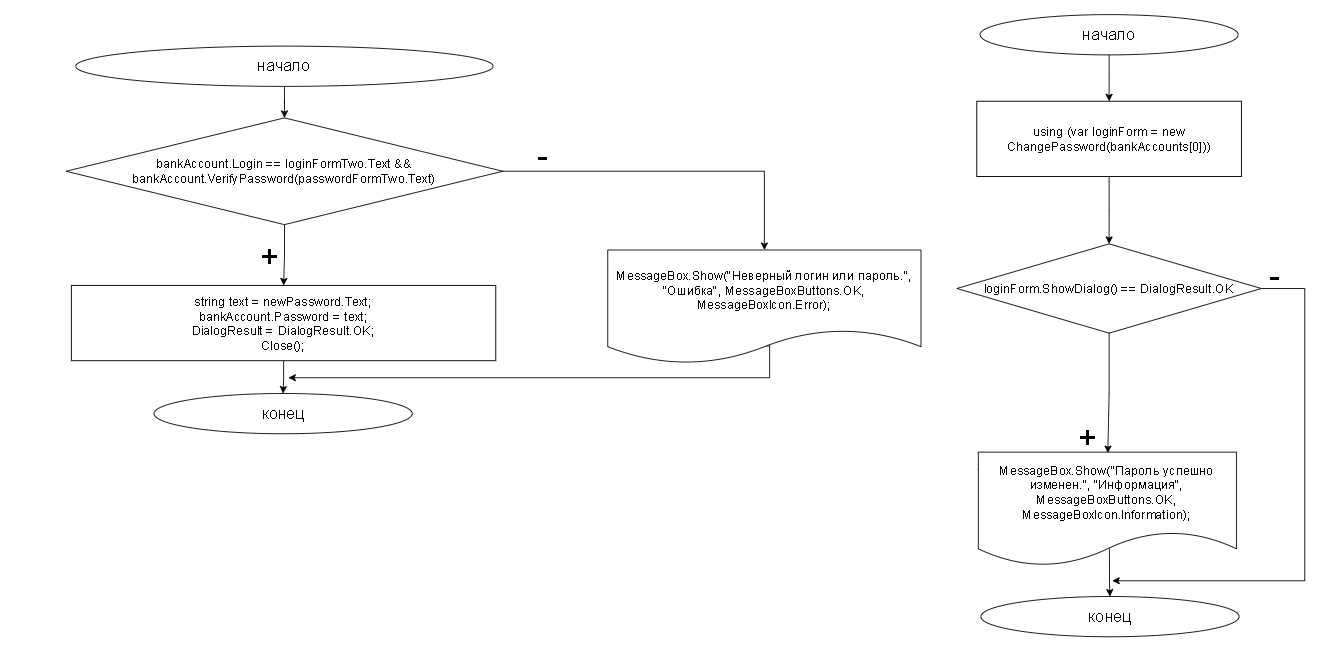


Рисунок 4 – Схемы алгоритмов эффекта «Смена пароля аккаунта»

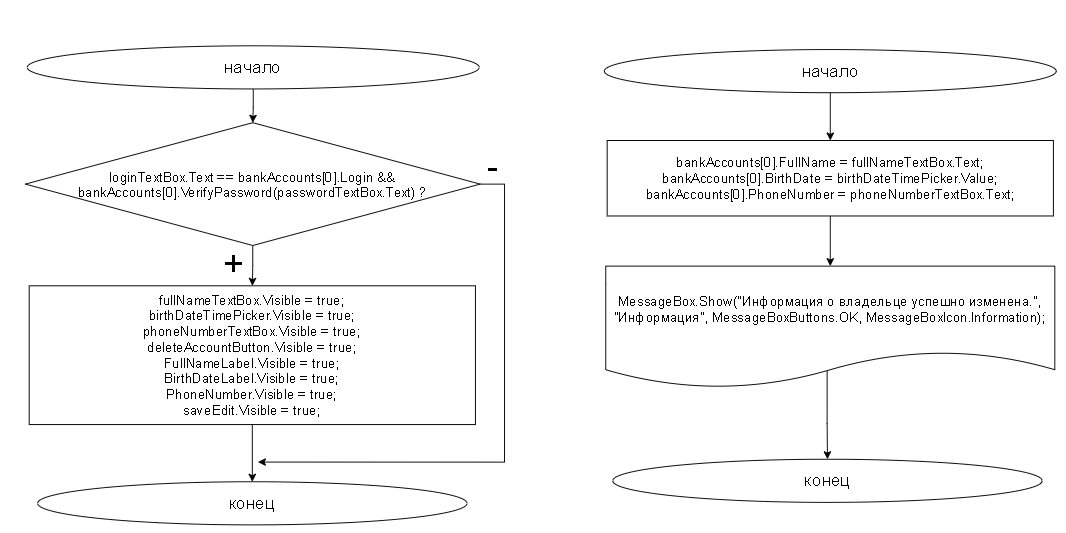


Рисунок 5 – Схемы алгоритмов эффекта «Редактирование данных и их сохранение»

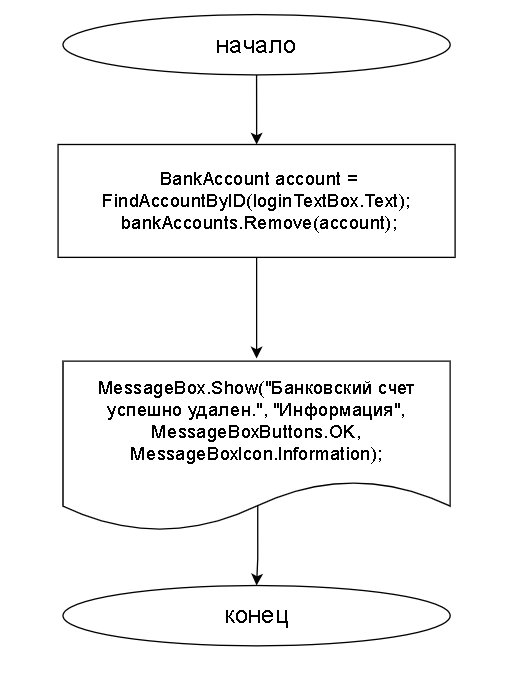


Рисунок 6 – Схема алгоритма эффекта «Удаление аккаунта путем нажатия соответствующей кнопки»

## 6. Подбор тестовых примеров

**Пример 1:**

Введём данные пользователя, чтобы создать аккаунт:

Логин: login

Пароль: password

ФИО: Иванов Иван Иванович

Дата рождения: 1 января 2000

Телефон: 89999999999

Нажав на кнопку создать аккаунт, у нас создаётся аккаунт, ID и отображаются кнопки для управления данными аккаунта

**Пример 2:**

Если пользователю меньше 16 лет то программа отказывает в регистрации аккаунта

**Пример 3:**

Чтобы увидеть введённый пароль, нужно нажать на кнопку показать пароль, тогда пользователь увидит все введённые символы в поле пароль

**Пример 4:**

Чтобы изменить пароль, пользователю нужно нажать на кнопку сменить пароль и ввести в отдельном окне свой логин и пароль, а также новый пароль

**Пример 5:**

Если пользователь при смене пароля вводит не свой логин или пароль то появляется ошибка

**Пример 6:**

Чтобы посмотреть логин, в поле для логина надо ввести уникальный ID и нажать кнопку показать логин, тогда откроется окно для ввода номера телефона, при успешном заполнении которого покажется логин пользователя

**Пример 7:**

Если в поле логин введён неправильный уникальный ID, то при нажатии на кнопку показа логина возникнет ошибка

**Пример 8:**

Чтобы отредактировать данные, пользователь должен ввести свой логин и пароль, нажать соответствующую кнопку, отредактировать нужное и нажать кнопку сохранить

**Пример 9:**

Чтобы удалить аккаунт, пользователь должен ввести свой логин и пароль, нажать кнопку редактировать профиль, а потом нажать кнопку удалить профиль

## 7. Листинг (код) составленного программного обеспечения

Form1:

namespace Rabota\_16

{

public partial class Account : Form

{

private List<BankAccount> bankAccounts;

public Account()

{

InitializeComponent();

bankAccounts = new List<BankAccount>();

passwordTextBox.UseSystemPasswordChar = true;

saveEdit.Visible = false;

deleteAccountButton.Visible = false;

showLoginButton.Visible = false;

changePasswordButton.Visible = false;

changeInfoButton.Visible = false;

}

private void createAccountButton\_Click(object sender, EventArgs e)

{

string login = loginTextBox.Text;

string password = passwordTextBox.Text;

string fullName = fullNameTextBox.Text;

DateTime birthDate = birthDateTimePicker.Value;

string phoneNumber = phoneNumberTextBox.Text;

if (birthDate.AddYears(16) > DateTime.Now)

{

MessageBox.Show("Пользователь должен быть старше 16 лет.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

BankAccount newAccount = new BankAccount(login, password, fullName, birthDate, phoneNumber);

bankAccounts.Add(newAccount);

MessageBox.Show($"Успешно создан банковский счет с ID: {newAccount.Id}", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

loginTextBox.Enabled = true;

passwordTextBox.Enabled = true;

deleteAccountButton.Enabled = true;

changePasswordButton.Enabled = true;

changeInfoButton.Enabled = true;

fullNameTextBox.Visible = false;

birthDateTimePicker.Visible = false;

phoneNumberTextBox.Visible = false;

createAccountButton.Visible = false;

FullNameLabel.Visible = false;

BirthDateLabel.Visible = false;

PhoneNumber.Visible = false;

showLoginButton.Visible = true;

changePasswordButton.Visible = true;

changeInfoButton.Visible = true;

}

private void showLoginButton\_Click(object sender, EventArgs e)

{

if (loginTextBox.Text == bankAccounts[0].Id.ToString())

{

using (var loginForm = new ShowLogin(bankAccounts[0]))

{

if (loginForm.ShowDialog() == DialogResult.OK)

{

if (bankAccounts[0] != null && loginTextBox.Text == bankAccounts[0].Id.ToString())

{

MessageBox.Show($"Логин банковского счета: {bankAccounts[0].Login}", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

else

{

MessageBox.Show("Неверный идентификатор счета.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void showPasswordButton\_Click(object sender, EventArgs e)

{

passwordTextBox.UseSystemPasswordChar = !passwordTextBox.UseSystemPasswordChar;

}

private void deleteAccountButton\_Click(object sender, EventArgs e)

{

BankAccount account = FindAccountByID(loginTextBox.Text);

bankAccounts.Remove(account);

MessageBox.Show("Банковский счет успешно удален.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

createAccountButton.Visible = true;

loginTextBox.Text = null;

passwordTextBox.Text = null;

birthDateTimePicker.Value = DateTime.Now;

phoneNumberTextBox.Text = null;

fullNameTextBox.Text = null;

saveEdit.Visible = false;

deleteAccountButton.Visible = false;

showLoginButton.Visible = false;

changePasswordButton.Visible = false;

changeInfoButton.Visible = false;

}

private BankAccount FindAccountByID(string id)

{

try

{

return bankAccounts.FirstOrDefault(account => account.Id.ToString() == id);

}

catch (Exception)

{

return null;

}

}

private void changePasswordButton\_Click(object sender, EventArgs e)

{

using (var loginForm = new ChangePassword(bankAccounts[0]))

{

if (loginForm.ShowDialog() == DialogResult.OK)

{

MessageBox.Show("Пароль успешно изменен.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

private void changeInfoButton\_Click(object sender, EventArgs e)

{

if (loginTextBox.Text == bankAccounts[0].Login && bankAccounts[0].VerifyPassword(passwordTextBox.Text))

{

fullNameTextBox.Visible = true;

birthDateTimePicker.Visible = true;

phoneNumberTextBox.Visible = true;

deleteAccountButton.Visible = true;

FullNameLabel.Visible = true;

BirthDateLabel.Visible = true;

PhoneNumber.Visible = true;

saveEdit.Visible = true;

}

}

private void saveEdit\_Click(object sender, EventArgs e)

{

bankAccounts[0].FullName = fullNameTextBox.Text;

bankAccounts[0].BirthDate = birthDateTimePicker.Value;

bankAccounts[0].PhoneNumber = phoneNumberTextBox.Text;

MessageBox.Show("Информация о владельце успешно изменена.", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

fullNameTextBox.Visible = false;

birthDateTimePicker.Visible = false;

phoneNumberTextBox.Visible = false;

saveEdit.Visible = false;

FullNameLabel.Visible = false;

BirthDateLabel.Visible = false;

PhoneNumber.Visible = false;

deleteAccountButton.Visible = false;

}

}

public partial class BankAccount

{

private string login;

private string password;

private string fullName;

private Guid id;

private DateTime birthDate;

private string phoneNumber;

public string Login

{

get { return login; }

}

public string Password

{

set { password = value; }

get { return password; }

}

public string FullName

{

get { return fullName; }

set { fullName = value; }

}

public Guid Id

{

get { return id; }

}

public DateTime BirthDate

{

get { return birthDate; }

set { birthDate = value; }

}

public string PhoneNumber

{

get { return phoneNumber; }

set { phoneNumber = value; }

}

public bool VerifyPassword(string password)

{

return Password == password;

}

public BankAccount(string login, string password, string fullName, DateTime birthDate, string phoneNumber)

{

this.login = login;

this.password = password;

this.fullName = fullName;

id = Guid.NewGuid();

this.birthDate = birthDate;

this.phoneNumber = phoneNumber;

}

}

}

Form2:

namespace Rabota\_16

{

public partial class ChangePassword : Form

{

private BankAccount bankAccount;

public ChangePassword(BankAccount account)

{

InitializeComponent();

bankAccount = account;

}

private void okButton\_Click(object sender, EventArgs e)

{

if (bankAccount.Login == loginForChangePassword.Text && bankAccount.VerifyPassword(passwordFormTwo.Text))

{

string text = newPassword.Text;

bankAccount.Password = text;

DialogResult = DialogResult.OK;

Close();

}

else

{

MessageBox.Show("Неверный логин или пароль.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Form3:

namespace Rabota\_16

{

public partial class ShowLogin : Form

{

private BankAccount bankAccount;

public ShowLogin(BankAccount account)

{

InitializeComponent();

bankAccount = account;

}

private void ConfirmButton\_Click(object sender, EventArgs e)

{

if (bankAccount.PhoneNumber == ShowLoginForm.Text)

{

DialogResult = DialogResult.OK;

Close();

}

else

{

MessageBox.Show("Неверный номер телефона.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

## 8. Графический пользовательский интерфейс программного обеспечения

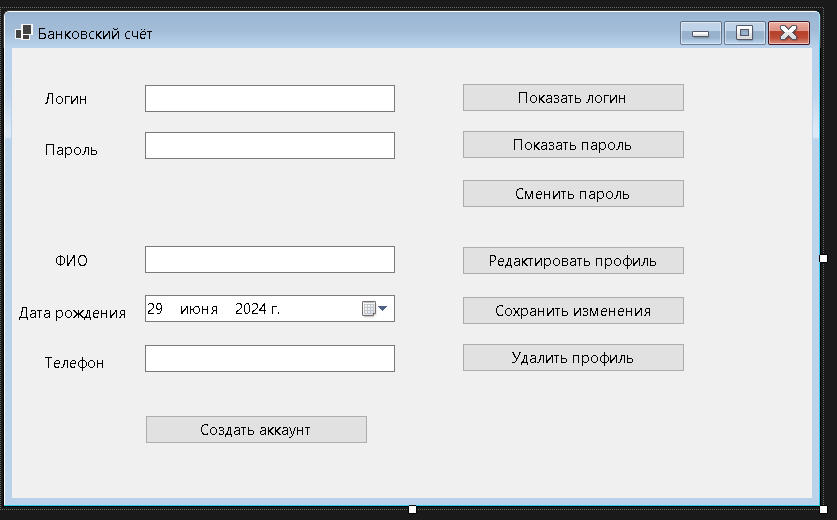
****

Рисунок 7 – Графический пользовательский интерфейс программного обеспечения

## 9. Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению

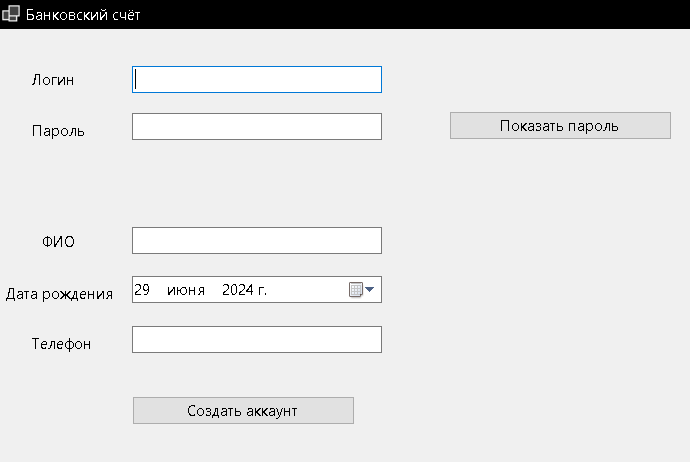
****

Рисунок 8 – Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению

## 10. Расчёт тестовых примеров с использованием составленного программного обеспечения

**Пример 1:**

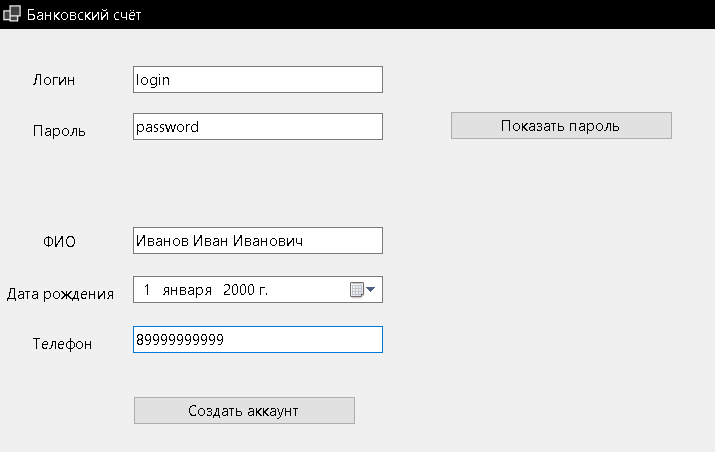


Рисунок 9 – Расчёт тестового примера 1 (1)

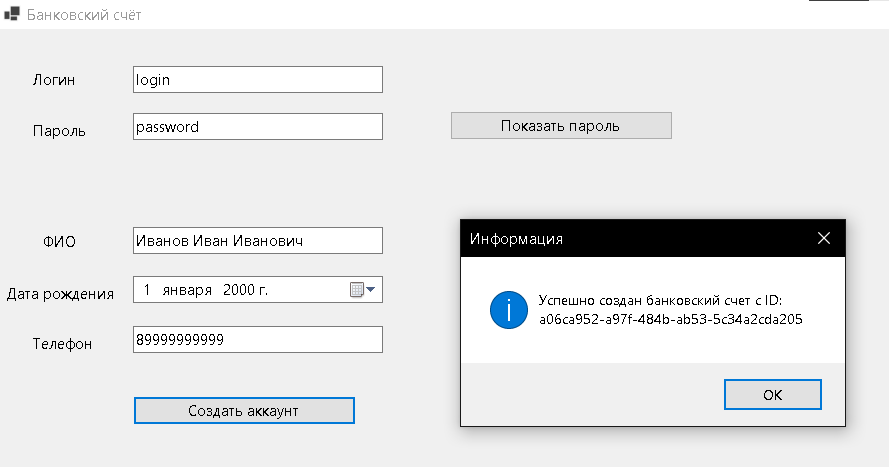


Рисунок 10 – Расчёт тестового примера 1 (2)

**Пример 2:**

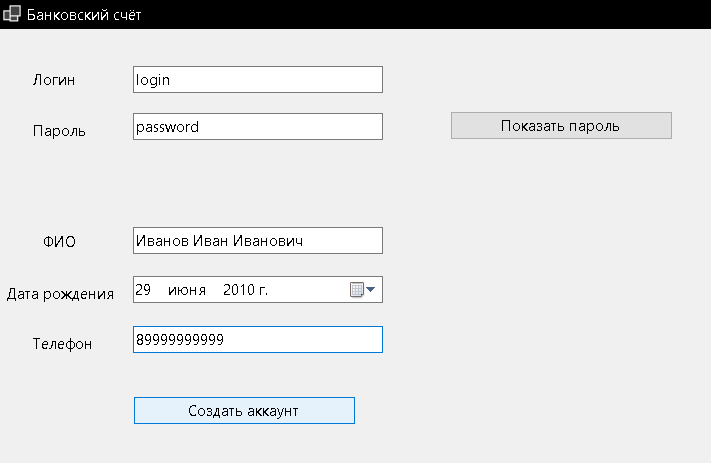


Рисунок 11 – Расчёт тестового примера 2 (1)

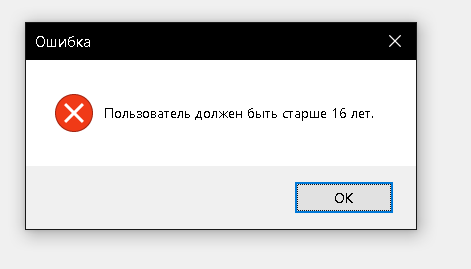


Рисунок 12 – Расчёт тестового примера 2 (2)

**Пример 3:**

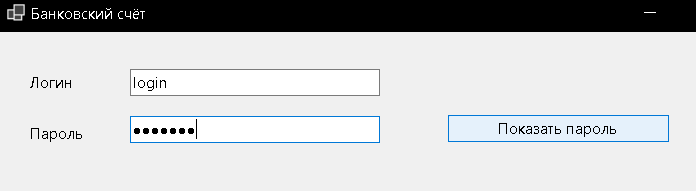


Рисунок 13 – Расчёт тестового примера 3 (1)

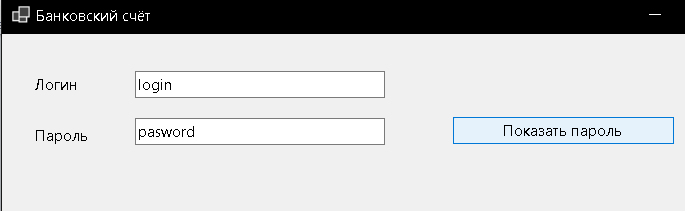


Рисунок 14 – Расчёт тестового примера 3 (2)

**Пример4:**

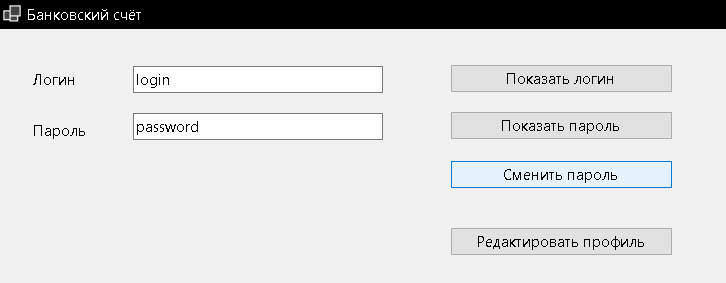


Рисунок 15 – Расчёт тестового примера 4 (1)

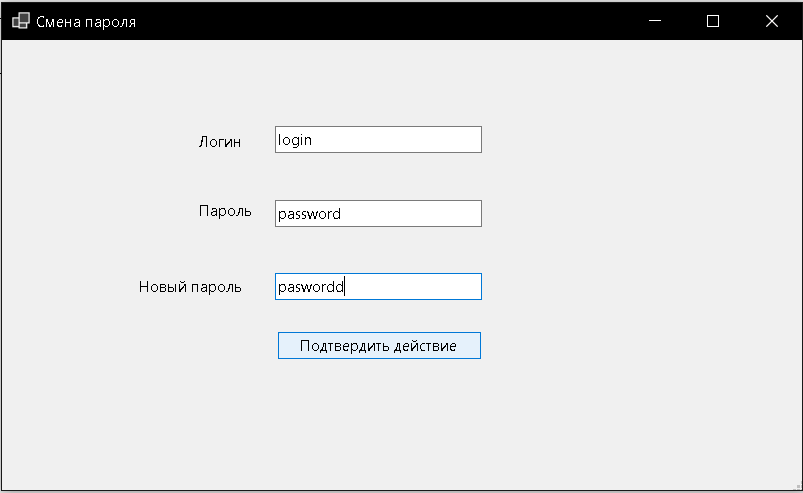


Рисунок 16 – Расчёт тестового примера 4 (2)

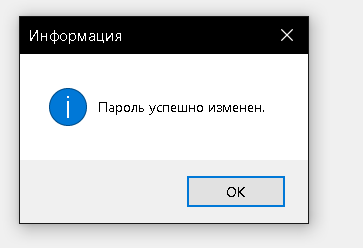


Рисунок 17 – Расчёт тестового примера 4 (3)

**Пример 5:**

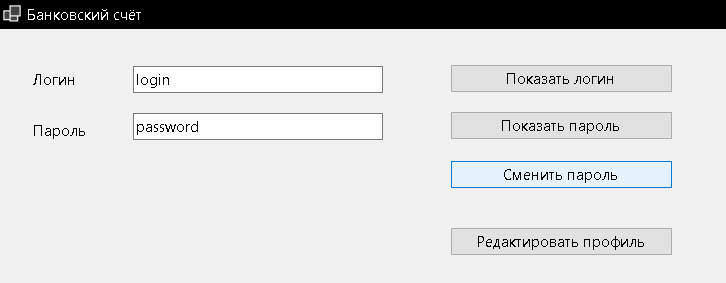


Рисунок 18 – Расчёт тестового примера 5 (1)

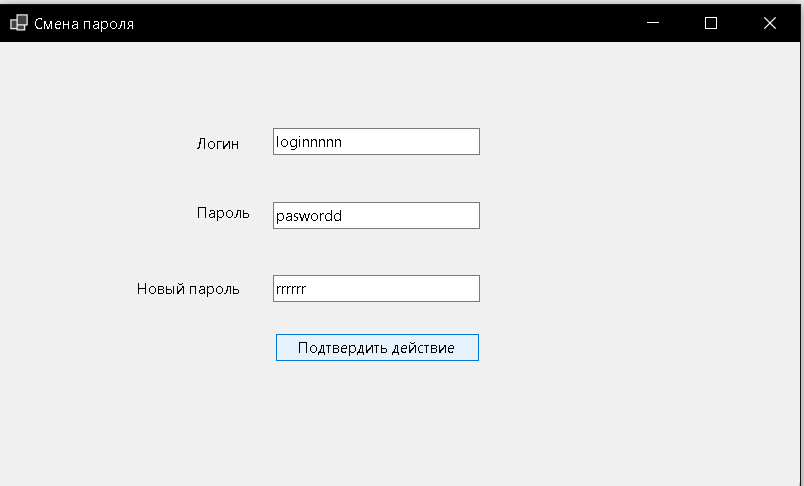


Рисунок 19 – Расчёт тестового примера 5 (2)

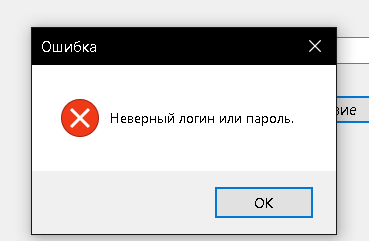


Рисунок 20 – Расчёт тестового примера 5 (3)

**Пример 6:**

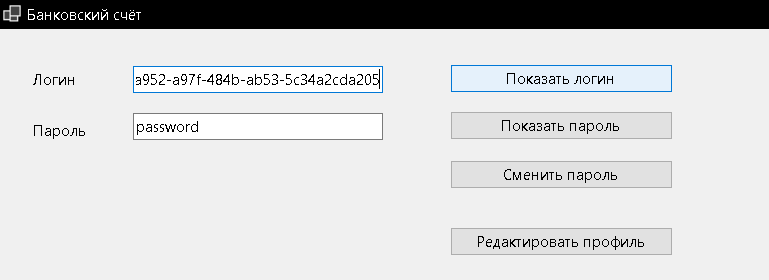


Рисунок 21 – Расчёт тестового примера 6 (1)

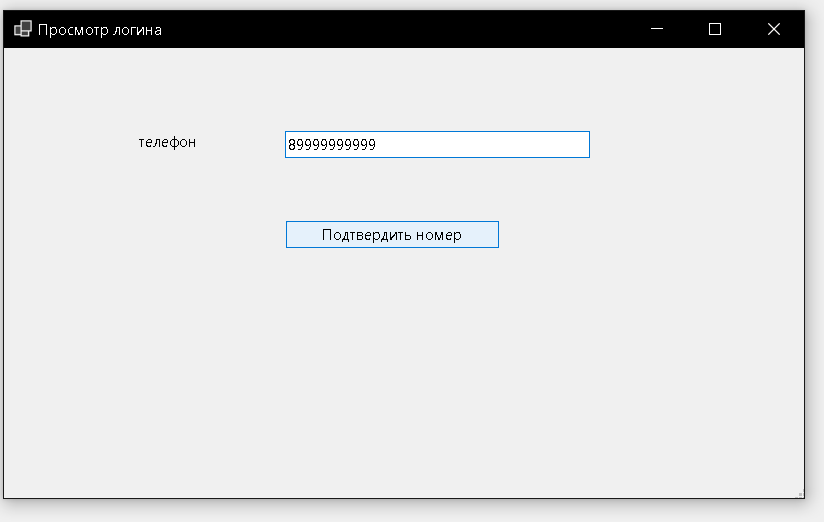


Рисунок 22 – Расчёт тестового примера 6 (2)

**Пример 7:**

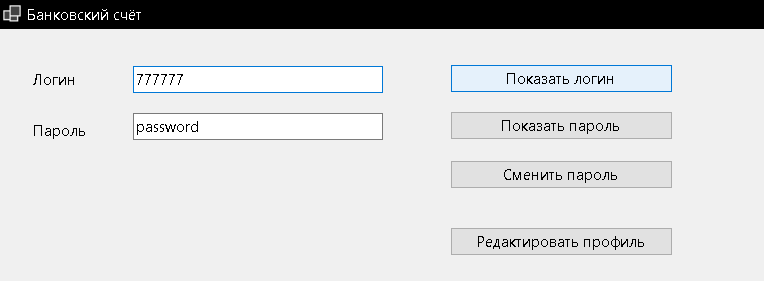


Рисунок 23 – Расчёт тестового примера 7 (1)

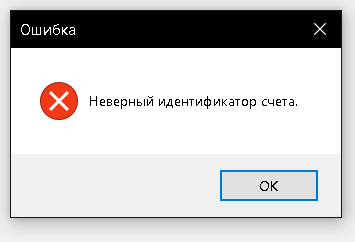


Рисунок 24 – Расчёт тестового примера 7 (2)

**Пример 8:**

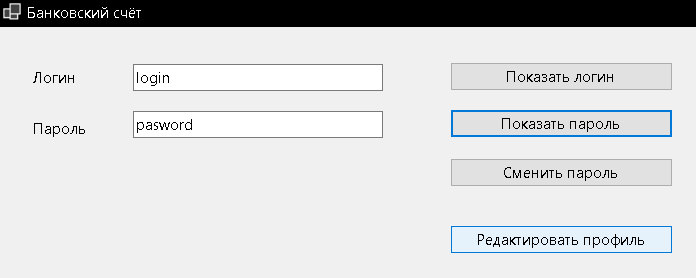


Рисунок 25 – Расчёт тестового примера 8 (1)

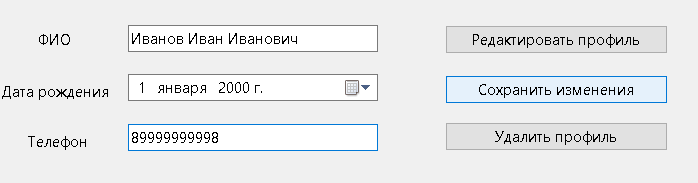


Рисунок 26 – Расчёт тестового примера 8 (2)

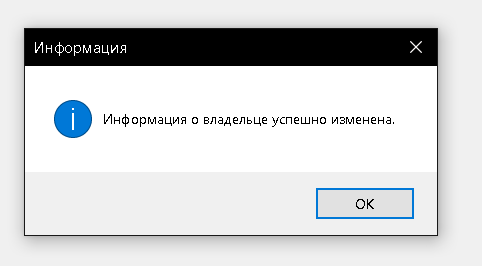


Рисунок 27 – Расчёт тестового примера 8 (3)

**Пример 9:**

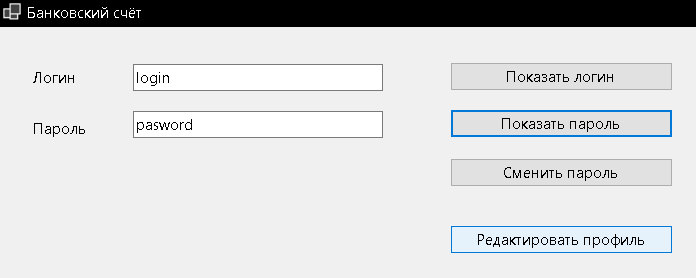


Рисунок 28 – Расчёт тестового примера 9 (1)

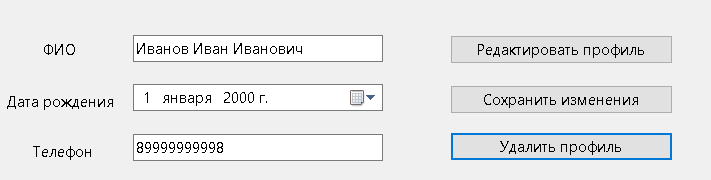


Рисунок 29 – Расчёт тестового примера 9 (2)

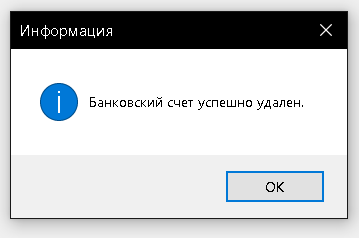


Рисунок 30 – Расчёт тестового примера 9 (3)

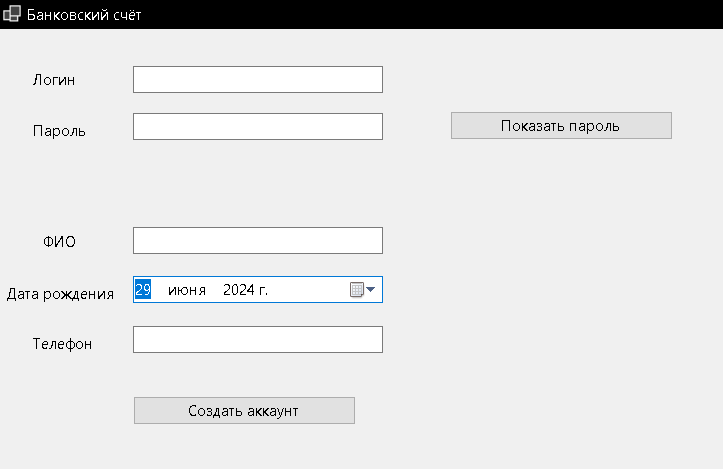


Рисунок 31 – Расчёт тестового примера 9 (4)

## 11. Вывод о проделанной работе

В ходе данной лабораторной работы научился решать задачу по реализации функционала аккаунта банковского счёта в Windows Forms. Научился инкапсулировать классы, научился лучше работать с объектами класса и методами. Также научился работать с несколькими формами в Windows Forms одновременно, что помогло мне в решении поставленной задачи.