МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВИЙ ПРОЕКТ**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: “Об’єктно-орієнтоване програмування”

на тему:

**“Моделювання банкомату”**

студента І курсу групи ІПЗ-19-1

спеціальності 121 “Інженерія програмного забезпечення”

Држевецького Микити Олеговича

(прізвище, ім’я та по-батькові)

Керівник: ст. викладач кафедри ІПЗ

Марчук Г.B

Дата захисту: “13” червня 2020р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Чижмотря

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В. Марчук

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Л. Левківський

(підпис) (прізвище та ініціали)

Житомир – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

“ЗАТВЕРДЖУЮ”

Зав. кафедри ІПЗ

\_\_\_\_\_\_\_\_\_\_\_\_ І.В. Пулеко

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 20\_\_р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Држевецькому Микиті Олеговичу

1. Тема роботи: “Моделювання банкомату” керівник роботи: ст. викладач кафедри ІПЗ Марчук Г.B

2. Строк подання студентом: “29” травня 2020р.

3. Вихідні дані до роботи: “Моделювання банкомату”

4. Зміст розрахунково-пояснювальної записки(перелік питань, які підлягають розробці)

* + - 1. Аналіз технічного завдання
      2. Пошук та дослідження аналогічних розробок
      3. Алгоритми роботи майбутньої програми

4. Опис роботи програми

5. Програмне дослідження

5. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)

1. Презентація до КП.

2. Посилання на репозиторій:

6. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1,2 | Марчук Г.B, ст. викладач каф. ІПЗ |  |  |

1. Дата видачі завдання “17” лютого 2020 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Постановка задачі | 15.04.20 -17.04.20 |  |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 18.04.20 -21.04.20 |  |
| 3 | Формулювання технічного завдання | 21.04.20 – 22.04.20 |  |
| 4 | Опрацювання літературних джерел | 22.04.20 – 05.05.20 |  |
| 5 | Проектування структури | 05.05.20 – 07.05.20 |  |
| 6 | Написання програмного коду | 07.05.20 – 29.05.20 |  |
| 7 | Налагодження | 01.06.20 – 10.06.20 |  |
| 8 | Написання пояснювальної записки | 10.06.20 –  12 06.20 |  |
| 9 | Захист | 13.06.20 |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.О Држевецький

(підпис) (прізвище та ініціали)

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В Марчук

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсового проекту на тему “Інформаційно – пошукова система: футбольні змагання” складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на **35** сторінках друкованого тексту.

Пояснювальна записка має **34** сторінки додатків. Список використаних джерел містить **10** найменувань і займає **1** сторінку. В роботі наведено **35** рисунків. Загальний обсяг роботи – **70** сторінки.

У першому розділі було обґрунтовано створення програми для зберігання, пошуку і відображення інформації про футбольні змагання.

У другому розділі проведено проектування і розробка програмного продукту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи створення програми для зберігання, пошуку і відображення інформації про змагання.

У додатку представлений лістинг розробленого програмного продукту.

Ключові слова: ООП, БАЗА ДАНИХ, ВІДОБРАЖЕННЯ ІНФОРМАЦІЇ, ДОСТУП, ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ, ПОШУК.

**ЗМІСТ**

[**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ** 6](#_Toc41609137)

[**ВСТУП** 7](#_Toc41609138)

[**РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ** 9](#_Toc41609139)

[1.1 Аналіз задачі, засобів та методів її вирішення 9](#_Toc41609140)

[1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи. 10](#_Toc41609141)

[1.3 Технічне завдання на курсову роботу 12](#_Toc41609142)

[Висновки до першого розділу 13](#_Toc41609143)

[**РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ** 14](#_Toc41609144)

[2.1 Проектування загального алгоритму роботи програми 14](#_Toc41609145)

[2.2 Розробка функціональних алгоритмів роботи програми 16](#_Toc41609146)

[2.3 Розробка програмного забезпечення 16](#_Toc41609147)

[Висновки до другого розділу 22](#_Toc41609148)

[**РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ** 23](#_Toc41609149)

[3.1 Опис роботи додатку та його тестування 23](#_Toc41609150)

[Висновки до третього розділу 33](#_Toc41609151)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 35](#_Toc41609152)

[**ДОДАТОК** 26](#_Toc41609153)

# **ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

VS – Visual Studio

ПЗ – Програмне забезпечення

БД – База даних

# **ВСТУП**

У цій курсовій роботі буде наведено приклад банкомату.

**Банкома́т** (скорочення від «банківський автомат», іноді **ATM**, від [англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *automated teller machine* — «автоматичний касовий апарат») — електронний програмно-технічний комплекс з вмонтованою спеціалізованою [ЕОМ](https://uk.wikipedia.org/wiki/%D0%95%D0%9E%D0%9C), призначений для здійснення автоматизованих операцій видачі наявних [грошових коштів](https://uk.wikipedia.org/wiki/%D0%93%D0%BE%D1%82%D1%96%D0%B2%D0%BA%D0%B0), зокрема з використанням [платіжних карток](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%BD%D0%BA%D1%96%D0%B2%D1%81%D1%8C%D0%BA%D0%B0_%D0%BF%D0%BB%D0%B0%D1%82%D1%96%D0%B6%D0%BD%D0%B0_%D0%BA%D0%B0%D1%80%D1%82%D0%BA%D0%B0), передачі розпоряджень [банку](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%BD%D0%BA) про перерахування грошових коштів з [банківського рахунку](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%BD%D0%BA%D1%96%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D1%80%D0%B0%D1%85%D1%83%D0%BD%D0%BE%D0%BA) клієнта та виконання інших операцій: оплати товарів, послуг; для автоматизованого складання документів, що підтверджують відповідні операції (видача паперових касових [чеків](https://uk.wikipedia.org/wiki/%D0%A7%D0%B5%D0%BA)). Інколи банкомати можуть також мати функцію прийому грошових коштів, в такому випадку вони називаються депозитними банкоматами.

# **РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ**

## Аналіз задачі, засобів та методів її вирішення

Задача полягає в тому, що потрібно:

1. Вибрати 1 – ну із запропонованих карток
2. Ввести правельний PIN
3. Обрати сумму для видачі готівки
4. Забрати гроші

В процесі створення банкомату брався будь-який аспект проекту, розроблялася його найлегша версія, а після того вдосконалювалася, чи повністю перероблювалася.

Було вирішено розробляти БД в Visual Studio за допомогою додатку WinForms, оскільки він має широкі можливості в створенні функціоналу програми.

## Аналіз існуючого програмного забезпечення за тематикою курсової роботи.

Для аналізу існуючого програмного забезпечення за тематикою курсової роботи я вибрав банкомат Приват банку, на мою думку це є найращий приклад.

Для збереження усіх данних користовувача я використав JSON.

JSON - це текстовий формат обміну даними між комп'ютерами

Переваги JSON

1) Компактність

2) Читабельність

3) Зрозумілість

1.3 Технічне завдання на курсову роботу

**Підстава для розробки:**

Підставою для розробки курсової роботи є завдання на курсову роботу.

**Призначення розробки:**

Програма призначена для індивідуального використання користувачем для зняття готівки

**Вимоги до програмного засобу:**

БД має інформацію про баланс вашої картки, номер телефону та PIN - код. У відповідь на некоректні дії користувача має з’явитись попередження та підказка разом з можливістю знову виконати ті ж самі дії вже коректним способом.

Програма повинна виконувати наступні функції:

1. Видача говітки.
2. Звязок з JSON файлом ​
3. Перевірка балансу
4. Можливість користуватися банкоматом без картки​

**Вимоги до надійності:**

Передбачити ситуації при виконанні некоректних дій як зі сторони звичайного користувача.

**Вимоги до інформаційної і програмної сумісності:**

Повинна бути встановлена Visual Studio 2013 і вище.

**Спеціальні вимоги**

Немає.

## Висновки до першого розділу

У ході виконання першого розділу було визначено дозволені можливості обох типів користувачів та їх зону роботи, можливість доступу до інформації з різними рівнями безпеки.

Далі було визначено необхідні інструменти для реалізації програмного забезпечення, а також інтерфейсу та функціонування БД, що створюється.

# **РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

## 2.1 Проектування загального алгоритму роботи програми

1.На початку программи користувач повинен обрати 1 з 4 – ох карток (на вибір). Обравши картку користувач повинен вставити картку у приймач. Після цього користувач повинен ввести правельний Pin код (який записаний у JSON).

Далі ви повинні вибрати сумму, яку ви хочете отримати. Також ви можете перевірити баланс вашої картки.

1. Якщо користувач хоче провести операцію без картки, він повинен зробити наступні дії. Користувач повинен вести свій номер телефону, цей номер телефону понен бути прив’язаний до 1 з 4 – ох карток. У разі неправильного вводу телефону, программа виведе попередження. Після того як користувач ввів номер телефону, відкриваеться наступна форма.

На ваш телефон приходить рандомний код, який ви повинні ввести. Після правильного вводу коду, ви можете роботи дії які були у 1 – му пункті.

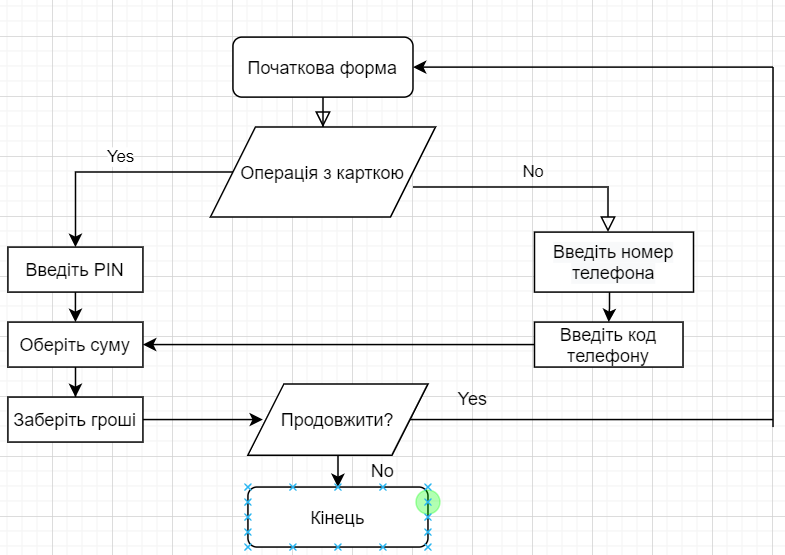


Рис.2.1. Блок-схема програми.

## 2.2 Розробка функціональних алгоритмів роботи програми

При розробці програми було важливо вирішити проблему зв’язку форм між собою, забезпечивши єдиний «банк» для зберігання інформації на кожному кроці користування банкоматом. В даному випадку було вирішено використовувати один з шаблонів проектування – Singleton, або одиначка. Також, окремий статичний клас для зберіганя опрацьованої картки.

Також, важливим завданням було забезпечити зберігання інформації про картки, моделюючи свого роду “банківську базу даних”. Це було досягнуте шляхом серіалізації об’єктів типу “Картка” в NoSQL базу даних на базі JSON

## 2.3 Розробка програмного забезпечення

2.3.1 Розробка інтерфейсу головного меню.

Для розробки інтерфейсу було використано Visual Studio 2019 та створено додаток WinForms. Оскільки він дозволяє реалізувати візуальні можливості зрозумілі як для адміністратора, так і для звичайного користувача.

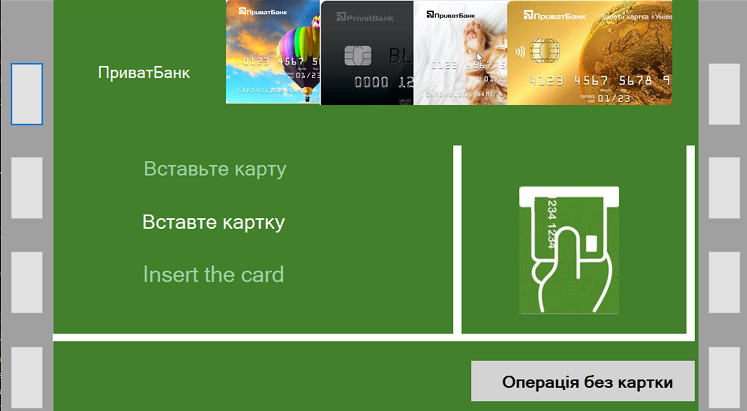


Рис.2.2

На початку роботи програми будь-якого користувача зустрічає вікно

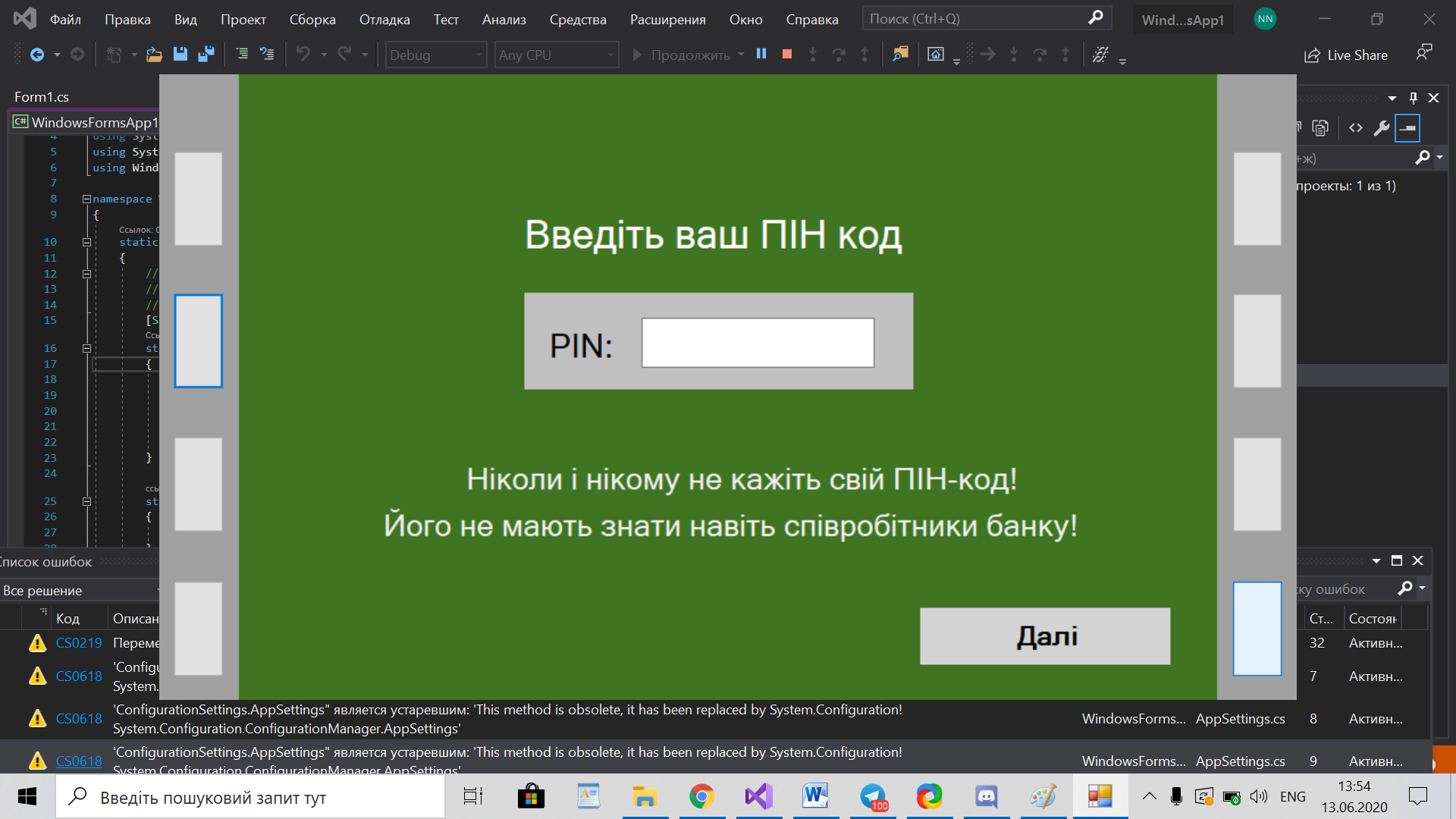


Рис.2.3 Введення PIN

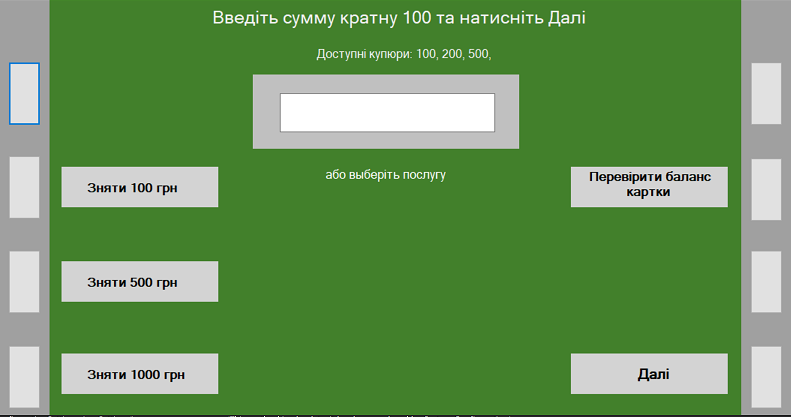


Рис.2.4. Вибір суми.

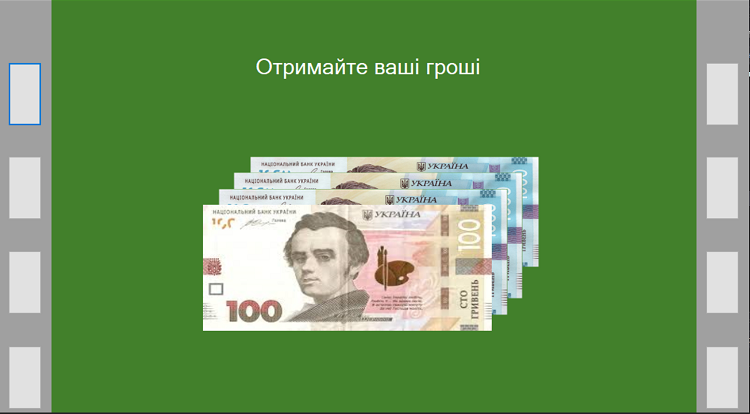


Рис.2.5.Видача готівки

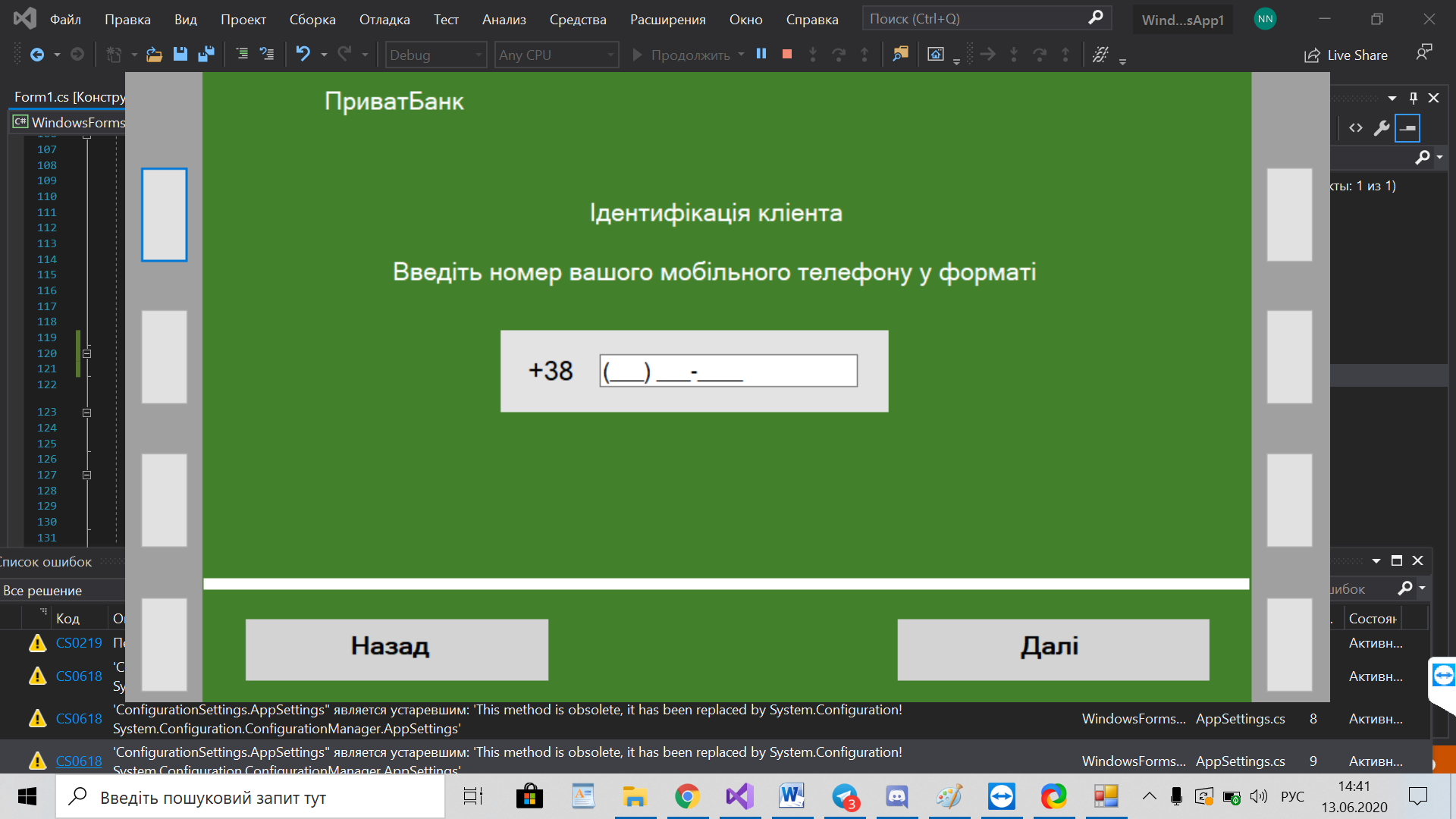


Рис.2.6. Введення телефону

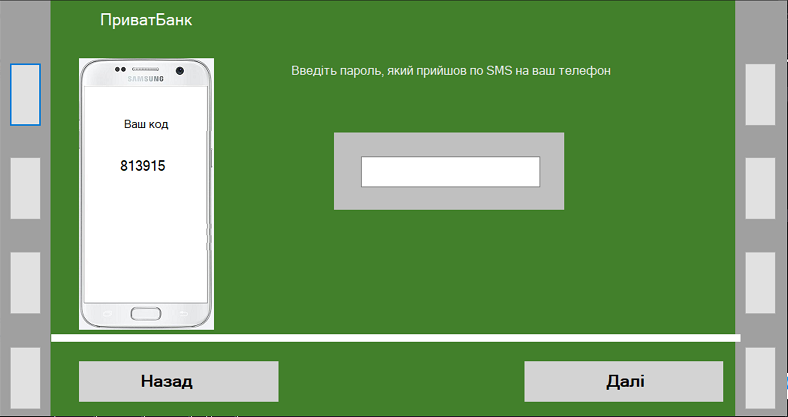


Рис.2.7. Перевірка коду.



Рис.2.8. Інформація JSON

Рис.2.10. Діаграма класів.

## Висновки до другого розділу

У ході виконання другого розділу було спроектовано, у вигляді блок-схеми, загальний функціонал програмного додатку, яка у повному обсязі її описує, а також створено діаграму класів та реалізовано графічний інтерфейс програми.

# **РОЗДІЛ 3. ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ**

## 3.1 Опис роботи додатку та його тестування

На даному етапі курсової роботи буде продемонстрована робота вже реалізованої БД, а також протестовано можливості при стандартній ситуації та при можливості некоректного введення даних.

Тож розпочнемо тестування, ілюструючи його відповідними рисунками.

При вході в БД користувача зустрічає вікно реєстрації.

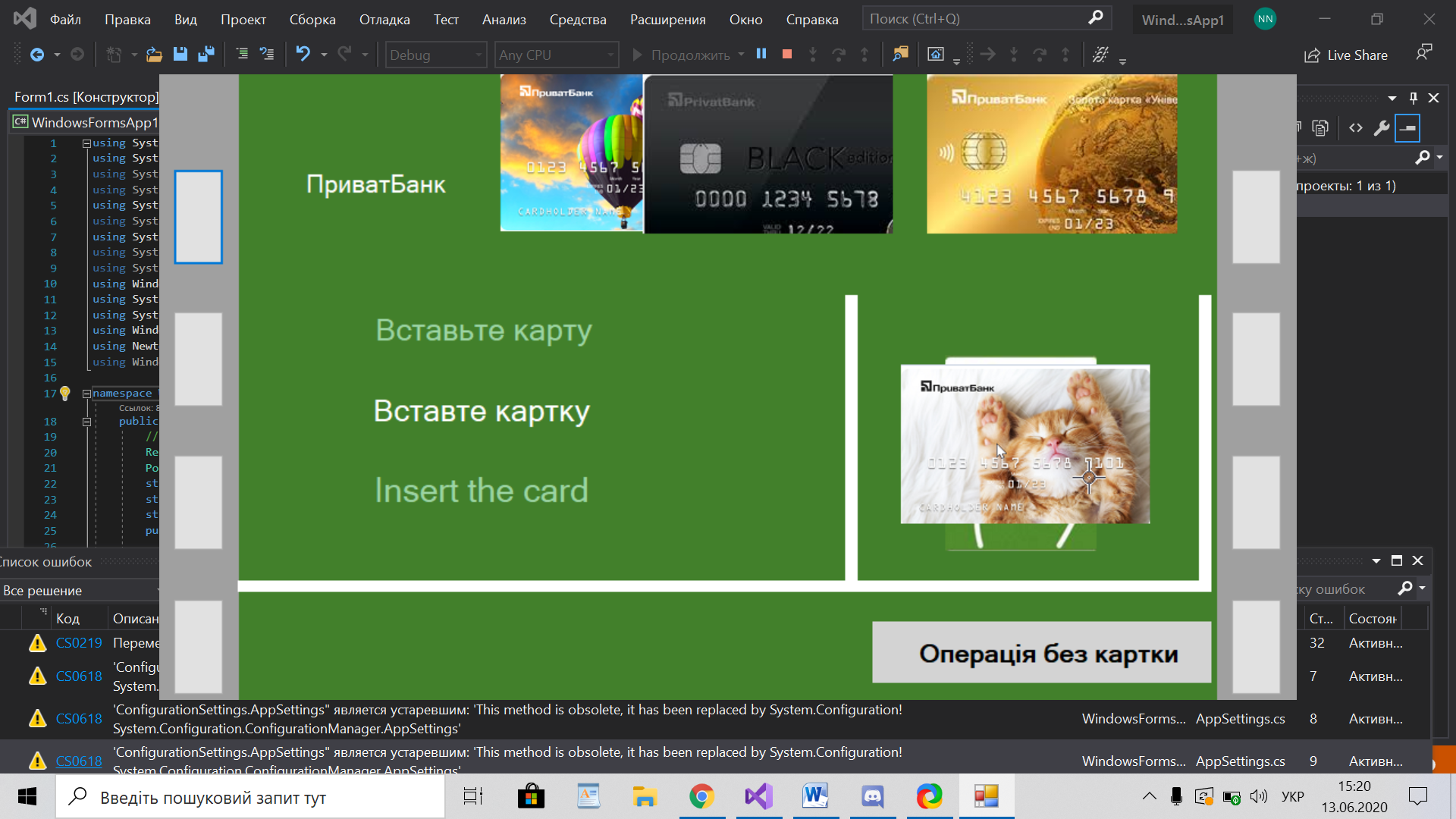


Рис.3.1.

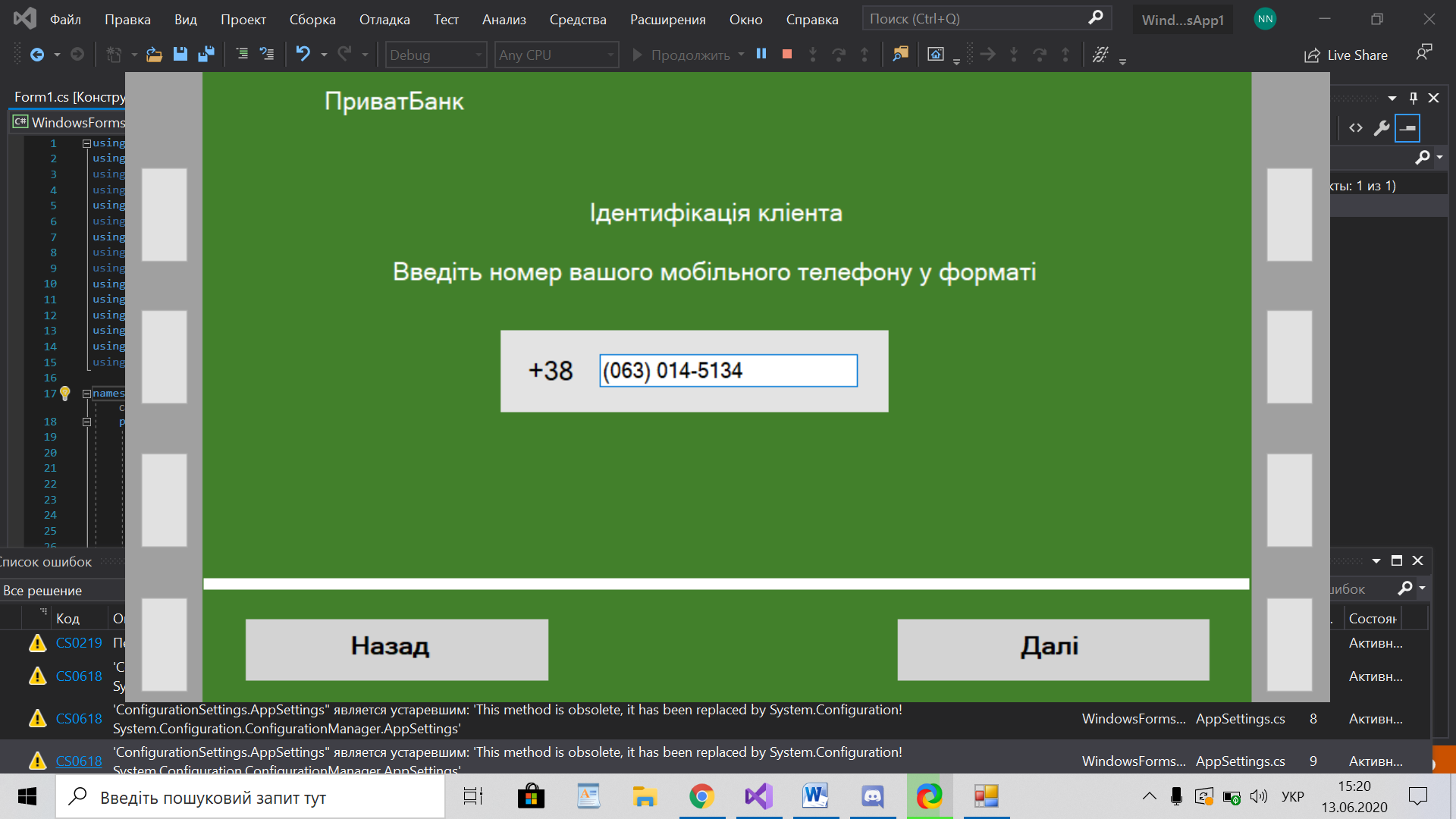


Рис.3.2. Введення пароля.

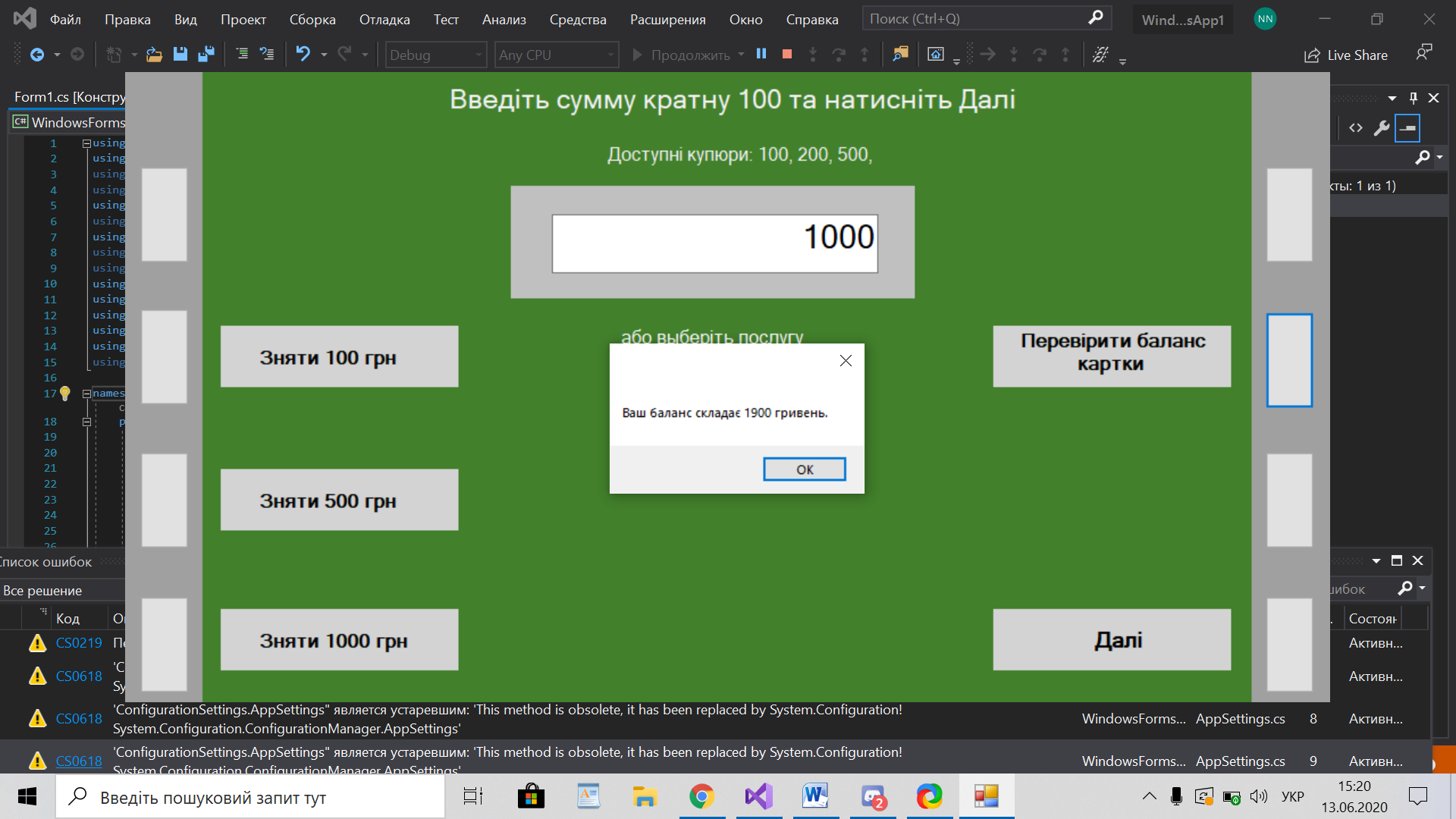


Рис.3.3. Вибір суми

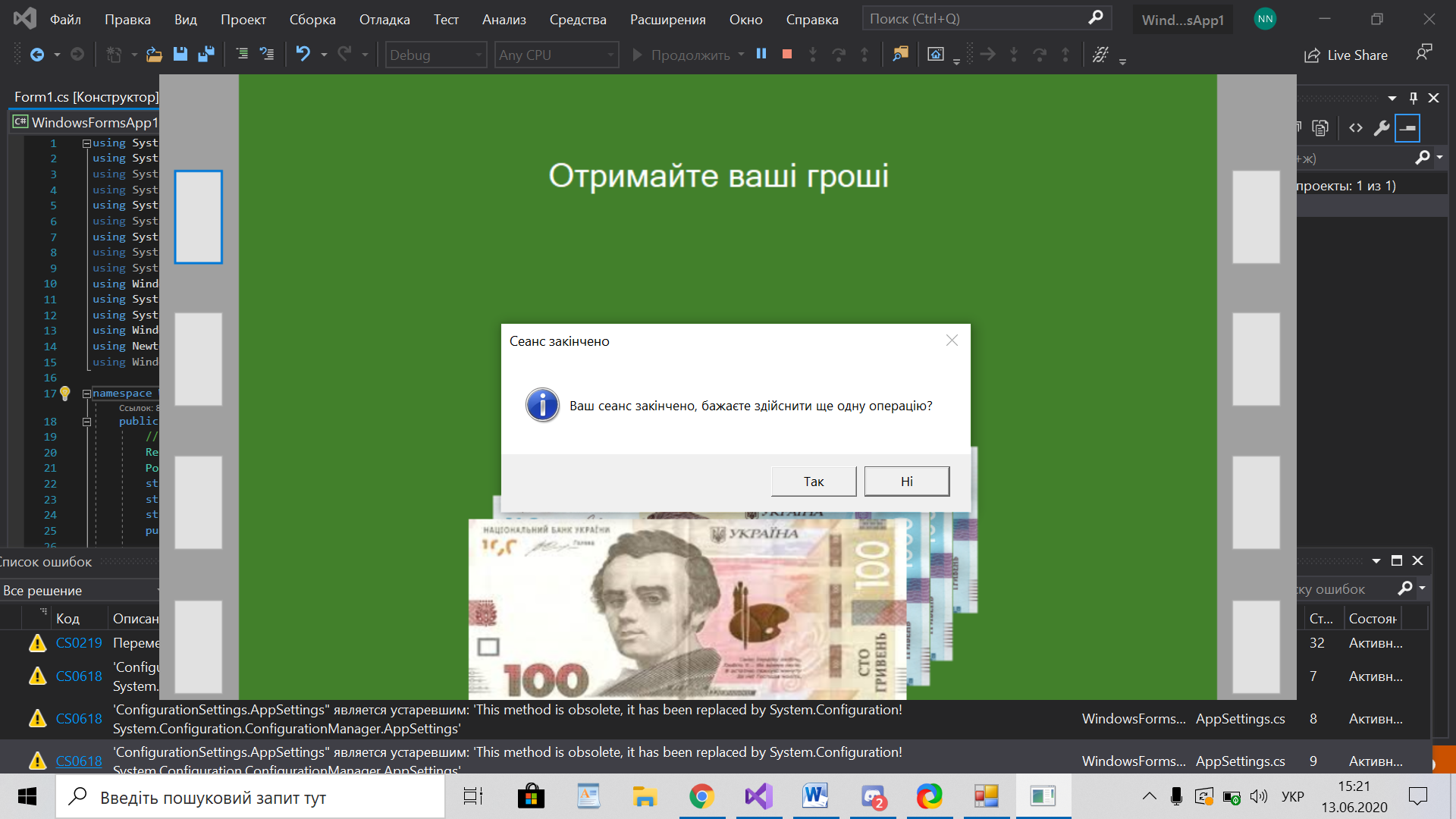
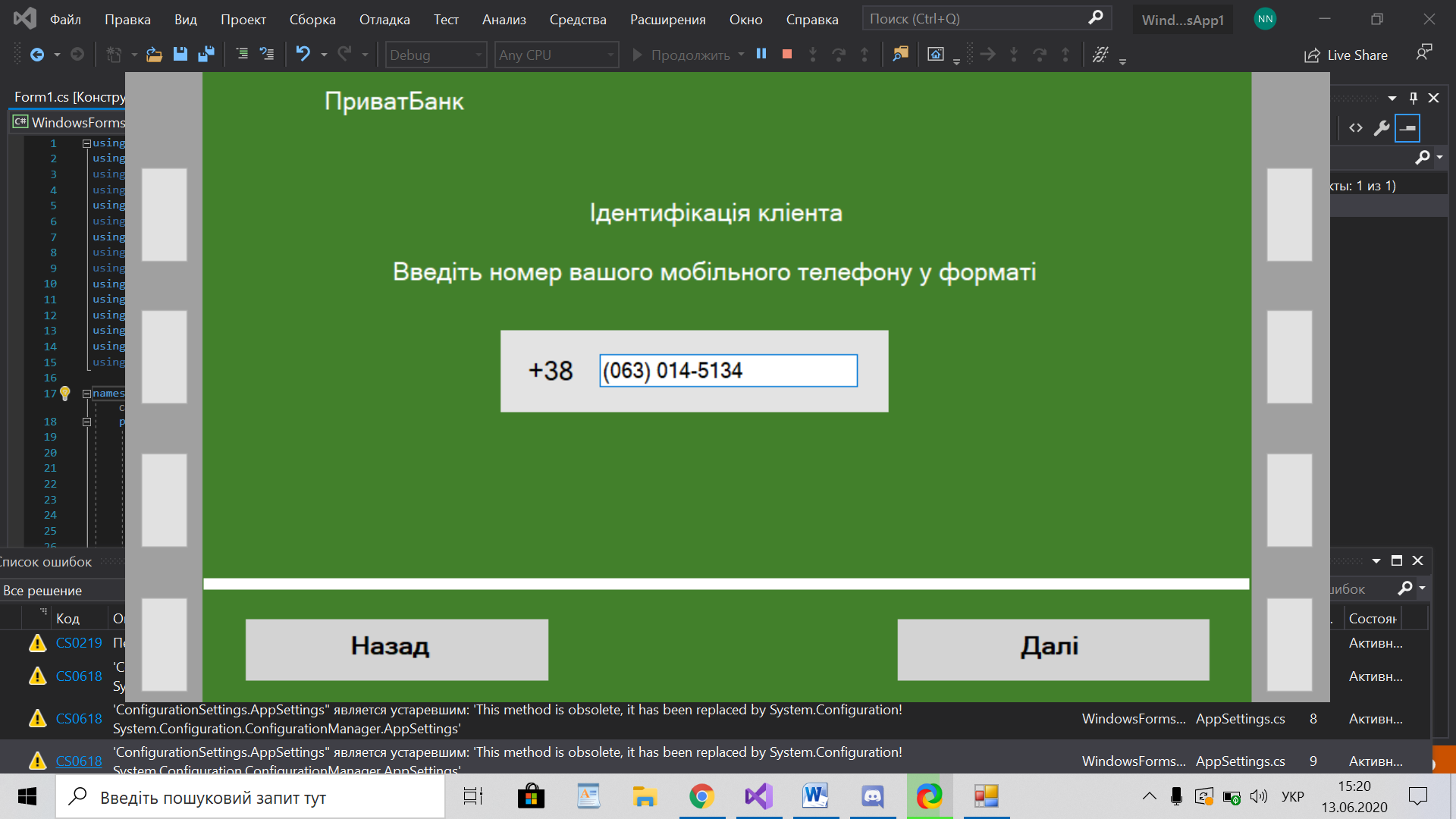
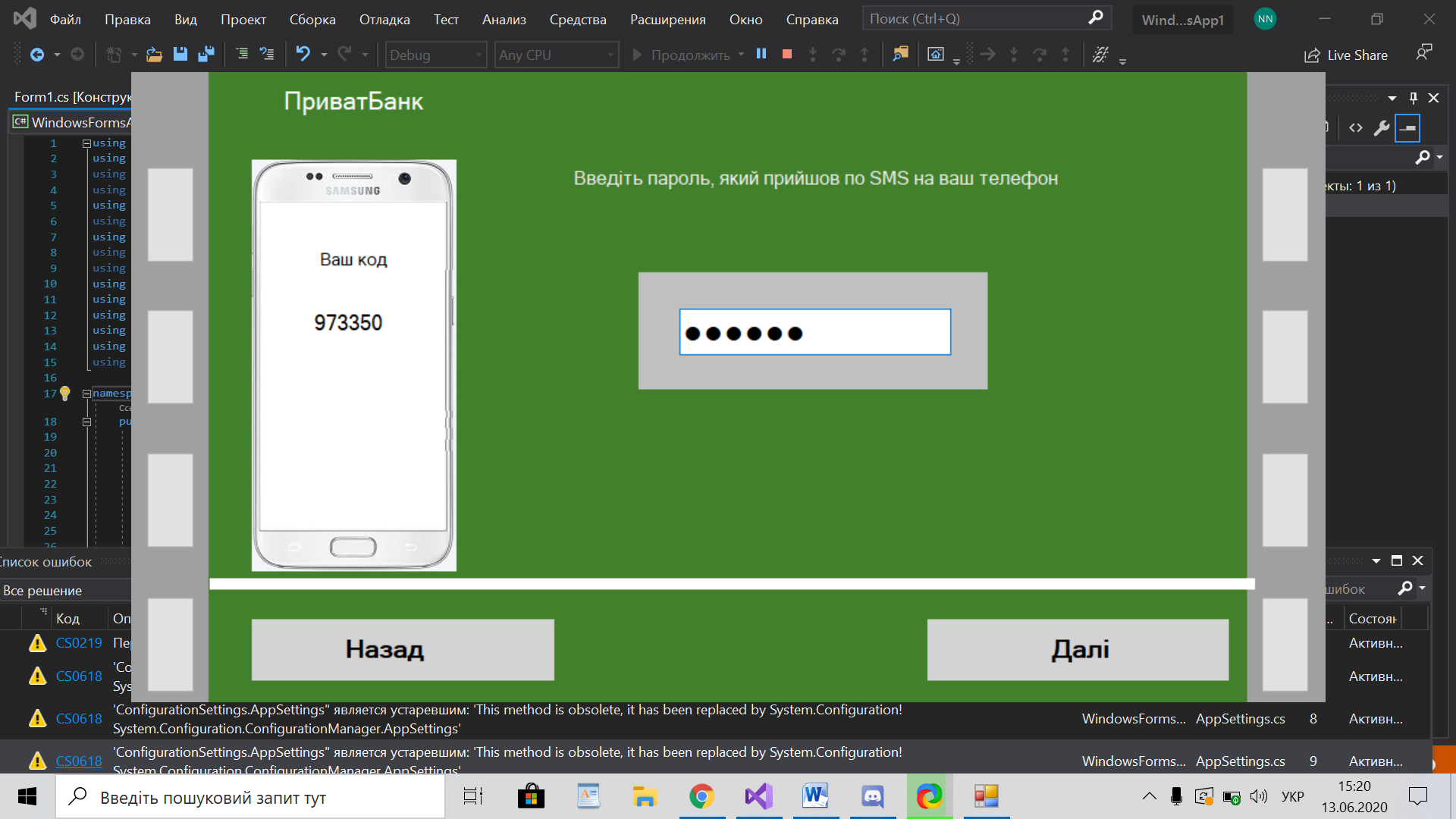
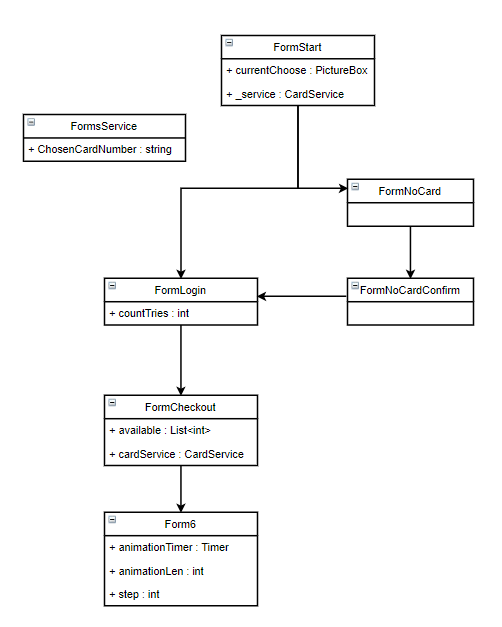
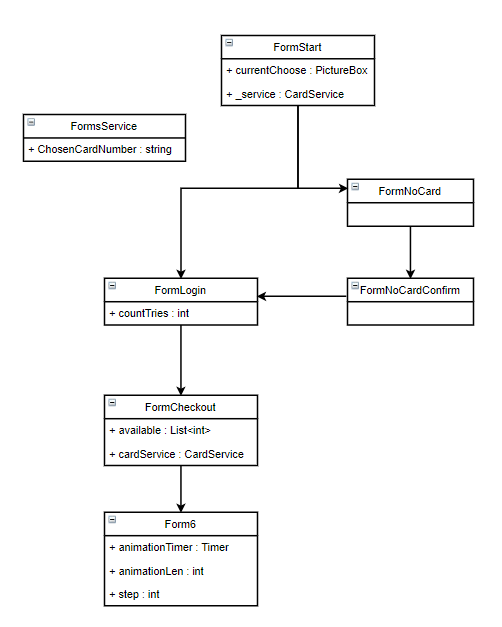


Рис.3.4. Видача готівки

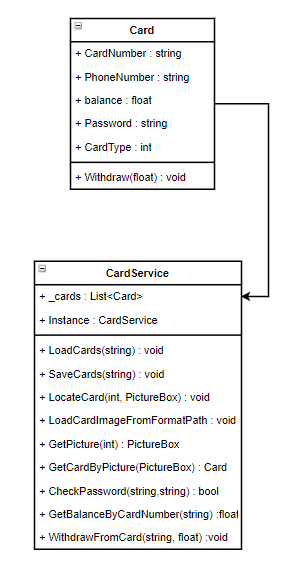
Рис.3.5. Вікно входу в програму.







**Діаграма Рис1**



**Діаграма Рис1**

відповідних рисунків, які ілюстрували це на практиці. Під час роботи різного роду проблем не виникало. БД є простою та зручною в користуванні.

**ВИСНОВКИ**

В першому розділі курсового проекту було сформовано уявлення про БД, що розроблюється та виявлено основні потреби для написання цієї БД.

В другому розділі курсового проекту було проведено розробку функціональних і загальних частин БД. При розробці функціонального алгоритму було проведено дослідження методів розробки програмного коду додатку. На заключному етапі було розроблено програмний код спроектованого додатку. В результаті було отримано готовий програмний продукт який виконував всі заплановані функції для усіх можливостей.

В третьому розділі курсового проекту було описано методи користування створеної БД, були пояснені і продемонстровані всі створені елементи інтерфейсу в формі. Також було проведено тестування, тобто БД було запущено в реальних умовах її користування. В ході тестування проблем не виникало.

В результаті виконання курсового проекту отримано БД, яка повністю відповідає запланованому функціоналу та поставленим на початку проектування вимогам. БД є простою та зручною в користуванні.

# **ДОДАТОК**

Додаток

Програмний код

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Configuration;

using WindowsFormsApp1.Services;

using System.Windows.Forms;

using System.IO;

using WindowsFormsApp1.Models;

using Newtonsoft.Json;

using WindowsFormsApp1.Properties;

namespace WindowsFormsApp1 {

public partial class FormStart : Form {

// Location 300:0, Size 220:140

Rectangle startRectangle = new Rectangle(300, 0, 220, 140);

Point startDragging;

static CardService cardService;

static Random rnd;

static PictureBox currentChoose;

public static int cardTypes;

public FormStart() {

InitializeComponent();

rnd = new Random((int)(DateTime.Now.Ticks));

var dirPath = AppSettings.jsonDirPath;

var fileName = AppSettings.jsonFileName;

int startIndex = 0;

PictureBox cardBox;

cardService = CardService.Instance;

cardService.LoadCards(dirPath + fileName);

for (int i = 0; i < cardService.Count(); i++) {

Rectangle current = startRectangle;

current.X += (i \* (500 / cardService.Count()));

cardService.LocateCard(i, current);

cardService.LoadCardImageFromFormatPath(i, AppSettings.cardImgFormat);

cardBox = cardService.GetPicture(i);

cardBox.MouseEnter += CardBox\_MouseEnter;

cardBox.MouseLeave += CardBox\_MouseLeave;

cardBox.MouseDown += CardBox\_MouseDown;

cardBox.MouseUp += CardBox\_MouseUp;

cardBox.MouseMove += CardBox\_MouseMove;

this.Controls.Add(cardBox);

cardBox.BringToFront();

//startIndex += startIndex == 0 ? Controls.GetChildIndex(cardBox) : 1;

//this.Controls.SetChildIndex(cardBox, startIndex);

}

}

private void FormStart\_Load(object sender, EventArgs e)

{

}

private void CardBox\_MouseUp(object sender, MouseEventArgs e) {

if (currentChoose.Bounds.IntersectsWith(Controls["cardInput"].Bounds)) {

FormsService.ChosenCardNumber = cardService.GetCardByPicture(currentChoose).CardNumber;

MoveToLogin();

}

}

private void CardBox\_MouseDown(object sender, MouseEventArgs e) {

if (e.Button == MouseButtons.Left)

startDragging = new Point(e.X, e.Y);

}

private void CardBox\_MouseMove(object sender, MouseEventArgs e) {

if (e.Button != MouseButtons.Left)

return;

currentChoose.Top = currentChoose.Top + (e.Y - startDragging.Y);

currentChoose.Left = currentChoose.Left + (e.X - startDragging.X);

}

public void MoveToLogin()

{

this.Visible = false;

FormLogin formLogin = new FormLogin();

formLogin.Show();

formLogin.TopLevel = true;

}

private void CardBox\_MouseLeave(object sender, EventArgs e) {

PictureBox left = sender as PictureBox;

var shifted = left.Location;

shifted.Y -= 30;

left.Location = shifted;

}

private void CardBox\_MouseEnter(object sender, EventArgs e) {

currentChoose = sender as PictureBox;

var shifted = currentChoose.Location;

shifted.Y += 30;

currentChoose.Location = shifted;

}

private void no\_cardButton\_Click(object sender, EventArgs e) {

FormNoCard form = new FormNoCard();

form.Show();

this.Visible = false;

form.Focus();

}

private void button8\_Click(object sender, EventArgs e) {

List<Card> cards = new List<Card>();

var dirPath = AppSettings.jsonDirPath;

var fileName = AppSettings.jsonFileName;

var typesCount = Directory.GetFiles("./Cards/").Length;

cards.Add(new Card(364, rnd.Next(1, typesCount)) {CardNumber = "1234567890", PhoneNumber="+380637012889", Password = "1234" });

cards.Add(new Card(100000, rnd.Next(1, typesCount)) { CardNumber = "6578493021", PhoneNumber = "+380660192339", Password = "1234" });

cards.Add(new Card(3000, rnd.Next(1, typesCount)) { CardNumber = "0192837465", PhoneNumber = "+380972134977", Password = "1234" });

cards.Add(new Card(5610, rnd.Next(1, typesCount)) { CardNumber = "1029384756", PhoneNumber = "+380123456789", Password = "1234" });

var jsonData = JsonConvert.SerializeObject(cards, Formatting.Indented);

if (!Directory.Exists(dirPath))

Directory.CreateDirectory(dirPath);

File.WriteAllText(dirPath + fileName, jsonData, Encoding.UTF8);

}

// Створює базовий json файл, для подальшої роботи програми.

// Виконується "адміністратором" при першому заупуску програми, для "доступу" до бд з картками

private void controlLeftPanel\_Paint(object sender, PaintEventArgs e) {

}

//private void pictureBox3\_Click(object sender, EventArgs e)

//{

// currentObject = sender;

//}

}

}

Form2

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using WindowsFormsApp1.Services;

namespace WindowsFormsApp1

{

public partial class FormLogin : Form

{

int countTries = 0;

public FormLogin()

{

InitializeComponent();

this.textPASS.AutoSize = false;

this.textPASS.Size = new Size(this.textPASS.Width, 44); // присвоюємо ширину text box

}

private void button8\_Click(object sender, EventArgs e)

{

Form Form3 = new FormCheckout();

Form3.Show();

Hide();

}

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

{

CardService cardService = CardService.Instance;

string password = textPASS.Text;

if (password.Length != 4)

{

MessageBox.Show(

"Введіть 4 цифри пінкода!",

"Помилка в введені паролю",

MessageBoxButtons.OK,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button2

);

textPASS.Text = "";

return;

}

countTries++;

if (!cardService.CheckPassword(FormsService.ChosenCardNumber, password))

{

if (countTries < 3)

MessageBox.Show(

$"Пінкод неправильний!\nСпроб залишилось: {3 - countTries}",

"Помилка в введені паролю",

MessageBoxButtons.OK,

MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button2

);

else

{

var res = MessageBox.Show(

"Пінкод неправильний! \nСпроб залишилось: 0. \nТікайте, поліцію викликано, а карту заблоковано!",

"Помилка в введені паролю",

MessageBoxButtons.OK,

MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button2

);

if (res == DialogResult.OK)

Application.Exit();

}

textPASS.Text = "";

return;

}

this.Hide();

FormCheckout formCheckout = new FormCheckout();

formCheckout.Show();

}

private void textPASS\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!char.IsDigit(number)&& number != 8)

e.Handled = true; // перевіряемо чи був оброблений ввод с клавіатури

}

}

}

Form3

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using WindowsFormsApp1.Services;

namespace WindowsFormsApp1

{

public partial class FormCheckout : Form

{

const string textFormatMultiply = @"Введіть сумму кратну \* та натисніть Далі ";

const string textFormatAvailable = @"Доступні купюри: \*";

List<int> available;

float balance;

CardService cardService = CardService.Instance;

public FormCheckout()

{

available = new List<int>() {

100, 200, 500 // Генерація буде

};

InitializeComponent();

}

private void FormCheckout\_Load(object sender, EventArgs e) {

available.Sort(); // Найменше - перше

string availableUahText = "";

foreach (var item in available) {

availableUahText += item.ToString() + ", ";

}

balance = cardService.GetBalanceByCardNumber(FormsService.ChosenCardNumber);

labelAvailable.Text = textFormatAvailable.Replace("\*", availableUahText);

labelMulti.Text = textFormatMultiply.Replace("\*", available[0].ToString());

}

private void Form3\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void textPASS\_KeyDown(object sender, KeyEventArgs e) {

e.Handled = (e.KeyCode < Keys.NumPad0 || e.KeyCode > Keys.NumPad9) ||

(e.KeyCode < Keys.D0 || e.KeyCode > Keys.D9);

e.Handled = e.KeyCode == Keys.Back || e.Handled;

}

private void buttonBalance\_Click(object sender, EventArgs e) {

MessageBox.Show(

$"Ваш баланс складає {Math.Round(balance, 2)} гривень."

);

}

private void button2\_Click(object sender, EventArgs e) {

textPASS.Text = "100";

}

private void button3\_Click(object sender, EventArgs e) {

textPASS.Text = "500";

}

private void button4\_Click(object sender, EventArgs e) {

textPASS.Text = "1000";

}

private void buttonWithdrown\_Click(object sender, EventArgs e) {

if(textPASS.Text.Length < 2) {

MessageBox.Show(

"Введіть коректну суму для виводу!"

);

return;

}

float.TryParse(textPASS.Text, out float amount);

if((int)amount % available[0] != 0) {

MessageBox.Show(

$"Введіть коректну суму для виводу, кратну {available[0]}!"

);

return;

}

if(balance < amount) {

MessageBox.Show(

"На балансі недостатньо коштів! \nГрусть, печаль, тоска :("

);

return;

}

cardService.WithdrawFromCard(FormsService.ChosenCardNumber, amount);

Form6 form = new Form6();

this.Hide();

form.Show();

form.Focus();

return;

}

private void textPASS\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!char.IsDigit(number) && number != 8)

e.Handled = true; // перевіряемо чи був оброблений ввод с клавіатури

}

}

}

Form4

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using WindowsFormsApp1.Services;

namespace WindowsFormsApp1

{

public partial class FormNoCard : Form

{

public FormNoCard()

{

InitializeComponent();

}

private void textPASS\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!char.IsDigit(number)&& number != 8)

e.Handled = true;//перевіряемо чи був оброблений ввод с клавіатури

}

private void button4\_Click(object sender, EventArgs e)

{

FormStart f1 = new FormStart();

f1.Show();

Hide();

}

private void button8\_Click(object sender, EventArgs e)

{

string phonenum = "+38" + new string(

(from s in maskedTextBoxPhone.Text.ToCharArray()

where !("()- ".Contains(s)) // Вибираємо всі символу, окрім (, ), - і пробілу

select s).ToArray()); // ToArray - щоб зліпити в кучу

// MessageBox.Show(phonenum);

try

{

FormsService.ChosenCardNumber = CardService.Instance.GetNumberByPhone(phonenum);

FormNoCardConfirm f5 = new FormNoCardConfirm();

f5.Show();

this.Visible = false;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Помилка при пошуку за телефоном");

}

}

private void maskedTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!char.IsDigit(number)&& number != 8)

e.Handled = true;//перевіряемо чи був оброблений ввод с клавіатури

}

}

}

**Form5**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class FormNoCardConfirm : Form

{

private Random rnd;

public FormNoCardConfirm()

{

rnd = new Random((int)DateTime.Now.Ticks);

InitializeComponent();

}

private void FormNoCardConfirm\_Load(object sender, EventArgs e)

{

codeBox.Text = GetNDigitRandNumber(6);

}

private string GetNDigitRandNumber(int n)

{

if(n == 0) return "";

return rnd.Next(0, 10).ToString() + GetNDigitRandNumber(n-1);

}

private void button4\_Click(object sender, EventArgs e)

{

FormNoCard f1 = new FormNoCard();

f1.Show();

Hide();

}

private void button8\_Click(object sender, EventArgs e)

{

if (textPASS.Text != codeBox.Text)

{

MessageBox.Show("Неправильний код для підтвердження, повторити спробу?");

codeBox.Text = GetNDigitRandNumber(6);

textPASS.Text = "";

return;

}

FormCheckout f1 = new FormCheckout();

this.Visible = false;

f1.Show();

f1.Focus();

}

private void textPASS\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

if (!char.IsDigit(number)&& number != 8)

e.Handled = true;//перевіряемо чи був оброблений ввод с клавіатури

}

}

}

**Form6**

using System;

using System.Drawing;

using System.Timers;

using System.Windows.Forms;

using Timer = System.Timers.Timer;

namespace WindowsFormsApp1

{

public partial class Form6 : Form

{

public Form6()

{

InitializeComponent();

}

private Point AnimationEndPoint;

private int animationLen = 150, step = 2;

Timer animationTimer;

private void Form6\_Load(object sender, EventArgs e)

{

animationTimer = new Timer(100);

animationTimer.SynchronizingObject = this;

animationTimer.Elapsed += AnimationTick; // функция, которая срабатывает каждые 100мс

animationTimer.Start();

// Указываем где должна будет остановиться панелька. По Х так же, а по У будет больше(ниже)

AnimationEndPoint = new Point(panelMoney.Location.X, panelMoney.Location.Y + animationLen);

}

private void AnimationTick(object sender, ElapsedEventArgs e)

{

var prevPoint = panelMoney.Location;

panelMoney.Location = new Point(prevPoint.X, prevPoint.Y + step); // Перемещаем панель с деньгами на {step} ниже

if(panelMoney.Location.Y >= AnimationEndPoint.Y) { // Если по У ушло ниже/на уровне конечной позиции - останов очка

(sender as Timer).Stop();

switch (MessageBox.Show("Ваш сеанс закінчено, бажаєте здійснити ще одну операцію?", "Сеанс закінчено",

MessageBoxButtons.YesNo, MessageBoxIcon.Information, MessageBoxDefaultButton.Button2, MessageBoxOptions.DefaultDesktopOnly,

false)) // MessageBox на выходе возвращает enum DialogResult, кнопочку, которую тыкнул пользователь

{ // Соответственно, если он нажал да, хочу ещё снять деньги - то переадресовываем его на старт

case DialogResult.Yes:

FormStart formStart = new FormStart();

this.Visible = false;

formStart.Show();

formStart.Focus();

break;

default: // Ну а если он нажал нет, или закрыл окно - то выходим из программы и усё

Application.Exit();

break;

}

}

}

}

}

**FormsApp1.Services**

WindowsFormsApp1.Services

using System.Windows.Forms;

namespace WindowsFormsApp1.Services {

public static class FormsService {

public static string ChosenCardNumber { get; set; }

}

}

**CardService**

using System;

using System.Collections.Generic;

using WindowsFormsApp1.Models;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using Newtonsoft.Json;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApp1.Services {

class CardService {

private List<Card> \_cards;

private static CardService cardService;

private CardService() {

\_cards = new List<Card>();

}

public static CardService Instance { // Сінглтон для того, щоб можна було працювати з одним об'єктом впродовж всієї програми

get {

return cardService ?? (cardService = new CardService());

}

}

public void LoadCards(string filePath) {

if (File.Exists(filePath)) {

string jsonData = File.ReadAllText(filePath);

\_cards = JsonConvert.DeserializeObject<List<Card>>(jsonData);

}

}

public void SaveCards(string filePath)

{

string jsonDataText = JsonConvert.SerializeObject(\_cards, Formatting.Indented); // Formatting.Indented для гарного вигляду json, а не в строчку

File.WriteAllText(filePath, jsonDataText);

}

public void LocateCard(int index, Rectangle rect) {

Card card;

card = \_cards[index];

if (card.cardBox == null)

card.cardBox = new PictureBox();

card.cardBox.Location = rect.Location;

card.cardBox.Size = rect.Size;

}

public void LoadCardImageFromFormatPath(int index, string formatPath) {

Card card = \_cards[index];

PictureBox pict = card.cardBox;

pict.ImageLocation = formatPath.Replace("\*", card.CardType.ToString());

}

public PictureBox GetPicture(int index) {

PictureBox cardImg = \_cards[index].cardBox;

return cardImg;

}

public Card GetCardByPicture(PictureBox box) {

foreach (Card card in \_cards)

if (box == card.cardBox)

return card;

return null;

}

public bool CheckPassword(string cardNumber, string password) {

foreach (var card in \_cards)

if (card.CardNumber == cardNumber)

return card.Password == password;

return false;

}

public float GetBalanceByCardNumber(string cardNumber) {

foreach (var card in \_cards)

if (card.CardNumber == cardNumber)

return card.Balance;

return -1;

}

public string GetNumberByPhone(string phone)

{

foreach (var card in \_cards)

if (card.PhoneNumber == phone)

return card.CardNumber;

throw new ArgumentException("Номер телефону не зареєстрований!");

}

public void WithdrawFromCard(string cardNumber, float amount) {

foreach (var card in \_cards)

if (card.CardNumber == cardNumber) {

card.Withdraw(amount);

return;

}

return;

}

public int Count() => \_cards.Count;

}

}