Міністерство освіти і науки України

Національний технічний університет

«Харківський політехнічний інститут»

Навчально-науковий інститут комп'ютерних наук та інформаційних технологій

Кафедра системного аналізу та інформаційно-аналітичних технологій

(САІТ)

Звіт

про виконання лабораторної роботи №1

«Основи розробки додатків у .NET»

за дисципліною

«Стек технологій .NET»

Виконав

студент групи КН-320Д Євген КУЛДОШИН

Перевірив

доцент кафедри СУ Олександр ЛИСЕНКО

Харків 2023

1. **Мета**

Навчитися створювати консольні та віконні додатки, ознайомтись з використання делегатів та подій у мові програмування C#.

1. **Завдання**

1. Опрацювати теорію (див. додаток А)

2. Написати програму для моделювання роботи банкомату.

Рішення має складатися з трьох проектів:

* бібліотека класів;
* віконний проект;
* консольний проект.

У бібліотеці класів реалізуйте загальні типи даних, які будуть використані у віконному та консольному проектах. У віконному додатку розробіть інтерфейс, який наближений до інтерфейсу традиційного банкомату. Консольний додаток повинен забезпечити ті ж самі можливості, що має віконний додаток, але за допомогою консольного меню.

Слід передбачити наступні можливості:

* аутентифікація (перевірка введення номеру картки та пін-коду);
* перегляд балансу на картці;
* перегляд історіі операцій з фільтруванням по проміжку часу (за поточний день, за поточний тіждень, за поточний місяц)
* зняття коштів;
* зарахування коштів на картку;
* перерахування коштів на картку із заданим номером;
* перегляд найближчих банкоматів;

Повинна бути передбачена можливість додавання власних обробників подій на виконання операцій аутентифікації, перегляду балансу, зняття коштів та перерахування коштів на іншу картку.

Додайте власні обробники подій, які будуть виводити повідомлення:

* для віконного додатку - за допомогою виклику:

MessageBox.Show("Повідомлення");

* для консольного додатку - за допомогою виклику

Console.WriteLine("Повідомлення");

При виконанні завдання створіть класи, які забезпечать роботу додатку:

* **Account** - клас, який є поданням карткового рахунку клієнта (властивості: номер

картки, прізвище та ім’я власника, баланс на рахунку, тощо);

* **AutomatedTellerMachine** - клас, який моделює роботу банкомата (містить

кількість грошей, які містяться у банкоматі, ідентифікатор банкомату, його адресу розташування, тощо);

* **Bank** - клас, що представляє банк (назва банку, перелік банкоматів, тощо).

Це не виключний перелік класів, за необхідності створюйте інші класи, які Вам будуть необхідні при реалізації завдання.

1. **ТЕОРІЯ**

Делегати, події та лямбди:

**Делегати** - це типи даних, які зберігають посилання на методи. Делегати можна використовувати для асоціації обробки подій з об'єктами або для передачі методів як параметрів.

**Події** - це механізм, який дозволяє об'єктам повідомляти один одному про певні події, такі як зміна значення або завершення завдання.

**Лямбда-вирази** - це короткі, однорядкові методи, які можна використовувати в якості делегатів. Лямбда-вирази часто використовуються для асоціації обробки подій з об'єктами.

* Делегати - це посилання на методи.
* Події - це механізм сповіщення про зміни.
* Лямбда-вирази - це короткі методи.

**Делегати та події**

Делегати та події часто використовуються разом. Делегати використовуються для асоціації обробки подій з об'єктами. Коли відбувається подія, делегат викликає метод, який обробляє подію.

1. **ВИКОНАННЯ**

Перше, що нас зустрічає в програмі – це привітання і вводимо данні (карта і пін-код) показано на рис. 4.1.

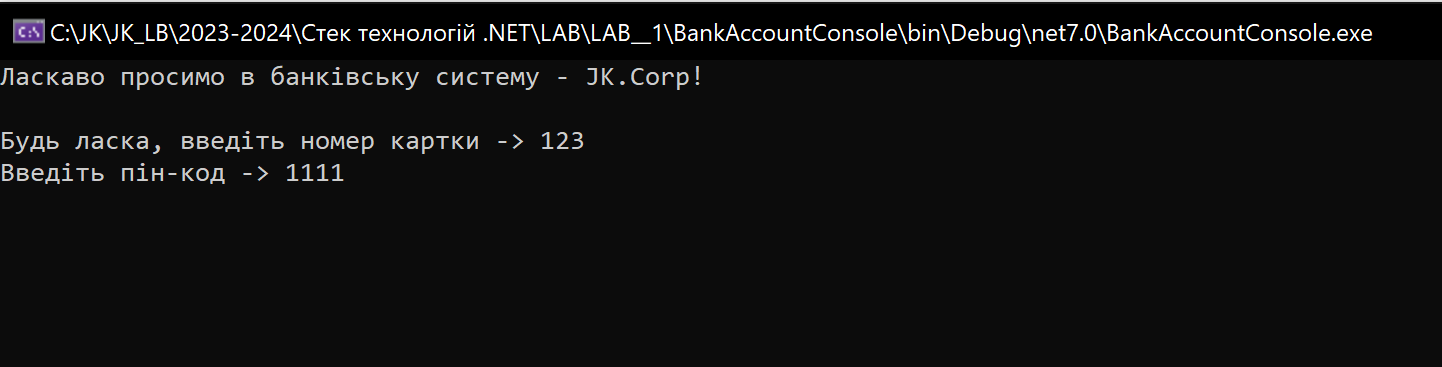


Рисунок 4.1 – Початкова сторінка

При вірно введених даних нас перекидає на меню нашого застосунку, показано на рис. 4.2.

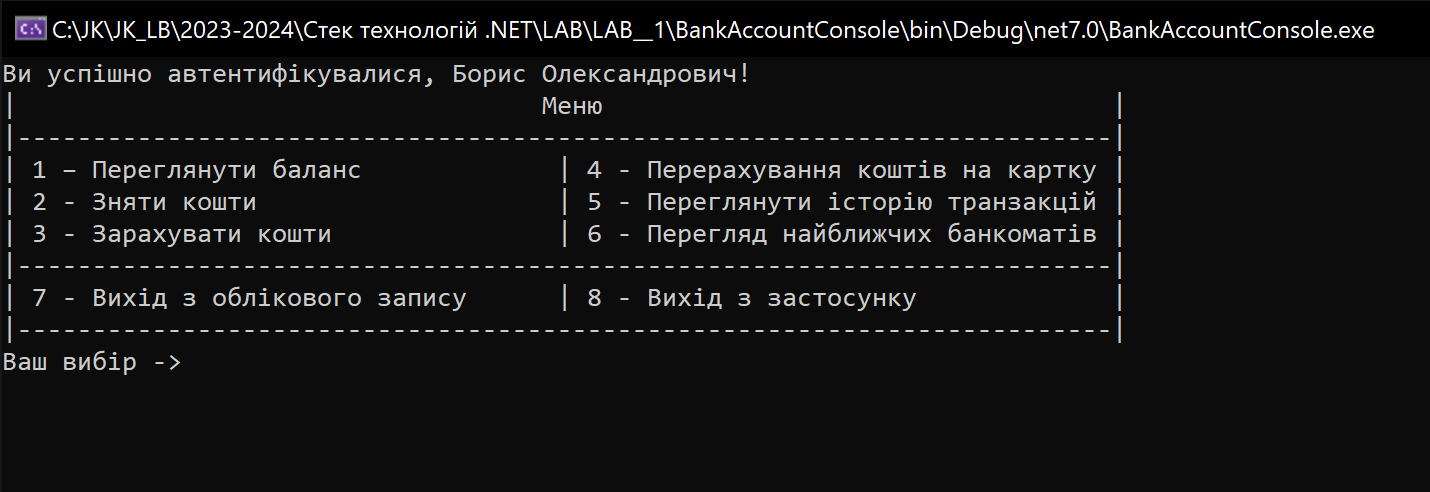


Рисунок 4.2 – Головне меню консольного застосунку

Вибираємо перший пункт, очищається консоль і виводить баланс нашого користувача, показано на рис. 4.3.

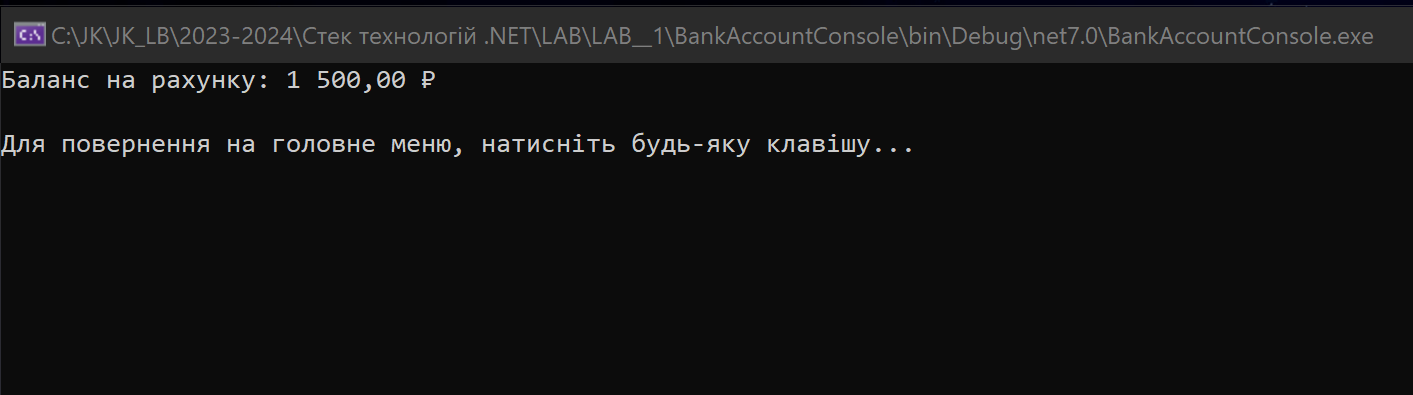


Рисунок 4.3 – Баланс користувача

Вибираємо другий пункт, очищається консоль і просить нас вести суму для зняття, показано на рис. 4.4.

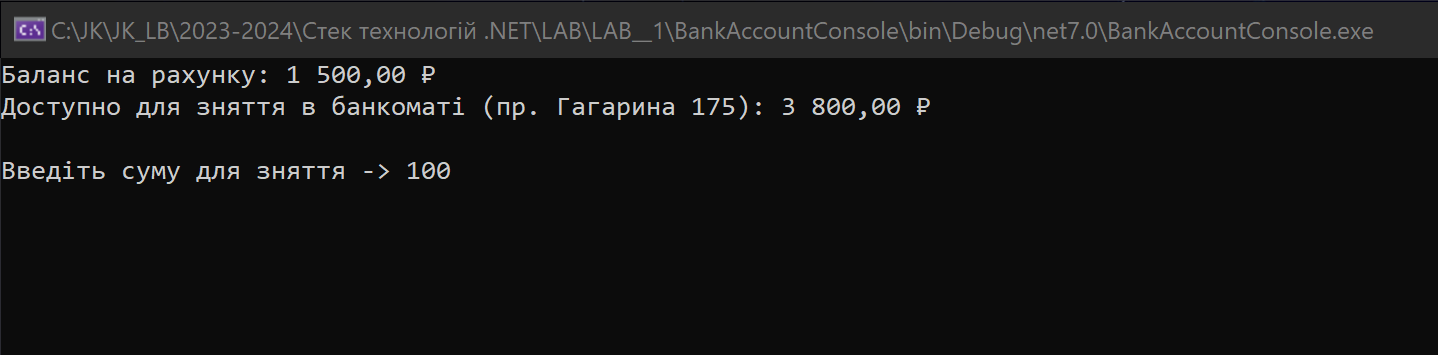


Рисунок 4.4 – Знімаємо гроші з банкомату

При успішному виконанню дії, виконується подія:

foreach (var account in bank.Accounts)

{

account.TransactionPerformed += transaction =>

{

Console.WriteLine($"Транзакція: {transaction.Type}, Сума: {transaction.Amount:C}");

};

}

І буде виведено інформацію про успішну операцію, показано на рис. 4.5.

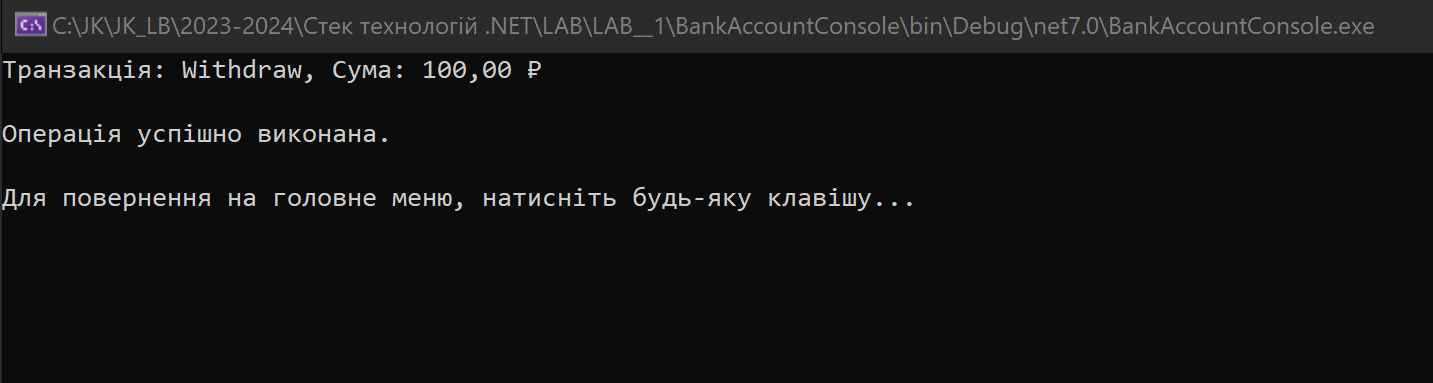


Рисунок 4.5 – Успішне зняття коштів з банкомату

Вибираємо третій пункт, очищається консоль і просить нас вести суму для додавання коштів на рахунок, показано на рис. 4.6.

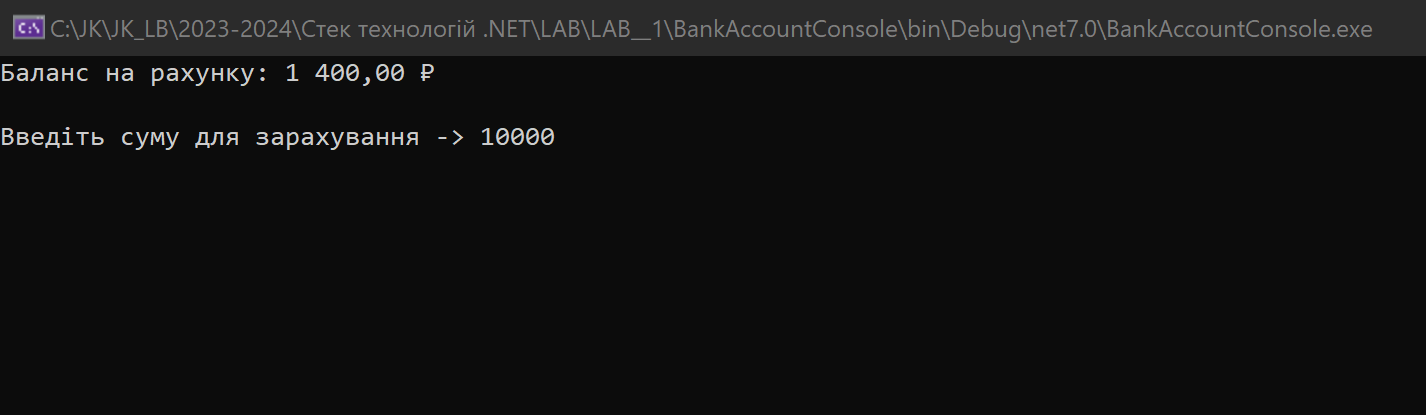


Рисунок 4.6 – Вводимо суму для зарахування на рахунок

І виводимо інформацію через подію про успішне виконання зарахування коштів на рахунок, показано на рис. 4.7.

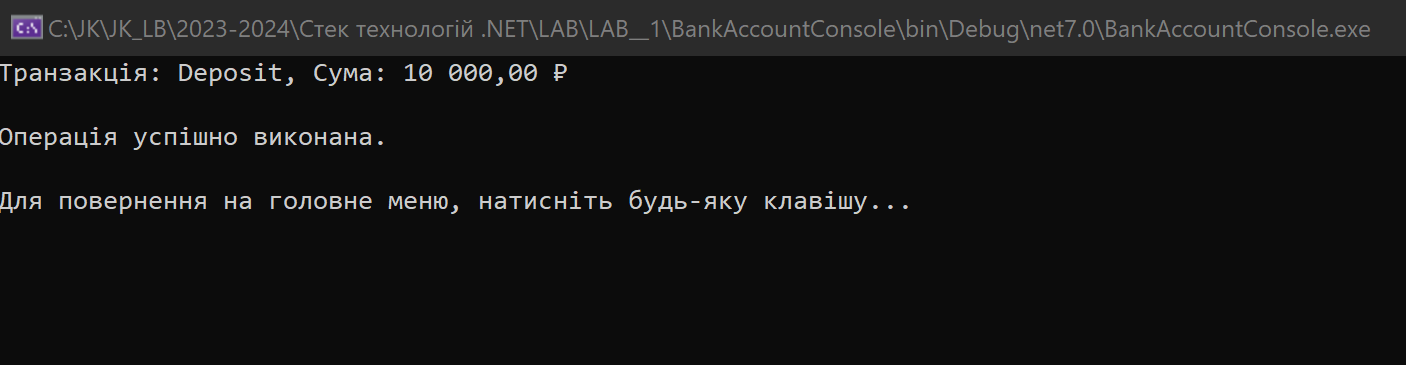


Рисунок 4.7 – Успішне зняття коштів з банкомату

Вибираємо четвертий пункт, очищається консоль і просить нас вести номер картки для перерахування, а надалі ми повинні ввести суму для зарахування, показано на рис. 4.8.

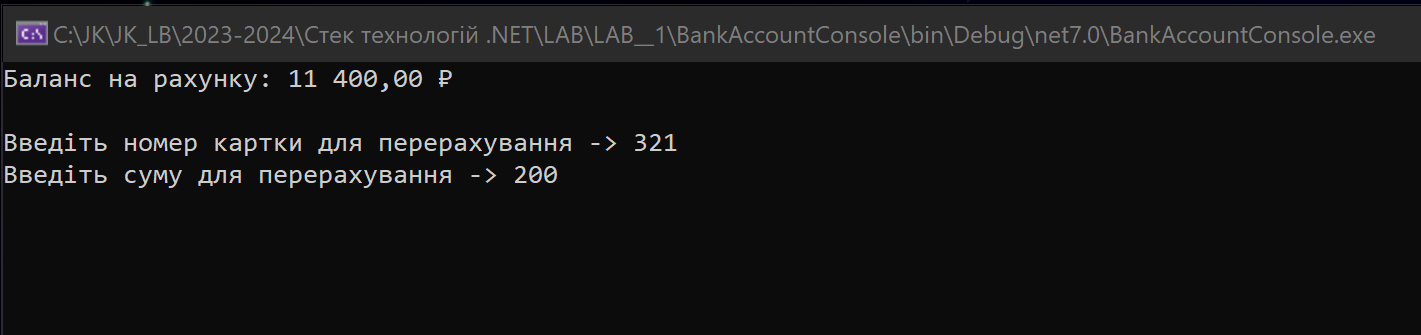


Рисунок 4.8 – Перерахування коштів на іншу карту.

І виводимо інформацію через подію про успішне виконання перерахування коштів на рахунок, показано на рис. 4.9.

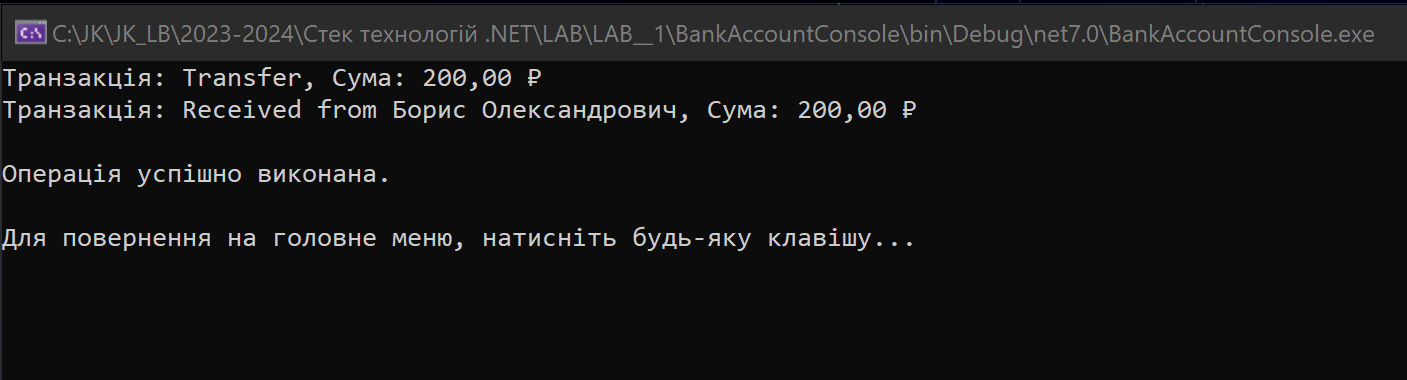


Рисунок 4.9 – Успішне перерахування коштів на картку

Вибираємо п’ятий пункт, для того щоб переглянути історію транзакцій, вибираємо (за день, за тиждень, за місяць або всі транзакції), показано на рис. 4.10.

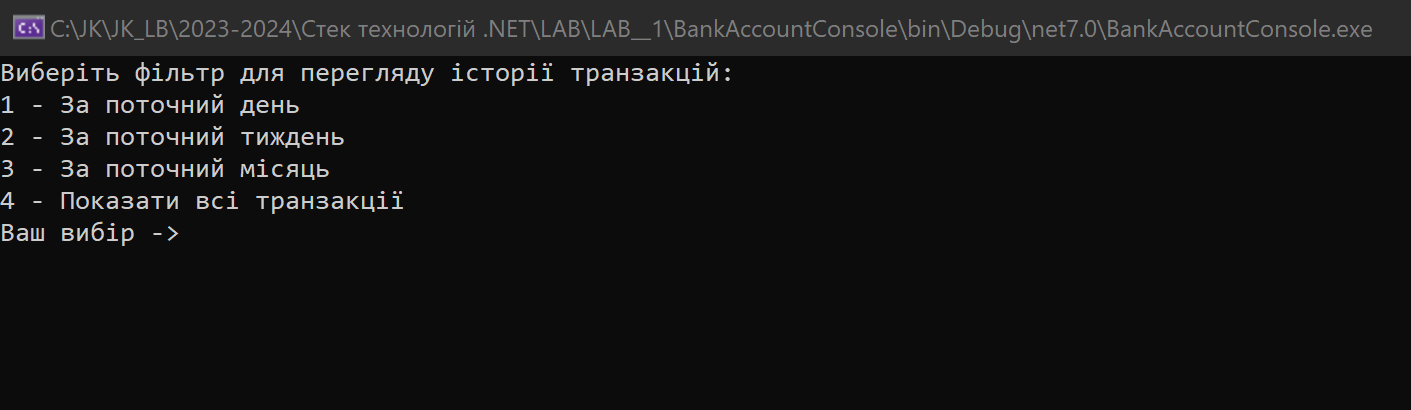


Рисунок 4.10 – Меню для перегляду історій транзакцій

Для прикладу вибрав за поточний день, показано на рис. 4.11.

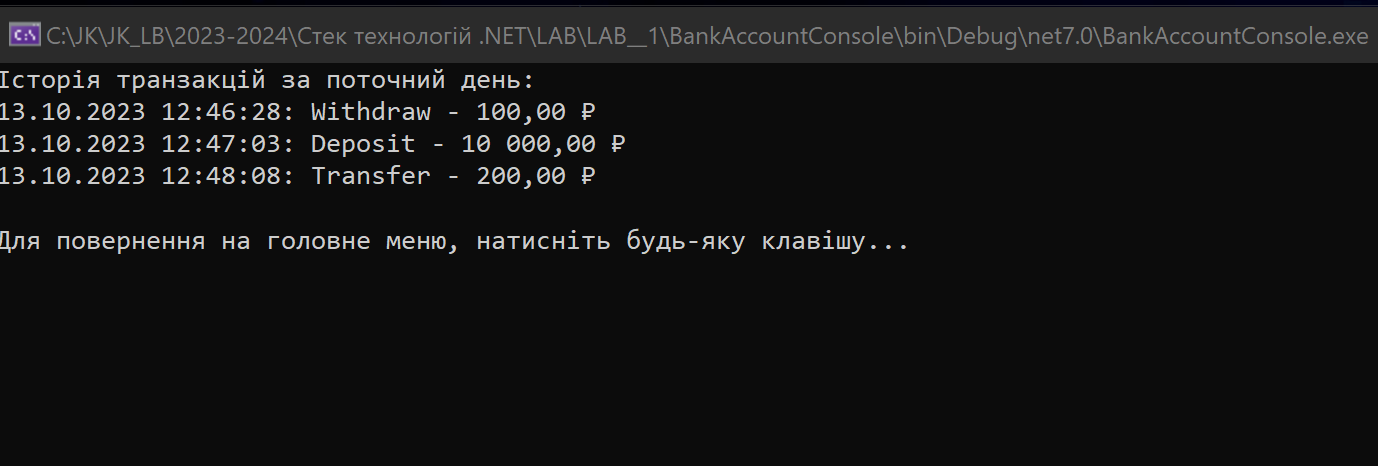
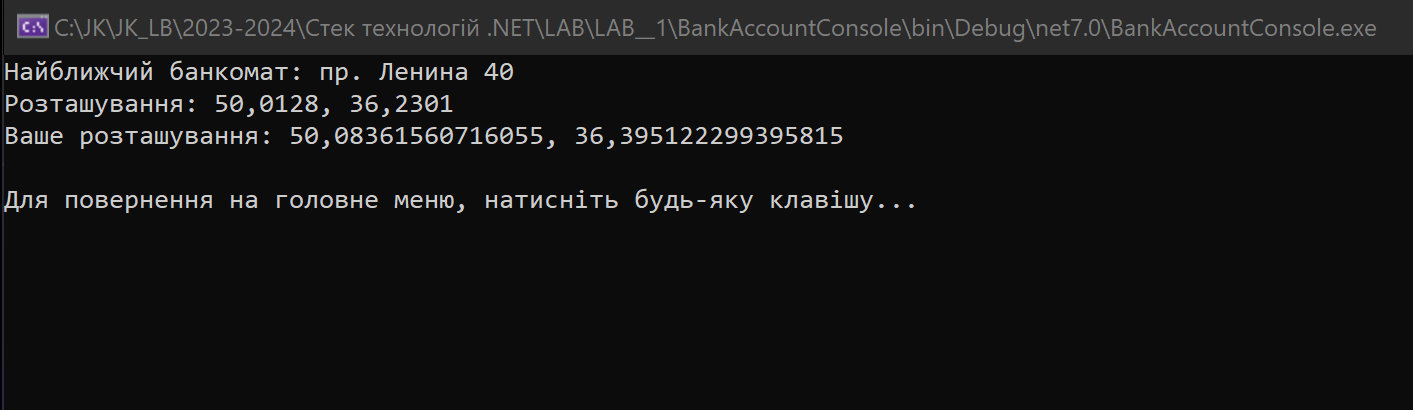


Рисунок 4.11 – Історія транзакцій за поточний день

Вибираємо шостий пункт, для того щоб переглянути найближчий банкомат, виводимо інформацію про (адресу банкомату, розташування в координатах, і координати користувача), показано на рис. 4.12.

Рисунок 4.12 – Найближчий банкомат

Наступним етапом нашої робити, є створення застосунку за допомогою Windows Forms. Нас зустрічає форма для авторизації, вводимо номер картки, пароль, показано на рис. 4.13.

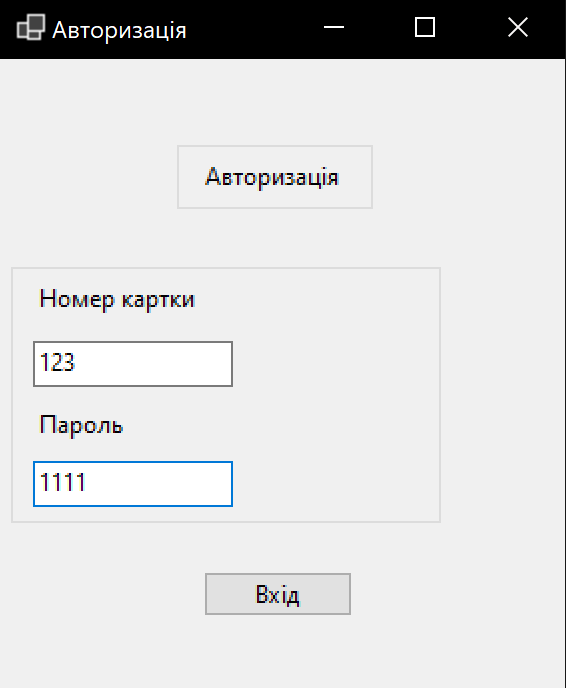


Рисунок 4.13 – Форма авторизації

Привітання у разі успішного входу до нашого аккаунту, показано на рис. 4.14.

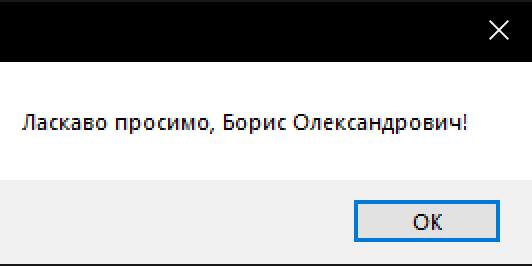


Рисунок 4.14 – Форма авторизації

Перекидає нас на головну сторінку нашого застосунку, показано на рис. 4.15. Для зняття коштів з рахунку, вводимо в поле суму, уразі якщо коштів не достатньо на рахунку або в банкоматі, буде виведено помилку. У разі успіху, буде виведено messagebox, показано на рис. 4.16. Для додавання коштів на рахунок, необхідно ввести суму в поле і натиснути кнопку для додавання і нас оповістить через messagebox як показано на рис. 4.17. Для того щоб перевести кошти на інший рахунок, необхідно ввести номер карти іншого, існуючого користувача і суму для переводу, але таку суму яка не буду перевищувати баланс поточного користувача. При успішно введених даних, нам через messagebox буде виведено як показано на рис. 4.18.

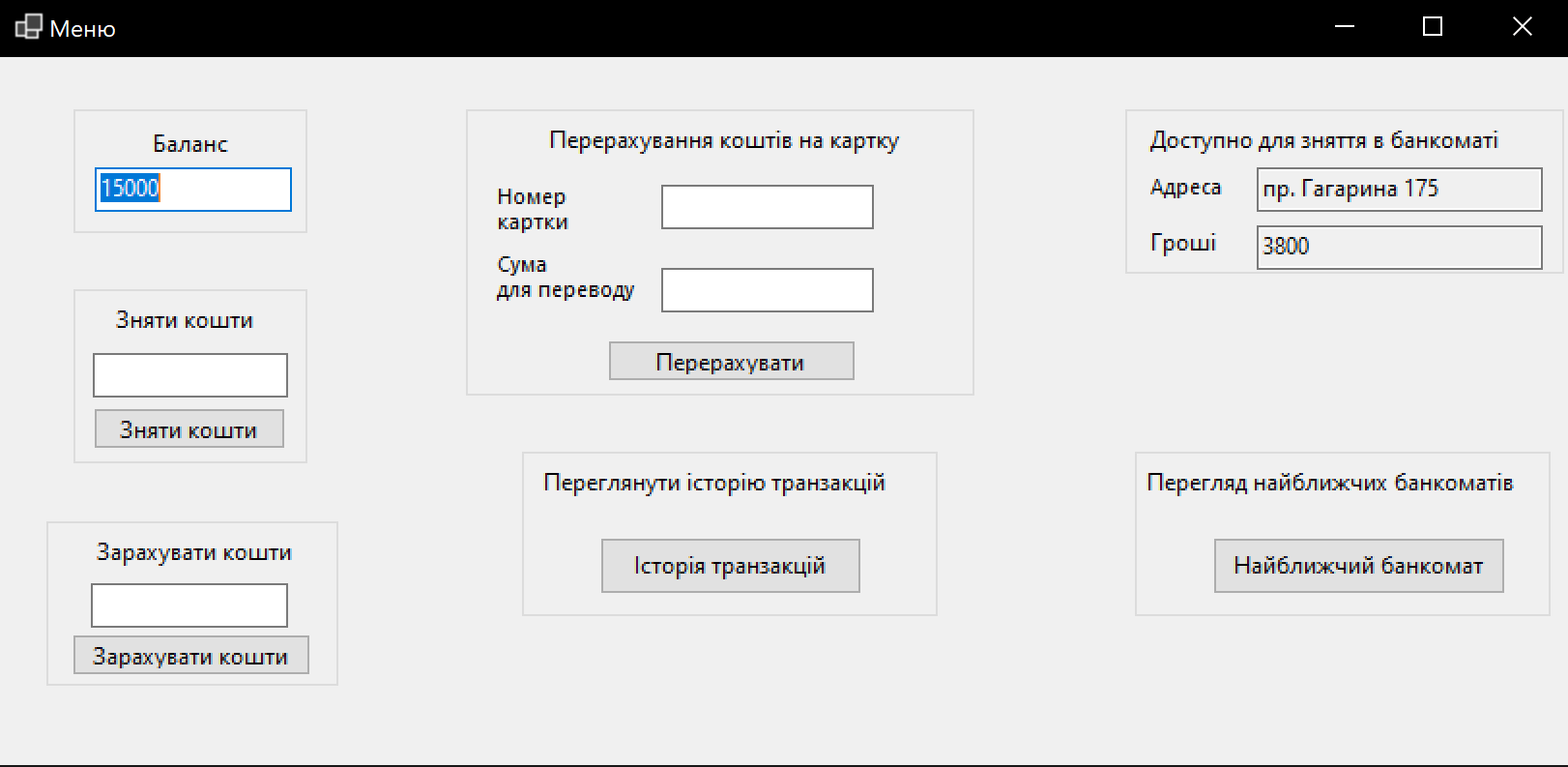


Рисунок 4.15 – Головна форма застосунку

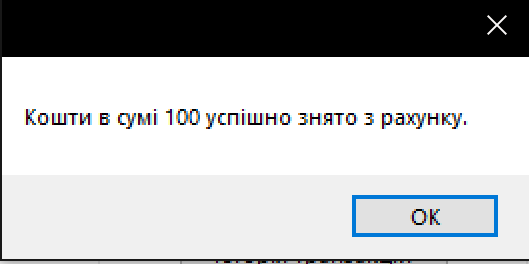


Рисунок 4.16 – Оповіщення про успішне зняття

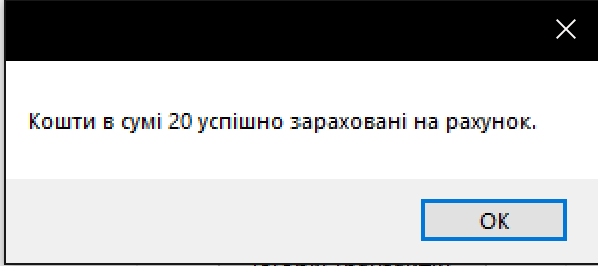


Рисунок 4.17 – Оповіщення про успішне зарахування

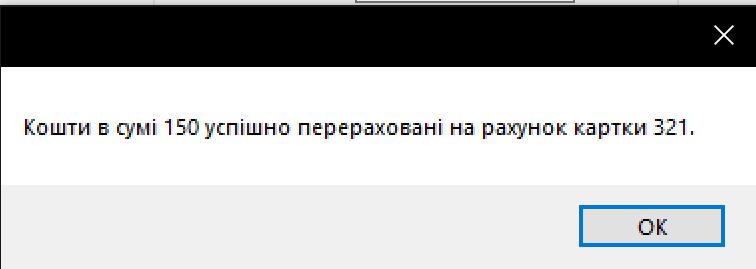


Рисунок 4.18 – Оповіщення про переведення коштів на інший рахунок

Наступна функція, перегляду історії транзакцій користувача, показано на рис. 4.19. Де ми можемо сортувати по дня, місяцям, і за весь період часу.

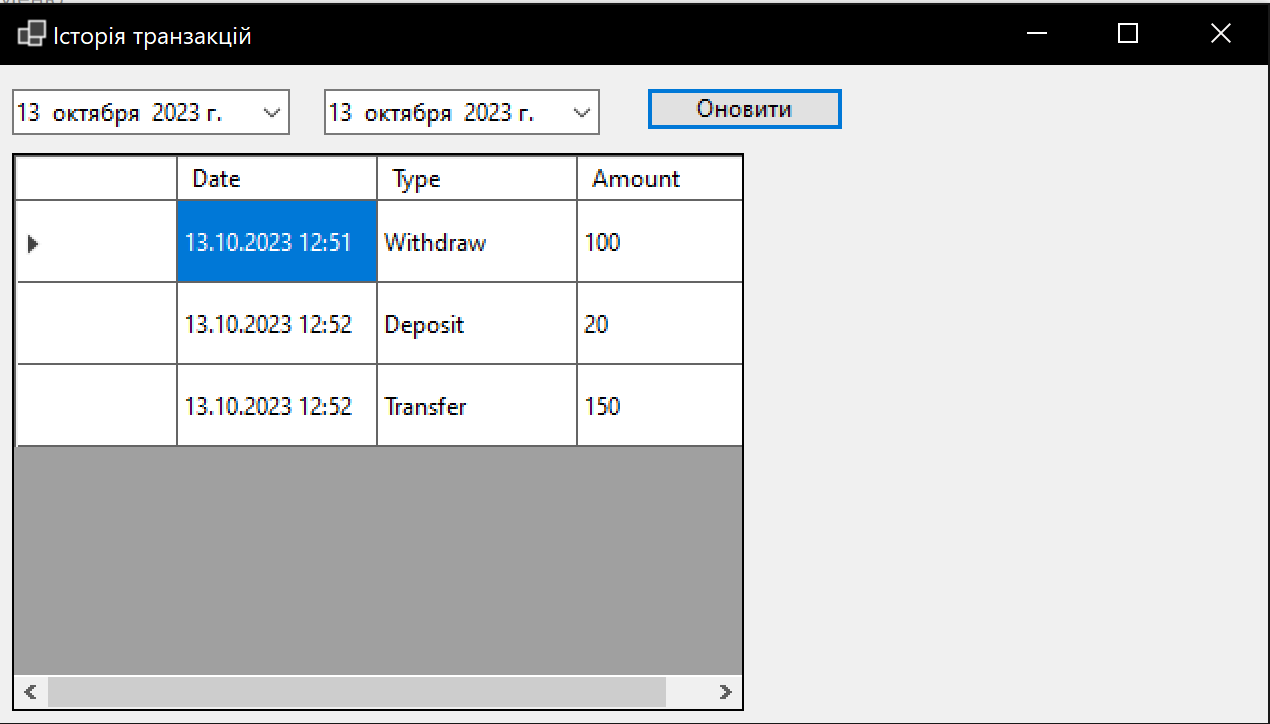


Рисунок 4.19 – Історія транзакцій

Остання функція, перегляд найближчих банкоматів до користувача, показано на рис. 4.20. Де ми можемо сортувати по дня, місяцям, і за весь період часу.

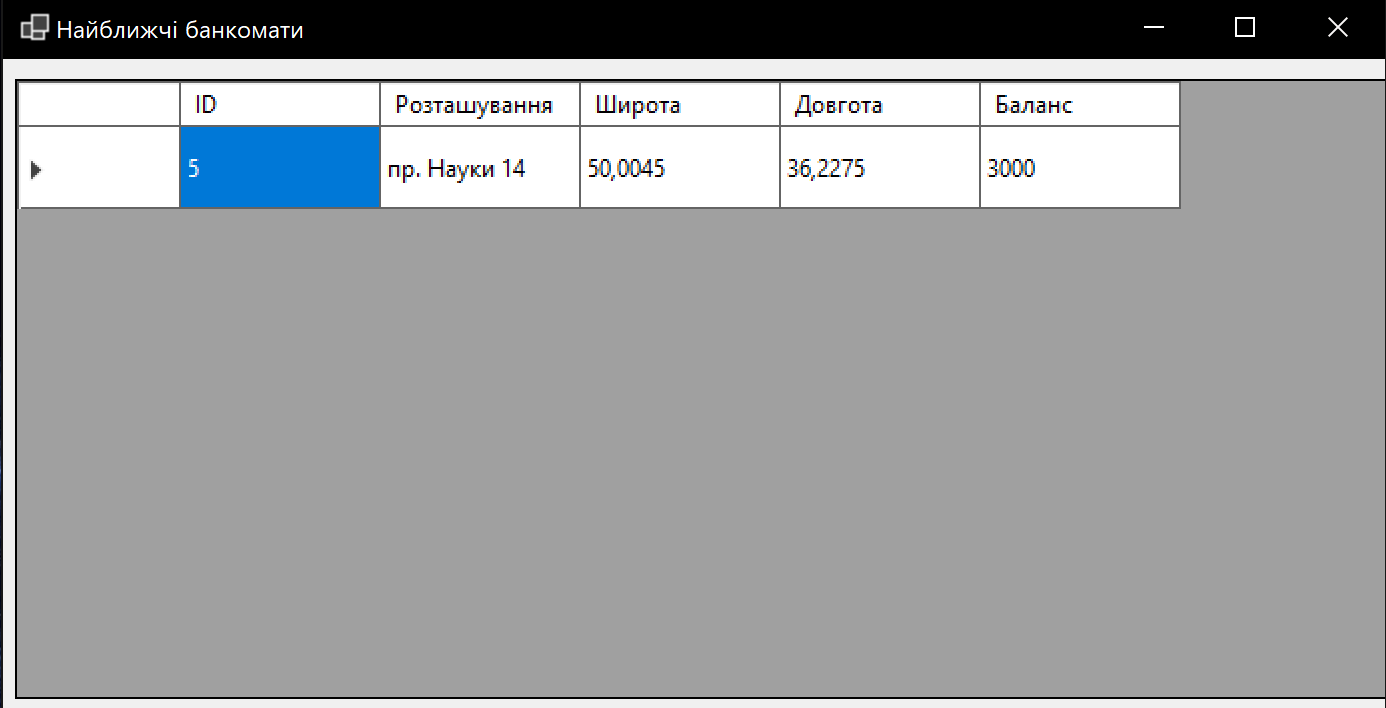


Рисунок 4.20 – Найближчі банкомати

1. **ВИСНОВОК**

У висновку можна підкреслити, що використання делегатів, подій та лямбда-виразів у програмуванні забезпечує ефективний та гнучкий спосіб управління подіями та обробкою даних. Делегати, як посилання на методи, дозволяють асоціювати обробку подій з конкретними об'єктами. Події надають можливість об'єктам сповіщати інші частини програми про зміни, дозволяючи реагувати на події у відповідний спосіб. Лямбда-вирази надають зручний і компактний спосіб визначення коротких методів, що можуть використовуватися як делегати для обробки подій. Застосування цих концепцій у поєднанні забезпечує зрозумілість, чистоту та гнучкість у розробці програмного забезпечення.