МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Обчислювальна техніка та програмування»

ПОЯСНЮВАЛЬНА ЗАПИСКА

до курсового проекту

на тему РОЗРОБКА МОДЕЛЮВАННЯ БАНКОМАТУ «MYBANKATM»

Керівник проекту

\_\_\_\_\_\_\_\_\_ Молчанов Г.І.

Виконавець

студент групи КІТ–26В

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лаврєнов Д.В.

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 р.

Харків 2018

РЕФЕРАТ

Пояснювальна записка до курсового проекту: 23 с., 7 рис., 6 джерел інформації.

Ключові слова: рахунок, платформа, NetBeans IDE.

У даному документі описуються особливості розробки програма, що дозволяє керувати власними коштами, забезпечує доступ до рахунку за ПІН-кодом, дізнаватися баланс коштів на картці, поповняти чи знімати кошти з рахунку.

Програму створено з використанням платформи Netbeans «MyBankAtm» інтегрованого середовища розробки (IDE) для платформи – NetBeans IDE 8.0.2.

ABSTRACT

Explanatory note to the course project: 23 p., 7 fig., 6 notes of informatio. Key words: account, platform, NetBeans IDE.

This document describes the development features of a program that allows you to manage your own funds, provides access to the account by the PIN-code, find out the balance of funds on the card, replenish or withdraw funds from the account.

The program was created using NetBeans's "MyBankAtm" Integrated Development Environment (IDE) platform platform - NetBeans IDE 8.0.2.

# 2

ЗМІСТ

ВСТУП 4

1. ОГЛЯД АНАЛОГІВ ТА ПОСТАНОВКА ЗАДАЧІ 5
   1. Огляд аналогічних систем 5
   2. Постановка задачі 6
   3. Вибір мови програмування… 7
   4. Вибір середовища розробки 8
2. [РОЗРОБКА ПРОГРАМИ 10](#_TOC_250001)
   1. Структура програми 10
   2. Алгоритм відображення інформації про клієнта 11
   3. Розробка класів ……………………………………………………………...11
   4. Опис GUI компонента BorderLayout ……………………………………….17
   5. Опис GUI компонента jPanel ……………………………………………….17
3. [ОПИС РОБОТИ ПРОГРАМИ](#_TOC_250000) …………………………………………………20
   1. Структура проекту «MyBankAtm» ………………………………………….20
   2. Вигляд головного меню програми 20
   3. Сторінка відображення даних про клієнта та виконання основних операції

……………………...…………………………………………………………..... 21

ВИСНОВКИ 22

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ 23

ВСТУП

В даний час широке застосування отримали банкомати. Майже на всіх підприємствах працівники отримують заробітну плату на банківський рахунок, до якого прив'язана дебетова карта. Це дозволяє підприємству економити кошти на виплати заробітних плат, так як готівкові гроші вимагають витрат на обслуговування. Безготівковий переказ в свою чергу здійснюється практично без витрат зусиль і часу.

Однак в силу швидких темпів технічного прогресу далеко не всі вміють правильно користуватися банкоматами для отримання готівки. Особливо великий цей відсоток серед людей старшого покоління.

Метою даної курсової роботи є розробка програмного забезпечення, що моделює роботу банкомату. Це програмне забезпечення може служити корисним елементом навчання власників банківських карток їх правильного використання.

1. ОГЛЯД АНАЛОГІВ, ПОСТАНОВКА ЗАВДАННЯ
   1. Огляд аналогічних систем

На сьогоднішній день не багато програм, які програмно моделют роботу банкомату.

Такими програмами є: Банкомат, Балтійський банк. Усі ці програми мають схожий інтерфейс користувача: поле вводу PIN-коду (Personal Identification Number), переглядом поточного стану рахунку, зняття грошей з рахунку; поповнення рахунку.

На рис. 1.1 зображення інтерфейс програм.

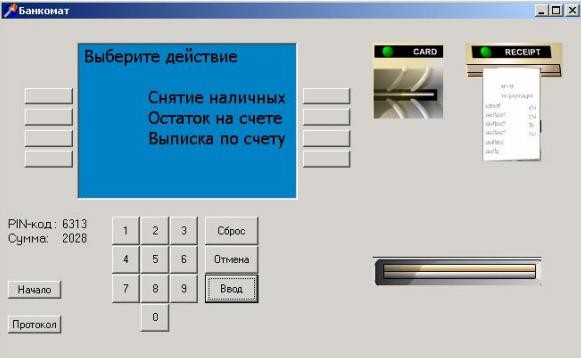
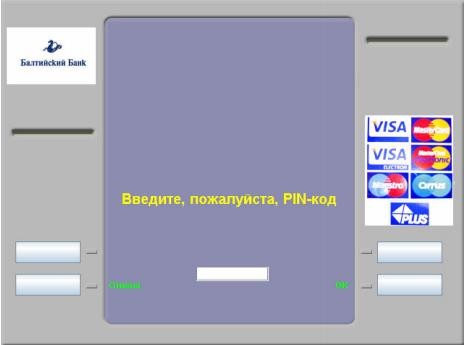
.

Рисунок 1.1. – Інтерфейси прикладів програм

Проте вони мають недолік застарілий інтерфейс та складний інтерфейс. Багатьом користувачам це дуже не подобається. 85% можуть закрити програму, якщо їм не сподобається дизайн інтерфейсу, 83% закроюють програму, якщо будуть змушені робити багато кликів, щоб знайти те, що їм потрібно, 40% ніколи не повернуться на сайт, якщо їм було важко його використовувати в перший раз.

В роботі необхідно розробити програму, яка проста и зручна у використанні, має дружелюбний і інтуїтивно зрозумілий інтерфейс.

* 1. Постановка задачі

Під банкоматом будемо розуміти автоматизований пристрій, що дозволяє здійснювати операції, пов'язані з

* + ідентифікацією користувача (власника рахунку в банку);
  + переглядом поточного стану рахунку;
  + зняттям грошей з рахунку;
  + поповнення рахунку зі банкомату.

З точки зору клієнта, взаємодія з емуляцією банкомату має виглядати так само, як і в реальності: він повинен мати можливість почати роботу з конкретною карткою, яка повинна бути ідентифікована пристроєм, а потім за допомогою меню здійснювати в будь-якій послідовності перераховані вище грошові операції. В реальному банкоматі ідентифікація карти відбувається шляхом автоматичного зчитування її номера. У даній роботі в якості цієї процедури буде використовуватися введення номера карти з клавіатури. Автентифікація, як правило, проводиться на основі введення клієнтом так званого PIN-коду (Personal Identification Number), який перевіряється сервером банку на відповідність вважає номером картки.

Меню має бути реалізовано на основі графічного інтерфейсу. Емуляція банкомату повинна буде дозволяти проводити всі перераховані вище операції з рахунком.

* 1. Вибір мови програмування

Для даного курсового проекту було обрану мову програмування Java тому, що Java об'єктно-орієнтована мова. Це дозволяє створювати модульні програми та багаторазовий код. Мова програмування Java не залежить від платформи. Один і той же код можна використовувати для різних операційних систем. Завдяки цьому джава-розробник пише і запускає джава- програму в будь-якому середовищі за допомогою JVM (Java Virtual Machine). Компанії програмного забезпечення Sun і Oracle доклали величезне зусилля, щоб удосконалювати Java ще більше. Код написаний на старій версії без проблем працює на оновленій. Адже немає нічого гіршого, ніж переписувати код, щоб він працював на новій версії. Марна трата часу для

розробника.

Безкоштовність. Важливий фактор для багатьох організацій. Адже коли компанія вибирає ту або іншу технологію, ціна грає не останню роль. Java - безкоштовна платформа. Коли компанія користується їй, їй потрібно лише накласти "джавіста" і заплатити за розробку додатків. Java проста в основному завдяки своїй екосистемі.

Ця мова з 1995 року вдосконалюється до того, який ми бачимо сьогодні: тонни хороших фреймворків і різноманітний набір стандартних бібліотек роблять Java простим у використанні.

Компілятор перевіряє код на відповідність правилам синтаксису мови, а потім записує байт-код у файли .class. Байт-код - це набір інструкцій, призначених для запуску на віртуальній машині Java (JVM). У додаванні цього рівня абстракції компілятор Java відрізняється від інших мовних компіляторів, які виписують інструкції, які підходять для набору мікросхем ЦП, на якому буде працювати програма.

Збирання сміття. Замість того, щоб змушувати не відставати від виділення пам'яті (або використовувати сторонню бібліотеку для цього), платформа Java забезпечує управління пам'яттю «з коробки».

Коли Java-додаток створює екземпляр об'єкта під час виконання, JVM автоматично виділяє простір пам'яті для цього об'єкта з купи - пул пам'яті, виділений для використання програмою. Складальник сміття Java працює у фоновому режимі, відстежуючи, які об'єкти більше не потрібні додатком, і звільняючи від них пам'ять. Такий підхід до обробки пам'яті називається неявним управлінням памятью, тому що це не вимагає від вас написання коду для обробки пам'яті. Прибирання сміття є однією з найважливіших характеристик продуктивності платформи Java.

Java Development Kit. Коли завантажуємо Java Development Kit (JDK), отримуємо - на додаток до компілятору і іншим інструментам - повну бібліотеку класів готових утиліт, яка допомагає в виконувати найпоширеніші завдання з розробки додатків. Кращий спосіб зрозуміти сферу застосування пакетів і бібліотек JDK - це ознайомитися з документацією по JDK API.

Java Runtime Environment. Середовище виконання Java (JRE; також відома як середовище виконання Java) включає в себе JVM, бібліотеки коду та компоненти, необхідні для запуску програм, написаних на мові Java. JRE доступна для декількох платформ. Можна вільно поширювати JRE разом зі своїми додатками відповідно до умов ліцензії JRE, щоб надати користувачам додатки платформу для запуску програмного забезпечення. JRE входить в JDK.

* 1. Вибір середовища розробки

Для даного курсового проекту було обрано інтегроване середовище розробки для платформи Netbeans – NetBeans IDE 8.0.2.

Платформа NetBeans є спільною платформою додатків Java для настільних комп'ютерів. Вона забезпечує "технічні роботи", який кожен розробник зазвичай повинен робити сам, - збереження стану, з'єднання дій з пунктами меню, елементи панелей інструментів, поєднання клавіш, управління вікнами і т.д.

Всі ці можливості вбудовані в платформу NetBeans. Тепер не потрібно вручну вводити код для тих чи інших основних функцій програми. Платформа NetBeans не займає великого обсягу в додатку, але допоможе заощадити величезну кількість часу і праці.

Кожна можливість в додатку на платформі NetBeans може забезпечуватися окремим модулем NetBeans, який можна порівняти з тим, що підключається модулем. Модуль NetBeans - це група класів Java, яка надає додаток з певними можливостями.

Користувачі також можуть створювати нові модулі для самого середовища IDE NetBeans. Наприклад, користувачі можуть створювати модулі, що забезпечують доступ до найбільш популярним новітніми технологіями для користувачів IDE NetBeans. Також можна створити, наприклад, модуль, який буде надавати функцію додаткового редактора.

Додавання нової функціональності в IDE наприклад, Junit, Tomcat, easyUML. Використання сторонніх технологій в якості частини середовища. Можлива публікація в репозиторії (на порталі) модулів для Netbeans

Модуль може працювати не тільки в складі IDE, але і як самостійний додаток Модуль - самостійний модуль, що не залежить від інших

Набір модулів - група залежних модулів

Оболонка - обертає функціональність сторонньої бібліотеки в модуль для подальшого імпорту цього модуля

# РОЗРОБКА ПРОГРАМИ

* 1. Структура програми

Програма повинна здійснювати операції, пов'язані з:

* ідентифікацією користувача (власника рахунку в банку);
* переглядом поточного стану рахунку;
* зняттям грошей з рахунку;
* поповнення рахунку

Структура програми представлена на рис.2.1

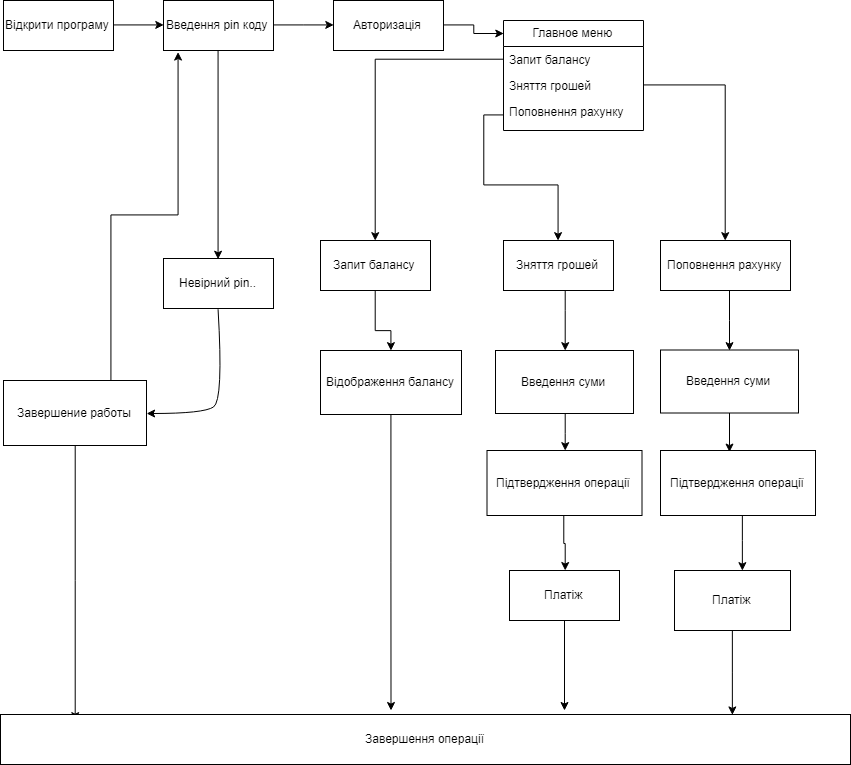


Рисунок 2.1 - Структура програми

* 1. Алгоритм відображення інформації про клієнта

Алгоритм відображення інформації про клієнта є простим алгоритмом, який є лінійним. У його реалізації використовуються методи і класи Java FX, а також класи створені для додаткової обробки даних, а саме класи Account, CheckingAccount, Customer.

Клас Account відповідає за ініціалізацію балансу.

Клас CheckingAccount відповідає за перевірку рахунку.

Алгоритм полягає в наступному:

* Вводим ідентифікатор користувача, який існує у нас в базі та натискаємо “Enter”
* виводиться ім'я і прізвище клієнта з відповідним ідентифікатором;
* вибираємо Сheck account balance;
* виводиться баланс клієнта
* вибираємо Make a deposit та обираємо кількість внеску на депозит ;
* виводитися новий баланс клієнта
* вибираємо Make a withdrawal та обираємо кількість зняття з депозиту ;
* виводитися новий баланс клієнта
  1. Розробка класів

Клас Account який буде відповідати за ініціалізацію балансу. Клас CheckingAccount який буде відповідати за перевірку рахунку.

Програма «MyBankAtm» повинна працювати з завантаження заставки, далі йде поява вікна з навігаційним меню поле для вводу, клавіатура (0-9), кнопки у яких працюють такі функції: Enter, “Make a deposit” та “Make a withdrawal”, панель відображення даних.

За відображення інформації про клієнта відповідає гілка коду, а саме:

historyArea.append("Balance of " + currentCustomer.getFirstName() + "'s first account is $" + currentAccount.getBalance());

if (currentAccount instanceof CheckingAccount) {

historyArea.append(". This is a Checking Account with overdraft protection $" + ((CheckingAccount) currentAccount).getOverdraftAmount() + "\n");

} else {

historyArea.append(". This is a Savings Account with interest rate " + ((SavingsAccount) currentAccount).getInterestRate() + "%\n");

}

statusField.setText("READY");

}

private void depositButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: try {

double amt = Double.parseDouble(amountField.getText()); currentAccount.deposit(amt);

historyArea.append("Deposit: $" + amt + ", new balance is $" + currentAccount.getBalance() + "\n"); statusField.setText("READY");

} catch (Exception e) {

historyArea.append("ERROR: can't complete deposit oiperation!\n"); statusField.setText("ERROR");

}

amountField.setText("");

}

private void withdrawButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: try {

double amt = Double.parseDouble(amountField.getText()); if (currentAccount.withdraw(amt)) {

historyArea.append("Deposit: $" + amt + ", new balance is $" + currentAccount.getBalance() + "\n"); statusField.setText("READY");

}

} catch (OverdraftException ex) { historyArea.append("ERROR: Insufficient funds!\n");

} catch (Exception e) {

historyArea.append("ERROR: can't complete deposit oiperation!\n"); statusField.setText("ERROR");

}

amountField.setText("");

}

private void enterButtonActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here: int customerID = 0;

try {

customerID = Integer.parseInt(amountField.getText()); currentCustomer = bank.getCustomer(customerID); currentAccount = currentCustomer.getAccount(0);

historyArea.append("Customer with ID = " + customerID + " is " + currentCustomer.getLastName() + ", " + currentCustomer.getFirstName() + "\n");

balanceButton.setEnabled(true); depositButton.setEnabled(true); withdrawButton.setEnabled(true); enterButton.setEnabled(false);

} catch (Exception ex) {

historyArea.append("ERROR: Customer not found or wrong Customer ID!\n");

}

amountField.setText("");

statusField.setText("Customer: "+currentCustomer.getLastName() + ", " + currentCustomer.getFirstName());

o Класс, що забезпечує початковий баланс

package com.mybank.domain; public class Account {

protected double balance;

/\*\*

* Constructor that provides initial balance

\*

* @param balance a positive account balance

\*/

protected Account(double balance) { if (balance >= 0) {

this.balance = balance;

} else {

this.balance = 0;

}

}

public Account() { this.balance = 0;

}

/\*\*

* Method to add money to account

\*

* @param amt a positive amount of money
* @return

\*/

public boolean deposit(double amt) { if (amt > 0) {

balance += amt; return true;

}

return false;

}

/\*\*

* Method to withdraw money

\*

* @param amt a positive amount of money

\*/

public boolean withdraw(double amt) throws Exception{ if (amt <= balance) {

balance -= amt; return true;

}

return false;

}

/\*\*

* Method to check account balance

\*

* @return the balance

\*/

public double getBalance() { return balance;

}

}

Клас для перевірки поточного рахунку

package com.mybank.domain;

public class CheckingAccount extends Account { private double overdraftAmount;

public CheckingAccount(double initBalance, double overdraftAmount) { this.balance=initBalance;

this.overdraftAmount=overdraftAmount;

}

public CheckingAccount(double initBalance) { this(initBalance, 0);

}

public boolean withdraw(double amt) throws OverdraftException{ if(amt<=balance+overdraftAmount){

balance=balance-amt; return true;

}

throw new OverdraftException(amt-balance-overdraftAmount, "Error! Insufficient funds");

}

public double getOverdraftAmount() { return overdraftAmount;

}

}

Клас для збереження даних про клієнта

package com.mybank.domain;

public class SavingsAccount extends Account { private double interestRate;

public SavingsAccount(double initBalance, double interestRate) { this.balance =initBalance;

this.interestRate=interestRate;

}

public void addInterestRate()

{

this.balance=this.balance\*(1+interestRate);

}

public double getInterestRate() {

return interestRate;

}

}

* 1. Опис GUI компонента BorderLayout

Ця процедура створена за допомогою JavaFx класу – Borderlayout. За замовчуванням в Swing використовується менеджер BorderLayout, в ньому визначені наступні константи для установки компонентів.

* BorderLayout.NORTH (верх)
* BorderLayout.SOUTH (низ)
* BorderLayout.EAST (праворуч)
* BorderLayout.WEST (зліва)
* BorderLayout.CENTER (заповнити середину до інших компонент або до країв)

За замовчуванням приймається константа Center.

* 1. Опис GUI компонента jPanel

Ця процедура створена за допомогою JavaFx класу – Jpanel.

В ньому визначені наступні установки компонентів.

GridLayout - об'єкт розміщує компоненти в сітці клітин. Кожен компонент займає весь доступний простір в своїй комірці, і кожна клітинка має точно такий же розмір.

JButton[BalanceButton, DepositButton,WithdrawButton] – кнопка представлена класом JButton; BalanceButton - перевіряє баланс; DepositButton - внесок на депозит; WithdrawButton – зняття з депозиту.

JTextField [AmountField] - дозволяє користувачам редагувати один рядок тексту.

Панель JPanel - це елемент управління, що являє собою прямокутне простір, на якому можна розміщувати інші елементи. Елементи додаються і видаляються методами, успадкованими від класу Container.

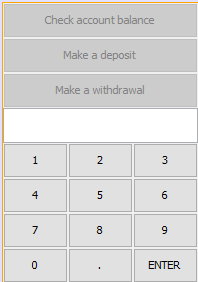
У нашому випадку даний компонент використовуваний в якості клавіатури банкомату, на рис 2.5 який наведений нижче

Рисунок 2.5 – Jpanel в якості клавіатури банкомату

# ОПИС РОБОТИ ПРОГРАМИ

* 1. Структура проекту «MyBankAtm»

Структура проекту «MyBankAtm» наведено на рис 3.1.

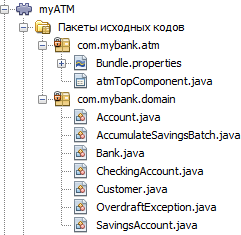


Рисунок 3.1 – Структура проекту

* 1. Вигляд головного меню програми наведено на рис. 3.2

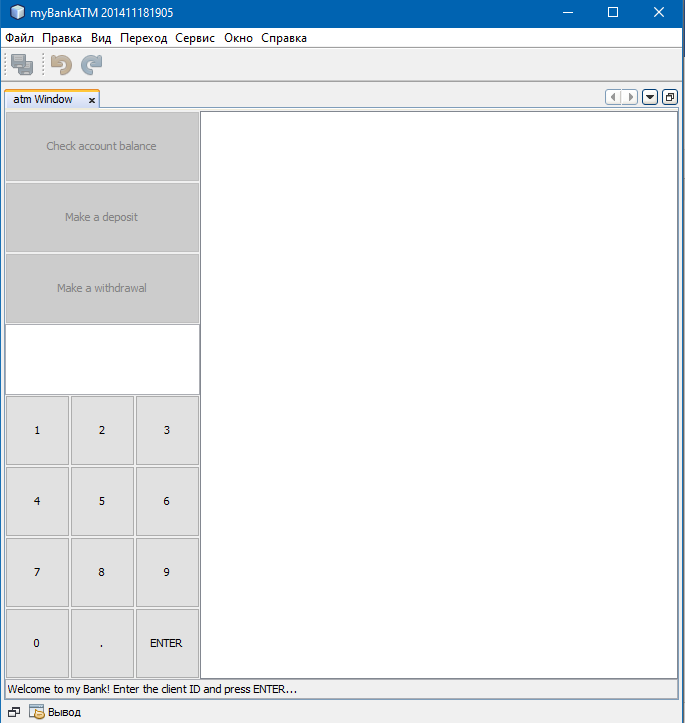


Рисунок 3.2 – Вигляд головного вікна програми

* 1. Сторінка відображення даних про клієнта та виконання основних операції (“Make a deposit ” та “Make a withdrawal”)

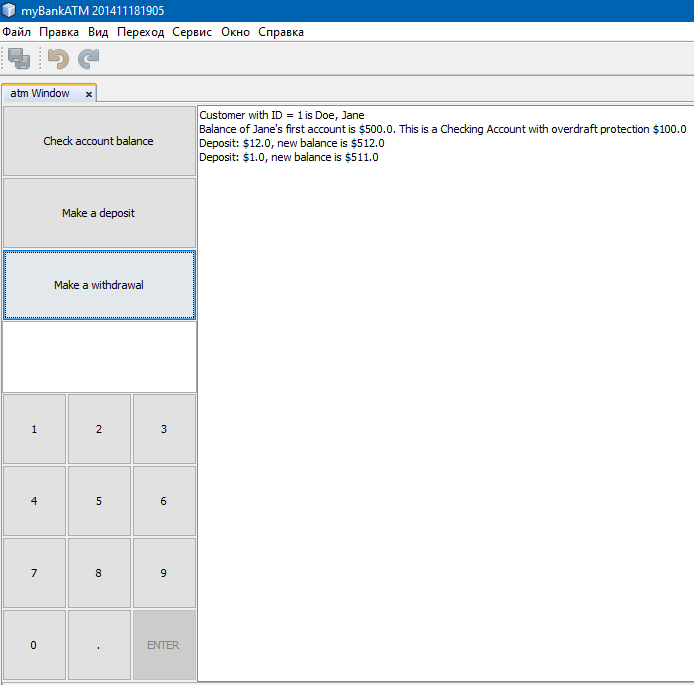


Рисунок 3.3 – Сторінка відображення даних про клієнта

ВИСНОВОК

В ході виконання роботи було проведено огляд програм аналогів, проаналізовано літературу, розроблено та створено програму, яка дозволяє здійснювати таки операції: ідентифікацію користувача, перегляд, поточного стану рахунку, зняттям грошей з рахунку, поповнення рахунку зі банкомату.

Програма проста і зручна у використанні, так як, має дружелюбний і інтуїтивно зрозумілий інтерфейс. Тестування ніяких недоліків не виявило. Всі поставлені завдання вирішені. Програма готова до використання.

Даний програмний продукт може бути використаний на платформах Windows 7/8/10, Linux та Mac OS.

В ході роботи були використані і вивчені наступні технології: JavaSE, JavaEE, JFC, JAI, JAXP.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

* + 1. Шилдт Г. Java 8. Полное руководство / - М.: Вильямс, 2015. - 720 c.
    2. Хабибуллин, И. Создание распределенных приложений на Java /- БХВ-Петербург, **2017**. - 704 c
    3. Хорстманн, Кей С. Java SE 8. Вводный курс / - М.: Диалектика / Вильямс, 2014. **- 898**c.
    4. Д. Гослинг, Б. Джой, Г. Стил, Г. Брача, А. Бакли. Язык программирования Java SE 8. Подробное описание / – М.: Вильямс. – 2015. – 672 c.
    5. Монахов В. В. Язык программирования Java и среда NetBeans. / – БХВ- СПб, 2011. –704 с.
    6. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приёмы объектно–ориентированного проектирования. Паттерны проектирования. Библиотека программиста. / – СПб.: Питер, 2010. – 366 с.