МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №4**

**Дисциплина: Платформо-независимое программирование**

Работу выполнил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чеуж А.А.

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. А. Приходько

Краснодар

2025

**Тема**: Библиотеки AWT и SWING для построения графического интерфейса пользователя. Обработка событий.

**Цель работы**: Освоить методы разработки с графическим интерфейсом Java, ознакомиться с различными компоновщиками. Научиться обрабатывать события пользовательского интерфейса.

**Ход работы:**

Задание 4.1:

1. Измените текст метки на «Я метка 1» или любой другой.

import javax.swing.\*;

public class **main** {

public static void **main**(**String**[] args) {

**JFrame** f = new **JFrame**();

**JLabel** lab = new **JLabel**("Я метка 1"); *//создать новый элемент "метка"*

f.**setSize**(300, 200);

f.**setLocation**(500, 200); *//задает расположение окна*

f.**add**(lab); *//присоединить "метку" к окну*

f.**setDefaultCloseOperation**(**JFrame**.EXIT\_ON\_CLOSE);

f.**setVisible**(true);

}

}

2. Напишите программу, создающую три-четыре метки (кнопки) в окне. Как они будут располагаться относительно окна и друг относительно друга? А если менять размер окна?

Если попытаться добавить кнопки по очереди следующим образом:

import javax.swing.\*;

public class **Buttons** {

public static void **main**(**String**[] args) {

**JFrame** frame = new **JFrame**("Кнопочки");

frame.**setDefaultCloseOperation**(**JFrame**.EXIT\_ON\_CLOSE);

frame.**setSize**(400, 200);

**JButton** butt1 = new **JButton**("Кнопка 1");

**JButton** butt2 = new **JButton**("Кнопка 1");

**JButton** butt3 = new **JButton**("Кнопка 1");

**JButton** butt4 = new **JButton**("Кнопка 1");

frame.**add**(butt1);

frame.**add**(butt2);

frame.**add**(butt3);

frame.**add**(butt4);

frame.**setVisible**(true);

}

}

То будет отображаться только одна кнопка при любом размере окна.



Рисунок 1 – Окно программы Buttons.

Напишите программу, создающую 8 9 меток (кнопки) в окне (только теперь добавляйте элементы в панель JPanel). Как они будут располагаться относительно окна и друг относительно друга? А если менять размер окна от минимального до максимального? А если вытянуть его в узкую вертикальную полоску?

Если мы добавим кнопка сначала на панель, а панель – позже на фрейм, то получим необходимый результат:

import javax.swing.\*;

public class **Buttons** {

public static void **main**(**String**[] args) {

**JFrame** frame = new **JFrame**("Пример кнопок");

frame.**setDefaultCloseOperation**(**JFrame**.EXIT\_ON\_CLOSE);

frame.**setSize**(400, 200);

**JPanel** panel = new **JPanel**();

panel.**add**(new **JButton**("Кнопка 1"));

panel.**add**(new **JButton**("Кнопка 2"));

panel.**add**(new **JButton**("Кнопка 3"));

panel.**add**(new **JButton**("Кнопка 4"));

frame.**add**(panel);

frame.**setVisible**(true);

}

}

Расположение осуществляется по потоку — слева направо.

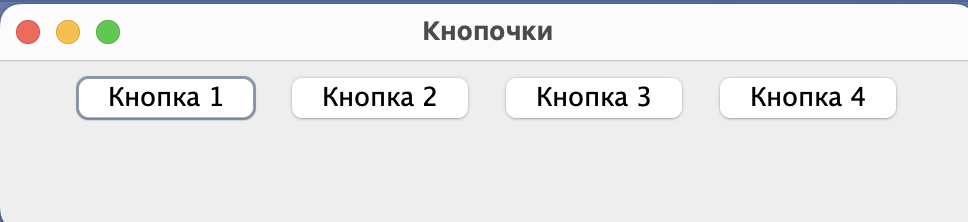


Рисунок 2 – Окно программы Buttons.

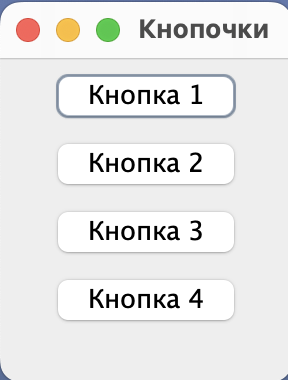


Рисунок 3 – Окно программы Buttons.

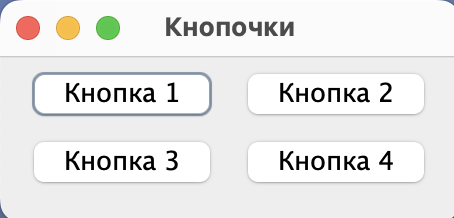


Рисунок 4 – Окно программы Buttons.

Если не хватает ширины окна — кнопки переносятся на следующую строку. При изменении размера окна они перестраиваются, чтобы уместиться.

Попробуем другие варианты компоновки.

Например, GridLayout расставляет компоненты в таблицу с заданным в конструкторе числом строк rows и столбцов columns.

import java.awt.\*;

import javax.swing.\*;

public class **Buttons** {

public static void **main**(**String**[] args) {

**JFrame** frame = new **JFrame**("Кнопочки");

frame.**setDefaultCloseOperation**(**JFrame**.EXIT\_ON\_CLOSE);

frame.**setSize**(400, 200);

*// 2 строки, 2 столбца*

frame.**setLayout**(new **GridLayout**(2, 2));

frame.**add**(new **JButton**("Кнопка 1"));

frame.**add**(new **JButton**("Кнопка 2"));

frame.**add**(new **JButton**("Кнопка 3"));

frame.**add**(new **JButton**("Кнопка 4"));

frame.**setVisible**(true);

}

}



Рисунок 5 – Окно программы Buttons.

В этом случае кнопки равномерно распределяются по ячейкам сетки. При изменении размера окна все кнопки меняют размер синхронно, оставаясь в своей ячейке.

Далее рассмотрим BorderLayout, он делит контейнер на пять неравных областей, полностью заполняя каждую область одним компонентом.

import java.awt.\*;

import javax.swing.\*;

public class **Buttons** {

public static void **main**(**String**[] args) {

**JFrame** frame = new **JFrame**("Кнопочки");

frame.**setDefaultCloseOperation**(**JFrame**.EXIT\_ON\_CLOSE);

frame.**setSize**(400, 200);

frame.**setLayout**(new **BorderLayout**());

frame.**add**(new **JButton**("Кнопка 1"), **BorderLayout**.NORTH);

frame.**add**(new **JButton**("Кнопка 2"), **BorderLayout**.SOUTH);

frame.**add**(new **JButton**("Кнопка 3"), **BorderLayout**.WEST);

frame.**add**(new **JButton**("Кнопка 4"), **BorderLayout**.EAST);

frame.**add**(new **JButton**("Кнопка 5"), **BorderLayout**.CENTER);

frame.**setVisible**(true);

}

}



Рисунок 6 – Окно программы Buttons.

Задание 4.2:

Реализовать визуальную работу банкомата — снятие и пополнение денег, вычисление остатка на счету и процентов по депозиту (класс Account и класс Bancomat).

Также была внедрена база данных для хранения информации о пользователях (номер их карты, пин-код, баланс счета).

В первую очередь рассмотрим UML-диаграмму классов в нашей программе:

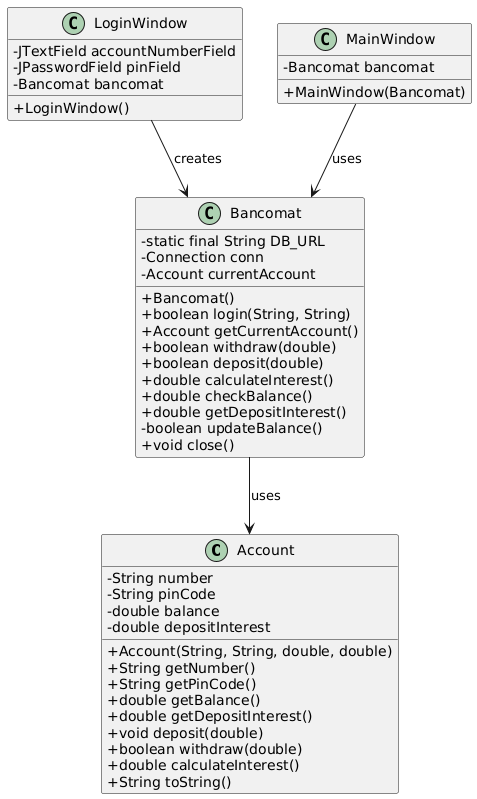


Рисунок 1 – UML-диаграмма классов

Данная программа представляет собой реализацию банкомата с графическим интерфейсом, предназначенного для взаимодействия с пользователями. Приложение позволяет выполнять основные банковские операции, такие как вход в систему по номеру счета и PIN-коду, просмотр баланса, пополнение счета, снятие средств и расчет процентов по депозиту. Все данные о пользователях хранятся в базе данных SQLite, к которой осуществляется подключение через JDBC.

Класс Account представляет банковский счет пользователя. Содержит информацию о номере счета, PIN-коде, текущем балансе и процентной ставке по депозиту. Также реализует функционал пополнения счета, снятие средств и расчет процентов. Класс Bancomatотвечает за работу с базой данных и управляет текущим активным пользователем (счетом):авторизация пользователя, выполнение операций со счетом через объект Account**,** синхронизация данных с БД.

Одним из ключевых сценариев использования банкомата является авторизация пользователя по номеру счёта и PIN-коду. Это событие реализуется при нажатии пользователем кнопки "Войти" в окне LoginWindow. Сценарий обеспечивает проверку доступа к системе и переход к основному окну работы с банкоматом только при успешной проверке введённых данных.

Рассмотрим диаграмму последовательности для обработки данного события:

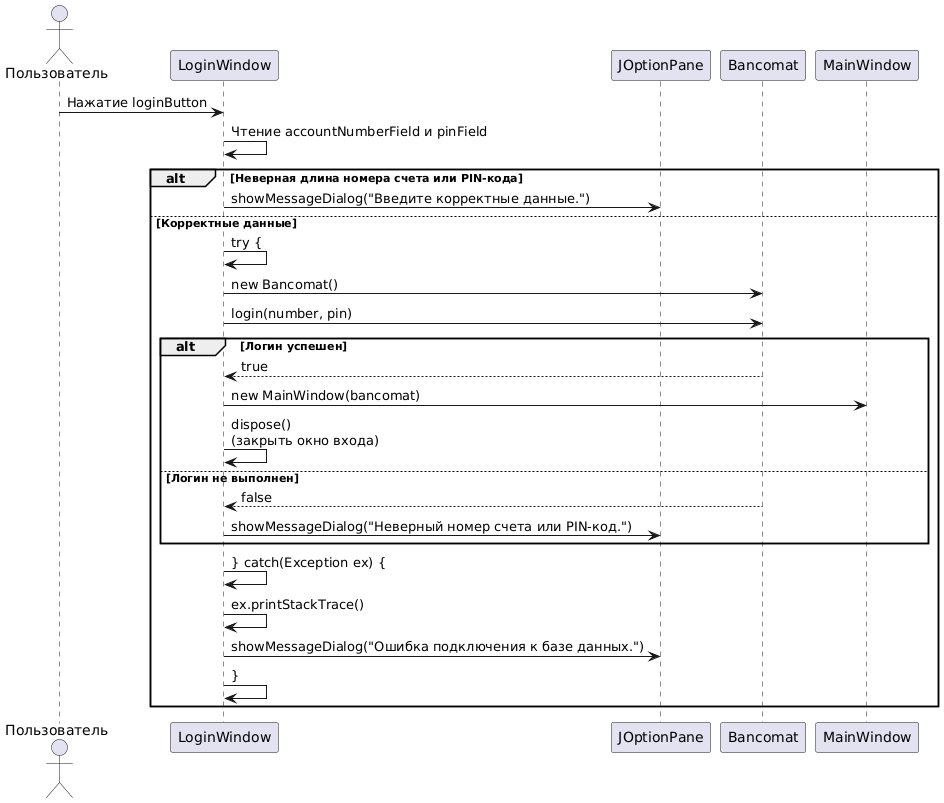
****

Рисунок 2 – Диаграмма последовательностей.

Листнинг программы:

Account.java:

public class Account {

private String number;

private String pinCode;

private double balance;

private double depositInterest;

public Account(String number, String pinCode, double balance, double depositInterest) {

this.number = number;

this.pinCode = pinCode;

this.balance = balance;

this.depositInterest = depositInterest;

}

public String getNumber() {

return number;

}

public String getPinCode() {

return pinCode;

}

public double getBalance() {

return balance;

}

public double getDepositInterest() {

return depositInterest;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

}

}

public boolean withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

return true;

} else {

return false;

}

}

public double calculateInterest() {

return balance \* (depositInterest / 100.0);

}

@Override

public String toString() {

return "Счет №" + number + "\nБаланс: " + balance + " руб.\nПроцент по депозиту: " + depositInterest + "%";

}

}

Bancomat.java:

import java.sql.\*;

public class Bancomat {

private static final String DB\_URL = "jdbc:sqlite:bancomat.db";

private Connection conn;

private Account currentAccount;

public Bancomat() {

try {

Class.forName("org.sqlite.JDBC");

conn = DriverManager.getConnection("jdbc:sqlite:bancomat.db");

} catch (Exception e) {

e.printStackTrace();

}

}

public boolean login(String number, String pinCode) {

String query = "SELECT \* FROM Accounts WHERE number = ? AND pin\_code = ?";

try (PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, number);

stmt.setString(2, pinCode);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

// Создание объекта Account

currentAccount = new Account(

rs.getString("number"),

rs.getString("pin\_code"),

rs.getDouble("balance"),

rs.getDouble("deposit\_interest")

);

return true;

}

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}

public Account getCurrentAccount() {

return currentAccount;

}

public boolean withdraw(double amount) {

if (currentAccount != null && currentAccount.withdraw(amount)) {

return updateBalance();

}

return false;

}

public boolean deposit(double amount) {

if (currentAccount != null) {

currentAccount.deposit(amount);

return updateBalance();

}

return false;

}

public double calculateInterest() {

if (currentAccount != null) {

return currentAccount.calculateInterest();

}

return 0.0;

}

public double checkBalance() {

if (currentAccount != null) {

return currentAccount.getBalance();

}

return 0.0;

}

public double getDepositInterest() {

if (currentAccount != null) {

return currentAccount.getDepositInterest();

}

return 0.0;

}

private boolean updateBalance() {

String update = "UPDATE Accounts SET balance = ? WHERE number = ?";

try (PreparedStatement stmt = conn.prepareStatement(update)) {

stmt.setDouble(1, currentAccount.getBalance());

stmt.setString(2, currentAccount.getNumber());

int affectedRows = stmt.executeUpdate();

return affectedRows == 1;

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}

public void close() {

try {

if (conn != null) conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

LoginWindow.java:

import java.awt.\*;

import javax.swing.\*;

public class LoginWindow extends JFrame {

private JTextField accountNumberField;

private JPasswordField pinField;

private Bancomat bancomat;

public LoginWindow() {

super("Вход в банкомат");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setSize(300, 200);

setLocationRelativeTo(null); // Центр экрана

// Элементы интерфейса

accountNumberField = new JTextField();

pinField = new JPasswordField();

JButton loginButton = new JButton("Войти");

// Layout

setLayout(new GridLayout(3, 2, 5, 5));

add(new JLabel("Номер счета (12 цифр):"));

add(accountNumberField);

add(new JLabel("PIN-код (4 цифры):"));

add(pinField);

add(new JLabel(""));

add(loginButton);

// Действие при нажатии на кнопку "Войти"

loginButton.addActionListener(e -> {

String number = accountNumberField.getText().trim();

String pin = new String(pinField.getPassword()).trim();

if (number.length() != 12 || pin.length() != 4) {

JOptionPane.showMessageDialog(this, "Введите корректные данные.");

return;

}

try {

bancomat = new Bancomat();

if (bancomat.login(number, pin)) {

new MainWindow(bancomat); // Переход в основное окно

dispose(); // Закрыть окно входа

} else {

JOptionPane.showMessageDialog(this, "Неверный номер счета или PIN-код.");

}

} catch (Exception ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(this, "Ошибка подключения к базе данных.");

}

});

setVisible(true);

}

}

MainWindow.java:

import java.awt.\*;

import javax.swing.\*;

public class MainWindow extends JFrame {

private Bancomat bancomat;

public MainWindow(Bancomat bancomat) {

super("Банкомат");

this.bancomat = bancomat;

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setSize(400, 250);

setLocationRelativeTo(null);

setLayout(new GridLayout(5, 1, 10, 10));

JButton balanceButton = new JButton("Посмотреть баланс");

JButton depositButton = new JButton("Пополнить счет");

JButton withdrawButton = new JButton("Снять деньги");

JButton interestButton = new JButton("Рассчитать проценты");

JButton exitButton = new JButton("Выход");

add(balanceButton);

add(depositButton);

add(withdrawButton);

add(interestButton);

add(exitButton);

balanceButton.addActionListener(e -> {

double balance = bancomat.checkBalance();

double interest = bancomat.getDepositInterest();

JOptionPane.showMessageDialog(this, "Ваш баланс: " + balance + " руб.\nПроцент по депозиту: " + interest);

});

depositButton.addActionListener(e -> {

String input = JOptionPane.showInputDialog(this, "Введите сумму для пополнения:");

try {

double amount = Double.parseDouble(input);

if (amount > 0) {

bancomat.deposit(amount);

JOptionPane.showMessageDialog(this, "Счет пополнен.");

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Неверный ввод.");

}

});

withdrawButton.addActionListener(e -> {

String input = JOptionPane.showInputDialog(this, "Введите сумму для снятия:");

try {

double amount = Double.parseDouble(input);

if (bancomat.withdraw(amount)) {

JOptionPane.showMessageDialog(this, "Деньги сняты.");

} else {

JOptionPane.showMessageDialog(this, "Недостаточно средств.");

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(this, "Неверный ввод.");

}

});

interestButton.addActionListener(e -> {

double interest = bancomat.calculateInterest();

JOptionPane.showMessageDialog(this, "Проценты по депозиту: " + interest + " руб.");

});

exitButton.addActionListener(e -> {

bancomat.close();

System.exit(0);

});

setVisible(true);

}

}

Main.java:

import javax.swing.SwingUtilities;

public class Main {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new LoginWindow());

}

}

Выводы: В ходе данной лабораторной работы были изучены библиотеки AWT и SWING для построения графического интерфейса пользователя. Освоены методы разработки с графическим интерфейсом Java, ознакомлены с различными компоновщиками.