Dokumentacja Projektu ZPOiF

Michał Matuszyk, Sebastian Pergała, Aleksander Mróz

January 26, 2024

Wprowadzenie

Ta kompleksowa aplikacja została stworzona jako projekt na przedmiot "Zaawansowane Programowanie Obiektowe i Funkcyjne". Celem projektu jest symulacja środowiska bankowego z uwzględnieniem różnych aspektów interakcji między klientami a bankiem, zarządzania kontami, operacji bankowych oraz interfejsu użytkownika. Projekt ten ma na celu zapewnienie praktycznego zrozumienia mechanizmów działania systemów bankowych.

Struktura Projektu

Folder application.component

public class AllAccountsTab extends javax.swing.JPanel

Pola:

- private List<Account> accounts;
- private JTable table;

Metody:

- public AllAccountsTab
- private void initComponents
- public void updateGUI

Opis: Strona z listą wszystkich utworzonych kont.

public class AllBanksTab extends javax.swing.JPanel

Pola:

- private final ArrayList<Bank> banks;

Metody:

- public AllBanksTab

- private void initComponents
- private void openBankGui
- public void updateGUI

Opis: Strona z listą wszystkich banków.

public class AllCustomersTab extends javax.swing.JPanel

Metody:

- public AllCustomersTab
- private void initComponents

Opis: Strona z listą wszystkich klientów banków.

public class BankingSystemTab extends javax.swing.JPanel

Pola:

- private BankingSystem bankingSystem;
- private CentralBankGUI centralBankGUI;
- private AllBanksGUI allBanksGUI;
- public AllAccountsGUI allAccountsGUI;
- private JButton startButton;
- private JButton stopButton;
- private JCheckBox createAccountsCheckbox;
- private JCheckBox createRandomTransfersCheckbox;
- private JSlider numberOfLoansSlider;
- private JSlider numberOfWithdrawalsSlider;
- private JSlider numberOfDepositsSlider;

Metody:

- public BankingSystemTab
- private void initComponents
- private void onButtonOrCheckboxChanged
- private void sliderChanged

Opis: Strona do sterowania parametrami systemu bankowego.

public class CentralBankTab extends javax.swing.JPanel

Pola:

- private CentralBank centralBank;
- private JLabel reserveRatioLabel;
- private JTextField reserveRatioField;
- private JLabel moneyInAllAccountsLabel;
- private JLabel numberOfBanksLabel;
- private JTextArea transactionsTextArea;

Metody:

- public CentralBankTab
- private void initComponents

Opis: Strona do sterowania parametrami banku centralnego.

public class DefaultForm extends javax.swing.JPanel

Pola:

- private javax.swing.JLabel jLabel1;

Metody:

- public DefaultForm
- private void initComponents

Opis: Domyślna strona wyświetlająca przekazany jej teks.

public class GiveLoanTab extends javax.swing.JPanel

Pola:

- private BankingSystem bankingSystem

Metody:

- public GiveLoanTab
- private void initComponents

Opis: Strona pozwalające na udzielanie pożyczek.

public class Header extends javax.swing.JPanel

Pola:

- private List<Account> accounts;
- private JTable table;

Metody:

- public Header
- @Override private void paintComponent
- private void initComponents

Opis: Nagłówek aplikacji.

public class InteractionsTab extends javax.swing.JPanel

Pola:

- private BankingSystem bankingSystem;

Metody:

- public InteractionsTab
- private void initComponents
- private void openTransactionGUI

Opis: Strona, za pomocą której można wywoływać interakcje między elementami systemu bankowego.

public class TransactionTab extends javax.swing.JPanel

Pola:

- private CentralBank centralBank;
- private List<Integer> accountIds;
- private JComboBox<Integer> originAccountDropdown;
- private JComboBox<Integer> destinationAccountDropdown;
- private JTextField amountTextField;
- private JButton sendButton;

Metody:

- public TransactionTab

- private void initComponents
- private void send
- private int parseAmount

Opis: Strona do wykonywania przelewów.

public class UserAccountInformationTab extends javax.swing.JPanel

Pola:

- private Account account;
- private Customer user;

Metody:

- public UserAccountInformationTab
- private void initComponents

Opis: Strona z informacjami o koncie z perspektywy użytkownika aplikacji.

public class UserGiveLoanTab extends javax.swing.JPanel

Pola:

- private BankingSystem bankingSystem;
- private Account userAccount;

Metody:

- public UserGiveLoanTab
- private void initComponents

Opis: Strona umożliwiająca branie kredytu z perspektywy użytkownika.

public class UserTransactionTab extends javax.swing.JPanel

Pola:

- private CentralBank centralBank;
- private List<Integer> accountIds;
- private JLabel originAccountDropdown;
- private JComboBox<Integer> destinationAccountDropdown;
- private JTextField amountTextField;

- private JButton sendButton;
- private Account userAccunt;

Metody:

- public UserTransactionTab
- private void initComponents
- private void send
- private int parseAmount

Opis: Strona umożliwiająca dokonywanie tranzakcji z perspektywy użytkownika.

Folder application.component.pictures

Opis: Folder z grafiką użytą w GUI aplikacji.

Folder application.main

public class Main extends javax.swing.JFrame

Pola:

- private javax.swing.JPanel body;
- private application.menu.Menu menu1;

Metody:

- public Main
- private void showForm
- private void initComponents

Opis: Klasa zarządzająca wyświetlaniem stron w aplikacji po ich wybraniu przez uzytkownika. Ponadto składa pozostałe funkcjonalne i graficzne komponenty aplikacji w całość.

Folder application.menu

public class Menu extends JComponent

Pola:

- private MenuEvent event;
- private MigLayout layout;

- private final String[ ][ ] menuItems = new String[ ][ ]{

  {"User's perspective", "Account information", "Make a transaction", "Take a loan"},

  {"System information", "Central Bank", "List of banks", "List of accounts", "List of customers"},

  {"System interactions", "Transfer money", "Give a loan"},

  {"God's interface", "Banking system", "Interactions"} };

Metody:

- public MenuEvent getEvent
- public void setEvent
- private void init
- private Icon getIcon
- private void addMenu
- private void addSubMenu
- private void hideMenu
- @Override protected void paintComponent

Opis: Klasa reprezentująca menu w aplikacji. Łączy w całość rozwijające się listy przycisków pokazujących konkretne strony z obszarem przeznaczonym na menu oraz za pomocą pola *menuItems* ustala zawartość menu.

public class MenuAnimation

Metody:

- public static void showMenu

Opis: Klasa z metodą odpowiadającą za rozwijanie i zwijanie się menu od strony graficznej.

public interface MenuEvent

Metody:

- void selected

Opis: Interfejs z metodą reprezentującą wybranie konkretnego elementu z menu.

public class MenuItem extends JButton

Pola:

- private BufferedImage shadow;
- private final int shadowSize = 10;
- private final int index;
- private final boolean subMenuAble;
- private int subMenuIndex;
- private int length;

Metody:

- public int getIndex
- public MenuItem
- private void createShadowImage
- public void initSubMenu
- @Override protected void paintComponent
- @Override public void setBounds

Opis: Klasa reprezentująca opcje widoczne w liście po rozwinięciu kategorii w menu, które kierują użytkownika do odpowiadających stron w aplikacji.

Folder application.scroll

public class ScrollBarWin11UI extends BasicScrollBarUI

Pola:

- private final int scrollSize = 10;

Metody:

- @Override public void installUI
- @Override protected JButton createIncreaseButton
- @Override protected JButton createDecreaseButton
- @Override protected void paintTrack
- @Override protected void paintThumb

Klasy wewnętrzne:

- private static class ScrollButton extends JButton

  Metody:

  - public ScrollButton

Opis: Klasa odpowiada za stworzenie graficznych komponentów paska przewijania w menu aplikacji.

public class ScrollPaneWin11 extends JScrollPane

Pola:

- private final int scrollSize = 10;

Metody:

- ScrollPaneWin11
- @Override public boolean isOptimizedDrawingEnabled
- @Override public void updateUI

Klasy wewnętrzne:

- private static class ScrollLayout extends ScrollPaneLayout

  Metody:

  - @Override public void layoutContainer

Opis: Klasa odpowiada za stworzenie graficznych komponentów paska przewijania w menu aplikacji jak i złożenie tych komponentów i nadanie elementowi wizualnego indykatora interaktywności.

Folder application.shadow

public class GraphicsUtilities

Metody:

- public GraphicsUtilities
- public static void getPixels
- public static void setPixels

Opis: Klasa zawierająca metody przydatne podczas graficznej obróbki komponentów aplikacji.

public class ShadowRenderer

Pola:

- private int size = 5;
- private float opacity = 0.5f;
- private Color color = Color.BLACK;

Metody:

- public ShadowRenderer
- public BufferedImage createShadow

Opis: Klasa odpowiadająca za tworzenie cieni pod wybranymi komponentami w aplikacji.

0.1 Folder Banking

Klasa: Account

Pola:

- private int accountId
- private int balance
- private String password

Opis: Zarządzanie informacjami konta.

Klasa: Bank

Pola:

- public int moneyOnHand
- public int minReserveRatio
- private CentralBank centralBank
- private int bankId
- public String bankName

Opis: Zarządzanie operacjami i informacjami bankowymi.

Klasa: BankingSystem

Pola:

- public CentralBank centralBank
- public BankingSystemGUI bankingSystemGUI
- public AllAccountsGUI allAccountsGUI
- private Timer timer
- public boolean createNewAccounts
- public boolean createRandomTransfers

Opis: Zarządzanie całym systemem, odpowiedzialny jest za symulacje.

Klasa: CentralBank

Pola:

- public BankingSystem bankingSystem

Opis: Jest odpowiedzialny za ustawianie reserveRatio, które pobierają wszystkie banki

Klasa: Loan

Pola:

- private Account account
- public int loanAmount
- private int loanPercentage
- private int outstandingAmount

Opis: Klasa reprezentująca pożyczkę bankową - domyślnie spłacana jest przez 20 lat, co miesiąc = 240 iteracji.

Klasa: Transaction

Pola:

- private int originAccountId
- private int destinationAccountId
- private int originBankId
- private int destBankId
- private int amount

Opis: Klasa opisująca tranzakcję między kontami.

Obsługa aplikacji

Do nawigacji w aplikacji służy boczne menu. Jego zawartość jest następująca:

- User's perspective
  - Account information
    - zawiera informacje o koncie z perspektywy użytkownika aplikacji
  - Make a transaction
    - pozwala na wykonanie przelewu z perspektywy użytkownika aplikacji
  - Take a loan
    - umożliwia wzięcie kredytu z perspektywy użytkownika aplikacji
- System information
  - Central Bank
    - ustawienie parametrów banku centralnego
  - List of banks
    - lista wszystkich banków z informacjami o nich, umożliwiająca pokazanie dodatkowych detali w osobnym oknie po kliknięciu na wiersz z danymi o konkretnym banku
  - List of accounts
    - lista wszytkich kont zawartych w systemie bankowym wraz z informacjami
  - List of customers
    - lista wszystkich klientów insniejących w systemie bankowym wraz z informacjami
- System interactions
  - Transfer money
    - strona umożliwiająca przelanie pieniędzy między dwoma dowolnymi kontami
  - Give loan
    - strona umożliwiająca udzielenie pożyczki dowolnemu kontu
- God's interface
  - Idea:
    - Umożliwienie użytkownikowi interakcji z całym system poprzez podgląd danych w każdej instancji banku oraz wywieranie wpływu na czynniki symulacji, jak np. liczba pożyczek udzielanych w każdym cyklu, liczba wypłat gotówki, liczba wpłat gotówki.
  - Banking system
    - strona pozwalająca na wyświetlanie informacji zawartych w kategorii *System information* w formie osobno otwartych okien oraz umożliwiająca przeprowadzenie symulacji działania systemu bankowego z wybranymi w na tej stronie parametrami

– Interactions

  - strona pozwalająca na wyświetlanie informacji zawartych w kategorii *System interactions* w formie osobno otwartych okien oraz umożliwiająca dodatkowe interakcje między wszystkii istniejącymi w systemie kontami, a systemem bankowych.

- Bank Run

  Poprzez ten projekt chcieliśmy zaprezentować jedno z największych niebezpieczeństw całego systemu bankowego, a mianowicie ryzyko tzw. "Bank run". Jest to zjawisko, w którym klienci banków chcą w jednym momencie wypłacić gotówkę, a bank jej nie posiada. Powoduje to eskalację problemu i ostatecznie prowadzi to do upadłości całego systemu. Dzięki naszemu projektowi, można zabawić się w prezesa banku centralnego i spróbować sterować rynkiem, tak aby żaden klient nie miał problemu z wypłatą gotówki. Możemy również pokazać co się dzieje, gdy wszyscy dokonają wypłaty gotówki. Możemy to zrobić poprzez naciśnięcie w panelu interakcji przycisku "Withdraw all cash from all accounts".

1 Możliwe ulepszenia

Niestety z ograniczonego czasu nasza aplikacja musiała dokonać wielu uproszczeń aby niezwykle skomplikowany system - jakim jest system bankowy - mógł zostać zasymulowany. Zatem poniżej przedstawiamy możliwe ulepszenia aby zwiększyć realizm tego projektu:

- **Wprowadzenie różnych typów kont bankowych**: Możliwość symulowania różnych typów kont bankowych, takich jak konta oszczędnościowe, konta walutowe, konta inwestycyjne itp. - dzięki skomplikowanym warunkom wyciągania pięniędzy z nich, jest możliwe użycie ich do stabilizacji systemu

- **Implementacja algorytmów zarządzania ryzykiem**: Dodanie algorytmów zarządzania ryzykiem, które mogą symulować działania banku w przypadku wystąpienia ryzyka bankowego, takiego jak ryzyko kredytowe, ryzyko operacyjne, niestety z racji na ich skomplikowaną naturę - ogromne zespoły w bankach - musieliśmy je znacząco uprościć

- **Analiza danych i raportowanie**: Rozwinięcie funkcjonalności aplikacji o mechanizmy analizy danych i raportowania, które umożliwią użytkownikowi lepsze zrozumienie zachowań klientów i sytuacji finansowej banku.

- **Symulacja kryzysu finansowego**: Dodanie możliwości symulacji kryzysu finansowego i jego wpływu na system bankowy, co pozwoliłoby użytkownikowi lepiej zrozumieć skutki takich zdarzeń i sposoby radzenia sobie z nimi - np. utrata wartości papierów wartościowych które posiada bank oraz przekład tego na sytuację finansową banku.