UNIWERSYTET RZESZOWSKI

WYDZIAŁ NAUK ŚCISŁYCH I TECHNICZNYCH

Klaudia Dudzińska

135052

Informatyka i ekonometria

Przygotowanie projektu Menadżer budżetu domowego w systemie LaTeX

Praca inżynierska

Praca wykonana pod kierunkiem
mgr inż. Ewa Żesławska

Rzeszów 2026

Serdecznie dziękuję za pomoc w przygotowaniach mgr. inż Ewie Żesławskiej

Spis treści

1. Wprowadzenie

Aplikacja „Budżet domowy” jest narzędziem programistycznym stworzonym w języku Java, służącym do ewidencjonowania i analizy finansów osobistych. Głównym zadaniem systemu jest umożliwienie użytkownikowi bieżącego monitorowania stanu portfela poprzez rejestrowanie wpływów i wydatków. Projekt wykorzystuje bibliotekę `Swing` do budowy interfejsu graficznego oraz zewnętrzną bibliotekę `JFreeChart` do wizualizacji danych w formie wykresów.

Aplikacja została zaprojektowana w sposób modułowy, dzieląc funkcjonalności na dedykowane panele odpowiedzialne za wprowadzanie danych, ich prezentację tabelaryczną oraz graficzną analizę. Dzięki zastosowaniu lekkiego interfejsu użytkownika, program pozwala na szybkie dodawanie transakcji oraz natychmiastowy podgląd aktualnego salda.

1.1. Cele pracy

Głównym celem pracy było zaprojektowanie i zaimplementowanie funkcjonalnej aplikacji desktopowej wspomagającej zarządzanie domowym budżetem. Realizacja tego celu wymagała rozwiązania szeregu problemów programistycznych związanych z obsługą zdarzeń, walidacją danych wejściowych oraz dynamicznym odświeżaniem widoków.

Szczegółowe cele projektu obejmowały:

- Stworzenie intuicyjnego interfejsu użytkownika (GUI) przy użyciu biblioteki Swing.
- Implementację mechanizmu przechowywania i modyfikacji danych transakcyjnych w pamięci operacyjnej.
- Integrację narzędzi do generowania wykresów statystycznych.

1.1.1. Zakres funkcjonalny

Aplikacja oferuje użytkownikowi następujące funkcjonalności:

- **Dodawanie wydatków i przychodów**: Możliwość wyboru predefiniowanych kategorii (np. Jedzenie, Transport, Pensja) oraz wprowadzenia opisu, daty i kwoty.
- **Prezentacja danych**: Wyświetlanie listy transakcji w formie tabeli oraz sumarycznego salda.
- **Wizualizacja**: Generowanie wykresów kołowych (struktura wydatków) i słupkowych (transakcje w czasie).
- **Edycja historii**: Możliwość usuwania błędnie wprowadzonych wpisów z automatyczną aktualizacją wykresów i sumy.

1.1.1.1. Walidacja danych

Istotnym elementem systemu jest weryfikacja poprawności danych wprowadzanych przez użytkownika. Zastosowano mechanizmy oparte o wyrażenia regularne, które blokują wprowadzenie błędnych formatów:

- Format daty jest wymuszany jako `YYYY-MM-DD`.
- Pola kwotowe akceptują wyłącznie znaki cyfrowe.

1.1.2. Struktura techniczna

Projekt został podzielony na pakiety, separując logikę widoku od punktu startowego aplikacji. Główna klasa `MainFrame` integruje poszczególne komponenty, takie jak `BudgetPanel` (obsługa salda), `GraphPanel` (wykresy) oraz `TransactionsPanel` (tabela danych), zapewniając przepływ informacji między nimi. Do budowania projektu i zarządzania zależnościami wykorzystano narzędzie `Make`.

1.2. Zawartość pracy

Niniejsza dokumentacja opisuje szczegółowo proces powstawania aplikacji „Budżet domowy". W kolejnych rozdziałach przedstawiono analizę poszczególnych modułów systemu. Omówiono strukturę klas odpowiedzialnych za interfejs użytkownika, w tym sposób organizacji paneli w klasie `MainFrame`. Przedstawiono również logikę obsługi zdarzeń (Listenery) oraz sposób wykorzystania biblioteki `JFreeChart` do dynamicznego generowania wykresów w klasie `GraphPanel`. Dokumentacja zawiera także opis formatu danych oraz mechanizmów walidacji zastosowanych w formularzach wejściowych.

2. Implementacja systemu

W rozdziale tym przedstawiono szczegółowe informacje dotyczące struktury plików źródłowych aplikacji oraz architektury klas zaimplementowanych w języku Java. Omówiono również metody kompilacji projektu z wykorzystaniem narzędzia *Make* oraz strukturę zależności między komponentami interfejsu graficznego.

2.1. Struktura projektu

Kod źródłowy aplikacji został podzielony na logiczne komponenty zgodnie z wzorcem architektonicznym separującym widok od logiki danych. Projekt składa się z następujących głównych elementów:

1. **Pakiet view** – zawierający klasy odpowiedzialne za interfejs graficzny:
    - `MainFrame.java` – główne okno integrujące wszystkie panele;
    - `TransactionsPanel.java` – tabela wyświetlająca historię operacji;
    - `GraphPanel.java` – wizualizacja danych (wykresy);
    - `BudgetPanel.java` – prezentacja aktualnego salda.
2. **Pliki konfiguracyjne** – `Makefile` służący do automatyzacji procesu budowania.

Poniżej przedstawiono fragment klasy `MainFrame.java`, ilustrujący sposób inicjalizacji interfejsu i podziału na panele:

Listing 2.1. Inicjalizacja głównego okna aplikacji

```
public class MainFrame extends JFrame {
    public MainFrame() {
        setTitle("Budżet domowy");
        setSize(1200, 800);

        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.X_AXIS));

        // Lewy panel (formularze i tabela)
        JPanel leftPanel = new JPanel();
        leftPanel.setLayout(new BoxLayout(leftPanel, BoxLayout.Y_AXIS));

        // Prawy panel (wykresy i saldo)
        JPanel rightPanel = new JPanel();
        // ... konfiguracja paneli
    }
}
```

Aplikacja nie korzysta ze skomplikowanych baz danych; stan aplikacji (lista transakcji) jest przechowywany w pamięci operacyjnej w modelu tabeli (`DefaultTableModel`). Klasy komunikują się ze sobą poprzez bezpośrednie wywołania metod, co jest widoczne w przekazywaniu referencji do `BudgetPanel` i `GraphPanel` w konstruktorze panelu transakcji.

2.2. Kompilacja i uruchomienie

Projekt wykorzystuje narzędzie `Make` do zarządzania procesem kompilacji, co eliminuje konieczność ręcznego wywoływania kompilatora `javac` z długą listą zależności. Plik `Makefile` definiuje zmienne środowiskowe oraz ścieżki do bibliotek zewnętrznych (`lib/*`).

Aby zbudować i uruchomić projekt, należy wykonać w terminalu następujące polecenia (zakładając, że znajdujemy się w głównym katalogu projektu):

```
# Kompilacja wszystkich plików źródłowych do katalogu out/
make all

# Uruchomienie aplikacji (klasa główna view.MainFrame)
make run

# Wyczyszczenie plików skompilowanych (.class)
make clean
```

Alternatywnie, bez użycia `make`, kompilacja wymagałaby ręcznego wskazania ‘classpath‘ dla biblioteki JFreeChart. Przykładowa komenda dla systemu Linux/Unix wyglądałaby następująco:

```
javac -d out -cp src:lib/* src/view/*.java src/Main.java
```

Podczas kompilacji kluczowe jest zachowanie struktury katalogów zgodnej z pakietami Java (`package view;`), dlatego pliki wynikowe `.class` trafiają do podkatalogu `out/view/`.

2.3. Wykorzystane technologie i biblioteki

Do implementacji projektu wybrano sprawdzone technologie ekosystemu Java, które zapewniają przenośność i łatwość tworzenia interfejsów okienkowych. Poniżej opisano kluczowe biblioteki wykorzystane w kodzie.

Biblioteki standardowe Java (JDK).

- **Swing (javax.swing)** – podstawowa biblioteka do budowy GUI. W projekcie wykorzystano komponenty takie jak `JFrame` (okno główne), `JPanel` (kontenery), `JTable` (prezentacja danych tabelarycznych) oraz `JComboBox` (wybór kategorii).
- **AWT (java.awt)** – wykorzystywana do zarządzania układami (`BoxLayout`, `BorderLayout`, `FlowLayout`) oraz obsługi zdarzeń.
- **Collections Framework** – do przechowywania i agregacji danych (np. `HashMap` i `LinkedHashMap` w klasie `MainFrame` do przygotowania danych dla wykresów).

Biblioteki zewnętrzne.

- **JFreeChart** – biblioteka służąca do generowania profesjonalnych wykresów w aplikacjach Java. W klasie `GraphPanel.java` wykorzystano ją do stworzenia:
  - Wykresu kołowego (`ChartFactory.createPieChart`) obrazującego podział wydatków na kategorie.
  - Wykresu słupkowego (`ChartFactory.createBarChart`) prezentującego sumy transakcji w poszczególnych dniach.

  Biblioteka ta wymaga dołączenia plików JAR znajdujących się w katalogu `lib/`.

Wykorzystanie zewnętrznej biblioteki graficznej pozwoliło na znaczące uproszczenie kodu odpowiedzialnego za wizualizację, przenosząc ciężar renderowania grafiki na gotowe komponenty `ChartPanel`.

2.4. Logika biznesowa i przepływ danych

Logika aplikacji opiera się na interakcji użytkownika z formularzami w klasie `MainFrame` oraz dynamicznym odświeżaniu widoków. Przygotowując implementację, zwrócono uwagę na kilka istotnych szczegółów:

- **Walidacja danych wejściowych**: Przed dodaniem transakcji system weryfikuje poprawność danych przy użyciu wyrażeń regularnych (Regex). Data musi być zgodna z formatem `YYYY-MM-DD`, a kwota musi składać się wyłącznie z cyfr (`\d+`). Błędy są zgłaszane za pomocą okien dialogowych `JOptionPane`.
- **Dynamiczna aktualizacja**: Dodanie nowego wydatku lub przychodu w `MainFrame` powoduje łańcuch wywołań: 1. Dodanie wiersza do `TransactionsPanel`. 2. Przeliczenie sumy całkowitej i aktualizację etykiety w `BudgetPanel`. 3. Pobranie zagregowanych danych i przerysowanie wykresów w `GraphPanel`.
- **Usuwanie rekordów**: Tabela transakcji posiada dedykowaną kolumnę z przyciskami "Usuń". Zostało to zrealizowane poprzez implementację własnego edytora komórek (`ButtonEditor` w klasie `TransactionsPanel`), który po kliknięciu usuwa wiersz z modelu i wymusza ponowne przeliczenie salda oraz odświeżenie wykresów.

Przepływ sterowania w aplikacji:

- Użytkownik klika "Dodaj"→ `ActionListener` weryfikuje dane.
- Poprawne dane → `transactionsPanel.addExpense()`.
- Aktualizacja widoku → `budgetPanel.setTotal()` oraz `graphPanel.updateData()`.

3. Implementacja komponentów systemu

3.1. Struktura logiczna aplikacji

Aplikacja „Budżet domowy” składa się z następujących modułów logicznych, które odwzorowują strukturę pakietów w projekcie:

- **view** – pakiet zawierający warstwę prezentacji (GUI),
- **MainFrame** – główne okno aplikacji integrujące wszystkie komponenty,
- **TransactionsPanel** – moduł odpowiedzialny za listę operacji i tabelę danych,
- **BudgetPanel** – panel wyświetlający podsumowanie finansowe,
- **GraphPanel** – moduł wizualizacji danych (wykresy),
- **resources** – zewnętrzne biblioteki (JFreeChart) wymagane do kompilacji.

3.2. Funkcjonalności i walidacja

Proces dodawania nowej transakcji zaimplementowany w klasie `MainFrame` przebiega w następujących krokach:

1. Pobranie danych z formularza (Kategoria, Opis, Data, Kwota);
2. Walidacja formatu daty (wymagany format YYYY-MM-DD);
3. Walidacja kwoty (sprawdzenie czy wartość jest numeryczna);
4. Aktualizacja modelu danych i odświeżenie widoków.

System obsługuje następujące typy operacji finansowych:

- **Wydatki** – pomniejszające saldo (wartości ujemne w tabeli);
- **Przychody** – powiększające saldo (wartości dodatnie w tabeli);
- **Usuwanie** – wycofanie błędnej transakcji z historii.

3.3. Formatowanie danych finansowych

Do wyświetlania kwot pieniężnych wykorzystano standardowe formatowanie liczb zmiennoprzecinkowych z dokładnością do dwóch miejsc po przecinku. W języku Java odpowiada za to metoda `String.format`, użyta w panelu budżetu:

```
String.format("%.2f zł", total)
```

Dzięki temu, nawet przy dużej liczbie miejsc po przecinku w obliczeniach, użytkownik widzi czytelną wartość, np.:

1234,56 zł

3.4. Wizualizacja danych i struktura klas

Wszystkie panele w aplikacji są osadzone w głównym kontenerze, co przedstawiono na Rys. 3.1. Klasa `MainFrame` zarządza układem `BoxLayout`, dzieląc okno na sekcję wprowadzania danych (lewą) oraz sekcję analityczną (prawą).

Ważne: Aby wykresy generowały się poprawnie, do projektu muszą być dołączone biblioteki `jcommon` oraz `jfreechart`. Brak tych plików w `CLASSPATH` uniemożliwi kompilację klasy `GraphPanel`.

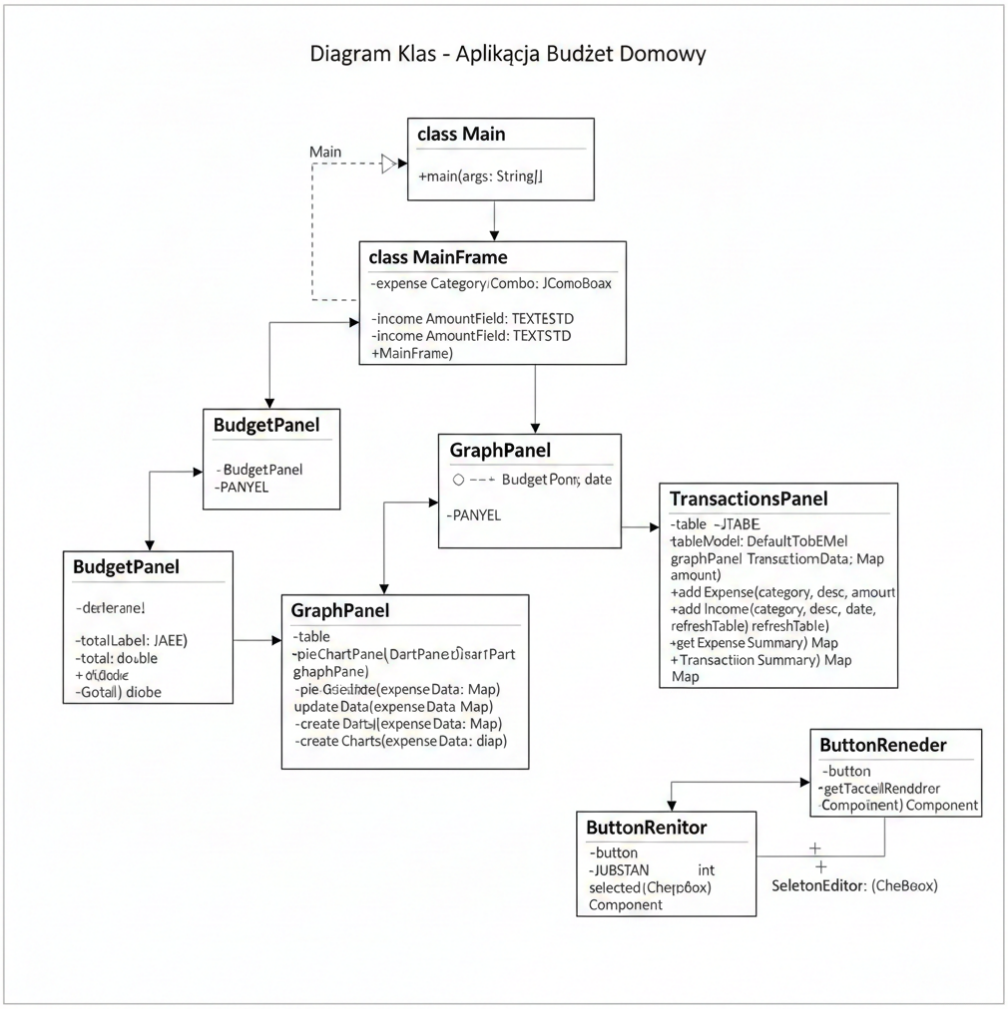
Schemat przepływu danych pomiędzy komponentami przedstawia się następująco: dane wprowadzone w `MainFrame` trafiają do `TransactionsPanel`, skąd po przetworzeniu zasilają wykresy w `GraphPanel`.

3.4.1. Struktura GUI

Interfejs graficzny oparty jest na bibliotece Swing. Główne elementy to:

- `JPanel` (kontenery),
- `JButton` (przyciski akcji),
- `JTable` (prezentacja danych),
- `JFreeChart` (zewnętrzny komponent wykresów).

Diagram klas (uproszczony) przedstawiono na Rys. 3.1.



Rys. 3.1. Diagram klas projektu Budżet Domowy.

Pakiet `view` zawiera podział na panele funkcjonalne, co pozwala na niezależne odświeżanie fragmentów interfejsu (zob. Rys. 3.2):

```
DefaultPieDataset pieDataset = new DefaultPieDataset();
for (Map.Entry<String, Double> entry : expenseData.entrySet()) {
    pieDataset.setValue(entry.getKey(), entry.getValue());
}
JFreeChart pieChart = ChartFactory.createPieChart(
        "Wydatki wg kategorii",
        pieDataset,
        false,
        true,
        false);
pieChartPanel = new ChartPanel(pieChart);
```

(a) Wydatki wg kategorii

```
DefaultCategoryDataset barDataset = new DefaultCategoryDataset();
for (Map.Entry<String, Double> entry : transactionData.entrySet()) {
    barDataset.addValue(entry.getValue(), "Kwota", entry.getKey());
}
JFreeChart barChart = ChartFactory.createBarChart(
        "Transakcje w czasie",
        "Data",
        "Kwota",
        barDataset,
        org.jfree.chart.plot.PlotOrientation.VERTICAL,
        false,
        true,
        false);
```

(b) Transakcje w czasie

Rys. 3.2. Wizualizacja danych w GraphPanel:

3.4.2. Definicja modelu danych

Tabela transakcji w `TransactionsPanel` wykorzystuje model `DefaultTableModel`. Struktura kolumn została przedstawiona w Tab. 3.1.

Tabela 3.1. Struktura tabeli transakcji.

Nazwa kolumny	Typ danych	Opis zawartości
Kategoria	String	Wybrana kategoria (np. Jedzenie, Pensja)
Opis	String	Krótki opis wprowadzony przez użytkownika
Data	String	Data w formacie YYYY-MM-DD
Kwota	Double	Wartość numeryczna (ujemna dla wydatków)
Akcja	JButton	Przycisk "Usuń"do kasowania wiersza

Kategorie wydatków i przychodów są zdefiniowane na sztywno w kodzie źródłowym, co przedstawia Tab. 3.2.

Tabela 3.2. Predefiniowane kategorie w systemie

Typ transakcji	Dostępne kategorie
Wydatki	Jedzenie, Transport, Mieszkanie, Rozrywka, Inne
Przychody	Pensja, Premia, Zwrot podatku, Inne

3.5. Algorytmy obliczeniowe

Podstawowym algorytmem w aplikacji jest obliczanie całkowitego salda S. Saldo jest sumą wszystkich zarejestrowanych transakcji t, gdzie $t \in T$ (zbiór wszystkich transakcji).

$$S = \sum_{i=0}^{n} t_i \tag{3.1}$$

gdzie: t_i może przyjmować wartości dodatnie (przychód) lub ujemne (wydatek).

W kontekście implementacji w języku Java, sumowanie odbywa się poprzez iterację po wierszach tabeli. Niech R będzie zbiorem wierszy w modelu tabeli:

$$S = \sum_{row=0}^{R.length} \text{parse}(R[row].\text{amount}) \tag{3.2}$$

gdzie funkcja parse konwertuje ciąg znaków na liczbę zmiennoprzecinkową typu `double`.

Dla wykresów statystycznych, dane są agregowane. Dla wykresu kołowego (Pie Chart) sumowane są wartości bezwzględne wydatków dla poszczególnych kategorii C:

$$\text{PieValue}(c) = \sum_{t \in T_c} |t_{amount}| \quad \text{dla} \quad t_{amount} < 0 \tag{3.3}$$

gdzie:

- c – kategoria wydatku,
- T_c – podzbiór transakcji należących do kategorii c.
-

3.6. Implementacja kodu źródłowego

Poniżej przedstawiono kluczowe fragmenty kodu aplikacji. Kod źródłowy znajduje się w katalogu `src/view`.

3.6.1. Panel wykresów (GraphPanel)

Klasa `GraphPanel` odpowiada za tworzenie wykresów przy użyciu biblioteki JFreeChart. Metoda `createCharts` inicjalizuje dwa niezależne panele wykresów.

Listing 3.1. Metoda tworząca wykresy w GraphPanel.java

```
private void createCharts(Map<String, Double> expenseData, Map<String, Double>
    transactionData) {

    DefaultPieDataset pieDataset = new DefaultPieDataset();
    for (Map.Entry<String, Double> entry : expenseData.entrySet()) {
        pieDataset.setValue(entry.getKey(), entry.getValue());
    }
    JFreeChart pieChart = ChartFactory.createPieChart(
            "Wydatki wg kategorii",
            pieDataset,
            false, true, false);
    pieChartPanel = new ChartPanel(pieChart);


    DefaultCategoryDataset barDataset = new DefaultCategoryDataset();
    for (Map.Entry<String, Double> entry : transactionData.entrySet()) {
        barDataset.addValue(entry.getValue(), "Kwota", entry.getKey());
    }

    add(pieChartPanel);
    add(barChartPanel);
}
```

3.6.2. Panel budżetu (BudgetPanel)

Panel ten jest odpowiedzialny za wyświetlanie aktualnego stanu konta. Wykorzystuje prosty mechanizm `JLabel` do aktualizacji tekstu.

Listing 3.2. Klasa BudgetPanel.java

```
public class BudgetPanel extends JPanel {
    private JLabel totalLabel;
    private double total = 0.0;

    public BudgetPanel() {
        setBackground(new Color(220, 255, 220));
        setLayout(new FlowLayout(FlowLayout.LEFT));
        add(new JLabel("Budzet domowy:"));
        totalLabel = new JLabel("0.00 zl");
        add(totalLabel);
    }

    public void setTotal(double total) {
        this.total = total;
        totalLabel.setText(String.format("%.2f zl", total));
```

```
255     }
256 }
```

257 3.6.3. Obsługa zdarzeń w MainFrame

258 Poniższy listing prezentuje obsługę przycisku dodawania wydatku, w tym walidację danych wejścio-
259 wych za pomocą wyrażeń regularnych (Regex).

Listing 3.3. Obsługa dodawania wydatku w MainFrame.java

```
260 addExpenseButton.addActionListener(e -> {
261     String category = expenseCategoryCombo.getSelectedItem().toString();
262     String desc = expenseDescField.getText();
263     String date = expenseDateField.getText();
264     String amount = expenseAmountField.getText();
265 
266     if (!amount.matches("\\d+")) {
267         JOptionPane.showMessageDialog(this, "Kwota␣moze␣zawierac␣tylko␣cyfry!",
268                 "Blad", JOptionPane.ERROR_MESSAGE);
269         return;
270     }
271 
272     if (!date.matches("\\d{4}-\\d{2}-\\d{2}")) {
273         JOptionPane.showMessageDialog(this, "Data␣musi␣byc␣w␣formacie␣YYYY-MM-DD!",
274                 "Blad", JOptionPane.ERROR_MESSAGE);
275         return;
276     }
277 
278     transactionsPanel.addExpense(category, desc, date, amount);
279 
280 });
```

Bibliografia

[1] Ada Europe. *Ada Reference Manual ISO/IEC 8652:200y(E) Ed. 3*, 2006.

[2] A. Burns and B. Dobbing. The Ravenscar Profile for real–time and high integrity systems. *CrossTalk*, 16(11):9–12, 2003.

[3] A. Burns, B. Dobbing, and T. Vardanega. Guide for the use of the ada ravenscar profile in high integrity systems. Technical Report YCS-2003-348, University of York, 2003.

[4] A. Diller. *LaTeX wiersz po wierszu*. Wydawnictwo Helion, Gliwice, 2000.

[5] L. Lamport. *LaTeX system przygotowywania dokumentów*. Wydawnictwo Ariel, Kraków, 1992.

[6] Object Refinery Limited. Jfreechart library documentation, 2024. Dostęp: 2024-09-16.

[7] Oracle. Java foundation classes (jfc) / swing documentation, 2024. Dostęp: 2024-09-16.

[8] J. Peleska, D. Große, A. E. Haxthausen, and R. Drechsler. Automated verification for train control systems. In *Proc. of the 5th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMAT 2004)*, pages 252–265, Braunschweig, Germany, December 2004.

Spis rysunków

Spis tabel

Spis listingów

Streszczenie pracy

Przygotowanie projektu Menadżer budżetu domowego w systemie LaTeX

Przedmiotem niniejszej pracy jest zaprojektowanie i implementacja aplikacji desktopowej służącej do zarządzania budżetem domowym, stworzonej w języku Java z wykorzystaniem biblioteki Swing. Głównym celem projektu było opracowanie narzędzia umożliwiającego efektywne monitorowanie przychodów i wydatków oraz ich wizualną analizę.

Aplikacja opiera się na modularnej architekturze podzielonej na panele funkcjonalne:

- **Moduł ewidencji transakcji**: umożliwia dodawanie, usuwanie oraz kategoryzację operacji finansowych, które są prezentowane w formie czytelnej tabeli.
- **Moduł analityczny**: wykorzystuje bibliotekę JFreeChart do generowania dynamicznych wykresów kołowych (podział wydatków na kategorie) oraz słupkowych (historia transakcji w czasie).
- **Moduł monitorowania salda**: odpowiada za bieżące przeliczanie i wyświetlanie całkowitego stanu budżetu.

W ramach pracy zrealizowano interfejs użytkownika zapewniający walidację wprowadzanych danych, w tym poprawność formatu dat oraz kwot numerycznych. System pozwala na bieżące śledzenie kondycji finansowej poprzez automatyczne odświeżanie widoków graficznych po każdej modyfikacji danych. Dokumentacja techniczna została przygotowana w systemie LaTeX z uwzględnieniem standardów opisu systemów informatycznych.

Słowa kluczowe: Java, Swing, JFreeChart, zarządzanie finansami, budżet domowy, wizualizacja danych.

Abstract

Thesis in LaTeX

The subject of this thesis is the design and implementation of a desktop application for home budget management, developed in the Java programming language using the Swing library. The primary objective of the project was to create a tool enabling effective monitoring of incomes and expenses, along with their visual analysis.

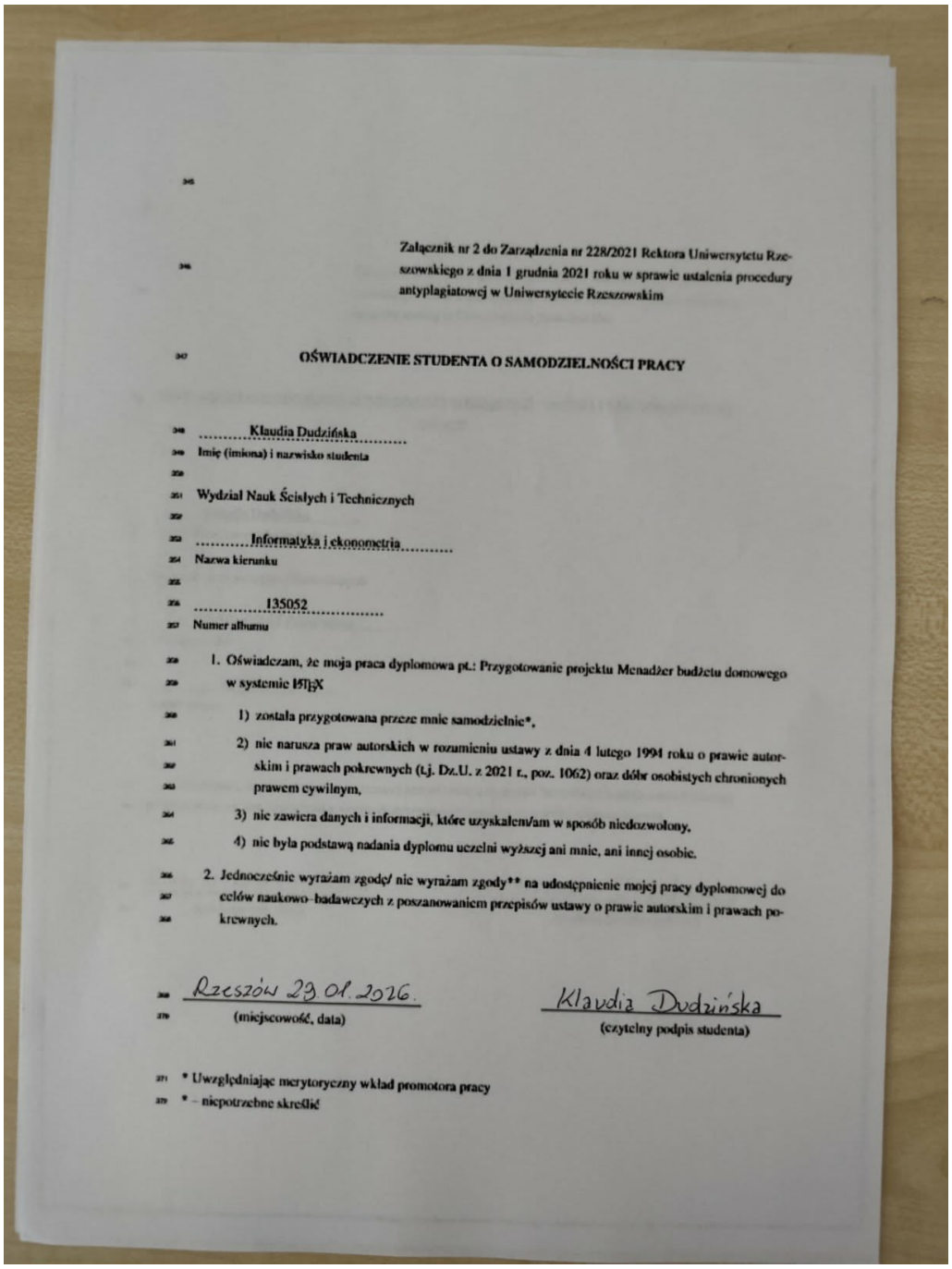
The application is based on a modular architecture divided into functional panels:

- **Transaction records module**: enables adding, deleting, and categorizing financial operations, which are presented in a clear table format.
- **Analytical module**: utilizes the JFreeChart library to generate dynamic pie charts (expense breakdown by category) and bar charts (transaction history over time).
- **Balance monitoring module**: responsible for the real-time calculation and display of the total budget status.

As part of the project, a user interface was implemented to provide data validation, including the correctness of date formats and numerical amounts. The system allows for continuous tracking of financial health through the automatic refreshing of graphical views after each data modification. The technical documentation was prepared using the LaTeX system, adhering to standard specifications for information systems description.

Keywords: Java, Swing, JFreeChart, financial management, home budget, data visualization.

Oświadczenie o samodzielności



Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OŚWIADCZENIE STUDENTA O SAMODZIELNOŚCI PRACY

Klaudia Dudzińska
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

Informatyka i ekonometria
Nazwa kierunku

135052
Numer albumu

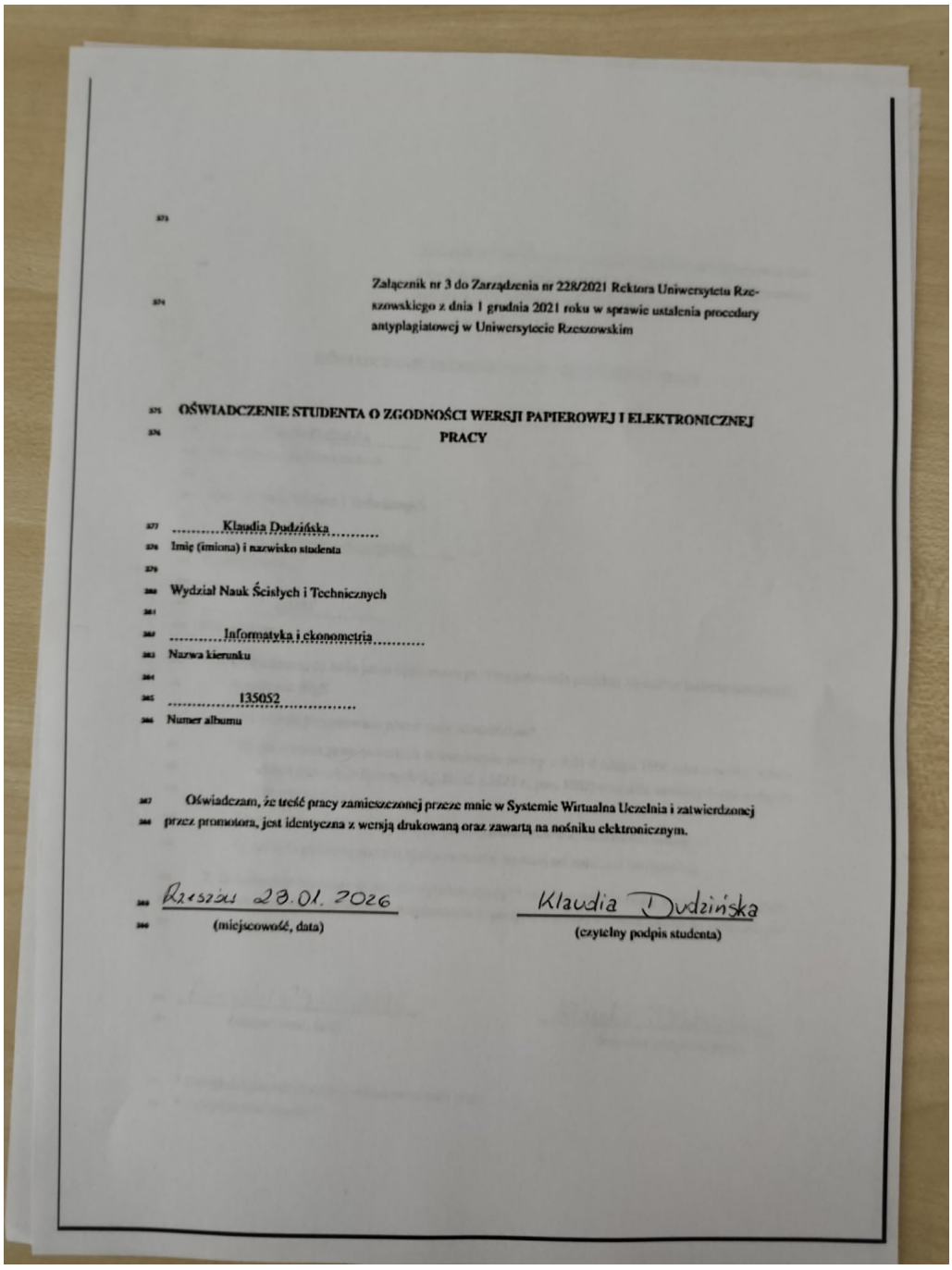
1. Oświadczam, że moja praca dyplomowa pt.: Przygotowanie projektu Menadżer budżetu domowego w systemie LaTeX
   1) została przygotowana przeze mnie samodzielnie*,
   2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
   3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
   4) nie była podstawą nadania dyplomu uczelni wyższej ani mnie, ani innej osobie.
2. Jednocześnie wyrażam zgodę/ nie wyrażam zgody** na udostępnienie mojej pracy dyplomowej do celów naukowo-badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

Rzeszów 23.01.2026
(miejscowość, data)

Klaudia Dudzińska
(czytelny podpis studenta)

* Uwzględniając merytoryczny wkład promotora pracy
** – niepotrzebne skreślić

Oświadczenie o udostępnieniu pracy



Załącznik nr 3 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OŚWIADCZENIE STUDENTA O ZGODNOŚCI WERSJI PAPIEROWEJ I ELEKTRONICZNEJ PRACY

Klaudia Dudzińska
Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

Informatyka i ekonometria
Nazwa kierunku

135052
Numer albumu

Oświadczam, że treść pracy zamieszczonej przeze mnie w Systemie Wirtualna Uczelnia i zatwierdzonej przez promotora, jest identyczna z wersją drukowaną oraz zawartą na nośniku elektronicznym.

Rzeszów 28.01.2026
(miejscowość, data)

Klaudia Dudzińska
(czytelny podpis studenta)