Zastosowanie języka JAVA w aplikacjach mobilnych i desktopowych

Zbigniew Kluczkowski

Spis treści

Wprowadzenie

W języku Java, tworzenie aplikacji z interfejsem graficznym (GUI) jest możliwe dzięki bibliotekom i frameworkom, które oferują różne narzędzia do tworzenia okien, przycisków, etykiet, pól tekstowych
i innych elementów interaktywnych.

AWT (Abstract Window Toolkit) oraz **Swing** to dwie główne biblioteki, które pozwalają na tworzenie takich interfejsów w Javie. Obie mają swoje specyficzne cechy, zalety i ograniczenia.

AWT – Abstract Window Toolkit

AWT to pierwsza biblioteka GUI w Javie, która została wprowadzona razem z pierwszymi wersjami języka. Służy do tworzenia interfejsów użytkownika z wykorzystaniem komponentów takich jak okna, przyciski, etykiety i inne.

Charakterystyka AWT:

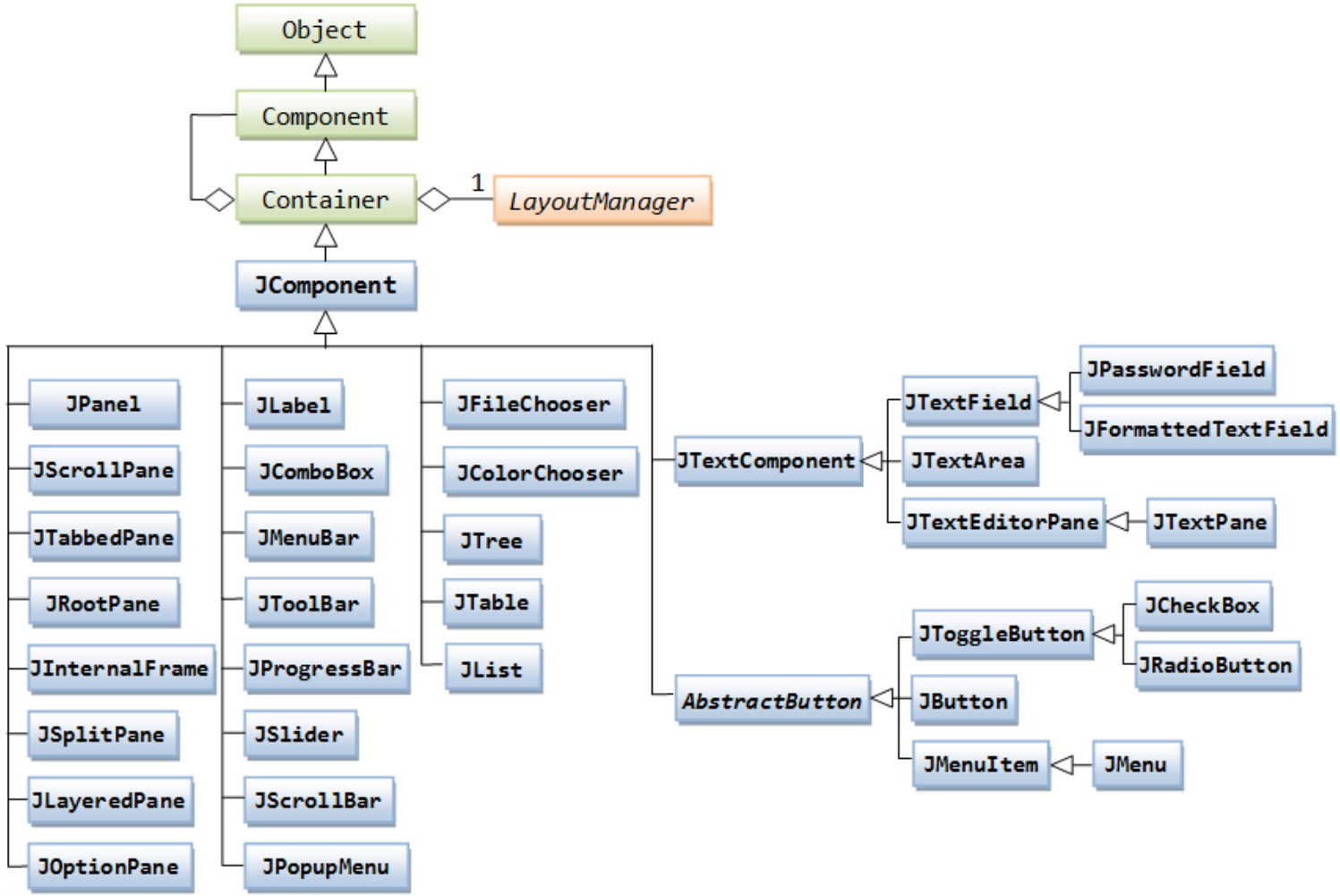
Platforma zależna: AWT korzysta z natywnych komponentów systemowych, co oznacza, że wygląd aplikacji zależy od systemu operacyjnego (np. różnice w wyglądzie przycisków w Windows i Linux).	**Komponenty:** AWT dostarcza podstawowe komponenty, takie jak Button, Label, TextField, List, Checkbox, TextArea, Frame.	**Event Handling:** Obsługuje zdarzenia (eventy) związane z interakcją użytkownika, jak kliknięcia, wprowadzanie tekstu, czy zmiany rozmiaru okna.

Przykładowy kod z omówieniem

```
import java.awt.*;
import java.awt.event.*;

public class AWTExample {
   public static void main(String[] args) {
      Frame frame = new Frame("Przykład AWT");
      Button button = new Button("Kliknij mnie");

      button.setBounds(50, 100, 150, 30);

      button.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            System.out.println("Przycisk został kliknięty!");
         }
      });

      frame.add(button);
      frame.setSize(300, 300);
      frame.setLayout(null);
      frame.setVisible(true);

      frame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent we) {
            System.exit(0);
         }
      });
   }
}
```

- Aplikacja składa się z okna oraz przycisku. Zastosowano następujące metody: add (dodaje) i set (ustawia).
- ActionListener to odpowiednik „nasłuchiwacza” w Javie. Po naciśnięciu przycisku okno jest zamykane.
- Przy ustawieniu przycisku dwie pierwsze współrzędne oznaczają lewy górny róg przycisku a kolejne to długość i wysokość elementu.

Drzewo komponentów



Swing – rozszerzenie AWT

Swing to nowsza i bardziej zaawansowana biblioteka GUI w Javie, która oferuje większą elastyczność oraz lepszą estetykę aplikacji. Swing jest niezależny od platformy, co oznacza, że wygląd aplikacji jest jednolity na różnych systemach operacyjnych.

Charakterystyka Swing:

Platforma niezależna: Swing nie korzysta z natywnych komponentów systemowych, zamiast tego implementuje własne, co zapewnia jednolity wygląd aplikacji na różnych platformach.	**Bogatszy zestaw komponentów:** Swing oferuje bardziej zaawansowane komponenty, takie jak JButton, JLabel, JTextField, JComboBox, JTable, JList, JMenuBar, JTree.	**Lepsza obsługa zdarzeń:** Swing oferuje bardziej rozbudowaną obsługę zdarzeń i lepsze mechanizmy układów (layout managers).	**Wydajność:** Swing, dzięki oddzieleniu logiki aplikacji od elementów graficznych, może oferować lepszą wydajność w bardziej zaawansowanych aplikacjach.

Przykład SWING z omówieniem

```
import javax.swing.*;
import java.awt.event.*;

public class SwingExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Przykład Swing");
        JButton button = new JButton("Kliknij mnie");

        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Przycisk został kliknięty!");
            }
        });

        frame.add(button);
        frame.setSize(300, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

- Prosta aplikacja z przyciskiem i zdarzeniem, którego efektem jest wyświetlenie komunikatu.
- SetDefaultCloseOperation ustawia w oknie przyciski funkcyjne charakterystyczne dla każdego systemu operacyjnego.

Różnice między AWT a Swing

Cecha	AWT	Swing
Zależność od platformy	Zależy od natywnych komponentów systemowych	Niezależne od platformy, własne komponenty GUI
Komponenty	Mniej zaawansowane, ograniczone	Bardziej zaawansowane, bogatszy zestaw
Wygląd	Różni się w zależności od systemu operacyjnego	Jednolity wygląd na różnych systemach
Wydajność	Mniej wydajne przy bardziej złożonych UI	Lepsza wydajność w bardziej złożonych aplikacjach
Obsługa zdarzeń	Prosta obsługa zdarzeń	Rozbudowana i bardziej elastyczna

Zalety i wady AWT i Swing

AWT:

- **Zalety:** Prosty w użyciu, szybki do nauki, natywne komponenty systemowe.
- **Wady:** Ograniczony zestaw komponentów, brak jednolitego wyglądu na różnych platformach.

Swing:

- **Zalety:** Bardziej zaawansowane komponenty, niezależność od platformy, lepsza kontrola nad wyglądem i zachowaniem aplikacji.
- **Wady:** Może być bardziej skomplikowany w użyciu i wymagać większej ilości kodu.

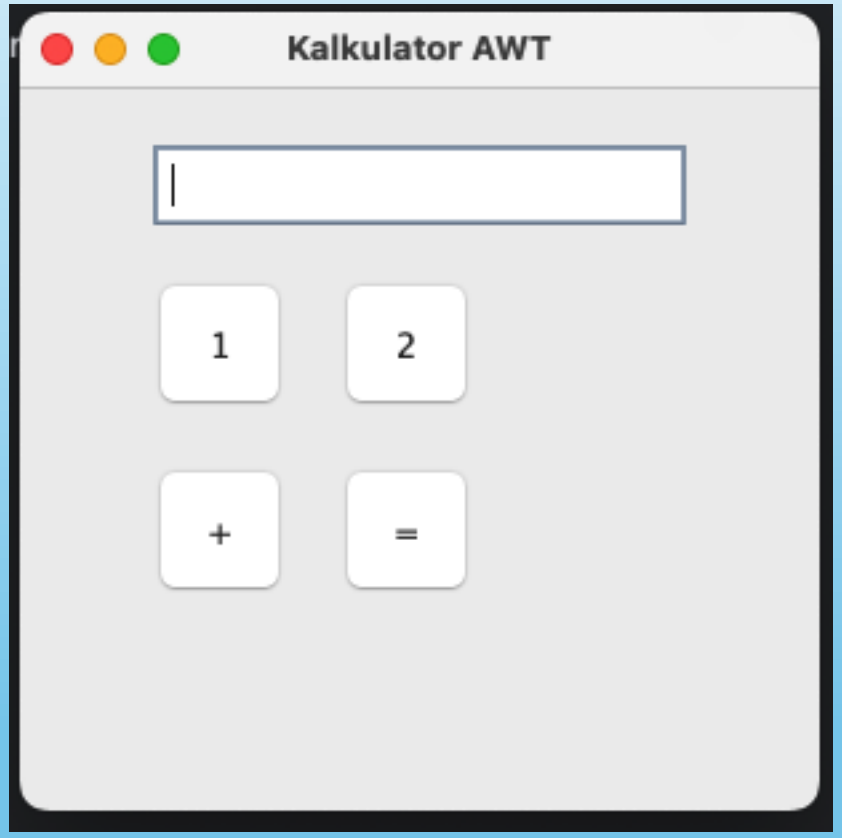
Prosty przykład – kod kalkulatora

```
import java.awt.*;
import java.awt.event.*;

public class AWTCalculator {
    public static void main(String[] args) {
        Frame frame = new Frame("Kalkulator AWT");
        TextField display = new TextField();
        display.setBounds(50, 50, 200, 30);

        Button button1 = new Button("1");
        Button button2 = new Button("2");
        Button buttonAdd = new Button("+");
        Button buttonEquals = new Button("=");

        button1.setBounds(50, 100, 50, 50);
        button2.setBounds(120, 100, 50, 50);
        buttonAdd.setBounds(50, 170, 50, 50);
        buttonEquals.setBounds(120, 170, 50, 50);
```

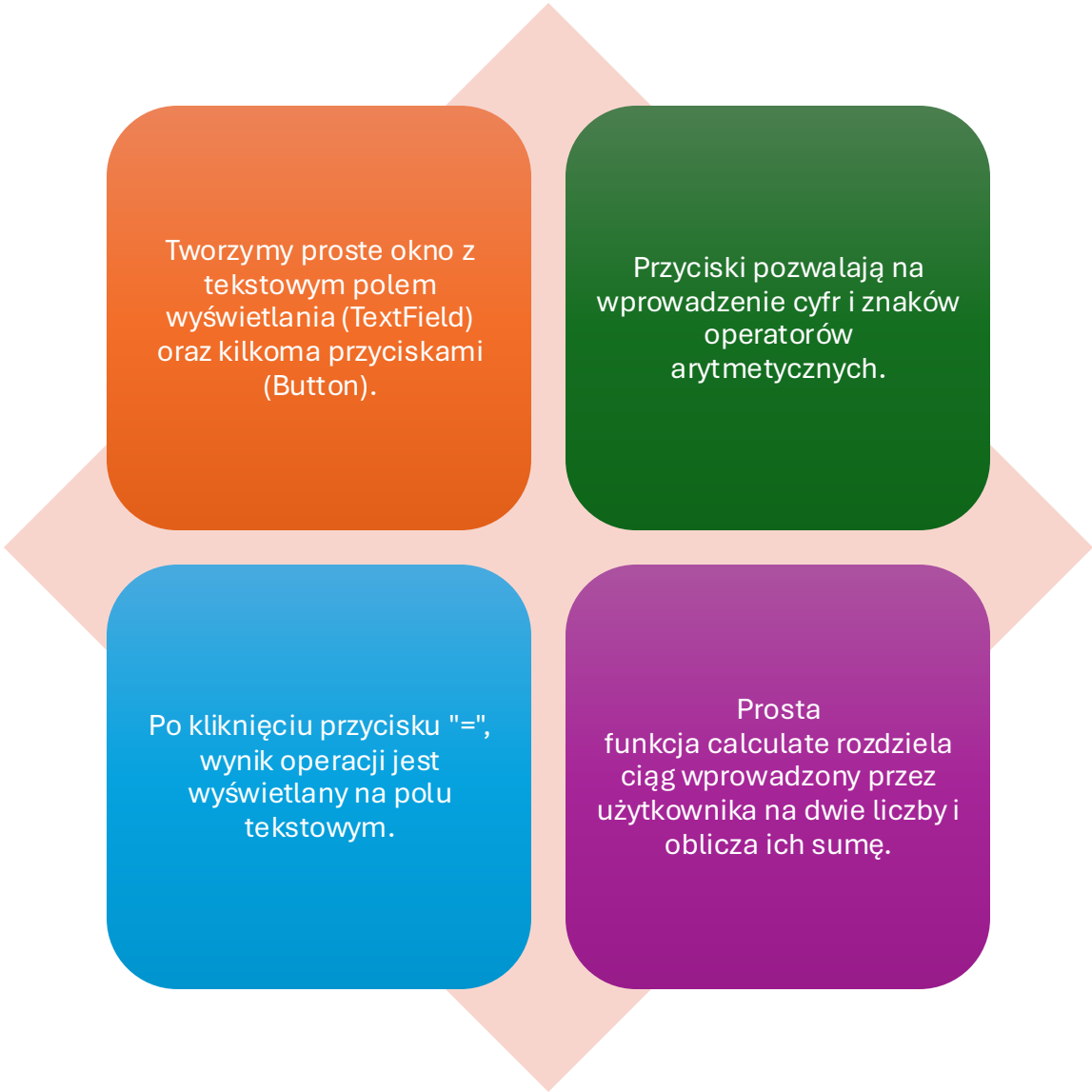
```
        frame.add(display);
        frame.add(button1);
        frame.add(button2);
        frame.add(buttonAdd);
        frame.add(buttonEquals);

        frame.setSize(300, 300);
        frame.setLayout(null);
        frame.setVisible(true);

        button1.addActionListener(e -> display.setText(display.getText() + "1"));
        button2.addActionListener(e -> display.setText(display.getText() + "2"));
        buttonAdd.addActionListener(e -> display.setText(display.getText() + "+"));
        buttonEquals.addActionListener(e -> {
            try {
                String result = calculate(display.getText());
                display.setText(result);
            } catch (Exception ex) {
                display.setText("Błąd");
            }
        });
```

Prosty przykład – kod kalkulatora

```
 frame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent we) {
            System.exit(0);
         }
      });
   }

   private static String calculate(String input) {
      String[] tokens = input.split("\\+");
      if (tokens.length == 2) {
         return String.valueOf(Integer.parseInt(tokens[0]) + Integer.parseInt(tokens[1]));
      }
      return "Błąd";
   }
}
```



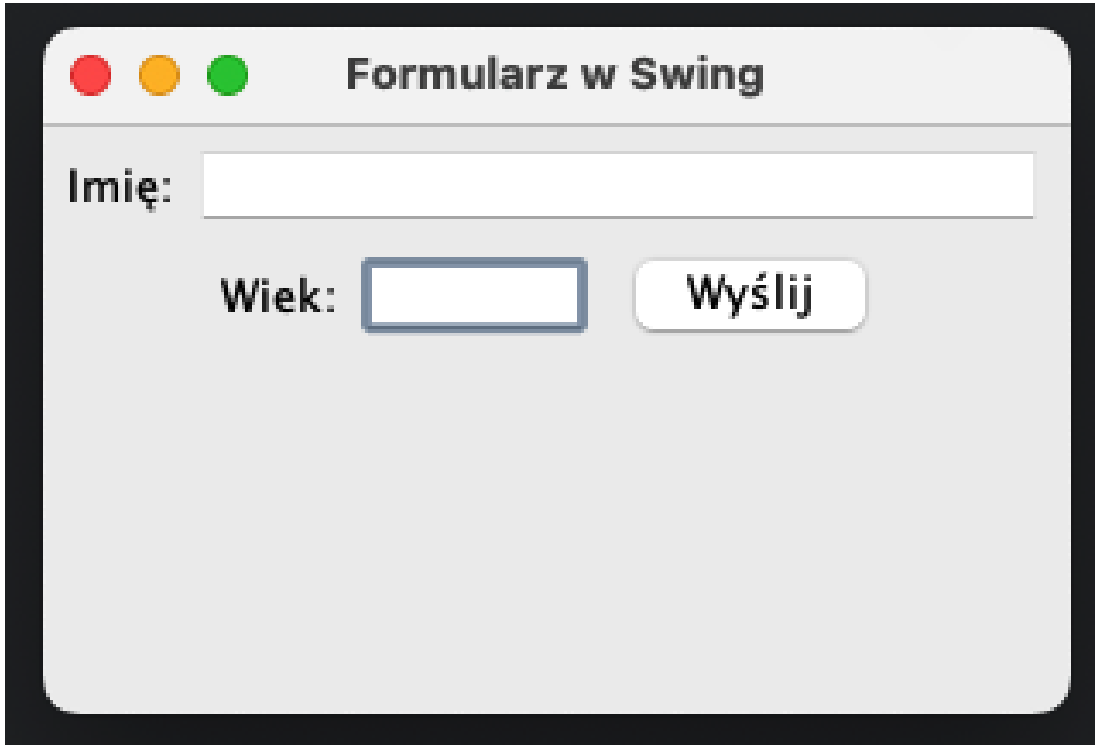
Omówienie:


Tworzymy proste okno z tekstowym polem wyświetlania (TextField) oraz kilkoma przyciskami (Button).
Przyciski pozwalają na wprowadzenie cyfr i znaków operatorów arytmetycznych.
Po kliknięciu przycisku "=", wynik operacji jest wyświetlany na polu tekstowym.
Prosta funkcja calculate rozdziela ciąg wprowadzony przez użytkownika na dwie liczby i oblicza ich sumę.

Prosty formularz w Swing

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingFormExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Formularz w Swing");
        frame.setLayout(new FlowLayout());

        JLabel nameLabel = new JLabel("Imię:");
        JTextField nameField = new JTextField(20);

        JLabel ageLabel = new JLabel("Wiek:");
        JTextField ageField = new JTextField(5);

        JButton submitButton = new JButton("Wyślij");

        frame.add(nameLabel);
        frame.add(nameField);
        frame.add(ageLabel);
        frame.add(ageField);
        frame.add(submitButton);

        submitButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText();
                String age = ageField.getText();
                JOptionPane.showMessageDialog(frame, "Witaj " + name + "! Masz " + age + " lat.");
            }
        });

        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Omówienie



Prosty formularz składający się z kilku dwóch pól tekstowych, przycisku oraz okna dialogowego uruchamianego przyciskiem.

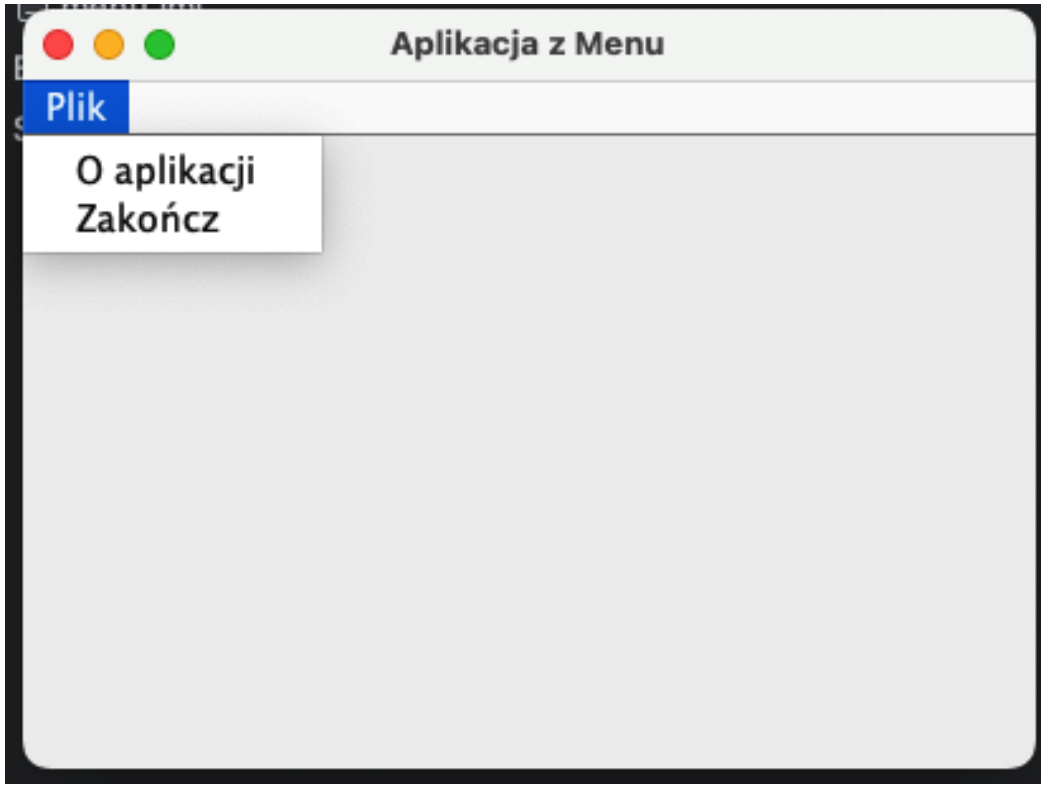
Aplikacja z menu

```
import javax.swing.*;
import java.awt.event.*;

public class SwingMenuExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Aplikacja z Menu");

        JMenuBar menuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("Plik");

        JMenuItem aboutItem = new JMenuItem("O aplikacji");
        JMenuItem exitItem = new JMenuItem("Zakończ");

        aboutItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(frame, "Aplikacja stworzona w Java Swing");
            }
        });
```

```
        exitItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });

        fileMenu.add(aboutItem);
        fileMenu.add(exitItem);
        menuBar.add(fileMenu);

        frame.setJMenuBar(menuBar);

        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Omówienie

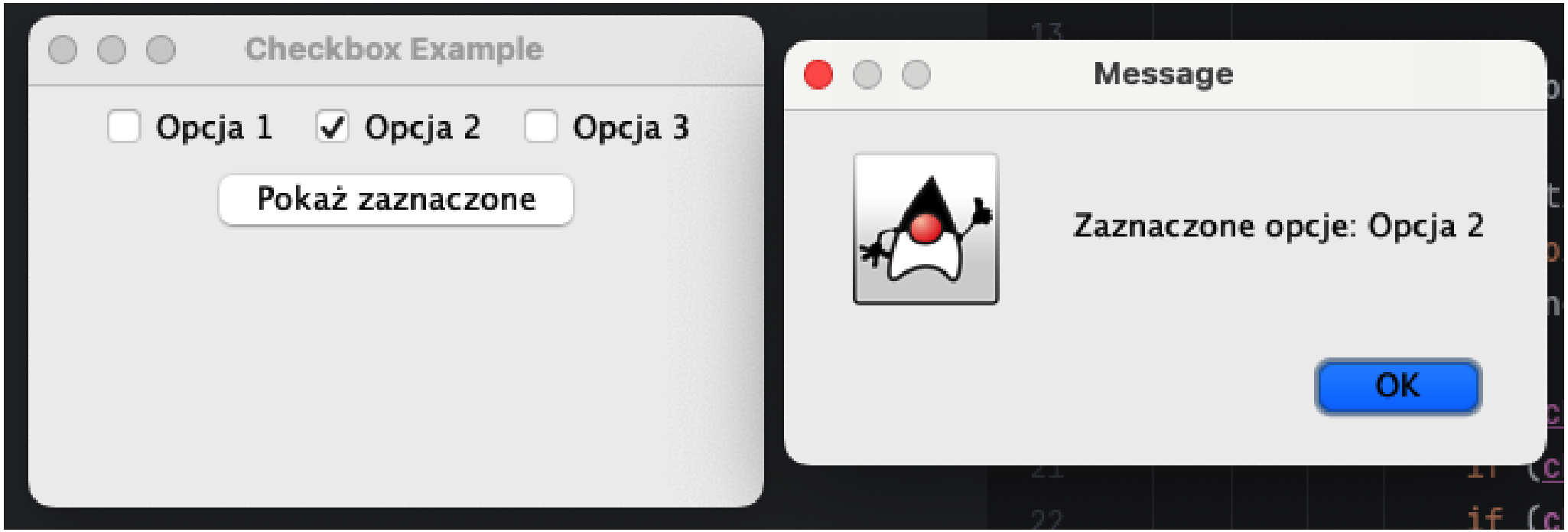
Proste Menu z dwoma „Itemami” oraz dopisaną do nich akcją. Zawiera trzy elementy aplikacji (JMenuBar, Jmenu i JMenuItem).



Aplikacja z wykorzystaniem pól typu checkbox

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingCheckboxExample {
   public static void main(String[] args) {
      JFrame frame = new JFrame("Checkbox Example");
      frame.setLayout(new FlowLayout());

      JCheckBox checkBox1 = new JCheckBox("Opcja 1");
      JCheckBox checkBox2 = new JCheckBox("Opcja 2");
      JCheckBox checkBox3 = new JCheckBox("Opcja 3");

      JButton button = new JButton("Pokaż zaznaczone");

      button.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            StringBuilder message = new StringBuilder("Zaznaczone opcje: ");

            if (checkBox1.isSelected()) message.append("Opcja 1 ");
            if (checkBox2.isSelected()) message.append("Opcja 2 ");
            if (checkBox3.isSelected()) message.append("Opcja 3 ");

            if (message.toString().equals("Zaznaczone opcje: ")) {
               message.append("Brak zaznaczeń.");
            }

            JOptionPane.showMessageDialog(frame, message.toString());
         }
      });

      frame.add(checkBox1);
      frame.add(checkBox2);
      frame.add(checkBox3);
      frame.add(button);

      frame.setSize(300, 200);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

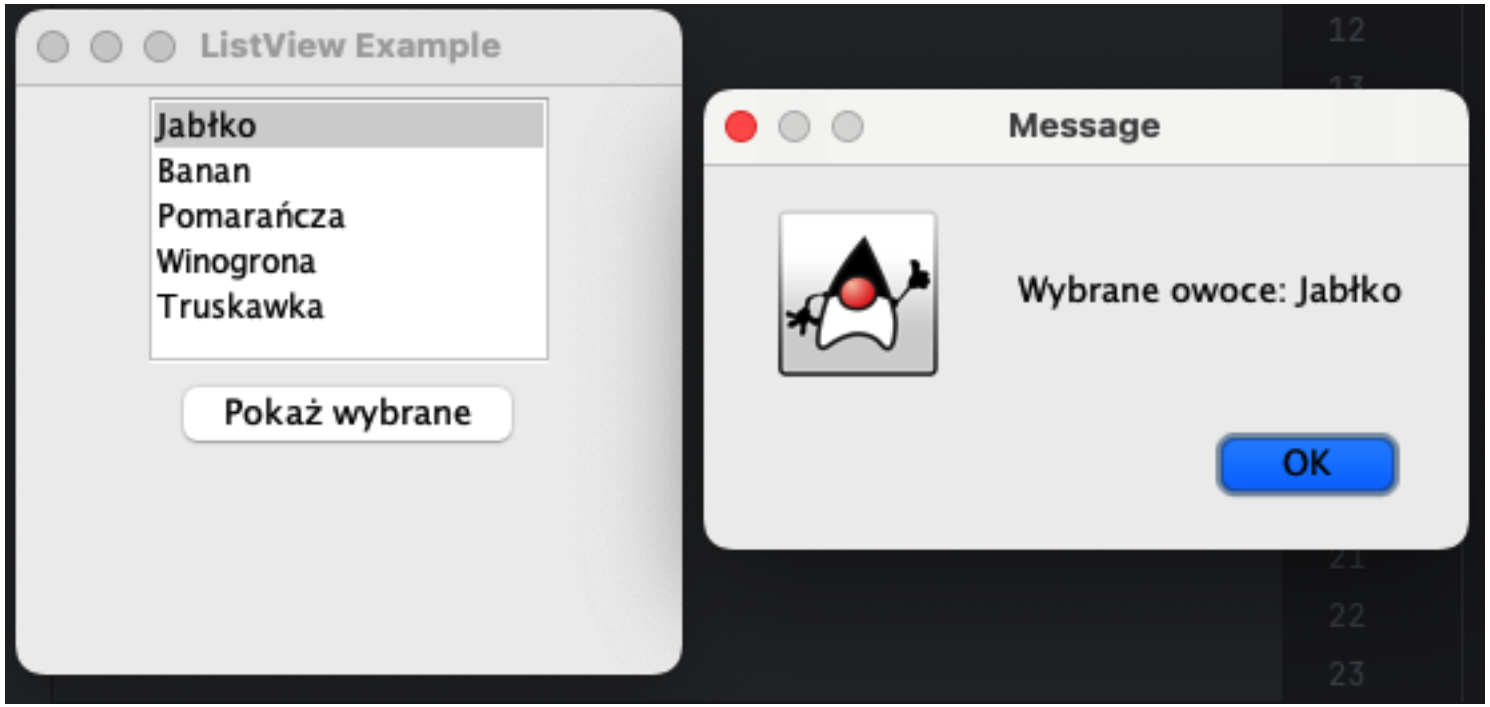
Efekt działania aplikacji



Aplikacja z wykorzystaniem ListView

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingListViewExample {
  public static void main(String[] args) {
    JFrame frame = new JFrame("ListView Example");

    String[] items = { "Jabłko", "Banan", "Pomarańcza", "Winogrona", "Truskawka" };
    JList<String> list = new JList<>(items);
    list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);

    JScrollPane listScroller = new JScrollPane(list);
    listScroller.setPreferredSize(new Dimension(150, 100));

    JButton button = new JButton("Pokaż wybrane");

    button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        ListSelectionModel selectionModel = list.getSelectionModel();
        int[] selectedIndices = list.getSelectedIndices();

        StringBuilder selectedItems = new StringBuilder("Wybrane owoce: ");
        if (selectedIndices.length == 0) {
          selectedItems.append("Brak wybranych.");
        } else {
          for (int i : selectedIndices) {
            selectedItems.append(items[i]).append(" ");
          }
        }

        JOptionPane.showMessageDialog(frame, selectedItems.toString());
      }
    });

    frame.setLayout(new FlowLayout());
    frame.add(listScroller);
    frame.add(button);

    frame.setSize(250, 250);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

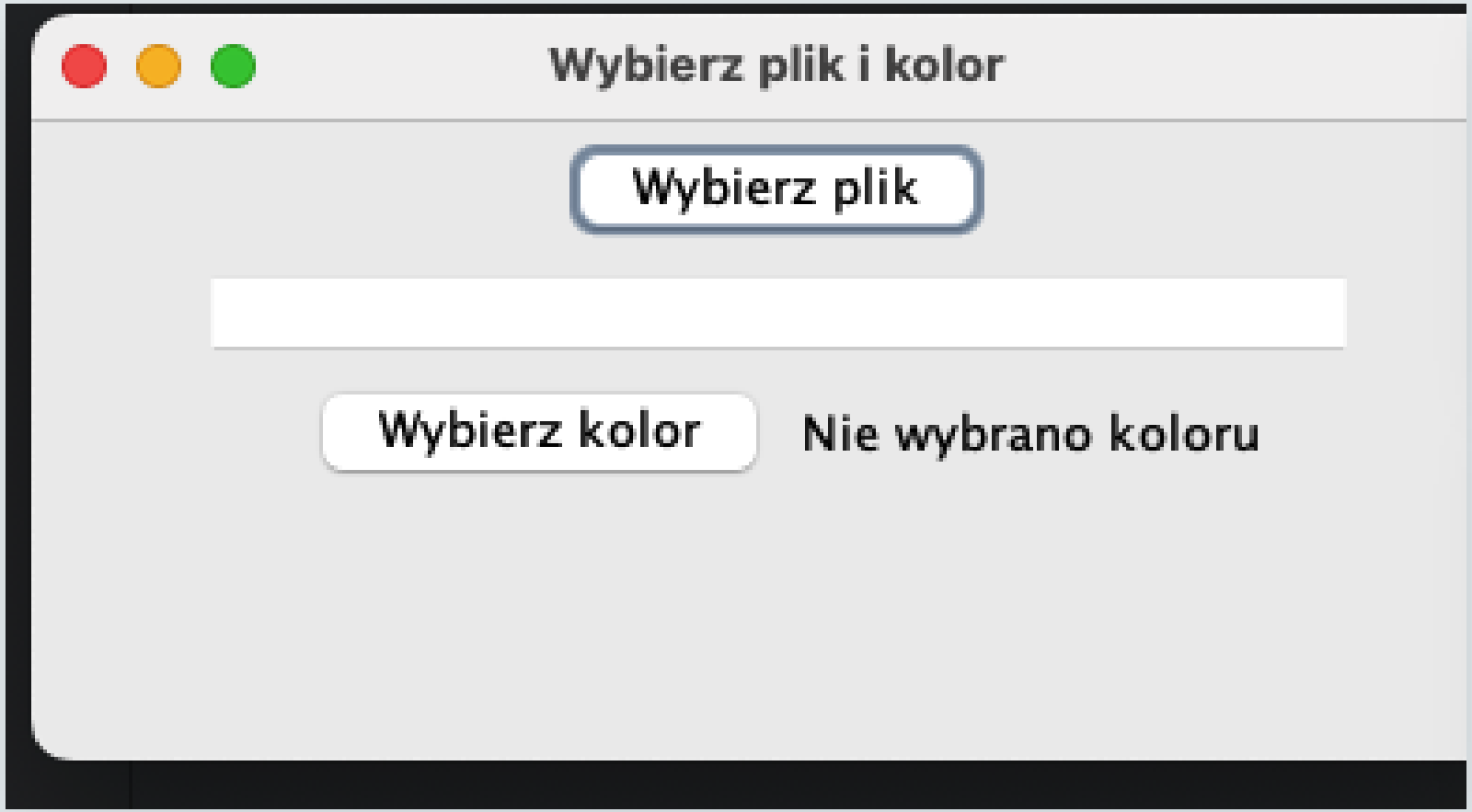
Efekt działania aplikacji

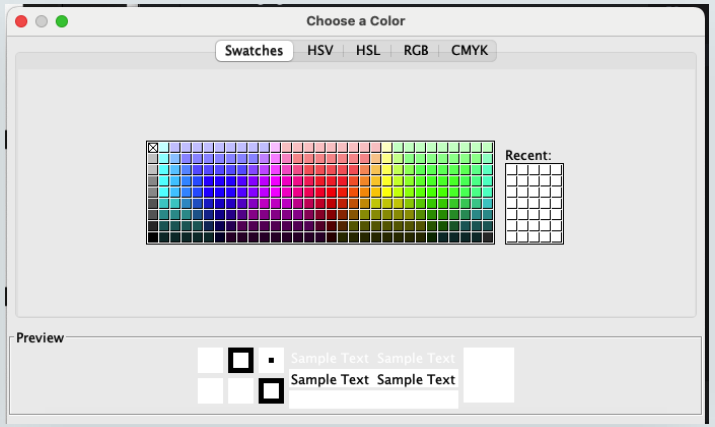


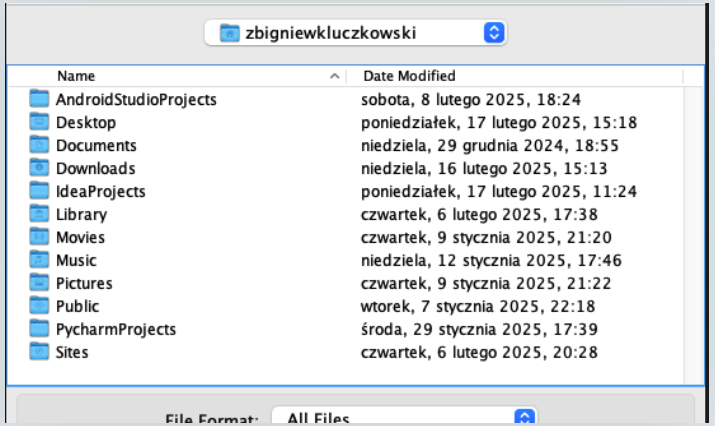
Aplikacja z wykorzystaniem JFileChooser i JColorChooser

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;

public class Wybierzplik {

    private JFrame frame;
    private JButton fileButton;
    private JButton colorButton;
    private JTextField filePathField;
    private JLabel colorLabel;

    public Wybierzplik() {
        frame = new JFrame("Wybierz plik i kolor");
        frame.setLayout(new FlowLayout());
        frame.setSize(400, 200);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        fileButton = new JButton("Wybierz plik");
        colorButton = new JButton("Wybierz kolor");

        filePathField = new JTextField(25);
        filePathField.setEditable(false);

        colorLabel = new JLabel("Nie wybrano koloru");

 fileButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                int result = fileChooser.showOpenDialog(frame);
                if (result == JFileChooser.APPROVE_OPTION) {
                    File file = fileChooser.getSelectedFile();
                    filePathField.setText(file.getAbsolutePath());
                }
            }
        });

        colorButton.addActionListener(new ActionListener()
{
            @Override
            public void actionPerformed(ActionEvent e) {
                Color color = JColorChooser.showDialog(frame,
"Choose a Color", Color.WHITE);
                if (color != null) {
                    colorLabel.setText("Color: " + color.toString());
                    colorLabel.setForeground(color);
                }
            }
        });

        frame.add(fileButton);
        frame.add(filePathField);
        frame.add(colorButton);
        frame.add(colorLabel);
    }

    public void display() {
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Wybierzplik().display();
            }
        });
    }
}
```

Efekt działania programu



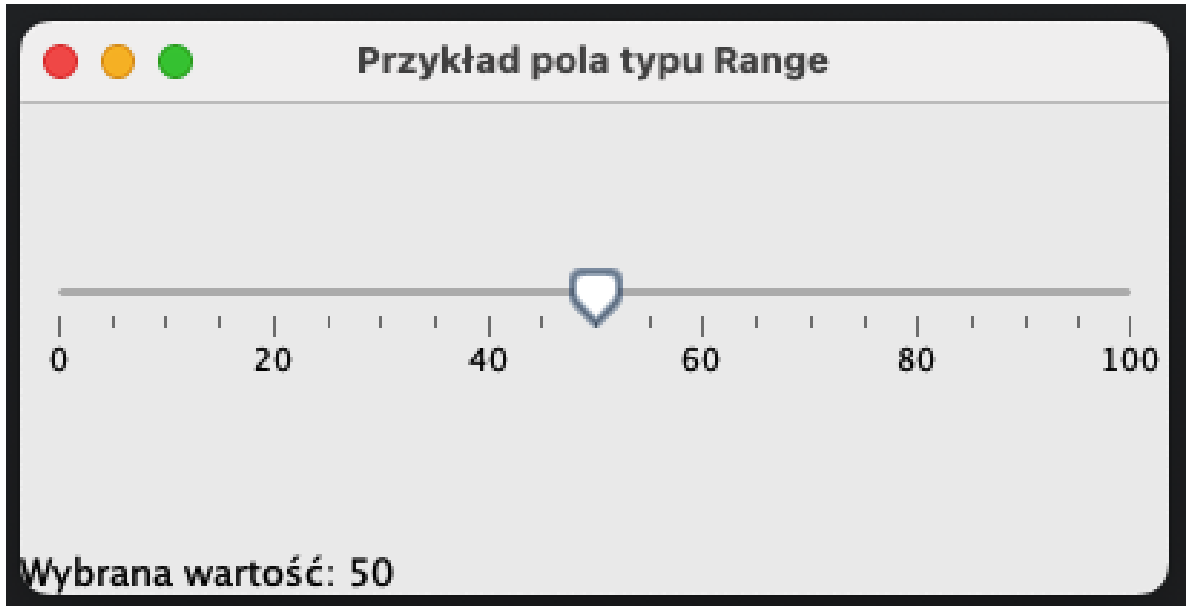




Aplikacja z wykorzystaniem JSlider i Range

```
import javax.swing.*;
import java.awt.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class RangeSlider extends JFrame {
    private JSlider slider;
    private JLabel label;

    public RangeSlider() {
        setTitle("Przykład pola typu Range");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        // Tworzenie suwaka
        slider = new JSlider(0, 100, 50); // Zakres od 0 do 100, domyślna wartość 50
        slider.setMajorTickSpacing(20); // Co 20 jednostek
        slider.setMinorTickSpacing(5); // Co 5 jednostek
        slider.setPaintTicks(true); // Rysuj ticki
        slider.setPaintLabels(true); // Rysuj etykiety


        label = new JLabel("Wybrana wartość: " + slider.getValue());

        slider.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                label.setText("Wybrana wartość: " + slider.getValue());
            }
        });


        setLayout(new BorderLayout());
        add(slider, BorderLayout.CENTER);
        add(label, BorderLayout.SOUTH);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            RangeSlider example = new RangeSlider();
            example.setVisible(true);
        });
    }
}
```

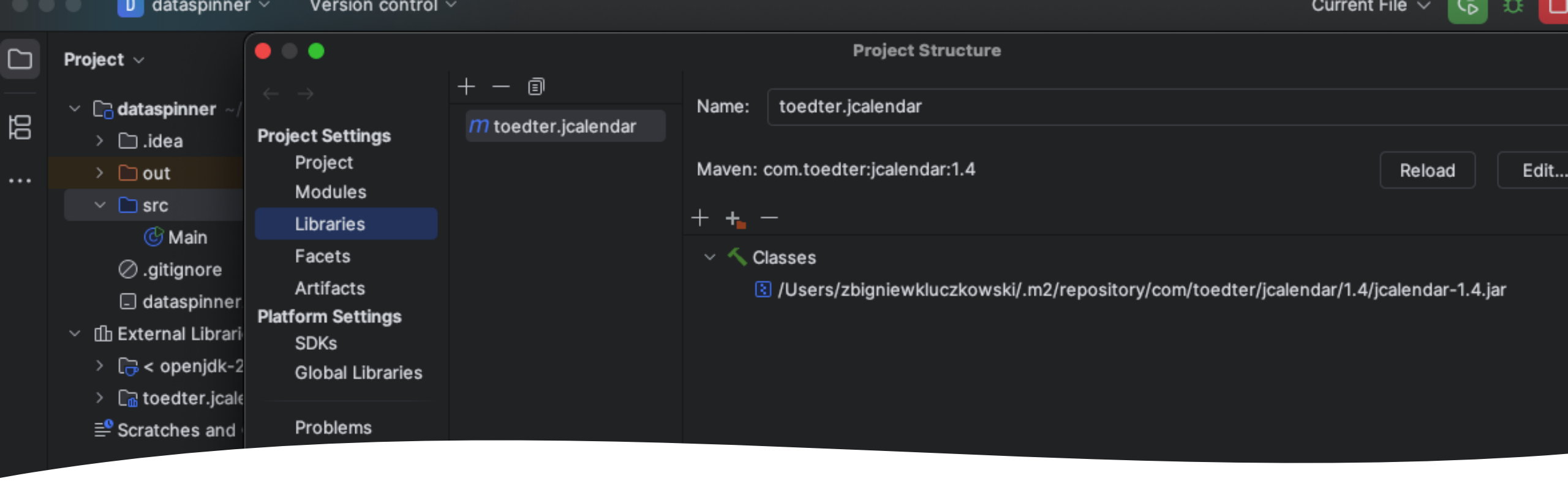
Efekt działania programu



Program pobiera wybraną wartość z suwaka i wypisują w linii tekstu.

Aplikacja z wykorzystaniem JDate, JSpinner i JScrollPane

```
import com.toedter.calendar.JDateChooser;

import javax.swing.*;
import java.awt.*;
import java.util.Date;

public class Main {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new
Main().createUI());
    }

    public void createUI() {
        JFrame frame = new JFrame("Swing z
JDateChooser, JSpinner i JScrollPane");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_C
LOSE);
        frame.setSize(500, 400);
        frame.setLocationRelativeTo(null);


        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel,
BoxLayout.Y_AXIS));

panel.setBorder(BorderFactory.createEmptyBorder(1
5, 15, 15, 15));


        JLabel dateLabel = new JLabel("Wybierz datę:");
        JDateChooser dateChooser = new
JDateChooser();
        dateChooser.setDate(new Date());


        JLabel spinnerLabel = new JLabel("Wybierz liczbę
uczestników:");
        JSpinner spinner = new JSpinner(new
SpinnerNumberModel(1, 1, 100, 1));


        JLabel textLabel = new JLabel("Opis
wydarzenia:");
        JTextArea textArea = new JTextArea(6, 30);
        textArea.setLineWrap(true);
        textArea.setWrapStyleWord(true);
        JScrollPane scrollPane = new
JScrollPane(textArea);


        JButton submitButton = new JButton("Zapisz");


        submitButton.addActionListener(e -> {
            Date selectedDate = dateChooser.getDate();
            int count = (Integer) spinner.getValue();
            String description = textArea.getText();

            JOptionPane.showMessageDialog(frame,
                    "Wybrana data: " + selectedDate +
                            "\nUczestnicy: " + count +
                            "\nOpis: " + description,
                    "Dane",
JOptionPane.INFORMATION_MESSAGE);
        });


        panel.add(dateLabel);
        panel.add(dateChooser);
        panel.add(Box.createRigidArea(new
Dimension(0, 10)));
        panel.add(spinnerLabel);
        panel.add(spinner);
        panel.add(Box.createRigidArea(new
Dimension(0, 10)));
        panel.add(textLabel);
        panel.add(scrollPane);
        panel.add(Box.createRigidArea(new
Dimension(0, 10)));
        panel.add(submitButton);

        frame.add(panel);
        frame.setVisible(true);
    }
}
```



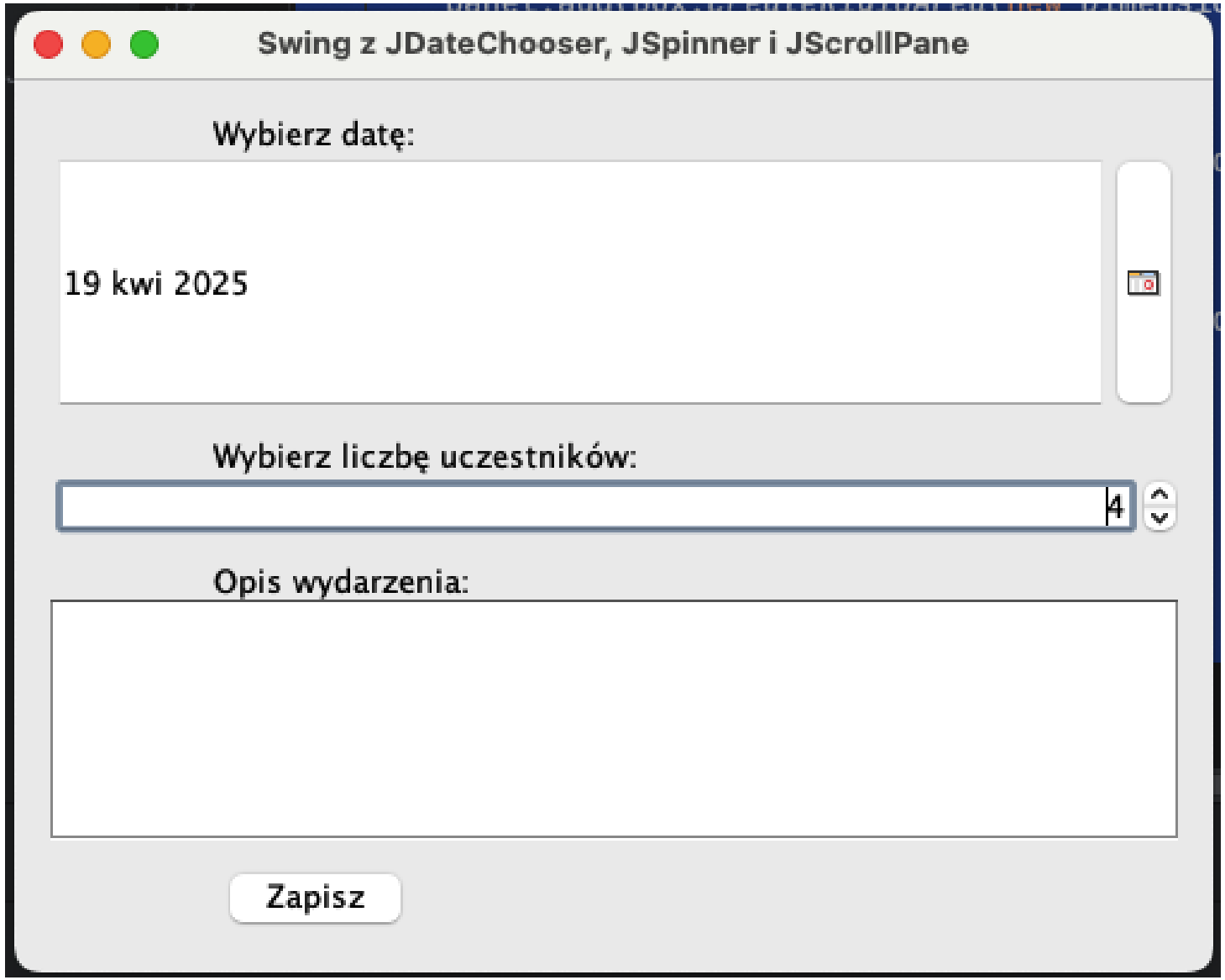
Co wykorzystaliśmy w projekcie?

Pole kalendarza wymaga zainstalowania biblioteki JCalendar. Można znaleźć w postaci ”paczki” i ją ręcznie dodać do projektu. Można również ją zainstalować z repozytorium MAVEN dodając ją do struktury projektu.

MAVEN pozwala nam korzystać z wielu rozszerzeń, które można wykorzystać do budowy aplikacji. Znajdziecie tam odpowiedniki słynnego „Connectora” do baz danych oraz wiele innych bibliotek, które po instalacji znajdą się wewnątrz projektu.

Efekt działania programu

- Pole kalendarza, scroll i pole tekstowe po częsty element aplikacji desktopowej.
- Opisywany przypadek to oczywiście niezbędne minimum.



Aplikacja z przełącznikiem


```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Main {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new Main().createUI());
    }

    public void createUI() {
        JFrame frame = new JFrame("Przełącznik Swing - JToggleButton");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);
        frame.setLocationRelativeTo(null);


        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout());

        JToggleButton toggleButton = new JToggleButton("OFF");

        JLabel statusLabel = new JLabel("Stan: OFF");


        toggleButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (toggleButton.isSelected()) {
                    toggleButton.setText("ON");
                    statusLabel.setText("Stan: ON");
                } else {
                    toggleButton.setText("OFF");
                    statusLabel.setText("Stan: OFF");
                }
            }
        });

        panel.add(toggleButton);
        panel.add(statusLabel);

        frame.add(panel);
        frame.setVisible(true);
    }
}
```

Zdjęcie, RANDOM i … egzamin

```
import java.util.Arrays;
import java.util.Random;

public class Main {

    private JLabel imageLabel;
    private JTextField numberField;
    private File[] imageFiles;
    private File selectedDirectory;
    private final Random random = new Random();

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new Main().createUI());
    }

    public void createUI() {
        JFrame frame = new JFrame("Losowanie zdjęcia z folderu");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(700, 550);
        frame.setLocationRelativeTo(null);

        JPanel panel = new JPanel(new BorderLayout(10, 10));
        panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));


        JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));

        JButton folderButton = new JButton(" Wybierz folder");
        numberField = new JTextField(5);
        JButton drawButton = new JButton(" Losuj zdjęcie");

        topPanel.add(folderButton);
        topPanel.add(new JLabel("Liczba zdjęć do losowania:"));
        topPanel.add(numberField);
        topPanel.add(drawButton);


        imageLabel = new JLabel("Brak zdjęcia", SwingConstants.CENTER);
        imageLabel.setPreferredSize(new Dimension(640, 400));
        imageLabel.setBorder(BorderFactory.createLineBorder(Color.GRAY));

        panel.add(topPanel, BorderLayout.NORTH);
        panel.add(imageLabel, BorderLayout.CENTER);
folderButton.addActionListener(e -> chooseFolder(frame));
        drawButton.addActionListener(e -> drawRandomImage());

        frame.setContentPane(panel);
        frame.setVisible(true);
    }

    private void chooseFolder(Component parent) {
        JFileChooser chooser = new JFileChooser();
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

        int result = chooser.showOpenDialog(parent);
        if (result == JFileChooser.APPROVE_OPTION) {
            selectedDirectory = chooser.getSelectedFile();
            imageFiles = selectedDirectory.listFiles((dir, name) ->
                    name.toLowerCase().matches(".*\\.(jpg|jpeg|png|gif|bmp)"));

            if (imageFiles != null && imageFiles.length > 0) {
                Arrays.sort(imageFiles);
                JOptionPane.showMessageDialog(parent,
                        "Znaleziono " + imageFiles.length + " zdjęć w folderze.",
                        "Sukces", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(parent,
                        "Brak zdjęć w tym folderze.",
                        "Błąd", JOptionPane.ERROR_MESSAGE);
                imageFiles = null;
            }
        }
    }

    private void drawRandomImage() {
        if (imageFiles == null || imageFiles.length == 0) {
            JOptionPane.showMessageDialog(null, "Najpierw wybierz folder ze zdjęciami!", "Brak
danych", JOptionPane.WARNING_MESSAGE);
            return;
        }

        int count;
        try {
            count = Integer.parseInt(numberField.getText().trim());
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Podaj prawidłową liczbę!", "Błąd",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (count < 1 || count > imageFiles.length) {
            JOptionPane.showMessageDialog(null, "Liczba musi być między 1 a " +
imageFiles.length, "Zakres nieprawidłowy", JOptionPane.ERROR_MESSAGE);
            return;
        }

        int index = random.nextInt(count); // losuj od 0 do (count - 1)
        File chosenFile = imageFiles[index];

        ImageIcon icon = new ImageIcon(chosenFile.getAbsolutePath());
        Image scaled = icon.getImage().getScaledInstance(640, 400, Image.SCALE_SMOOTH);
        imageLabel.setIcon(new ImageIcon(scaled));
        imageLabel.setText(null);
    }
}
```

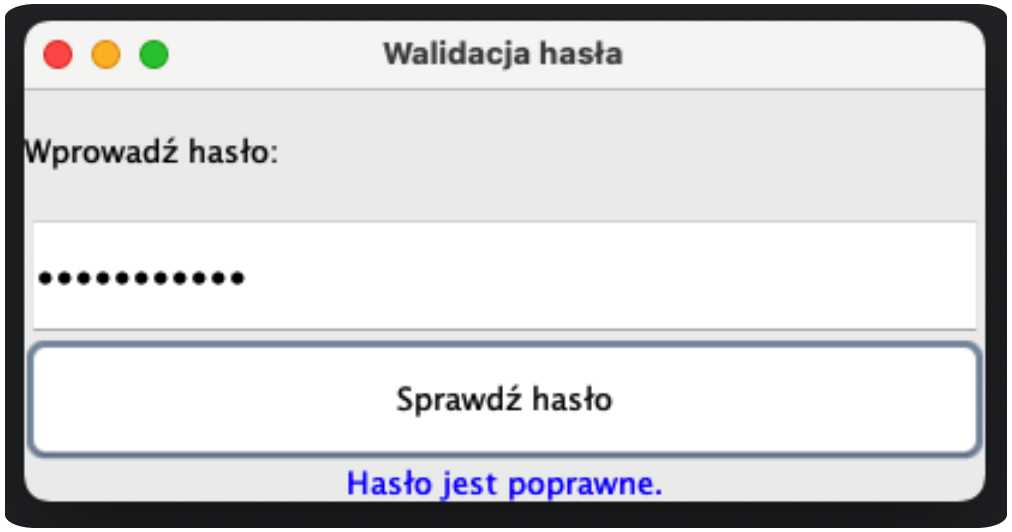
Efekt działania programu

Aplikacja pobiera dane z folderu, użytkownik wpisuje dowolną liczbę a następnie program na podstawie wpisu losuje wybrany element. Liczba odpowiada końcowemu elementowi zbioru. Zatem z 5 zdjęć zostaje wyświetlone jedno przypadkowe. Mechanizm wykorzystuje się w wielu zadaniach egzaminacyjnych, w których należy wykonać aplikację na telefon.

Aplikacja z walidacją hasła

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Main {
  public static void main(String[] args) {

    JFrame frame = new JFrame("Walidacja hasła");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 200);

    JPanel panel = new JPanel();
    panel.setLayout(new GridLayout(3, 1));

    JLabel label = new JLabel("Wprowadź hasło:");
    JPasswordField passwordField = new JPasswordField();

    JButton validateButton = new JButton("Sprawdź hasło");

    JLabel resultLabel = new JLabel("", SwingConstants.CENTER);

    panel.add(label);
    panel.add(passwordField);
    panel.add(validateButton);

    frame.add(panel, BorderLayout.CENTER);
    frame.add(resultLabel, BorderLayout.SOUTH);

    validateButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        String password = new String(passwordField.getPassword());

        if (isValidPassword(password)) {
          resultLabel.setText("Hasło jest poprawne.");
          resultLabel.setForeground(Color.BLUE);
        } else {
          resultLabel.setText("Hasło jest niepoprawne. Musi mieć co najmniej 8 znaków i zawierać cyfrę.");
          resultLabel.setForeground(Color.RED);
        }
      }

      private boolean isValidPassword(String password) {
        return password.length() >= 8 && password.matches(".*\\d.*");
      }
    });

    frame.setVisible(true);
```

Omówienie i efekt działania aplikacji

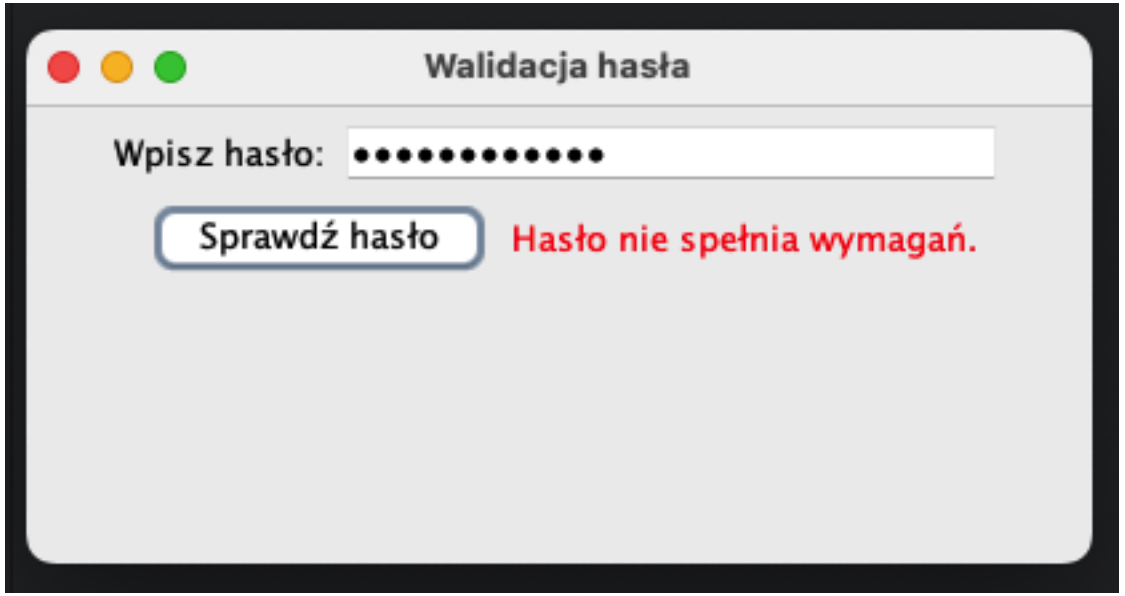


Aplikacja sprawdza długość hasła. Jeżeli jego długość jest równa lub większa niż 8 znaków wtedy pojawia się komunikat, że hasło jest poprawne.

Walidacja hasła z użyciem REGEX

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class PasswordValidationExample extends JFrame implements ActionListener {
    private JPasswordField passwordField;
    private JLabel resultLabel;

    public PasswordValidationExample() {
        setTitle("Walidacja hasła");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        JLabel label = new JLabel("Wpisz hasło:");
        add(label);

        passwordField = new JPasswordField(20);
        add(passwordField);

        JButton validateButton = new JButton("Sprawdź hasło");
        validateButton.addActionListener(this);
        add(validateButton);

        resultLabel = new JLabel("");
        add(resultLabel);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String password = new String(passwordField.getPassword());
        if (validatePassword(password)) {
            resultLabel.setText("Hasło jest poprawne.");
            resultLabel.setForeground(Color.GREEN);
        } else {
            resultLabel.setText("Hasło nie spełnia wymagań.");
            resultLabel.setForeground(Color.RED);
        }
    }

    private boolean validatePassword(String password) {
        // REGEX: Co najmniej 8 znaków, jedna wielka litera, jedna cyfra
        String regex = "^(?=.*[A-Z])(?=.*\\d).{8,}$";
        return password.matches(regex);
    }

    public static void main(String[] args) {
        new PasswordValidationExample();
    }
}
```

Efekt działania programu



Program pobiera od użytkownika hasło.
Będzie prawidłowe jeżeli jego składnia będzie zawierać:

- minimum 8 znaków;
- co najmniej jedną wielką literę;
- co najmniej jedną cyfrę.

Zdarzenia na mysz

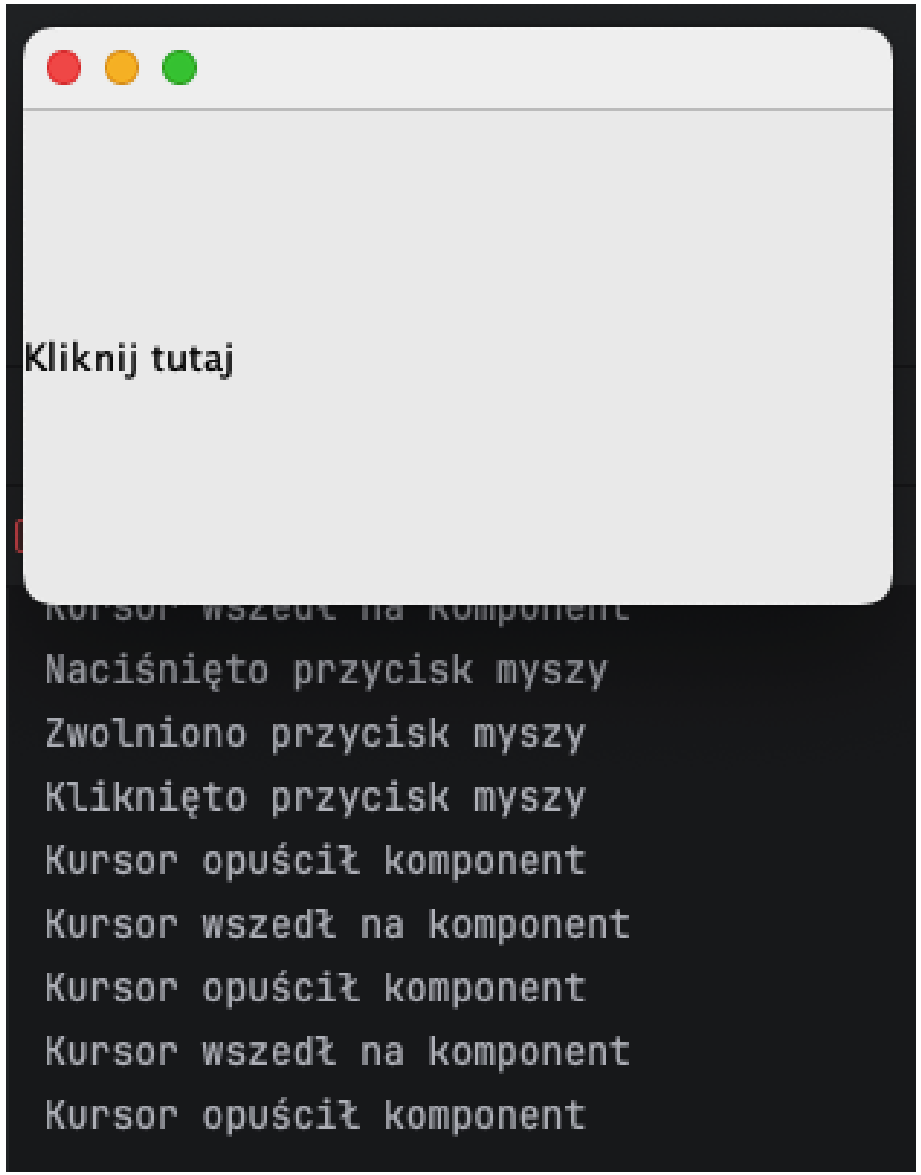
Metoda	Opis
mouseClicked(MouseEvent e)	Wywoływana po kliknięciu (naciśnięciu i zwolnieniu) przycisku myszy.
mousePressed(MouseEvent e)	Wywoływana po naciśnięciu przycisku myszy.
mouseReleased(MouseEvent e)	Wywoływana po zwolnieniu przycisku myszy.
mouseEntered(MouseEvent e)	Wywoływana, gdy kursor wchodzi na komponent.
mouseExited(MouseEvent e)	Wywoływana, gdy kursor opuszcza komponent.

Przykład?

Kilka zdarzeń na mysz ...

```
import javax.swing.*;
import java.awt.event.*;

public class MouseListenerExample extends JFrame
implements MouseListener {
  public MouseListenerExample() {
    JLabel label = new JLabel("Kliknij tutaj");
    label.addMouseListener(this);
    add(label);

    setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
  }

  public void mouseClicked(MouseEvent e) {
    System.out.println("Kliknięto przycisk myszy");
  }

  public void mousePressed(MouseEvent e) {
    System.out.println("Naciśnięto przycisk myszy");
  }

  public void mouseReleased(MouseEvent e) {
    System.out.println("Zwolniono przycisk myszy");
  }

  public void mouseEntered(MouseEvent e) {
    System.out.println("Kursor wszedł na komponent");
  }

  public void mouseExited(MouseEvent e) {
    System.out.println("Kursor opuścił komponent");
  }

  public static void main(String[] args) {
    new MouseListenerExample();
  }
}
```

Efekt działania programu



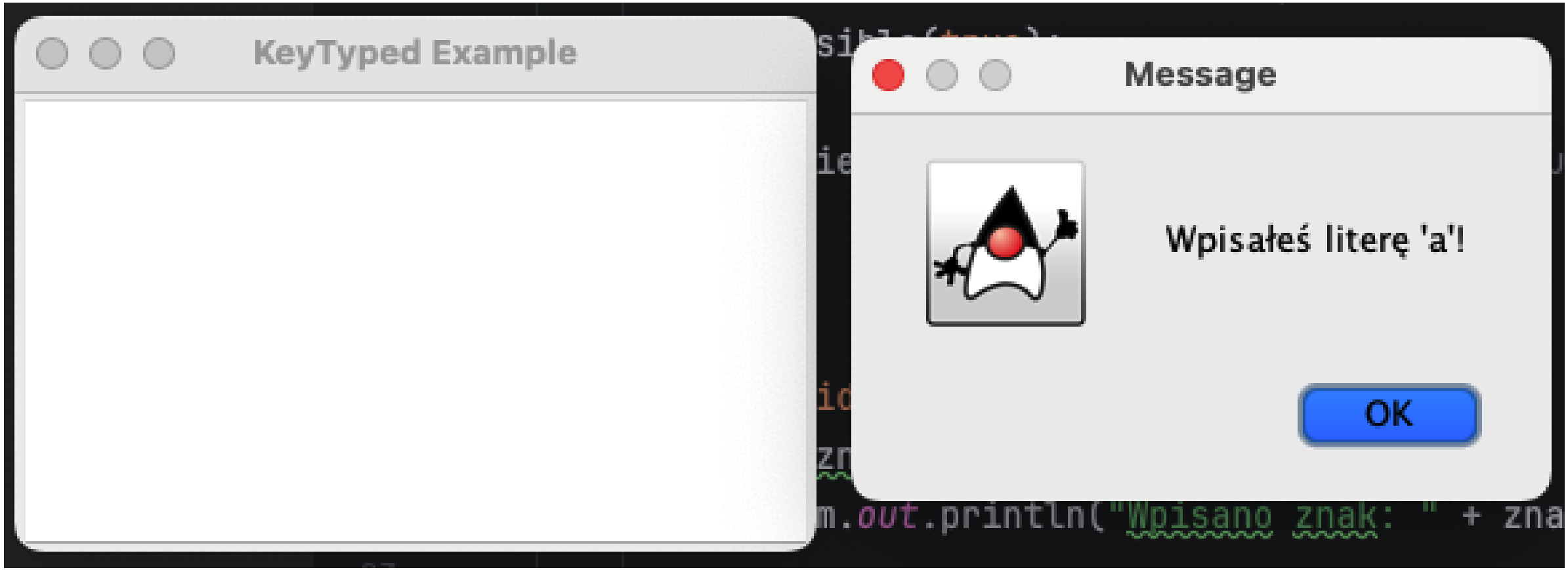
Zdarzenia na klawiaturę

Metoda	Opis
keyPressed(KeyEvent e)	Wywoływana po naciśnięciu klawisza.
keyReleased(KeyEvent e)	Wywoływana po zwolnieniu klawisza.
keyTyped(KeyEvent e)	Wywoływana po wprowadzeniu znaku (np. litery).

Przydatne stałe w KeyEvent:
KeyEvent.VK_ENTER – klawisz Enter
KeyEvent.VK_ESCAPE – klawisz Escape
KeyEvent.VK_SPACE – spacja
KeyEvent.VK_UP, KeyEvent.VK_DOWN, KeyEvent.VK_LEFT, KeyEvent.VK_RIGHT – strzałki
KeyEvent.VK_A, KeyEvent.VK_Z – litery
KeyEvent.VK_0, KeyEvent.VK_9 – cyfry

Przykład zdarzenia na klawiaturę

```
import javax.swing.*;
import java.awt.event.*;


public class KeyTypedExample extends JFrame
implements KeyListener {
    public KeyTypedExample() {
        JTextField textField = new JTextField();
        textField.addKeyListener(this);
        add(textField);


        setTitle("KeyTyped Example");
        setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null); // Center the window
        setVisible(true);


        textField.requestFocusInWindow();
}


    @Override
    public void keyTyped(KeyEvent e) {
        char znak = e.getKeyChar();
        System.out.println("Wpisano znak: " + znak);
        if (znak == 'a') {
            JOptionPane.showMessageDialog(this, "Wpisałeś
literę 'a'!");
        }
    }


    @Override
    public void keyPressed(KeyEvent e) {


    }


    @Override
    public void keyReleased(KeyEvent e) {


    }


    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new
KeyTypedExample());
    }
}
```



Efekt działania programu

Po wpisaniu litery „a” z klawiatury uruchamia się drugie okienko z komunikatem.

GRID w języku JAVA

```
import javax.swing.*;
import java.awt.*;
```

Importujesz

Siatka 2x2

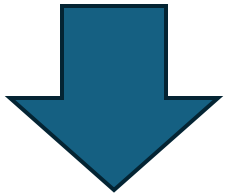
```
JPanel panel = new JPanel(new GridLayout(2, 2));
```

```
panel.add(new JButton("Przycisk 1"));
panel.add(new JButton("Przycisk 2"));
panel.add(new JButton("Przycisk 3"));
panel.add(new JButton("Przycisk 4"));
```

Dodajesz przyciski

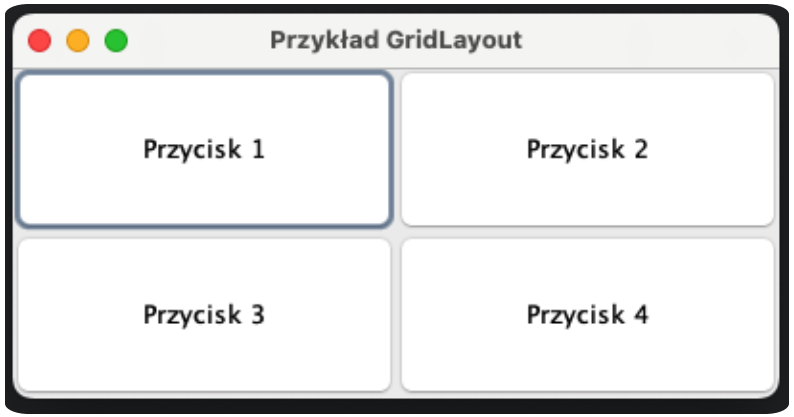
GRID w języku JAVA

Dodajesz do ramki



- JFrame frame = new JFrame("Przykład GridLayout");
- frame.add(panel);
- frame.setSize(400, 200);
- frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
- frame.setVisible(true);

Cały przykład




```
import javax.swing.*;
import java.awt.*;

public class GridExample {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Przykład GridLayout");
        frame.setLayout(new GridLayout(2, 2)); // 2 wiersze i 2 kolumny

        frame.add(new JButton("Przycisk 1"));
        frame.add(new JButton("Przycisk 2"));
        frame.add(new JButton("Przycisk 3"));
        frame.add(new JButton("Przycisk 4"));

        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

GRID dynamiczny

```
import javax.swing.*;
import java.awt.*;

public class GridExample {
   public static void main(String[] args) {
      JFrame frame = new JFrame("Przykład GridLayout z 9
przyciskami");
      frame.setLayout(new GridLayout(3, 3)); // 3 wiersze i 3
kolumny

      for (int i = 1; i <= 9; i++) {
         frame.add(new JButton("Przycisk " + i));
      }

      frame.setSize(400, 400);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

Parametry konstruktora GRIDLayout

Konstruktor GridLayout umożliwia dostosowanie siatki:

new GridLayout(int rows, int cols, int hgap, int vgap);

- **rows**: Liczba wierszy.
- **cols**: Liczba kolumn.
- **hgap**: Odstęp poziomy między komórkami (w pikselach).
- **vgap**: Odstęp pionowy między komórkami (w pikselach).

Przykład z odstępami:

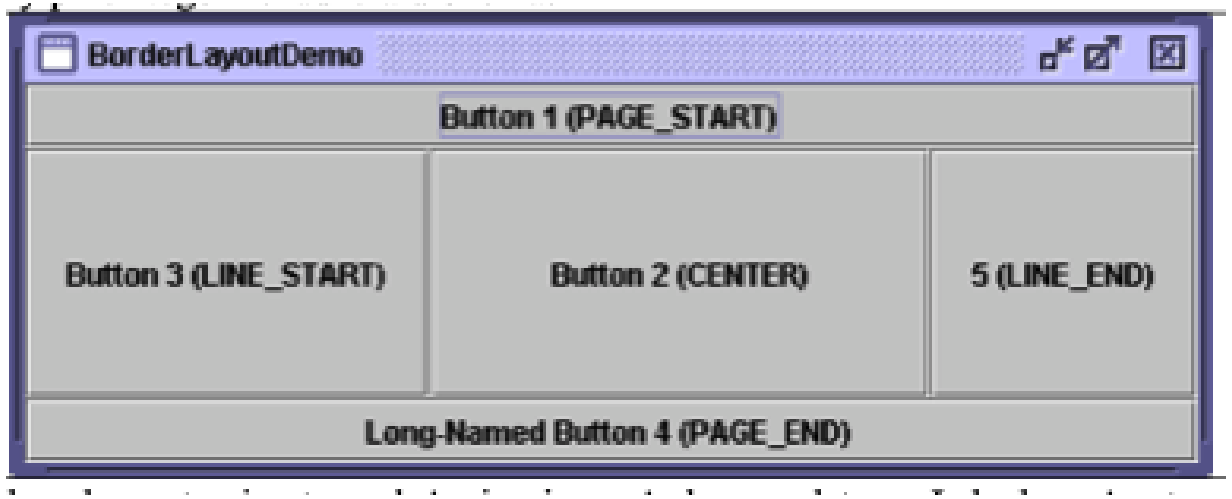
JPanel panel = new JPanel(new GridLayout(2, 3, 10, 15));

Siatka z 2 wierszami, 3 kolumnami, 10 pikseli odstępu poziomego i 15 pikseli odstępu pionowego.

Porównanie z innymi układami

Menedżer Układu	Kluczowe Cechy
GridLayout	Równe rozmiary komórek, siatka.
FlowLayout	Komponenty ułożone w wierszach od lewej do prawej.
BorderLayout	Podział na pięć regionów (North, South, East, West, Center).
BoxLayout	Układ w jednej osi (pionowej lub poziomej).
GridBagLayout	Zaawansowany i elastyczny układ siatki.

Inne rodzaje layout:



BorderLayout

Wygląd zapewniany przez tego menadżera ułożenia:

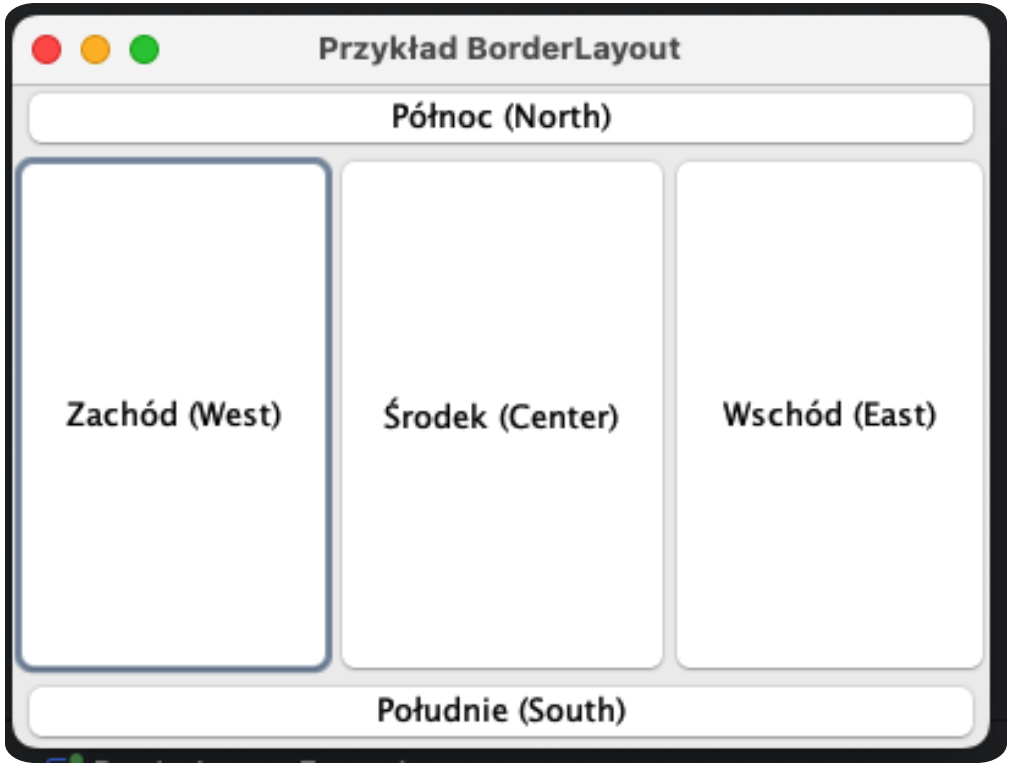
Domyślnie pomiędzy elementami w tym ułożeniu nie ma żadnego odstępu.

Jednak można te odstępy zdefiniować, podając wartość odstępów w poziomie i pionie (wyrażoną w punktach):

1. 2. w konstruktorze BorderLayout(int horizontalGap, int verticalGap),

albo w metodach void setHgap(int); void setVgap(int).

Sposób użycia BorderLayout



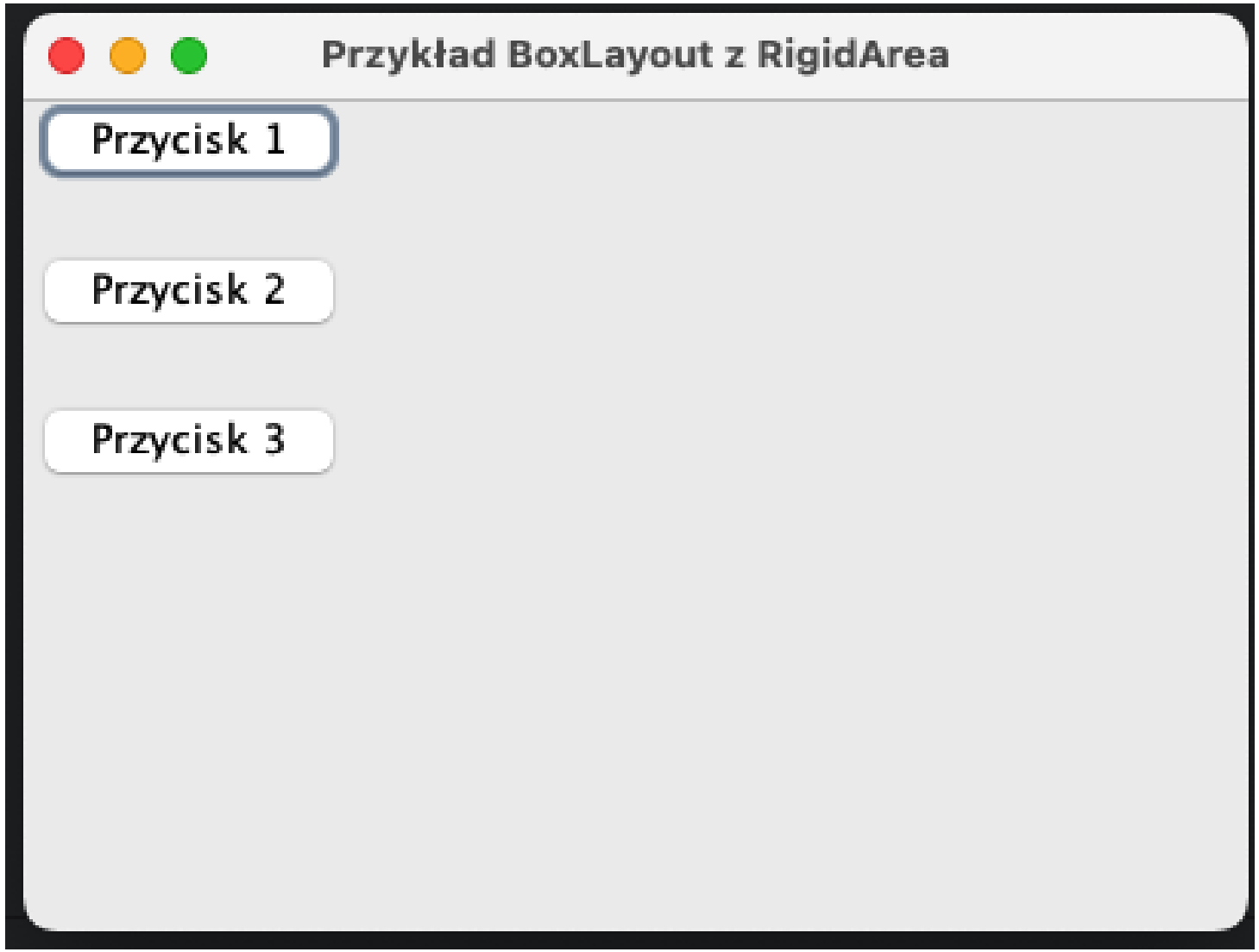

```
import javax.swing.*;
import java.awt.*;

public class BorderLayoutExample {
   public static void main(String[] args) {

      JFrame frame = new JFrame("Przykład BorderLayout");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(400, 300);

      frame.setLayout(new BorderLayout());

      JButton northButton = new JButton("Północ (North)");
      JButton southButton = new JButton("Południe (South)");
      JButton eastButton = new JButton("Wschód (East)");
      JButton westButton = new JButton("Zachód (West)");
      JButton centerButton = new JButton("Środek (Center)");

      frame.add(northButton, BorderLayout.NORTH);
      frame.add(southButton, BorderLayout.SOUTH);
      frame.add(eastButton, BorderLayout.EAST);
      frame.add(westButton, BorderLayout.WEST);
      frame.add(centerButton, BorderLayout.CENTER);

      frame.setVisible(true);
   }
}
```

BOXLayout

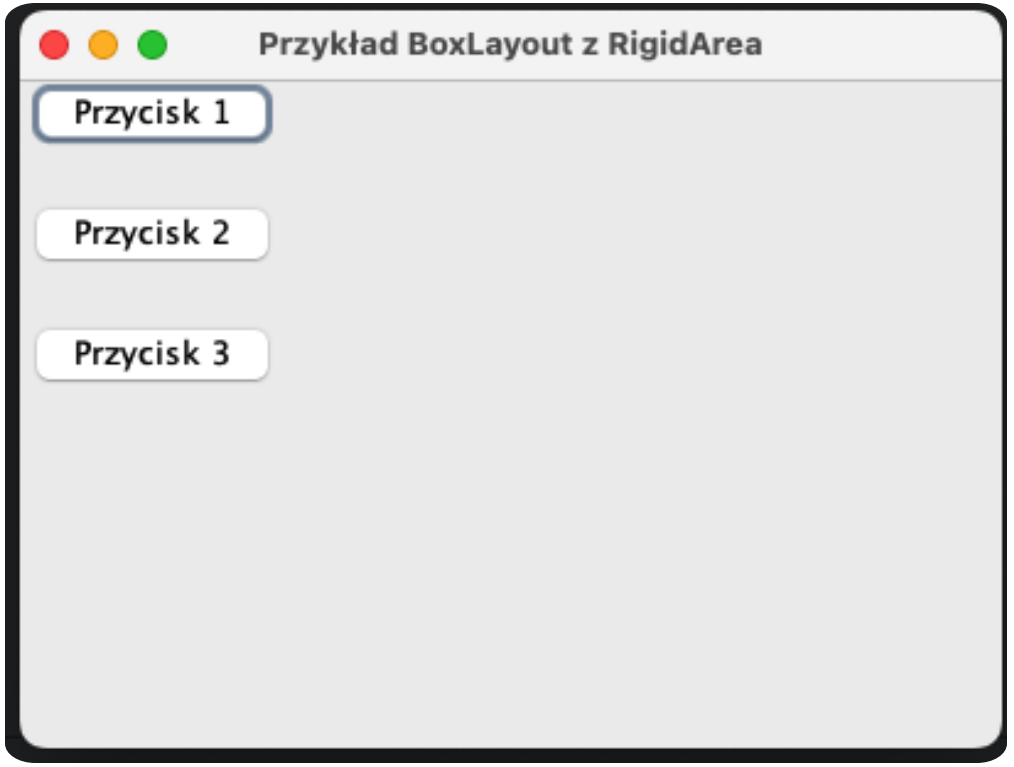
Wygląd zapewniany przez tego menadżera ułożenia to: albo pojedyncza kolumna, albo pojedynczy wiersz

komponentów. BoxLayout respektuje maksymalny żądany rozmiar komponentu oraz pozwala na

wyrównywanie komponentów.



Przykładowy kod BoxLayout



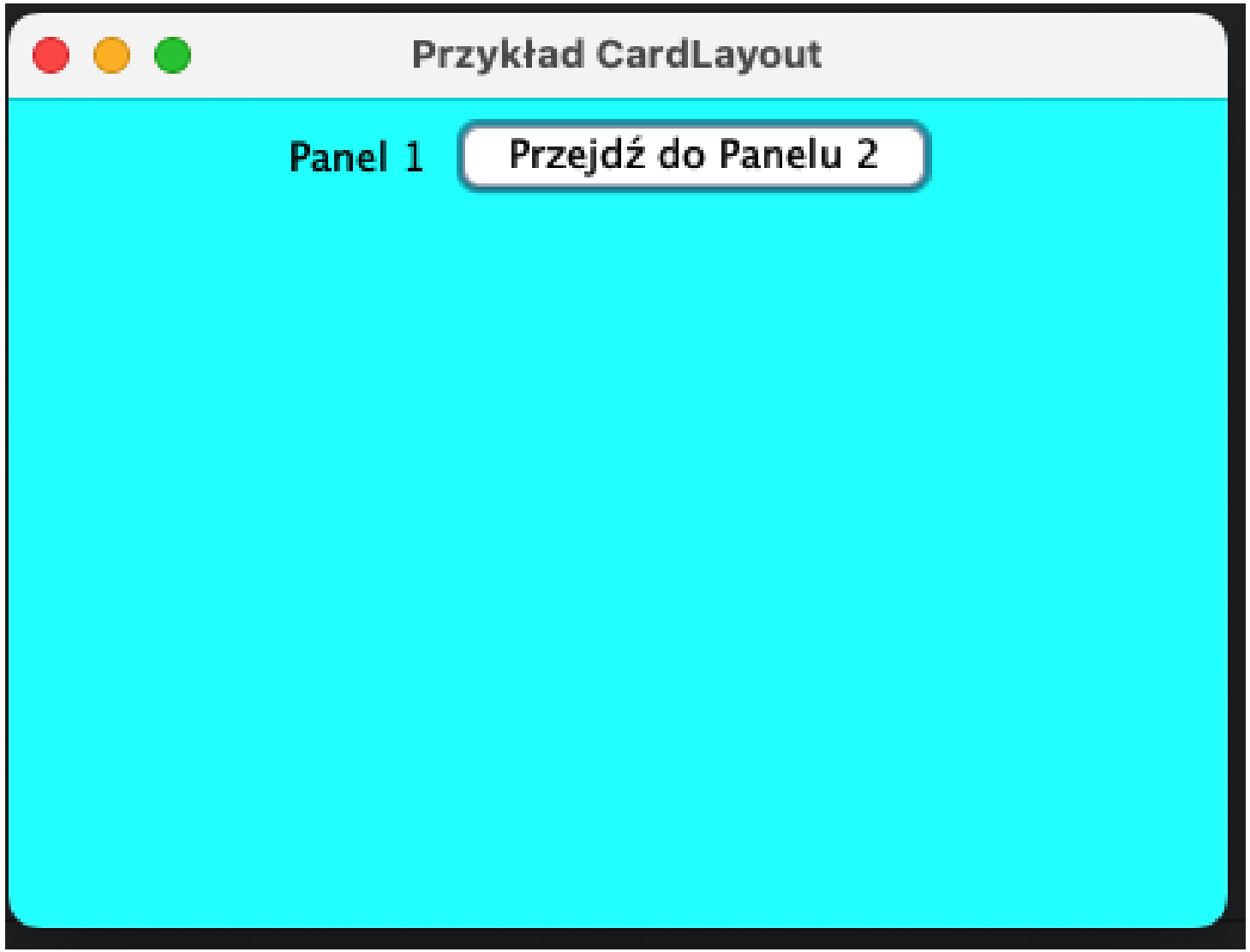

```
import javax.swing.*;
import java.awt.*;

public class BoxLayoutRigidArea {
   public static void main(String[] args) {

      JFrame frame = new JFrame("Przykład BoxLayout z RigidArea");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(400, 300);

      JPanel panel = new JPanel();
      panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

      JButton button1 = new JButton("Przycisk 1");
      JButton button2 = new JButton("Przycisk 2");
      JButton button3 = new JButton("Przycisk 3");

          panel.add(button1);
      panel.add(Box.createRigidArea(new Dimension(0, 20)));
      panel.add(button2);
      panel.add(Box.createRigidArea(new Dimension(0, 20)));
      panel.add(button3);

      frame.add(panel);

      frame.setVisible(true);
   }
}
```

Card Layout

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CardLayoutApps {
   public static void main(String[] args) {

      JFrame frame = new JFrame("Przykład CardLayout");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(400, 300);


      CardLayout cardLayout = new CardLayout();
      JPanel mainPanel = new JPanel(cardLayout);


      JPanel panel1 = new JPanel();
      panel1.setBackground(Color.CYAN);
      panel1.add(new JLabel("Panel 1"));
      JButton goToPanel2Button = new JButton("Przejdź do Panelu 2");
      panel1.add(goToPanel2Button);


      JPanel panel2 = new JPanel();
      panel2.setBackground(Color.PINK);
      panel2.add(new JLabel("Panel 2"));
      JButton goToPanel1Button = new JButton("Przejdź do Panelu 1");
      panel2.add(goToPanel1Button);

      mainPanel.add(panel1, "Panel1");
      mainPanel.add(panel2, "Panel2");


      goToPanel2Button.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            cardLayout.show(mainPanel, "Panel2");
         }
      });

      goToPanel1Button.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            cardLayout.show(mainPanel, "Panel1");
         }
      });


      frame.add(mainPanel);

       frame.setVisible(true);
   }
}
```

Efekt działania programu

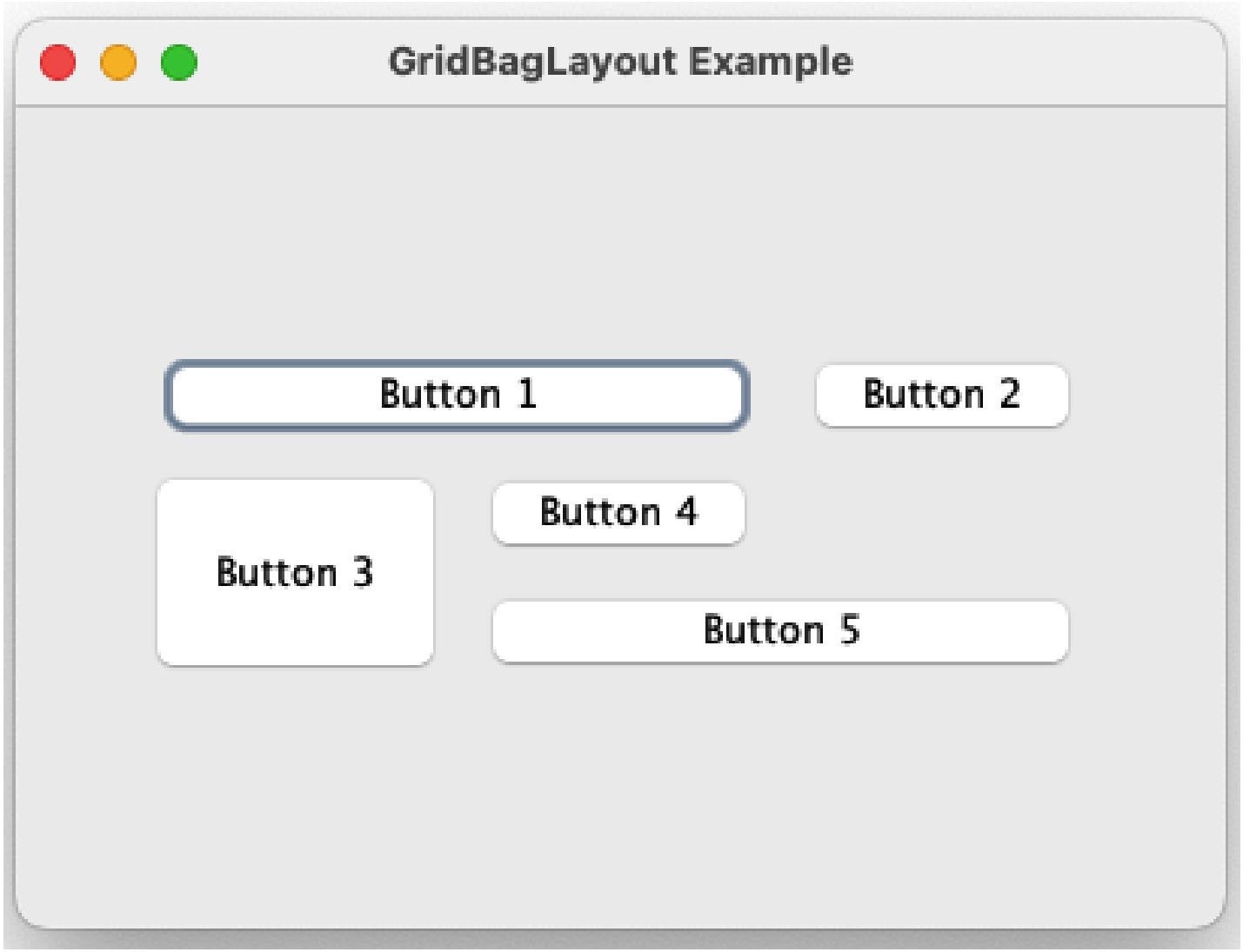
Aplikacja zawiera dwa przełączane panele w oknie. Na każdym z nich może znajdować się dowolna zawartość.



GridBagLayout – wszechstronne narzędzie

```
import javax.swing.*;
import java.awt.*;

public class Main {
   public static void main(String[] args) {
      SwingUtilities.invokeLater(() -> {
         JFrame frame = new JFrame("GridBagLayout
Example");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLO
SE);
         frame.setSize(400, 300);
         frame.setLayout(new GridBagLayout());

         GridBagConstraints gbc = new
GridBagConstraints();
         gbc.insets = new Insets(5, 5, 5, 5); // Marginesy
między komponentami

         JButton button1 = new JButton("Button 1");
         JButton button2 = new JButton("Button 2");
         JButton button3 = new JButton("Button 3");
         JButton button4 = new JButton("Button 4");
         JButton button5 = new JButton("Button 5");

         // Konfiguracja przycisków w siatce

         // Button 1 - zajmuje 2 kolumny
         gbc.gridx = 0;
         gbc.gridy = 0;
         gbc.gridwidth = 2;
         gbc.fill = GridBagConstraints.HORIZONTAL;
         frame.add(button1, gbc);

         // Button 2
         gbc.gridx = 2;
         gbc.gridy = 0;
         gbc.gridwidth = 1;
         frame.add(button2, gbc);

         // Button 3 - zajmuje 3 wiersze w jednej kolumnie
         gbc.gridx = 0;
         gbc.gridy = 1;
         gbc.gridheight = 3;
         gbc.fill = GridBagConstraints.VERTICAL;
         frame.add(button3, gbc);

         // Button 4
         gbc.gridx = 1;
         gbc.gridy = 1;
         gbc.gridheight = 1;
         frame.add(button4, gbc);

         // Button 5 - zajmuje 2 kolumny
         gbc.gridx = 1;
         gbc.gridy = 2;
         gbc.gridwidth = 2;
         gbc.fill = GridBagConstraints.HORIZONTAL;
         frame.add(button5, gbc);

         frame.setVisible(true);
      });
   }
}
```

Efekt działania programu

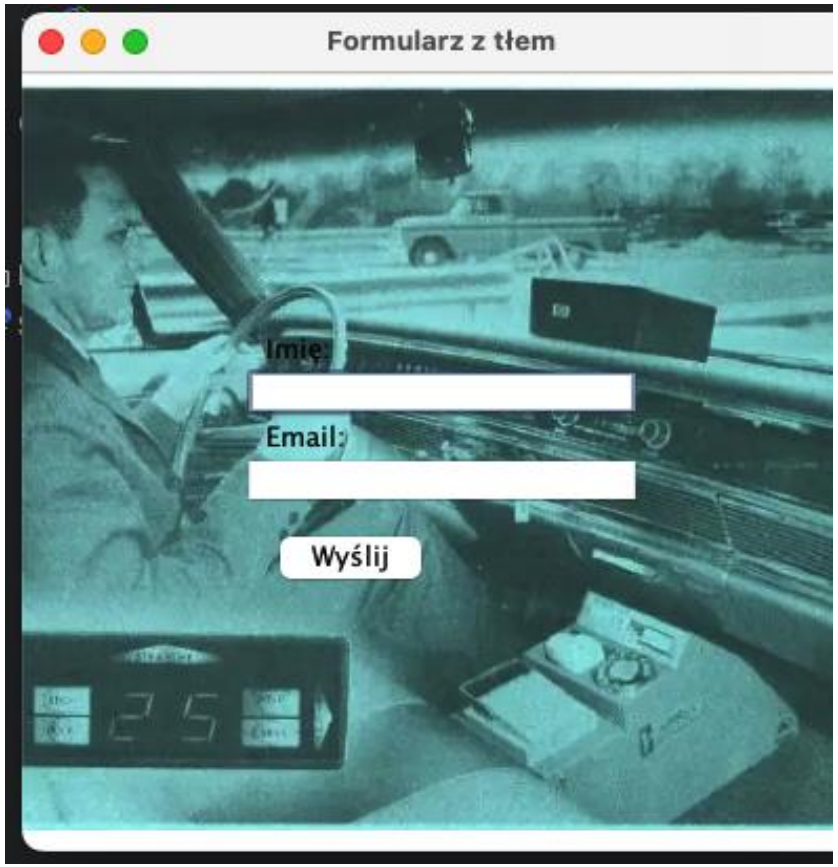


Właściwości GridBagLayout

Metoda	Opis
gridx, gridy	Określają pozycję w siatce.
gridwidth, gridheight	Definiują, ile kolumn lub wierszy komponent zajmuje.
fill	Określa sposób rozszerzania komponentu (HORIZONTAL, VERTICAL, BOTH, NONE).
weightx, weighty	Proporcje rozciągania komponentu.
insets	Ustawia marginesy między komponentami.
GridBagConstraints.BOTH	Rozciąga komponent w obu kierunkach.
insets	Margines wokół komponentu (new Insets(top, left, bottom, right)).
GridBagConstraints.CENTER (domyślnie)	Wyśrodkowanie.
GridBagConstraints.NORTH, SOUTH, EAST, WEST	Wyrównanie do danej krawędzi.
GridBagConstraints.NORTHEAST, NORTHWEST, SOUTHEAST, SOUTHWEST	Wyrównanie do rogu.

Obrazek jako tło okna w programie

```
import javax.swing.*;
import java.awt.*;

class BackgroundImageForm {
   public static void main(String[] args) {

      JFrame frame = new JFrame("Formularz z tłem");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(400, 400);


      ImageIcon backgroundIcon = new
ImageIcon("obrazek.png");
      JLabel backgroundLabel = new JLabel(backgroundIcon);
      backgroundLabel.setLayout(new GridBagLayout());
      frame.setContentPane(backgroundLabel);

      JPanel panel = new JPanel();
      panel.setOpaque(false);
      panel.setLayout(new BoxLayout(panel,
BoxLayout.Y_AXIS));

      JLabel nameLabel = new JLabel("Imię:");
      JTextField nameField = new JTextField(15);
      JLabel emailLabel = new JLabel("Email:");
      JTextField emailField = new JTextField(15);
      JButton submitButton = new JButton("Wyślij");
      panel.add(nameLabel);
      panel.add(nameField);
      panel.add(emailLabel);
      panel.add(emailField);
      panel.add(Box.createRigidArea(new Dimension(0, 10)));
      panel.add(submitButton);


      backgroundLabel.add(panel, new GridBagConstraints());

      frame.setVisible(true);
   }
}
```

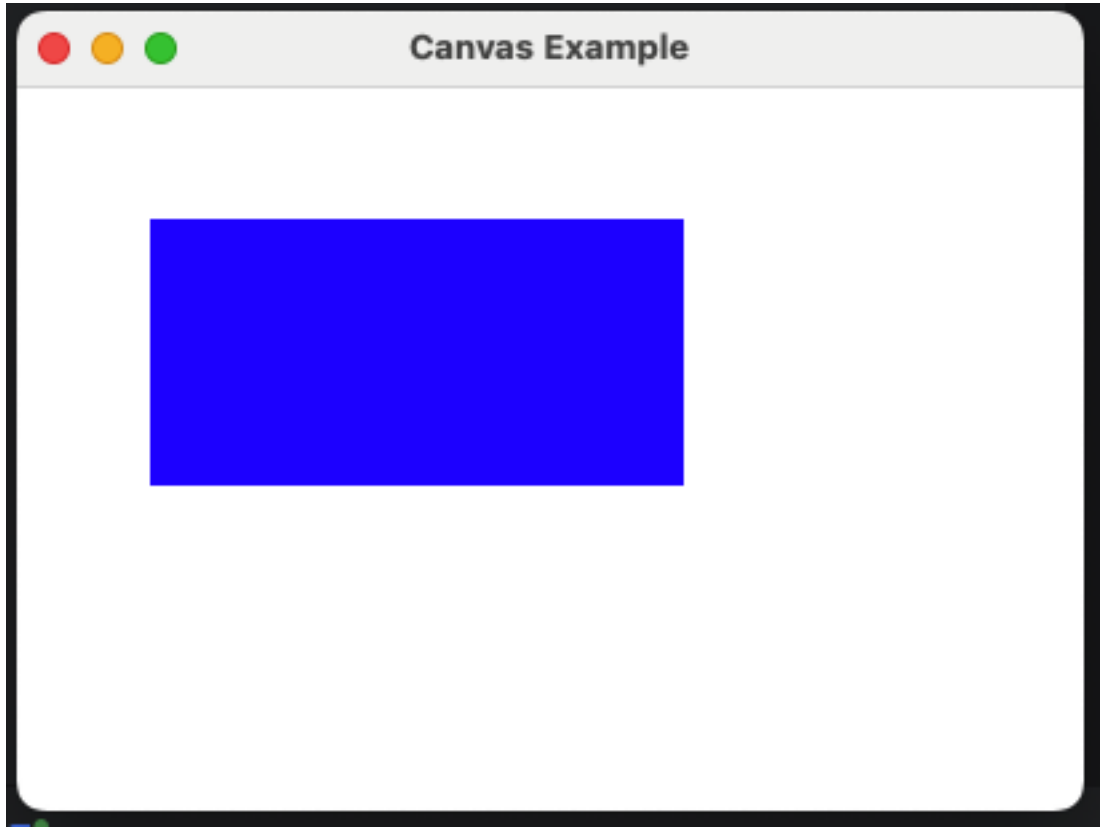


Gradient w oknie


```
import javax.swing.*;

import java.awt.*;

import java.awt.geom.*;


public class GradientExample extends JPanel {


@Override

   protected void paintComponent(Graphics g)
{

      super.paintComponent(g);


     Graphics2D g2d = (Graphics2D) g;



g2d.setRenderingHint(RenderingHints.KEY_AN
TIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);


Color color1 = Color.BLUE;    // Początkowy
kolor

     Color color2 = Color.GREEN;   // Końcowy
kolor

     GradientPaint gradient = new
GradientPaint(0, 0, color1, 0, getHeight(),
color2);

     g2d.setPaint(gradient);


     g2d.fillRect(0, 0, getWidth(), getHeight());

   }


   public static void main(String[] args) {

      JFrame frame = new JFrame("Gradient w
Swing");


frame.setDefaultCloseOperation(JFrame.EXIT
_ON_CLOSE);


      frame.add(new GradientExample());


     frame.setSize(400, 400);


     frame.setVisible(true);

   }
}
```

Przykład Canvas w Javie

```
import javax.swing.*;
import java.awt.*;

public class MyCanvas extends Canvas {
   public MyCanvas() {
      setBackground(Color.WHITE);
      setForeground(Color.BLUE);
   }

   @Override
   public void paint(Graphics g) {
      g.setColor(getForeground());
      g.fillRect(50, 50, 200, 100);
   }

   public static void main(String[] args) {
      JFrame frame = new JFrame("Canvas Example");
      MyCanvas canvas = new MyCanvas();
      frame.add(canvas);
      frame.setSize(400, 300);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```



Wybrane metody Canvas

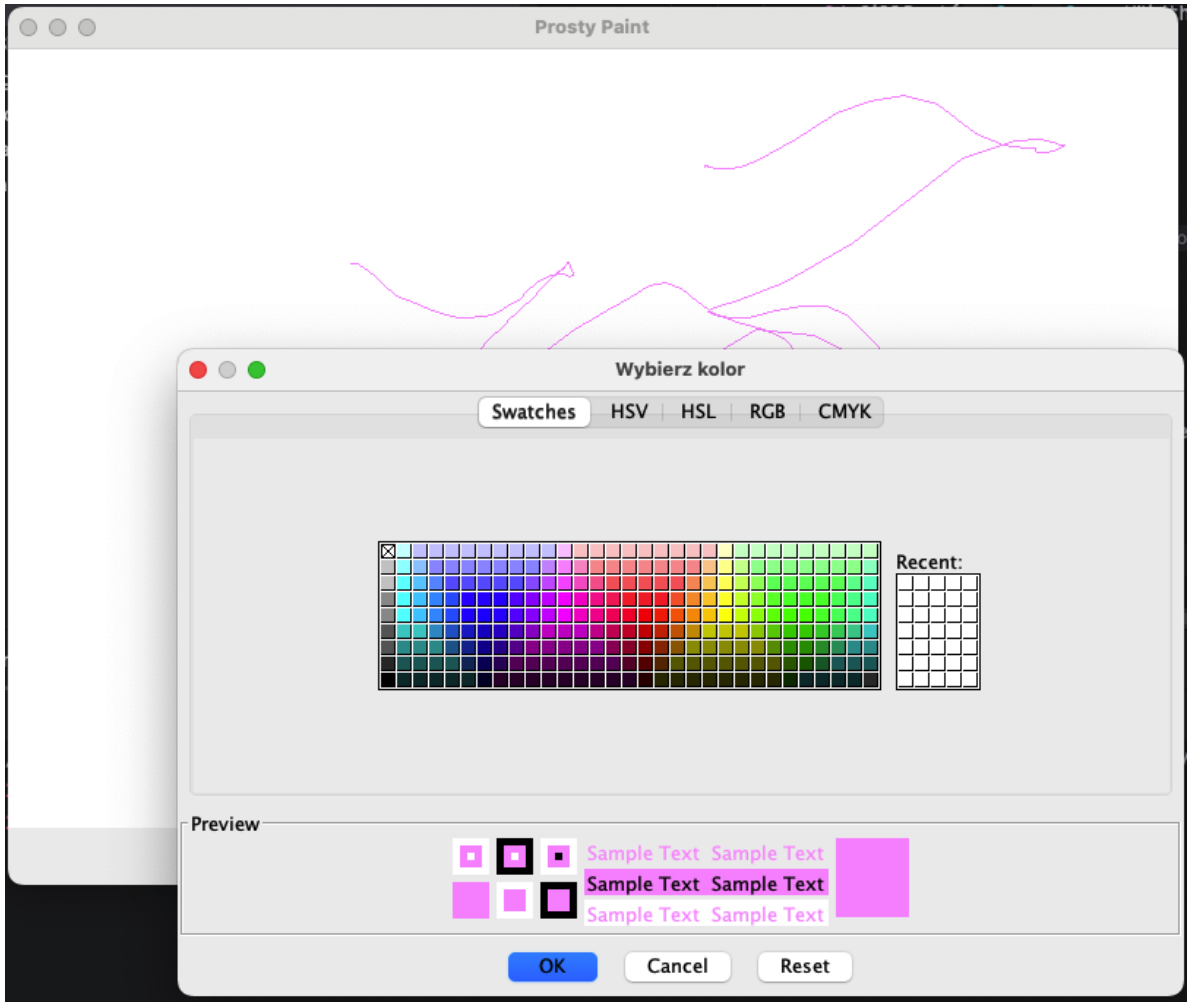
void addNotify()	•Wywoływana, gdy komponent jest dodawany do kontenera. Może być używana do inicjalizacji zasobów.
void paint(Graphics g)	•Metoda, która jest wywoływana, gdy komponent wymaga przerysowania. Używana do rysowania grafiki naCanvas.
void update(Graphics g)	•Domyślnie wywołuje paint(Graphics g), ale może być nadpisana, aby dostosować sposób aktualizacji komponentu.
void setSize(int width, int height)	•Ustawia rozmiar komponentu.
Dimension getPreferredSize()	•Zwraca preferowany rozmiar komponentu.
void setBackground(Color color)	•Ustawia kolor tła komponentu.
Color getBackground()	•Zwraca kolor tła komponentu.
void setForeground(Color color)	•Ustawia kolor rysowania (kolor, który będzie używany do rysowania tekstu i kształtów).
Color getForeground()	•Zwraca kolor rysowania komponentu.
void repaint()	•Żąda przerysowania komponentu. Może być używane do aktualizacji wyświetlanej grafiki.
void addMouseListener(MouseListener l)	•Dodaje nasłuchiwacza zdarzeń myszy do komponentu.
void addKeyListener(KeyListener l)	•Dodaje nasłuchiwacza zdarzeń klawiatury do komponentu.
void addComponentListener(ComponentListener l)	•Dodaje nasłuchiwacza zdarzeń zmiany rozmiaru komponentu.
void setVisible(boolean b)	•Ustawia widoczność komponentu.
void requestFocus()	•Żąda, aby komponent uzyskał fokus.

Prosty Paint z użyciem Canvas

```
import java.awt.event.*;
import java.awt.image.BufferedImage;

public class SimplePaint extends JFrame {
    private MyCanvas canvas;
    private Color currentColor = Color.BLACK;

    public SimplePaint() {
        setTitle("Prosty Paint");
        setSize(800, 600);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
;
        setLocationRelativeTo(null);

        canvas = new MyCanvas();
        add(canvas, BorderLayout.CENTER);

JPanel panel = new JPanel();
        JButton clearButton = new JButton("Wyczyść");
        JButton colorButton = new JButton("Wybierz
kolor");


        clearButton.addActionListener(e ->
canvas.clear());


        colorButton.addActionListener(e -> {
            Color newColor =
JColorChooser.showDialog(this, "Wybierz kolor",
currentColor);
            if (newColor != null) {
                currentColor = newColor;
                canvas.setCurrentColor(currentColor);
            }
        });
        panel.add(colorButton);
        panel.add(clearButton);
        add(panel, BorderLayout.SOUTH);
    }

class MyCanvas extends Canvas {
        private Image image;
        private Graphics2D g2d;
        private int lastX, lastY;

        public MyCanvas() {
            setBackground(Color.WHITE);
            setPreferredSize(new Dimension(800, 600));
            image = new BufferedImage(800, 600,
BufferedImage.TYPE_INT_ARGB);
            g2d = (Graphics2D) image.getGraphics();
            g2d.setColor(Color.WHITE);
            g2d.fillRect(0, 0, 800, 600);
            g2d.setColor(currentColor);

addMouseListener(new MouseAdapter() {
                @Override
                public void mousePressed(MouseEvent e) {
                    lastX = e.getX();
                    lastY = e.getY();
                }

                @Override
                public void mouseReleased(MouseEvent e) {
                    lastX = -1;
                    lastY = -1;
                }
            });

            addMouseMotionListener(new
MouseMotionAdapter() {
                @Override
                public void mouseDragged(MouseEvent e) {
                    int x = e.getX();
                    int y = e.getY();
                    g2d.drawLine(lastX, lastY, x, y);
                    lastX = x;
                    lastY = y;
                    repaint();
                }
            });
        }

        public void setCurrentColor(Color color) {
            currentColor = color;
            g2d.setColor(currentColor);
        }

        public void clear() {
            g2d.setColor(Color.WHITE);
            g2d.fillRect(0, 0, getWidth(), getHeight());
            g2d.setColor(currentColor);
            repaint();
        }

        @Override
        public void paint(Graphics g) {
            g.drawImage(image, 0, 0, null);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            SimplePaint paintApp = new SimplePaint();
            paintApp.setVisible(true);
        });
    }
```

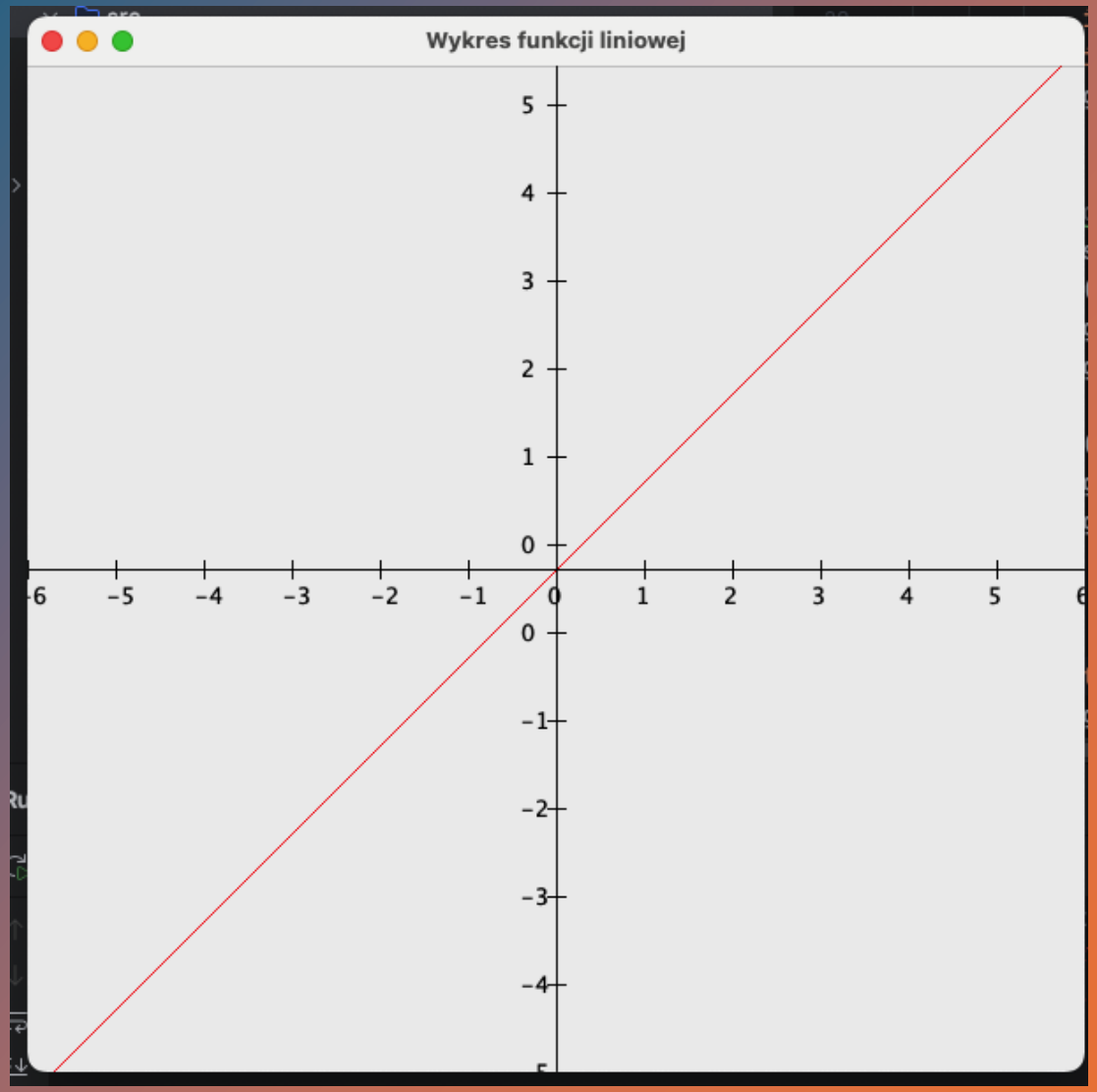
Efekt działania programu

Program pozwala wybrać kolor oraz narysować dowolny kształt za pomocą ruchu myszy. Pozwala również skasować narysowane elementy.



Rysowanie układu współrzędnych w Canvas

```
import javax.swing.*;
import java.awt.*;

public class LinearFunctionPlot extends JPanel {
  private double a, b;

  public LinearFunctionPlot(double a, double b) {
    this.a = a;
    this.b = b;
  }

  @Override
  protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D) g;

    int width = getWidth();
    int height = getHeight();
    int originX = width / 2;
    int originY = height / 2;

    g2d.drawLine(0, originY, width, originY);
    g2d.drawLine(originX, 0, originX, height);

    g2d.setColor(Color.RED);

    for (int x = -originX; x < originX; x++) {
      int y1 = (int) (a * x + b);
      int y2 = (int) (a * (x + 1) + b);
      g2d.drawLine(originX + x, originY - y1, originX + x + 1, originY - y2);
    }

    g2d.setColor(Color.BLACK);
    for (int i = -originX; i <= originX; i += 50) {
      g2d.drawLine(originX + i, originY - 5, originX + i, originY + 5);
      g2d.drawString(String.valueOf(i / 50), originX + i - 5, originY + 20);
    }
    for (int i = -originY; i <= originY; i += 50) {
      g2d.drawLine(originX - 5, originY - i, originX + 5, originY - i);
      g2d.drawString(String.valueOf(i / 50), originX - 20, originY - i + 5);
    }
  }

  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
      JFrame frame = new JFrame("Wykres funkcji liniowej");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(600, 600);
      frame.add(new LinearFunctionPlot(1, 0));
      frame.setVisible(true);
    });
  }
}
```



Efekt działania programu

Program rysuje wykres funkcji liniowej y=ax+b. Wraz ze zwiększaniem rozmiarów okna automatycznie zwiększa się obszar wykresu.

Chcąc uzyskać punkt (0,0) należy zablokować rozmiar i zachowanie okna.

SET na tysiąc sposobów (JFrame)

Metoda	Opis
setTitle(String title)	Ustawia tytuł okna.
setSize(int width, int height)	Ustawia rozmiar okna.
setVisible(boolean visible)	Ustawia widoczność okna.
setLocation(int x, int y)	Ustawia pozycję okna na ekranie.
setDefaultCloseOperation(int operation)	Ustawia zachowanie przy zamknięciu okna.
setResizable(boolean resizable)	Ustawia możliwość zmiany rozmiaru okna.
setLayout(LayoutManager layout)	Ustawia menedżera układu dla okna.
setIconImage(Image image)	Ustawia ikonę aplikacji w oknie.
setExtendedState(int state)	Ustawia stan okna (zminimalizowane, zmaksymalizowane).
setAlwaysOnTop(boolean alwaysOnTop)	Ustawia okno zawsze na wierzchu.
setUndecorated(boolean undecorated)	Ustawia okno bez ramki tytułowej.
setOpacity(float opacity)	Ustawia przezroczystość okna (0.0 - 1.0).

SET na tysiąc sposobów (JLabel)

Metoda	Opis
setText(String text)	Ustawia tekst etykiety.
setFont(Font font)	Ustawia czcionkę tekstu.
setForeground(Color color)	Ustawia kolor tekstu.
setBackground(Color color)	Ustawia kolor tła.
setOpaque(boolean opaque)	Ustawia nieprzezroczystość tła.
setIcon(Icon icon)	Ustawia ikonę w etykiecie.
setHorizontalAlignment(int alignment)	Ustawia poziome wyrównanie tekstu.
setVerticalAlignment(int alignment)	Ustawia pionowe wyrównanie tekstu.
setHorizontalTextPosition(int textPosition)	Ustawia pozycję tekstu względem ikony.
setVerticalTextPosition(int textPosition)	Ustawia pionową pozycję tekstu względem ikony.
setToolTipText(String text)	Ustawia tekst podpowiedzi.

SET na tysiąc sposobów (JButton)

Metoda	Opis
setText(String text)	Ustawia tekst na przycisku.
setEnabled(boolean enabled)	Ustawia aktywność przycisku.
setIcon(Icon icon)	Ustawia ikonę na przycisku.
setDisabledIcon(Icon icon)	Ikona przycisku, gdy jest nieaktywny.
setPressedIcon(Icon icon)	Ikona przycisku podczas naciśnięcia.
setRolloverIcon(Icon icon)	Ikona po najechaniu kursorem.
setMnemonic(int mnemonic)	Ustawia klawisz skrótu (Alt + klawisz).
setToolTipText(String text)	Ustawia podpowiedź na przycisku.
setBorderPainted(boolean painted)	Ustawia widoczność ramki przycisku.
setContentAreaFilled(boolean filled)	Ustawia wypełnienie przycisku.
setFocusPainted(boolean painted)	Ustawia efekt fokusu.
setHorizontalAlignment(int alignment)	Poziome wyrównanie tekstu i ikony.
setVerticalAlignment(int alignment)	Pionowe wyrównanie tekstu i ikony.

SET na tysiąc sposobów (JTextField)

Metoda	Opis
setText(String text)	Ustawia tekst w polu tekstowym.
setEditable(boolean editable)	Ustawia edytowalność pola.
setColumns(int columns)	Ustawia liczbę widocznych kolumn tekstu.
setCaretPosition(int position)	Ustawia pozycję kursora.
setHorizontalAlignment(int alignment)	Wyrównanie tekstu (LEFT, CENTER, RIGHT).
setSelectionStart(int start)	Ustawia początek zaznaczenia tekstu.
setSelectionEnd(int end)	Ustawia koniec zaznaczenia tekstu.
setToolTipText(String text)	Ustawia podpowiedź dla pola tekstowego.
setBackground(Color color)	Ustawia kolor tła.
setForeground(Color color)	Ustawia kolor tekstu.
setFont(Font font)	Ustawia czcionkę tekstu.

SET na tysiąc sposobów (JTextArea)

Metoda	Opis
setText(String text)	Ustawia tekst w obszarze tekstowym.
setEditable(boolean editable)	Ustawia edytowalność obszaru.
setLineWrap(boolean wrap)	Ustawia automatyczne łamanie linii.
setWrapStyleWord(boolean word)	Ustawia łamanie linii przy słowach.
setCaretPosition(int position)	Ustawia pozycję kursora.
setRows(int rows)	Ustawia liczbę widocznych wierszy.
setColumns(int columns)	Ustawia liczbę widocznych kolumn tekstu.
setTabSize(int size)	Ustawia liczbę spacji dla tabulatora.

SET na tysiąc sposobów (JTable)

Metoda	Opis
setModel(TableModel model)	Ustawia model danych tabeli.
setEnabled(boolean enabled)	Włącza lub wyłącza możliwość edycji tabeli.
setRowSelectionAllowed(boolean allowed)	Określa, czy użytkownik może zaznaczać całe wiersze.
setColumnSelectionAllowed(boolean allowed)	Określa, czy użytkownik może zaznaczać kolumny.
setSelectionMode(int mode)	Ustawia tryb zaznaczania (np. ListSelectionModel.SINGLE_SELECTION)
setRowHeight(int rowHeight)	Ustawia wysokość wierszy.
setColumnSelectionAllowed(boolean columnSelectionAllowed)	Ustawia możliwość zaznaczania kolumn.
setAutoResizeMode(int mode)	Ustawia tryb automatycznego zmieniania rozmiaru kolumn (JTable.AUTO_RESIZE_ALL_COLUMNS).
setShowGrid(boolean showGrid)	Określa, czy siatka tabeli ma być widoczna.
setIntercellSpacing(Dimension dim)	Ustawia odstęp między komórkami.
setGridColor(Color color)	Ustawia kolor siatki tabeli.
setTableHeader(JTableHeader tableHeader)	Ustawia nagłówek tabeli.

SET na tysiąc sposobów (JCheckbox)

Metoda	Opis
setSelected(boolean selected)	Ustawia zaznaczenie pola wyboru.
setText(String text)	Ustawia tekst przy polu wyboru.
setEnabled(boolean enabled)	Ustawia aktywność pola wyboru.
setIcon(Icon icon)	Ustawia ikonę dla pola wyboru.
setSelectedIcon(Icon icon)	Ikona dla zaznaczonego pola.
setHorizontalAlignment(int alignment)	Wyrównanie poziome tekstu i ikony.
setBorderPainted(boolean painted)	Ustawia widoczność ramki wokół pola wyboru.
setContentAreaFilled(boolean filled)	Ustawia wypełnienie pola wyboru.
setFocusPainted(boolean painted)	Ustawia efekt fokusu.
setToolTipText(String text)	Ustawia podpowiedź dla pola wyboru.

SET na tysiąc sposobów (JComboBox)

Metoda	Opis
setSelectedItem(Object item)	Ustawia wybrany element.
setEditable(boolean editable)	Ustawia edytowalność listy.
setModel(ComboBoxModel model)	Ustawia model danych.
setMaximumRowCount(int count)	Ustawia maksymalną liczbę widocznych elementów po rozwinięciu.
setRenderer(ListCellRenderer renderer)	Ustawia niestandardowy renderer dla pozycji.
setPopupVisible(boolean visible)	Ustawia widoczność rozwijanego menu.
setActionCommand(String command)	Ustawia komendę akcji dla komponentu.
setToolTipText(String text)	Ustawia tekst podpowiedzi dla listy rozwijanej.
setBackground(Color color)	Ustawia kolor tła listy rozwijanej.
setForeground(Color color)	Ustawia kolor tekstu w polu wyboru.

SET na tysiąc sposobów (JList)

Metoda	Opis
setListData(Object[] listData)	Ustawia dane listy.
setSelectionMode(int mode)	Ustawia tryb zaznaczania (SINGLE_SELECTION, MULTIPLE_INTERVAL_SELECTION).
setSelectedIndex(int index)	Ustawia zaznaczony indeks.
setCellRenderer(ListCellRenderer renderer)	Ustawia niestandardowy renderer dla elementów.
setFixedCellHeight(int height)	Ustawia stałą wysokość wiersza.
setFixedCellWidth(int width)	Ustawia stałą szerokość komórki listy.
setFixedCellHeight(int height)	Ustawia stałą wysokość komórki listy.
setCellRenderer(ListCellRenderer<? super E> renderer)	Ustawia renderer komórek listy.
setLayoutOrientation(int orientation)	Ustawia orientację listy (VERTICAL, HORIZONTAL_WRAP).
setFixedCellHeight(int height)	Ustawia stałą wysokość komórki listy.
setCellRenderer(ListCellRenderer<? super E> renderer)	Ustawia renderer komórek listy.
setLayoutOrientation(int orientation)	Ustawia orientację listy (VERTICAL, HORIZONTAL_WRAP).

SET na tysiąc sposobów (JPanel)

Metoda	Opis
setLayout(LayoutManager layout)	Ustawia menedżera układu.
setBackground(Color color)	Ustawia kolor tła.
setBorder(Border border)	Ustawia obramowanie panelu.
setPreferredSize(Dimension preferredSize)	Ustawia preferowany rozmiar.
setOpaque(boolean opaque)	Ustawia przezroczystość panelu.
setMinimumSize(Dimension dimension)	Ustawia minimalny rozmiar panelu.
setMaximumSize(Dimension dimension)	Ustawia maksymalny rozmiar panelu.
setAlignmentX(float alignmentX)	Ustawia poziome wyrównanie panelu.
setAlignmentY(float alignmentY)	Ustawia pionowe wyrównanie panelu.
setCursor(Cursor cursor)	Ustawia kursor myszy po najechaniu na panel.
setComponentOrientation(ComponentOrientation orientation)	Ustawia orientację komponentów (np. lewo-prawo).
setEnabled(boolean enabled)	Ustawia aktywność panelu i jego komponentów.

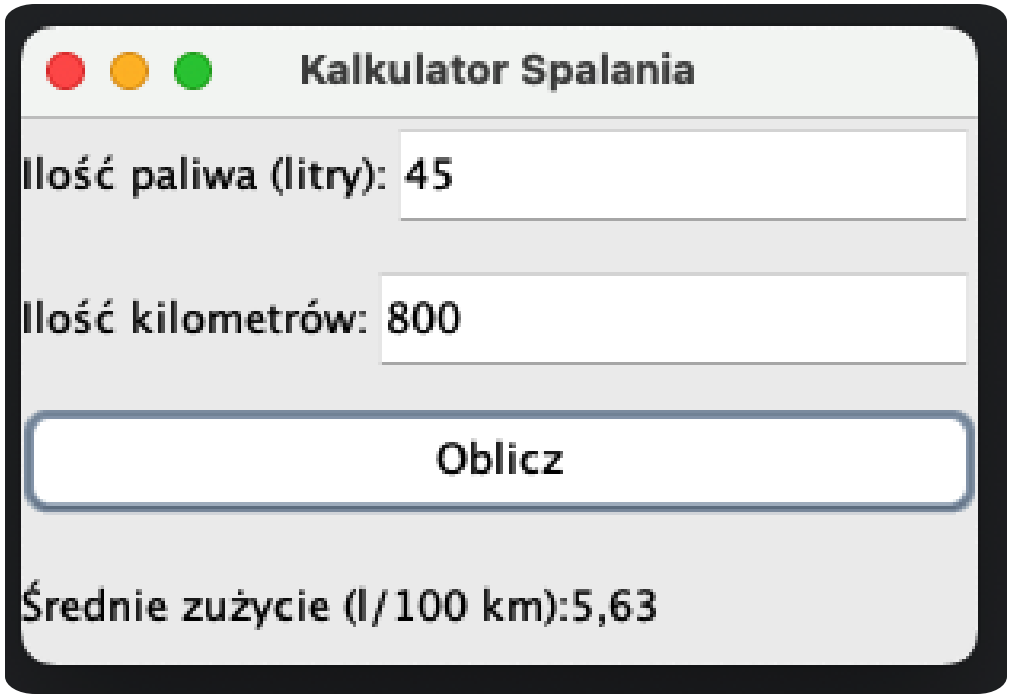
Pytania i polecenia

1. Wymień przynajmniej 3 różnice pomiędzy Swing a AWT.
2. Wymień przynajmniej 5 komponentów „Jdrzewa”.
3. Do czego służą metody setLayout, setVisible i setSize?
4. Co daje zastosowanie DefaultCloseOperation?
5. Czy do czego służy flowLayout?
6. Wymień przynajmniej 5 rodzajów zdarzeń w języku JAVA?
7. Do czego służy MessageDialog?
8. Ile w prezentacji wymieniono zdarzeń dla myszy?
9. Wymień przynamniej 5 właściwości JButton, JPanel i JComboBox.
10. Wymień przynajmniej 5 właściwości SET dla JButton, Jpanel i Jcombobox.
11. Wymień przynajmniej 5 metod Canvas.

Zadania do samodzielnego wykonania

Zadanie 1

W oparciu o poznany kod kalkulatora rozwiń projekt tak aby aplikacja wykonywała 4 podstawowe działania a użytkownik podawał liczby za pomocą naciskania przycisków.



Zadania do samodzielnego wykonania

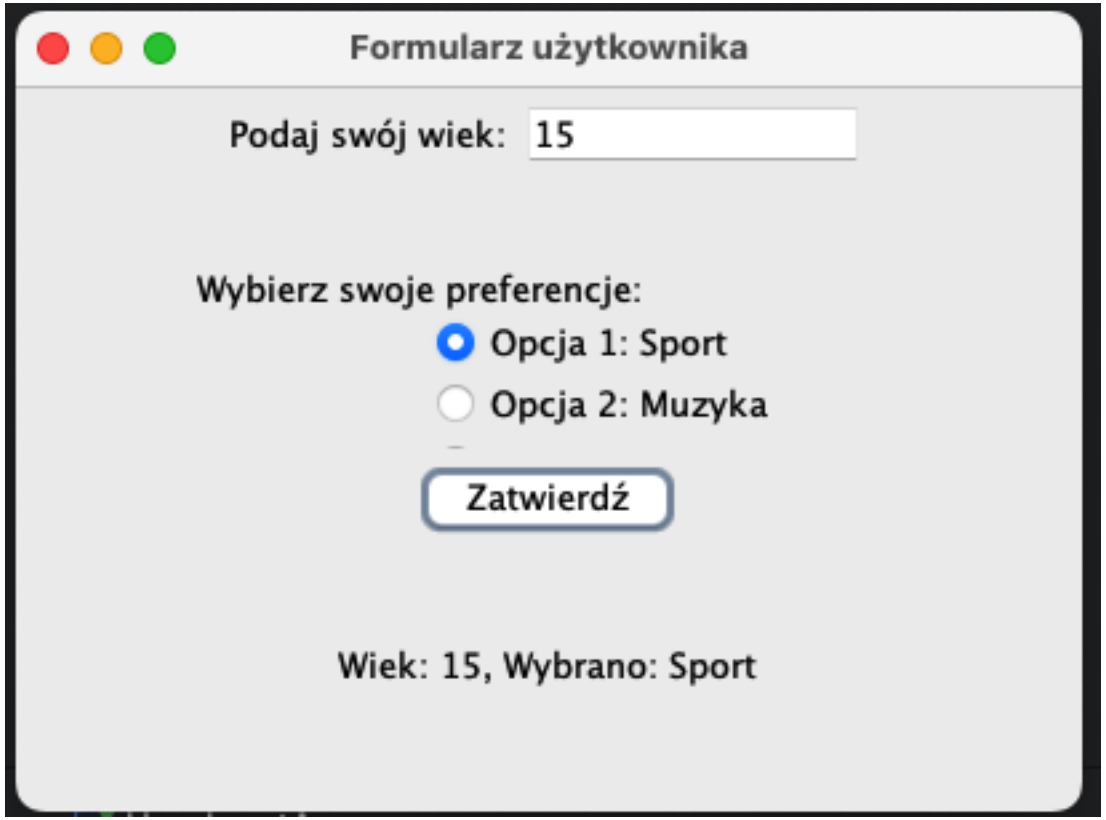
Zadanie 2

Napisz program, którego zadaniem będzie wyliczenie średniego zużycia paliwa w oparciu o dane: wielkość zbiornika paliwa oraz zasięg.

Zadania do samodzielnego wykonania

Zadanie 3

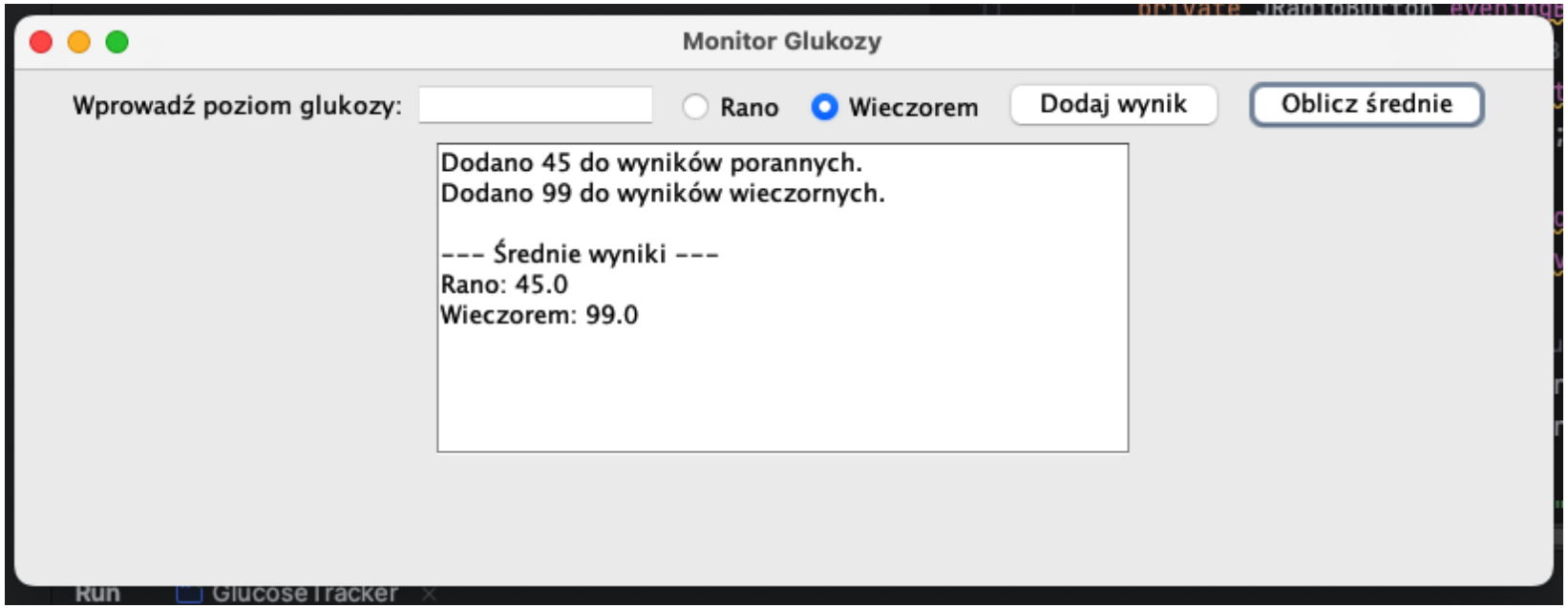
Napisz formularz, w którym użytkownik wpisuje wiek oraz wybiera jedną z dwóch opcji. Po naciśnięciu przycisku wpisane dane zostają wyświetlone w dolnej części formularza.



Zadania do samodzielnego wykonania

Zadanie 4

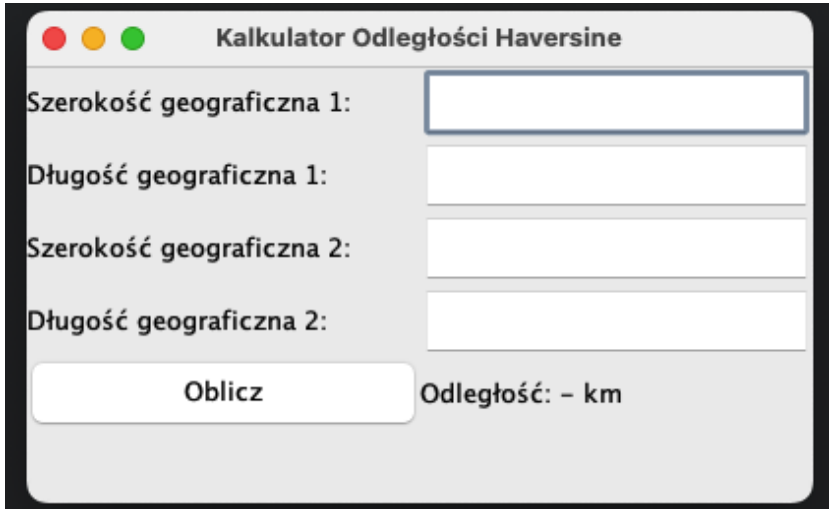
Napisz program, w którym użytkownik podaje wartość poziomu cukru i wybiera porę dnia. Program do naciśnięciu przycisku dodaje wynik do okna. Przycisk „Oblicz średnie” oblicza wartości oddzielnie dla pory dnia.



Zadania do samodzielnego wykonania

Zadanie 5

Napisz program, który oblicza odległość na podstawie długości i szerokości geograficznej. Wykorzystano wzór Haversina.

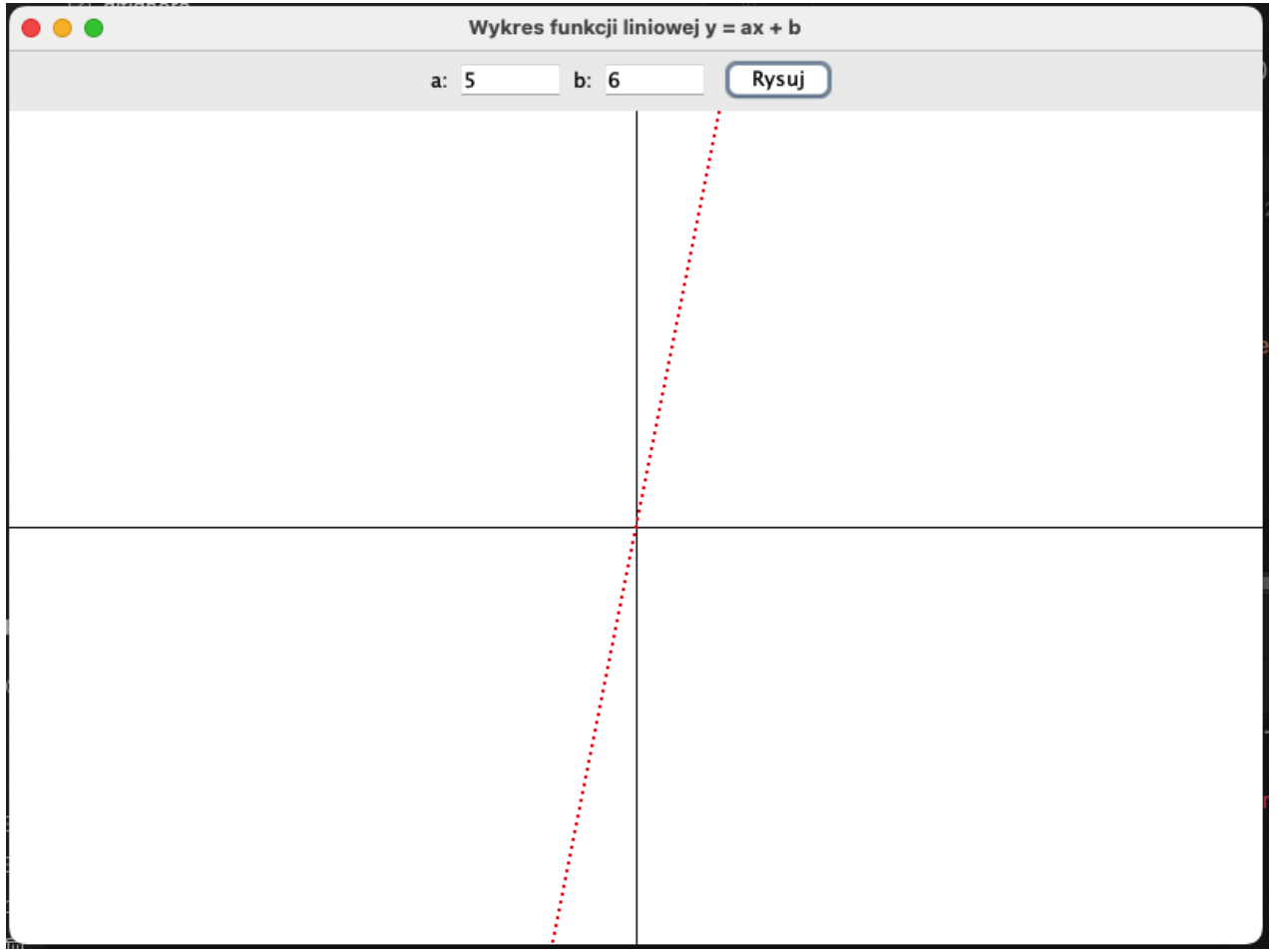



```
double a = Math.pow(Math.sin(dLat / 2), 2) + Math.cos(lat1) * Math.cos(lat2) *
Math.pow(Math.sin(dLon / 2), 2);
double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

double radius = 6371;
double distance = radius * c;
```

Zadania do samodzielnego wykonania

Zadanie 6

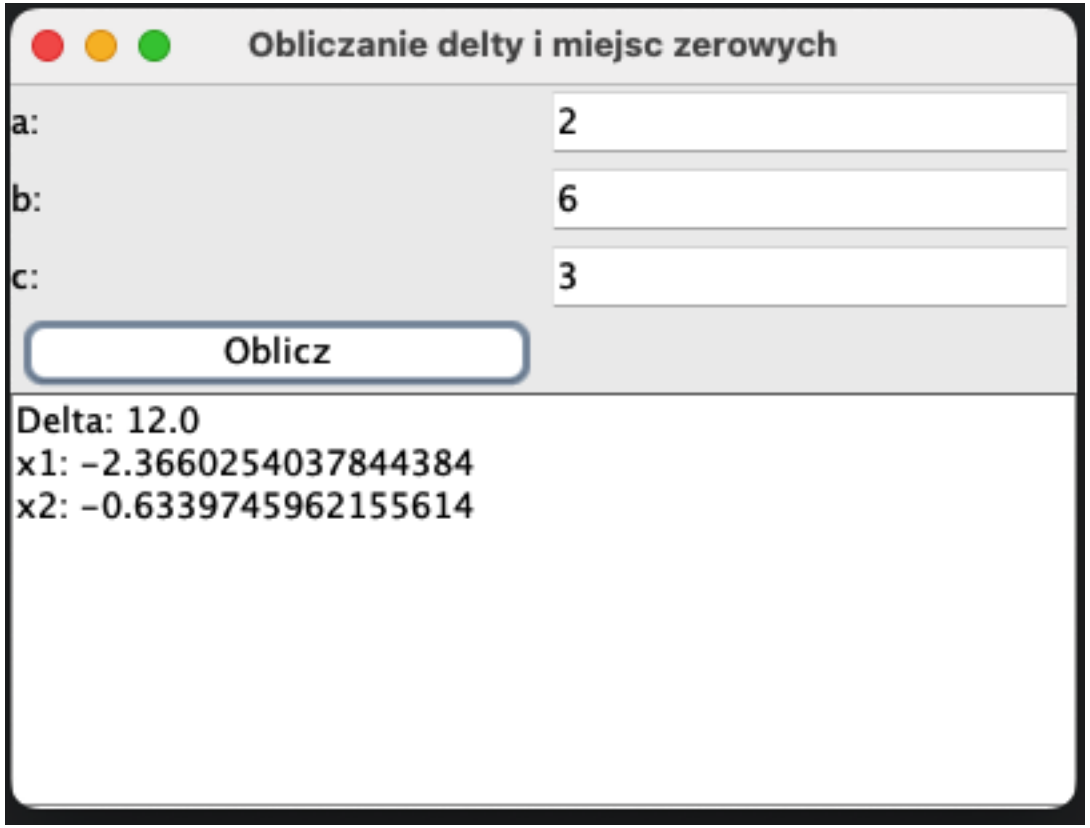
Napisz program, w którym użytkownik podaje wartość parametrów funkcji liniowej a następnie program rysuje prostą na układzie współrzędnych.



Zadania do samodzielnego wykonania

Zadanie 7

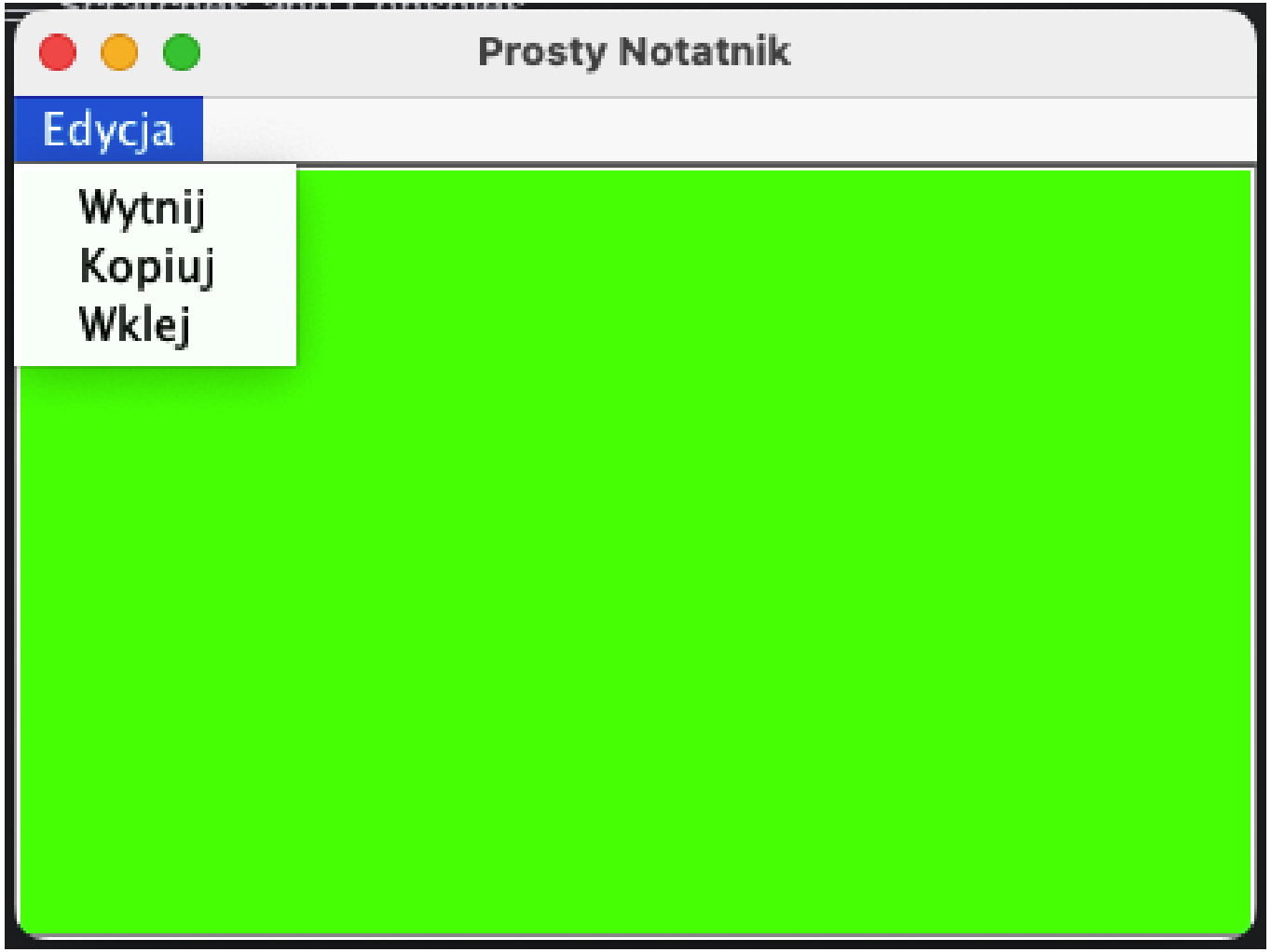
Napisz aplikację obliczającą „Deltę” oraz X0, X1 i X2 . Dane wprowadzane są przez użytkownika.



Zadania do samodzielnego wykonania

Zadanie 8

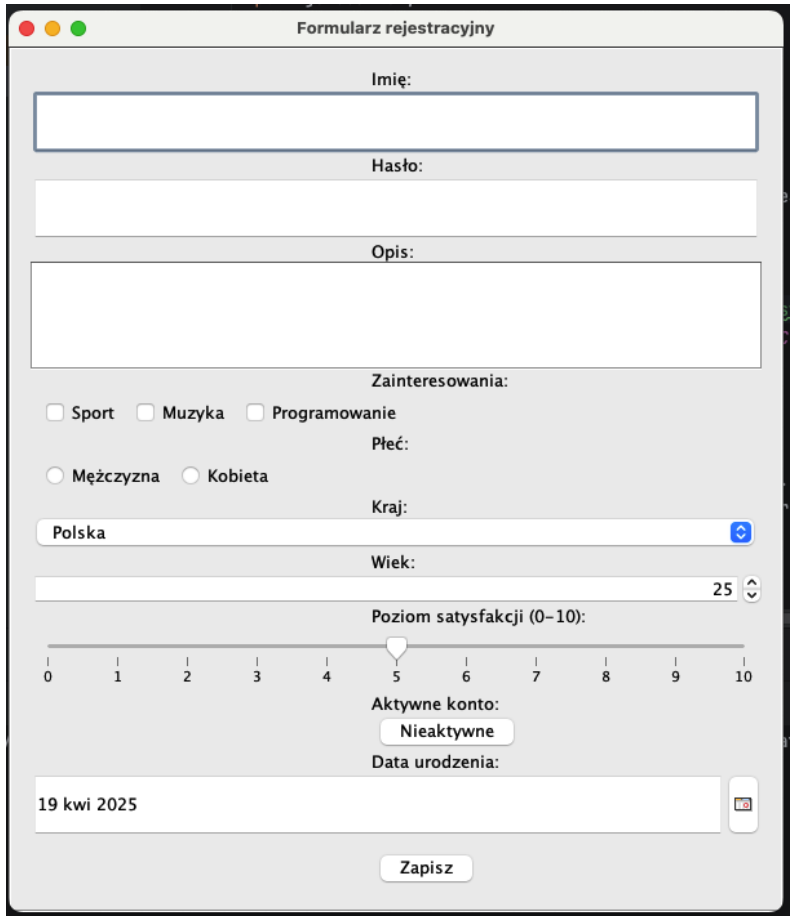
Napisz prostą aplikację notatnika z użyciem menu i metod copy, cut i paste.



Zadania do samodzielnego wykonania

Zadanie 9

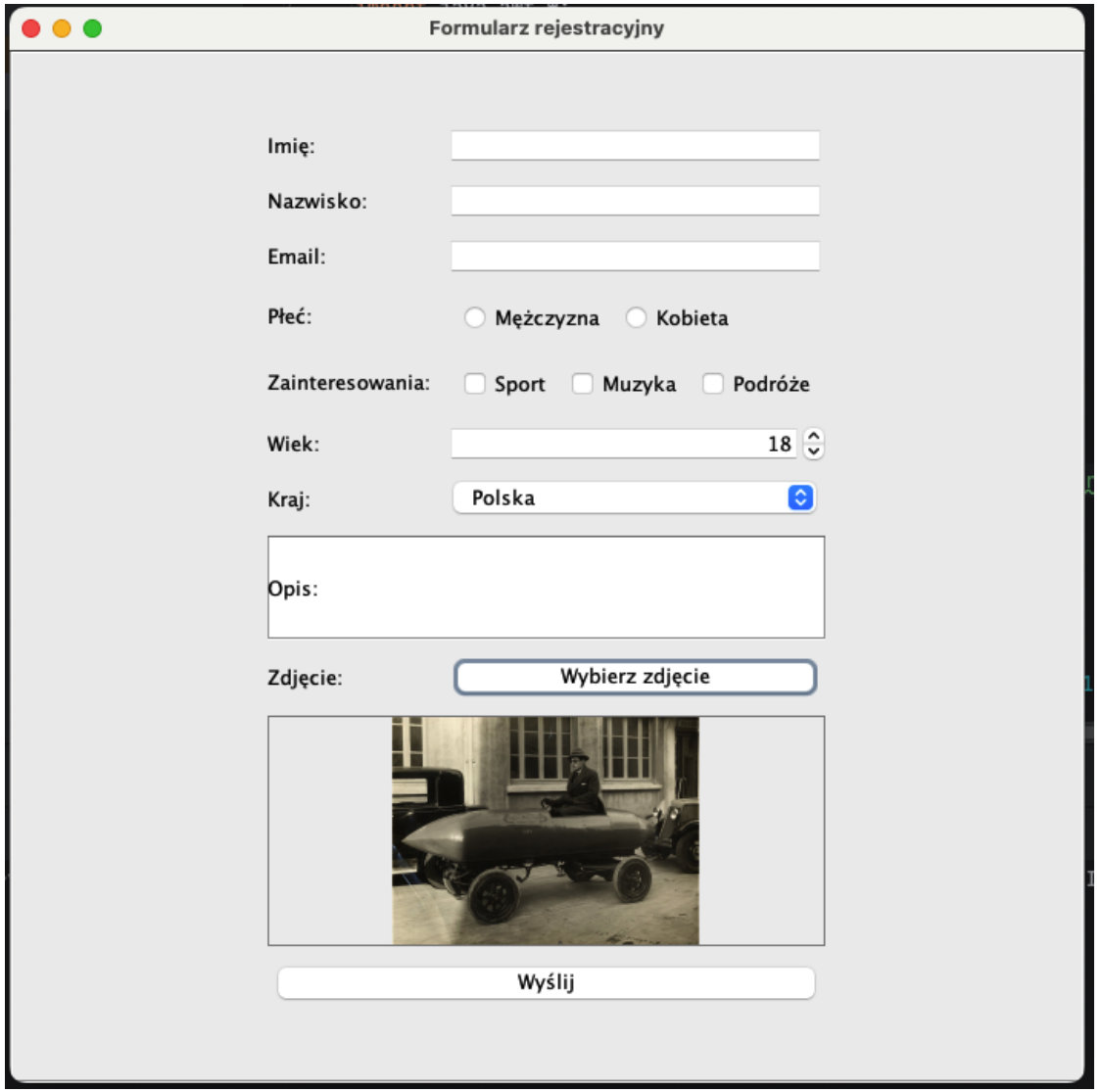
Napisz kod formularza rejestracyjnego według wzoru. Po naciśnięciu przycisku „Zapisz” pojawia się drugie okno z danymi wpisanymi przez użytkownika.



Zadania do samodzielnego wykonania

Zadanie 10

Na podstawie kodu z zadania 9 utwórz formularz z możliwością dodania zdjęcia oraz zastosuj GridBagLayout dla poprawnego wyświetlenia elementów zgodnie z wzorem.



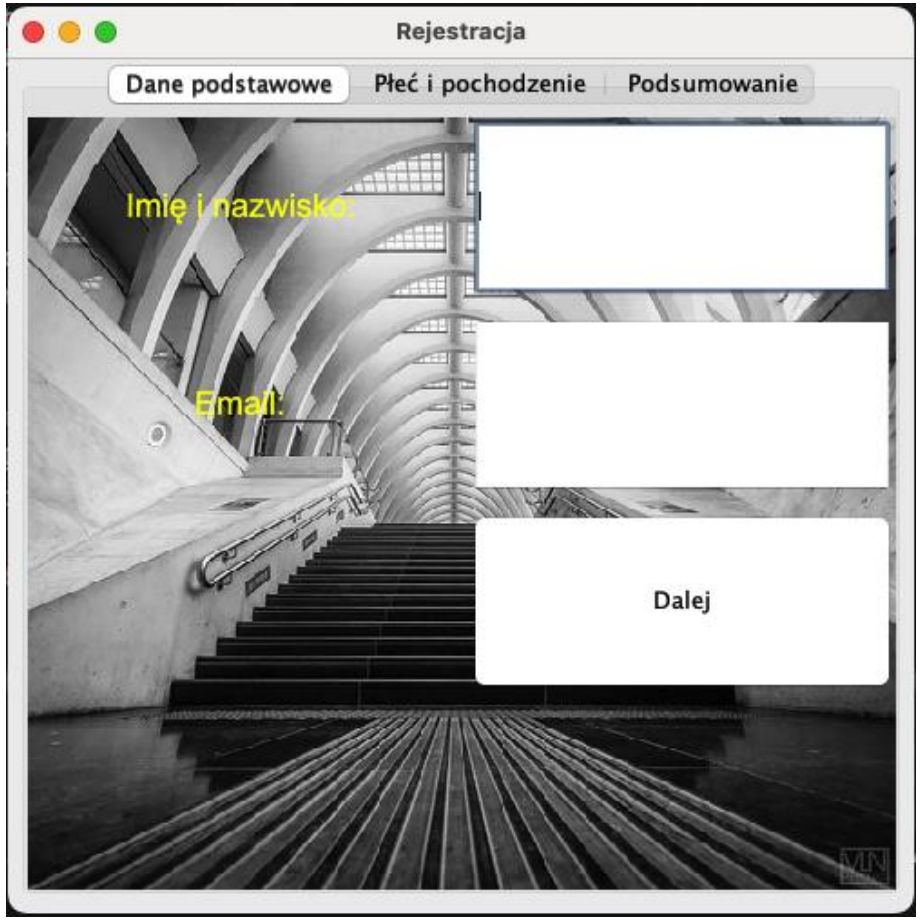
Zadanie do samodzielnego wykonania na 5!

Zadanie na 5:

Aplikacja w formie dzielonego pomiędzy karty formularza zbierającego dane i wyświetlającego dane na ostatniej karcie. Konieczne wykorzystanie zdjęcia jako tło oraz GRIDlayout. Należy użyć formatowania etykiet tekstowych oraz właściwości formatujących pola tekstowe.

Przykład znajdziecie na następnej stronie:

Zadanie do samodzielnego wykonania na 5!

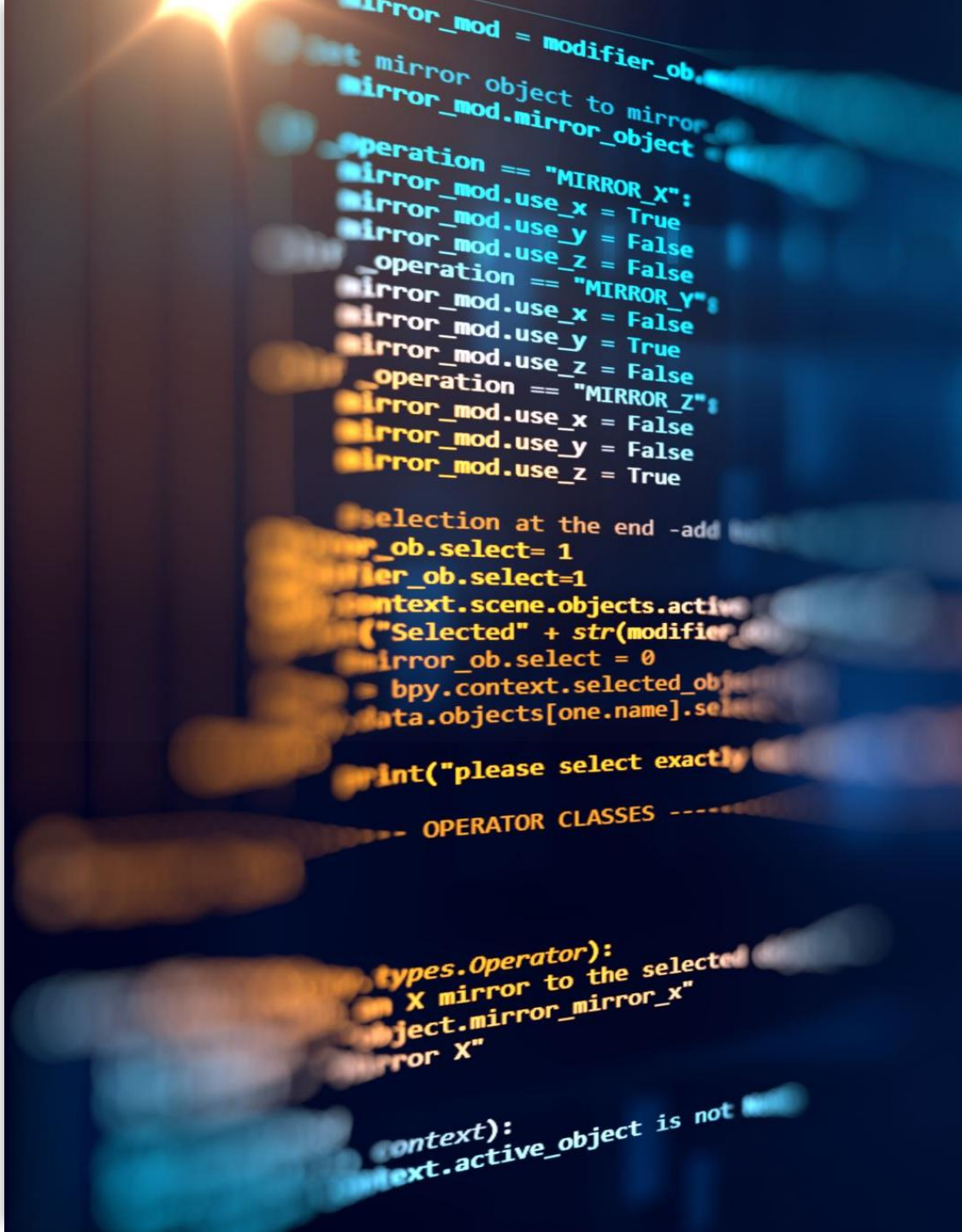


Zadanie do samodzielnego wykonania na 5!

Plik – miejsce gromadzenia danych aplikacji

Chcąc zapisać dane z formularza do pliku w Javie, można użyć klas takich
jak FileWriter, BufferedWriter, lub PrintWriter.
Poniżej znajduje się przykład, jak zapisać dane w pliku tekstowym:

1. **Zbierz dane z formularza**.
2. **Stwórz plik** (jeśli nie istnieje).
3. **Zapisz dane do pliku**.



Biblioteki i obiekt potrzebne do zapisu

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException; ...

File file = new File("formularz.txt");
BufferedWriter writer =new BufferedWriter
(new FileWriter(file, true));
```

Całość procesu zapisywania

Nasłuchiwanie zdarzenia do przycisku:

```
submitButton.addActionListener(e -> {
String name = nameField.getText();
String email= emailField.getText();
String age = ageField.getText();
String gender = maleButton.isSelected() ? "Mężczyzna" : femaleButton.isSelected() ? "Kobieta" : "Inne";
String country = (String) countryBox.getSelectedItem();
booleanacceptedTerms = termsCheckBox.isSelected();


summaryArea.setText("Podsumowanie:\n
Imię i nazwisko: " + name + "\n
Email: " + email + "\n
Wiek: " + age + "\n
Płeć: "+ gender + "\n
Kraj: " + country + "\n
Akceptacja regulaminu: " + (acceptedTerms ? "Tak" : "Nie"));
tabbedPane.setSelectedIndex(2);
```

Całość procesu zapisywania

```
try {
File file = new File("formularz.txt");
BufferedWriter writer = new BufferedWriter(new FileWriter(file, true));
writer.write("Imię i nazwisko: " + name + "\n");
writer.write("Email: " + email + "\n");
writer.write("Wiek: " + age + "\n");
writer.write("Płeć: " + gender + "\n");
writer.write("Kraj: " + country + "\n");
writer.write("Akceptacja regulaminu: " + (acceptedTerms ? "Tak" : "Nie") + "\n");
writer.write("---------------------------------\n");
writer.close();
JOptionPane.showMessageDialog(this, "Dane zostały zapisane do pliku!");
}
catch(IOException ex) { ex.printStackTrace();
JOptionPane.showMessageDialog(this, "Błąd podczas zapisywania do pliku!", "Błąd", JOptionPane.ERROR_MESSAGE); } });
```

Objaśnienie:

Zbieranie danych: Dane są pobierane z pól formularza (np. nameField.getText(), emailField.getText(), itp.).

Tworzenie pliku: Jeśli plik o nazwie formularz.txt nie istnieje, zostanie utworzony w bieżącym katalogu. Jeśli plik już istnieje, nowe dane zostaną dopisane do niego (dzięki opcji true w konstruktorze FileWriter).

Zapis do pliku: Dane są zapisywane w formacie tekstowym, linia po linii, a po zapisaniu danych do pliku, plik jest zamykany.

Obsługa wyjątków: Jeśli wystąpi błąd podczas zapisu, program wyświetli komunikat o błędzie.

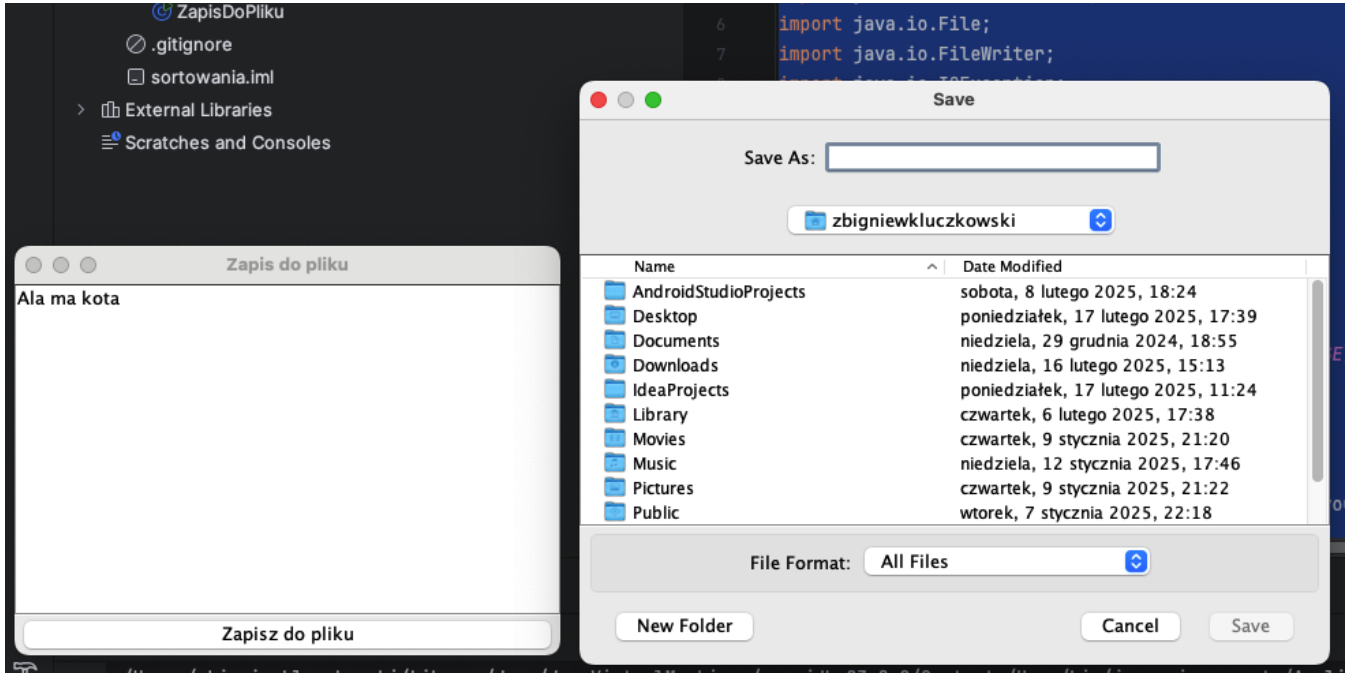
Przykład programu z dostępem do pliku

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class ZapisDoPliku extends JFrame {
    private JTextArea poleTekstowe;
    private JButton przyciskZapisz;

    public ZapisDoPliku() {
        setTitle("Zapis do pliku");
        setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
;
        setLayout(new BorderLayout());

        poleTekstowe = new JTextArea();
        add(new JScrollPane(poleTekstowe),
BorderLayout.CENTER);

        przyciskZapisz = new JButton("Zapisz do pliku");
        add(przyciskZapisz, BorderLayout.SOUTH);

        przyciskZapisz.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String tekst = poleTekstowe.getText();

                JFileChooser wybierzPlik = new
JFileChooser();
                int wynik = wybierzPlik.showSaveDialog(null);

                if (wynik == JFileChooser.APPROVE_OPTION)
{
                    File plik = wybierzPlik.getSelectedFile();
                    try (BufferedWriter writer = new
BufferedWriter(new FileWriter(plik))) {
                        writer.write(tekst);
                        JOptionPane.showMessageDialog(null,
"Zapisano do pliku: " + plik.getAbsolutePath());
                    } catch (IOException ex) {
                        JOptionPane.showMessageDialog(null,
"Błąd zapisu: " + ex.getMessage());
                    }
                }
            }
        });

        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new
ZapisDoPliku());
    }
}
```

Wynik działania programu

Objaśnienie:

Aplikacja notatnika z funkcją zapisu danych do pliku. Po naciśnięciu przycisku użytkownik może wpisać nazwę i zapisać plik w dowolnym miejscu. Po zapisie wyskakuje okno z miejscem zapisu pliku.



Wyszukiwarka z pliku

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;

public class ProstaWyszukiwarka extends JFrame {

    private JTextField poleSzukaj;
    private JTextArea poleWynikow;
    private JButton przyciskSzukaj;

    public ProstaWyszukiwarka() {
        setTitle("Wyszukiwarka z pliku");
        setSize(500, 400);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Panel górny
        JPanel panelGora = new JPanel();
        panelGora.setLayout(new FlowLayout());

        panelGora.add(new JLabel("Wyszukaj:"));
        poleSzukaj = new JTextField(20);
        panelGora.add(poleSzukaj);

        przyciskSzukaj = new JButton("Szukaj");
        panelGora.add(przyciskSzukaj);
        add(panelGora, BorderLayout.NORTH);

        // Pole wyników
        poleWynikow = new JTextArea();
        poleWynikow.setEditable(false);
        JScrollPane scrollPane = new JScrollPane(poleWynikow);
        add(scrollPane, BorderLayout.CENTER);

        // Obsługa przycisku
        przyciskSzukaj.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                szukajWPliku("dane.txt", poleSzukaj.getText());
            }
        });

        setVisible(true);
    }

    private void szukajWPliku(String nazwaPliku, String zapytanie) {
        poleWynikow.setText("");
        if (zapytanie.isEmpty()) {
            poleWynikow.setText("Wpisz frazę do wyszukania.");
            return;
        }
        try (BufferedReader reader = new BufferedReader(new FileReader(nazwaPliku))) {
            String linia;
            boolean znaleziono = false;
            while ((linia = reader.readLine()) != null) {
                if (linia.toLowerCase().contains(zapytanie.toLowerCase())) {
                    poleWynikow.append(linia + "\n");
                    znaleziono = true;
                }
            }
            if (!znaleziono) {
                poleWynikow.setText("Brak wyników dla: " + zapytanie);
            }
        } catch (IOException ex) {
            poleWynikow.setText("Błąd odczytu pliku: " + ex.getMessage());
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new ProstaWyszukiwarka());
    }
}
```

Efekt działania programu

Uwaga!

Uwaga na ścieżkę dostępu do pliku. Ponadto, należy odpowiednio formatować dane. Konstrukcja wyszukiwarki powinna zawierać możliwość „szukania po znaku”. To najprostszy sposób na napisanie prostego i funkcjonalnego kodu.

Dodatkowe możliwości:

Możesz zmienić format zapisywanych danych (np. JSON, CSV) w zależności od potrzeb.

Możesz określić ścieżkę do pliku, np. new File("C:/path/to/file/formularz.txt"), aby zapisać plik w konkretnym katalogu.

... można również zapisać dane w bazie SQL. To jednak ”wyższa szkoła jazdy”.

Baza danych SQL i okno w Javie

Zainstaluj odpowiedni sterownik JDBC: Jeśli używasz MySQL, musisz mieć zainstalowany sterownik JDBC (np. mysql-connector-java).

Utwórz połączenie z bazą danych: Skonfiguruj połączenie z bazą danych, używając JDBC.

Zapisz dane do bazy: Skorzystaj z przygotowanego zapytania SQL, aby wstawić dane z formularza do odpowiednich tabel.

Połączenie z bazą danych

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public Connection getConnection() throws SQLException {
   String url = "jdbc:mysql://localhost:3306/nazwa_bazy_danych;
   String user = "root";
   String password = "hasło";

   return DriverManager.getConnection(url, user, password);
}
```

Przykład zadania z zapisem do bazy danych

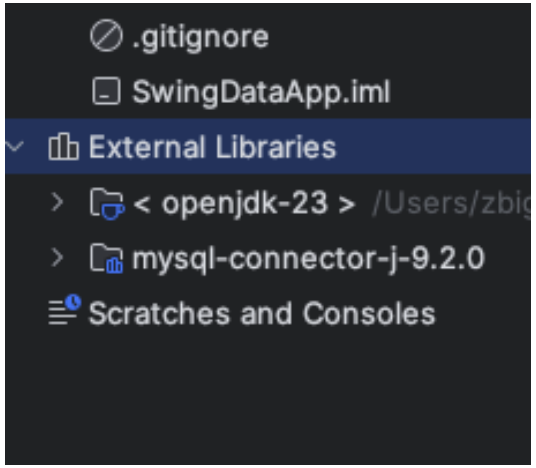
```
import java.awt.*;
import java.sql.*;

public class SwingDatabaseApp {
    private static final String DB_URL =
"jdbc:mysql://localhost:8889/test_db";
    private static final String DB_USER = "root";
    private static final String DB_PASS = "root";

    public static void main(String[] args) {
        Frame frame = new Frame("Aplikacja SQL w AWT");
        frame.setSize(400, 300);
        frame.setLayout(null);

        Label nameLabel = new Label("Imię:");
        nameLabel.setBounds(20, 50, 50, 20);
        frame.add(nameLabel);

        TextField nameField = new TextField();
        nameField.setBounds(80, 50, 200, 20);
        frame.add(nameField);

        Label emailLabel = new Label("Email:");
        emailLabel.setBounds(20, 80, 50, 20);
        frame.add(emailLabel);

        TextField emailField = new TextField();
        emailField.setBounds(80, 80, 200, 20);
        frame.add(emailField);

        Button saveButton = new Button("Zapisz");
        saveButton.setBounds(80, 110, 80, 30);
        frame.add(saveButton);

        Button loadButton = new Button("Wyświetl dane");
        loadButton.setBounds(180, 110, 120, 30);
        frame.add(loadButton);

        TextArea textArea = new TextArea();
        textArea.setBounds(20, 150, 340, 100);
        textArea.setEditable(false);
        frame.add(textArea);

        saveButton.addActionListener(e ->
saveToDatabase(nameField.getText(),
emailField.getText()));

        loadButton.addActionListener(e ->
textArea.setText(loadFromDatabase()));

        frame.addWindowListener(new
java.awt.event.WindowAdapter() {
            public void
windowClosing(java.awt.event.WindowEvent
windowEvent) {
                System.exit(0);
            }
        });

        frame.setVisible(true);
    }

    private static void saveToDatabase(String name, String
email) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver"); //
Załaduj sterownik MySQL
            Connection conn =
DriverManager.getConnection(DB_URL, DB_USER,
DB_PASS);
            String sql = "INSERT INTO users (name, email)
VALUES (?, ?)";
            PreparedStatement pstmt =
conn.prepareStatement(sql);
            pstmt.setString(1, name);
            pstmt.setString(2, email);
            pstmt.executeUpdate();
            pstmt.close();
            conn.close();
            System.out.println("Dane zapisane!");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    private static String loadFromDatabase() {
        StringBuilder result = new StringBuilder();
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn =
DriverManager.getConnection(DB_URL, DB_USER,
DB_PASS);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM
users");
            while (rs.next()) {
                result.append(rs.getInt("id")).append(") ")
                      .append(rs.getString("name")).append(" - ")
                      .append(rs.getString("email")).append("\n");
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return result.toString();
    }
}
```

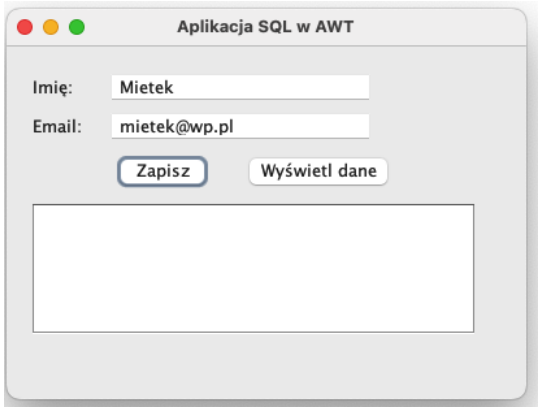
Przykład zadania z zapisem do bazy danych

Oczywiście potrzebujemy prostej bazy danych:

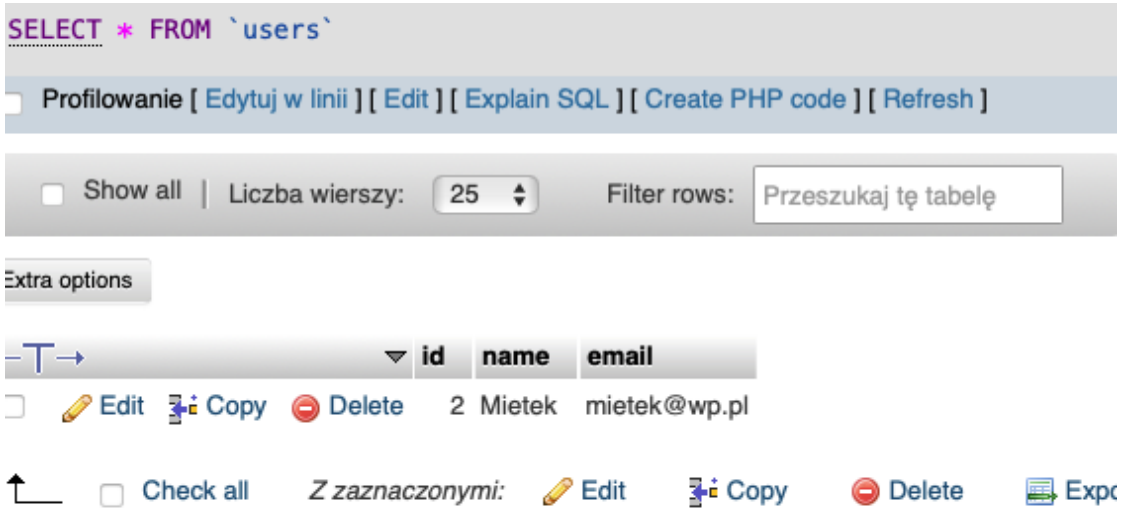
```
CREATE DATABASE test_db;
USE test_db;
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255)
);
```

Działanie programu





Konieczne jest JDBC umieszczone w katalogu “Libraries”, gdzie znajdują się wszystkie biblioteki projektu.



Opis działania?

Wpisujemy w formularz. Po naciśnięciu przycisku zapisujemy w pliku.

Przykład zadania z odczytem z bazy

(wyszukiwarka)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class FormularzBazySwing extends JFrame {
    private JTextField poleImie, poleNazwisko, poleSzukaj;
    private JTextArea obszarWynikow;
    private JButton przyciskZapisz, przyciskPokaz, przyciskSzukaj;

    // Ustawienia bazy danych
    private final String DB_URL = "jdbc:mysql://localhost:3306/twojabaza"; // Zmień nazwę bazy
    private final String DB_USER = "root";
    private final String DB_PASS = "root"; // Dostosuj hasło

    public FormularzBazySwing() {
        setTitle("Formularz użytkownika - baza danych");
        setSize(450, 450);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        setLocationRelativeTo(null);

        // Panel formularza
        JPanel panelForm = new JPanel(new GridLayout(4, 2, 10, 10));
        panelForm.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        panelForm.add(new JLabel("Imię:"));
        poleImie = new JTextField();
        panelForm.add(poleImie);

        panelForm.add(new JLabel("Nazwisko:"));
        poleNazwisko = new JTextField();
        panelForm.add(poleNazwisko);

        przyciskZapisz = new JButton("Zapisz");
        przyciskPokaz = new JButton("Pokaż dane");

        // Pole wyszukiwania
        panelForm.add(new JLabel("Szukaj (imie/nazwisko):"));
        poleSzukaj = new JTextField();
        panelForm.add(poleSzukaj);

        przyciskSzukaj = new JButton("Szukaj");

        panelForm.add(przyciskZapisz);
        panelForm.add(przyciskPokaz);
        panelForm.add(przyciskSzukaj);

        add(panelForm, BorderLayout.NORTH);

        // Obszar wyników
        obszarWynikow = new JTextArea();
        obszarWynikow.setEditable(false);
        add(new JScrollPane(obszarWynikow), BorderLayout.CENTER);

        // Obsługa przycisków
        przyciskZapisz.addActionListener(e -> zapiszDoBazy());
        przyciskPokaz.addActionListener(e -> pokazDaneZBazy());
        przyciskSzukaj.addActionListener(e -> szukajDanych());

        setVisible(true);
    }

    private void zapiszDoBazy() {
        String imie = poleImie.getText().trim();
        String nazwisko = poleNazwisko.getText().trim();

        if (imie.isEmpty() || nazwisko.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Wpisz imię i nazwisko.");
            return;
        }

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS)) {
            String sql = "INSERT INTO uzytkownicy (imie, nazwisko) VALUES (?, ?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, imie);
            stmt.setString(2, nazwisko);
            stmt.executeUpdate();

            JOptionPane.showMessageDialog(this, "Zapisano dane!");
            poleImie.setText("");
            poleNazwisko.setText("");
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Błąd zapisu do bazy.");
        }
    }

    private void pokazDaneZBazy() {
        obszarWynikow.setText("");

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS)) {
            String sql = "SELECT * FROM uzytkownicy";
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                int id = rs.getInt("id");
                String imie = rs.getString("imie");
                String nazwisko = rs.getString("nazwisko");
                obszarWynikow.append(id + ". " + imie + " " + nazwisko + "\n");
            }

        } catch (SQLException ex) {
            ex.printStackTrace();
            obszarWynikow.setText("Błąd odczytu z bazy.");
        }
    }

    private void szukajDanych() {
        String zapytanie = poleSzukaj.getText().trim();

        if (zapytanie.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Wpisz dane do wyszukania.");
            return;
        }

        obszarWynikow.setText("");

        try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS)) {
            String sql = "SELECT * FROM uzytkownicy WHERE imie LIKE ? OR nazwisko LIKE ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, "%" + zapytanie + "%");
            stmt.setString(2, "%" + zapytanie + "%");

            ResultSet rs = stmt.executeQuery();

            while (rs.next()) {
                int id = rs.getInt("id");
                String imie = rs.getString("imie");
                String nazwisko = rs.getString("nazwisko");
                obszarWynikow.append(id + ". " + imie + " " + nazwisko + "\n");
            }

        } catch (SQLException ex) {
            ex.printStackTrace();
            obszarWynikow.setText("Błąd wyszukiwania w bazie.");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(FormularzBazySwing::new);
    }
}
```

Wykorzystanie klasy HashSET

HashSet to klasa w Java, która jest częścią kolekcji frameworku. Służy do przechowywania unikalnych elementów i bazuje na strukturze haszującej.

Cechy HashSet:

- Przechowuje unikalne elementy.
- Nie gwarantuje zachowania kolejności elementów.
- Dopuszcza null jako wartość.
- Szybka operacja wstawiania, usuwania i wyszukiwania elementów (O(1)).

Najważniejsze metody HashSET

- add(E e) – Dodaje element do zestawu.
- remove(Object o) – Usuwa element z zestawu.
- contains(Object o) – Sprawdza, czy element istnieje w zestawie.
- size() – Zwraca liczbę elementów w zestawie.
- clear() – Usuwa wszystkie elementy.

Przykład użycia HashSET

```
import java.util.HashSet;

public class HashSetExample {
    public static void main(String[] args) {

        HashSet<String> set = new HashSet<>();

        set.add("Java");
        set.add("Python");
        set.add("C++");
        set.add("JavaScript");

        System.out.println("Elementy w HashSet: " + set);

        if (set.contains("Java")) {
            System.out.println("HashSet zawiera element 'Java'.");
        }

        set.remove("C++");
        System.out.println("Po usunięciu 'C++': " + set);
    }
}
```

Omówienie:
Tworzymy zbiór (kolekcję) danych, następnie wyświetlamy je, sprawdzamy czy dany element istnieje i usuwamy.

Prawda, że proste?

Zapis do pliku z HashSET

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashSet;

public class HashSetToFile {
    public static void main(String[] args) {
        HashSet<String> set = new HashSet<>();
        set.add("Apple");
        set.add("Orange");
        set.add("Banana");
        try (BufferedWriter writer = new BufferedWriter(new FileWriter("hashset_output.txt"))) {
            for (String element : set) {
                writer.write(element);
                writer.newLine();
            }
            System.out.println("Elementy zapisano do pliku.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Przykład zapisu do pliku z wieloma kolumnami

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashSet;


public class HashSetToMultiColumnTableFile {
   public static void main(String[] args) {
      // Tworzymy HashSet i dodajemy dane w formacie
      [element:kategoria]
      HashSet<String> set = new HashSet<>();
      set.add("1:Apple:Fruit");
      set.add("2:Orange:Fruit");
      set.add("3:Carrot:Vegetable");
      set.add("4:Grapes:Fruit");
      set.add("5:Potato:Vegetable");

 // Plik, do którego zapisujemy
      String fileName =
      "hashset_multicolumn_table.txt";
       try (BufferedWriter writer = new
BufferedWriter(new FileWriter(fileName))) {
   // Zapisujemy nagłówki tabeli
   writer.write("+----+--------------+-------------+");
   writer.newLine();
   writer.write("| ID | Element      | Kategoria   |");
   writer.newLine();
   writer.write("+----+--------------+-------------+");
   writer.newLine();

   // Zapisujemy dane z HashSet do tabeli
   for (String data : set) {
      // Rozdzielamy element na kolumny (ID,
Element, Kategoria)
      String[] parts = data.split(":");
      String id = parts[0];
      String element = parts[1];
      String category = parts[2];

      // Formatowanie wiersza
      writer.write(String.format("| %-2s | %-12s | %-
11s |", id, element, category));
      writer.newLine();
      writer.write("+----+--------------+-------------+");
      writer.newLine();
   }

   System.out.println("Elementy zapisano w
formie tabeli z wieloma kolumnami do pliku: " +
fileName);
   } catch (IOException e) {
      e.printStackTrace();
   }
  }
}
```

Co to są te „procenty” i „eS-y”?

Dzięki temu „wynalazkowi” można sformatować dane w komórce tabeli, np.:

%[wypełnienie][wyrównanie][szerokość][.precyzja]typ

Przykład:

%2s: Szerokość pola wynosi 2, wypełnienie spacjami.
%-2s: Szerokość pola wynosi 2, wyrównanie do lewej.
%05d: Wyrównanie liczbowe, wypełnienie zerami do szerokości 5.

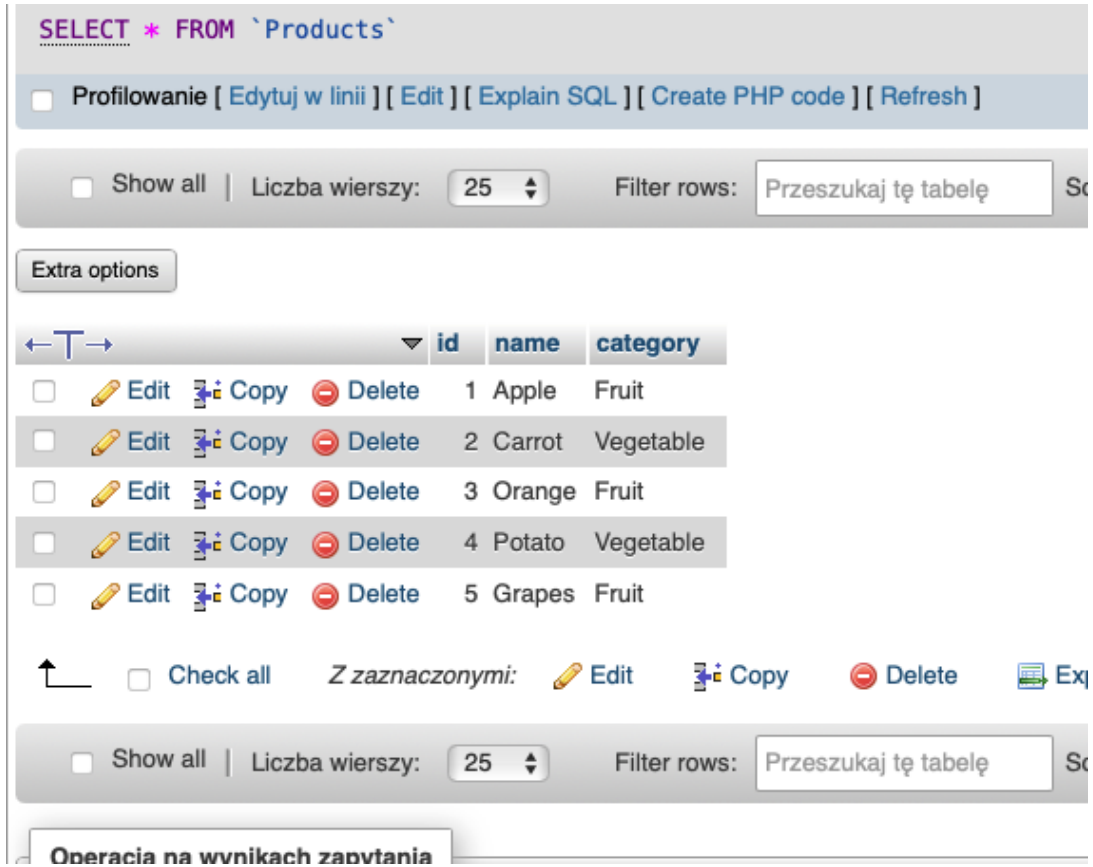
Wybrane specyfikatory

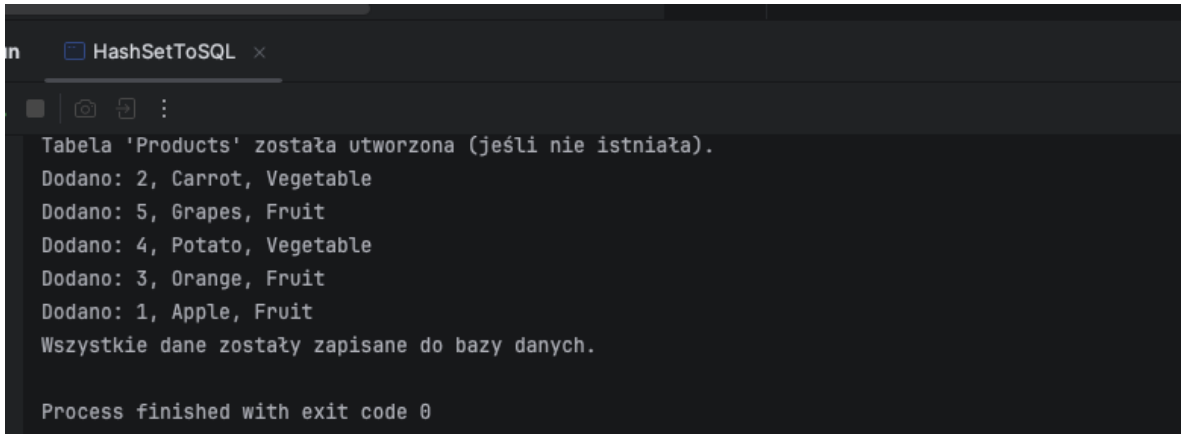
Specyfikator	Opis
%s	Ciąg znaków (String)
%d	Liczba całkowita (int)
%f	Liczba zmiennoprzecinkowa
%x	Liczba całkowita w formacie szesnastkowym
%o	Liczba całkowita w formacie ósemkowym
%n	Znak nowej linii (platformowo niezależny, np. \n)
%5d	Liczba całkowita wyrównana do prawej z szerokością 5 znaków
%-5s	Ciąg znaków wyrównany do lewej z szerokością 5 znaków
%05d	Liczba całkowita wyrównana do prawej, wypełniona zerami (szerokość 5)

Zapis HashSET do bazy SQL

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.Statement;
import java.util.HashSet;

public class HashSetToSQL {
    public static void main(String[] args) {
        // HashSet przechowuje dane w formacie: ID:Nazwa:Kategoria
        HashSet<String> set = new HashSet<>();
        set.add("1:Apple:Fruit");
        set.add("2:Carrot:Vegetable");
        set.add("3:Orange:Fruit");
        set.add("4:Potato:Vegetable");
        set.add("5:Grapes:Fruit");

        // Parametry połączenia z bazą danych
        String url = "jdbc:mysql://localhost:8889/testdb"; // Zmień 'testdb' na swoją bazę danych
        String user = "root"; // Zmień na swoją nazwę użytkownika MySQL
        String password = "root"; // Zmień na swoje hasło MySQL

        // SQL do tworzenia tabeli
        String createTableSQL = "CREATE TABLE IF NOT EXISTS Products (" +
                "id INT PRIMARY KEY, " +
                "name VARCHAR(50), " +
                "category VARCHAR(50)" +
                ");";

        // SQL do wstawiania danych
        String insertSQL = "INSERT INTO Products (id, name, category) VALUES (?, ?, ?)";

        try (Connection connection = DriverManager.getConnection(url, user, password);
             Statement statement = connection.createStatement();
             PreparedStatement preparedStatement = connection.prepareStatement(insertSQL)) {

            // Tworzenie tabeli
            statement.execute(createTableSQL);
            System.out.println("Tabela 'Products' została utworzona (jeśli nie istniała).");

            // Iterowanie po HashSet i zapisywanie danych do tabeli
            for (String data : set) {
                String[] parts = data.split(":"); // Rozdzielamy dane na kolumny
                int id = Integer.parseInt(parts[0]);
                String name = parts[1];
                String category = parts[2];

                // Ustawianie wartości w zapytaniu SQL
                preparedStatement.setInt(1, id);
                preparedStatement.setString(2, name);
                preparedStatement.setString(3, category);

                // Wykonanie zapytania
                preparedStatement.executeUpdate();
                System.out.println("Dodano: " + id + ", " + name + ", " + category);
            }

            System.out.println("Wszystkie dane zostały zapisane do bazy danych.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Zapis HashSET do bazy SQL - efekt





Achtung!

Prezentowany kod napisano pod MacOS oraz serwer MAMP.

W skrócie:

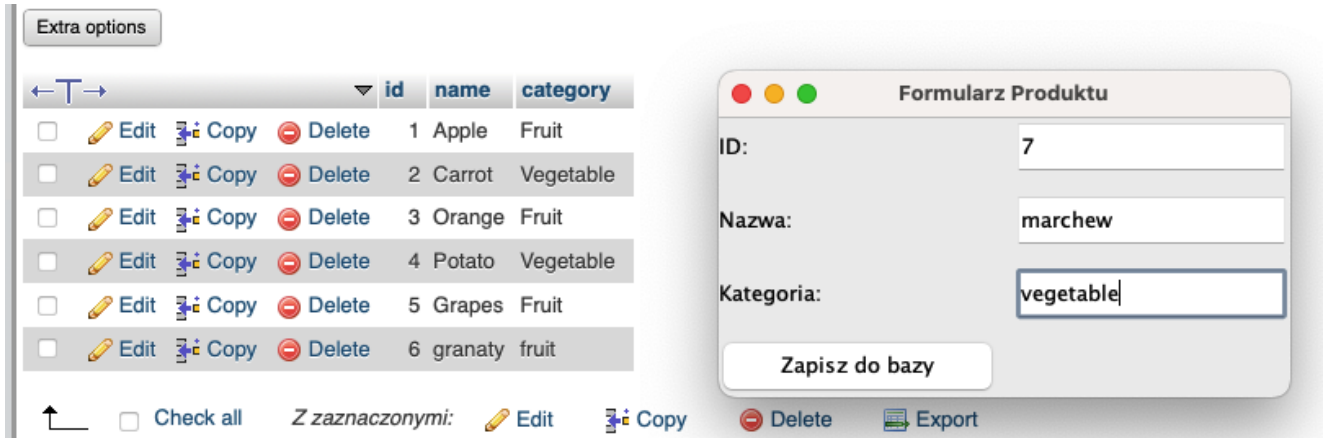
- wymaga domyślnego hasła „root”, inne serwery nie potrzebują;
- wymaga innych numerów portów – zamiast 3306 i 3307 jest 8888 i 8889;
- Konieczny jest mysql-connector umieszczony w „libraries”.

... reszta bez zmian.

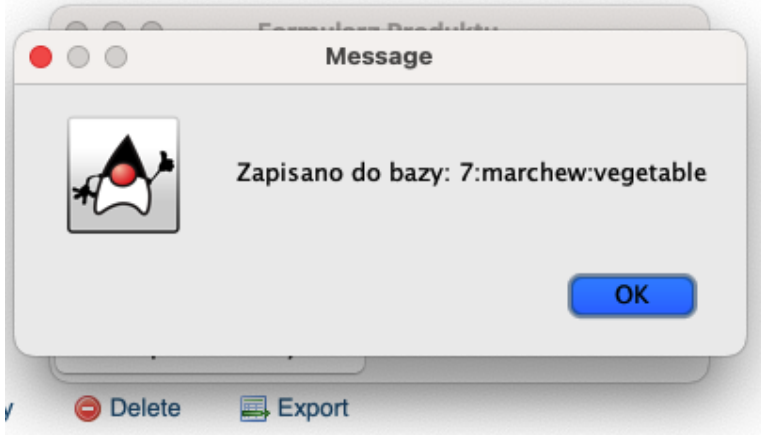
HashSET+baza+formularz

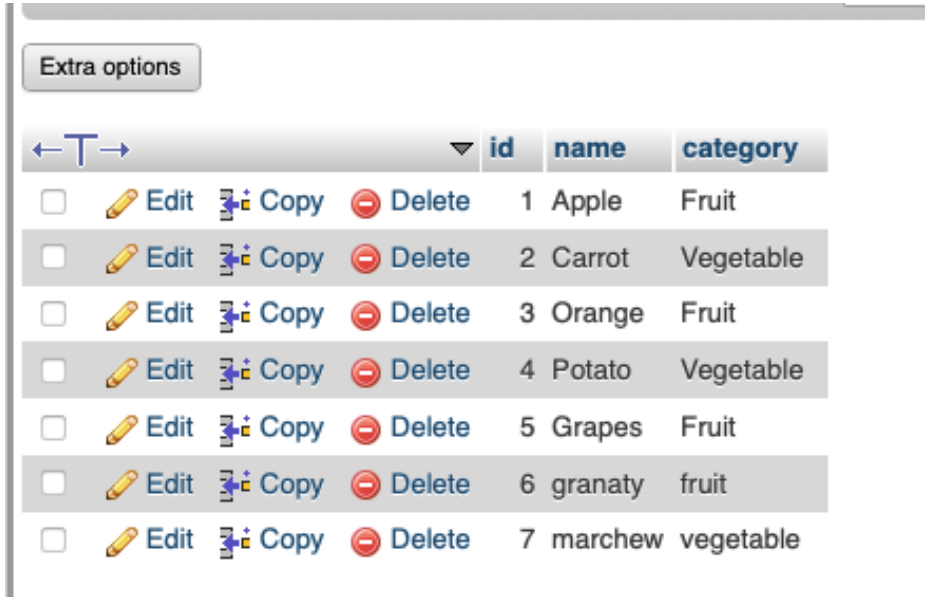
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.util.HashSet;

public class HashSetFormToSQL extends JFrame {

    private JTextField idField, nameField, categoryField;
    private JButton saveButton;
    private HashSet<String> dataSet = new
HashSet<>();

    // Połączenie z bazą
    private final String url =
"jdbc:mysql://localhost:8889/testdb"; // Zmień
port/bazę jeśli trzeba
    private final String user = "root";
    private final String password = "root"; // lub "root"
na MAMP

    public HashSetFormToSQL() {
        super("Formularz Produktu");


setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(350, 200);
        setLayout(new GridLayout(4, 2, 10, 10));

        // Pola formularza
        add(new JLabel("ID:"));
        idField = new JTextField();
        add(idField);

        add(new JLabel("Nazwa:"));
        nameField = new JTextField();
        add(nameField);

        add(new JLabel("Kategoria:"));
        categoryField = new JTextField();
        add(categoryField);

        // Przycisk zapisu
        saveButton = new JButton("Zapisz do bazy");
        add(saveButton);

        // Puste pole dla wyrównania siatki
        add(new JLabel(""));

        saveButton.addActionListener(new
ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveData();
            }
        });

        // Tworzenie tabeli przy starcie
        createTableIfNotExists();

        setVisible(true);
    }

    private void createTableIfNotExists() {
        String createTableSQL = "CREATE TABLE IF NOT
EXISTS Products (" +
                "id INT PRIMARY KEY, " +
                "name VARCHAR(50), " +
                "category VARCHAR(50)" +
                ");";
        try (Connection conn =
DriverManager.getConnection(url, user, password);
             Statement stmt = conn.createStatement()) {
            stmt.execute(createTableSQL);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Błąd
tworzenia tabeli: " + e.getMessage());
        }
    }

    private void saveData() {
        String idText = idField.getText().trim();
        String name = nameField.getText().trim();
        String category = categoryField.getText().trim();

        if (idText.isEmpty() || name.isEmpty() ||
category.isEmpty()) {
            JOptionPane.showMessageDialog(this,
"Wszystkie pola muszą być wypełnione!");
            return;
        }

        try {
            int id = Integer.parseInt(idText);
            String dataString = id + ":" + name + ":" +
category;

            if (dataSet.contains(dataString)) {
                JOptionPane.showMessageDialog(this, "Ten
wpis już istnieje!");
                return;
            }

            dataSet.add(dataString);

            // Zapis do bazy danych
            String insertSQL = "INSERT INTO Products (id,
name, category) VALUES (?, ?, ?)";
            try (Connection conn =
DriverManager.getConnection(url, user, password);
                 PreparedStatement pstmt =
conn.prepareStatement(insertSQL)) {

                pstmt.setInt(1, id);
                pstmt.setString(2, name);
                pstmt.setString(3, category);

                pstmt.executeUpdate();
                JOptionPane.showMessageDialog(this,
"Zapisano do bazy: " + dataString);

                // Czyścimy pola
                idField.setText("");
                nameField.setText("");
                categoryField.setText("");
            }

        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "ID
musi być liczbą!");
        } catch (SQLIntegrityConstraintViolationException
ex) {
            JOptionPane.showMessageDialog(this, "ID już
istnieje w bazie!");
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Błąd
zapisu: " + ex.getMessage());
        }
    }

    public static void main(String[] args) {
        // Załaduj sterownik JDBC (dla starszych wersji
JDK)
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (Exception e) {
            System.out.println("Nie można załadować
sterownika JDBC");
        }

        SwingUtilities.invokeLater(() -> new
HashSetFormToSQL());
    }
}
```

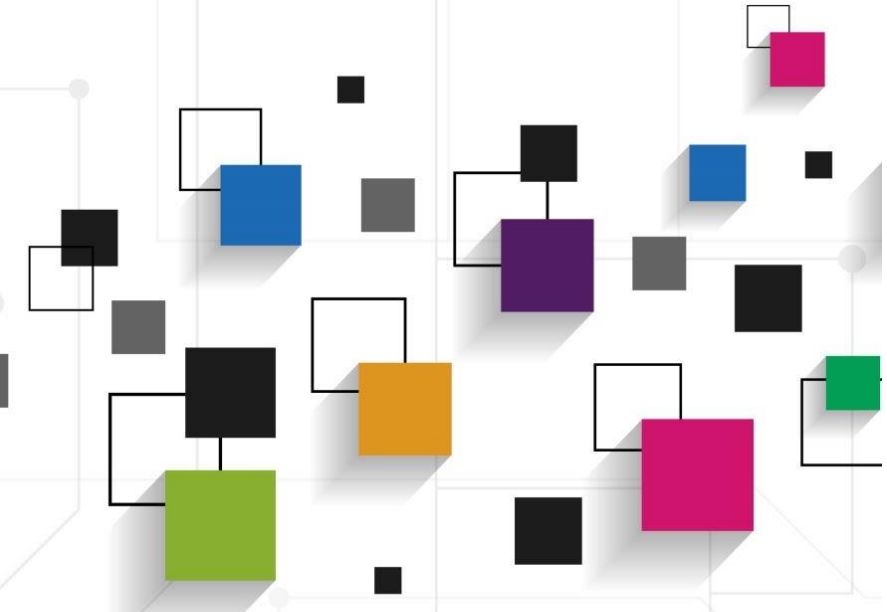
Efekt?


Extra options
id
name
category
Edit Copy Delete 1 Apple Fruit
Edit Copy Delete 2 Carrot Vegetable
Edit Copy Delete 3 Orange Fruit
Edit Copy Delete 4 Potato Vegetable
Edit Copy Delete 5 Grapes Fruit
Edit Copy Delete 6 granaty fruit
Check all
Z zaznaczonymi:
Edit
Copy
Delete
Export

Formularz Produktu
ID:
7
Nazwa:
marchew
Kategoria:
vegetable
Zapisz do bazy


Message
Zapisano do bazy: 7:marchew:vegetable
OK
Delete
Export


Extra options
id
name
category
Edit Copy Delete 1 Apple Fruit
Edit Copy Delete 2 Carrot Vegetable
Edit Copy Delete 3 Orange Fruit
Edit Copy Delete 4 Potato Vegetable
Edit Copy Delete 5 Grapes Fruit
Edit Copy Delete 6 granaty fruit
Edit Copy Delete 7 marchew vegetable



DAO

DAO (Data Access Object) to wzorzec projektowy, który oddziela logikę aplikacji od szczegółów dostępu do danych, takich jak połączenia z bazą danych. Jego głównym celem jest zapewnienie abstrakcji, dzięki której aplikacja może łatwiej zarządzać danymi, niezależnie od tego, czy pochodzą one z bazy danych, pliku, czy innego źródła.

Cechy DAO

- **Abstrakcja dostępu do danych** – DAO ukrywa szczegóły związane z komunikacją z bazą danych (np. SQL) przed resztą aplikacji. Pozwala to na łatwą zmianę sposobu przechowywania danych (np. zmiana bazy danych) bez konieczności modyfikowania reszty aplikacji.
- **Izolacja logiki biznesowej** – DAO oddziela kod odpowiedzialny za manipulację danymi (np. zapytania SQL) od logiki aplikacji. Dzięki temu kod jest bardziej modularny i łatwiejszy do testowania.
- **Wykonuje operacje CRUD** – DAO oferuje metody do wykonania podstawowych operacji na danych:
- **C**reate (tworzenie danych)
- **R**ead (odczyt danych)
- **U**pdate (aktualizacja danych)
- **D**elete (usuwanie danych)
- **Obsługuje połączenie z bazą danych** – DAO zarządza połączeniami z bazą danych i zapewnia, że połączenie jest otwierane i zamykane w odpowiednich momentach.

Przykład

```
package dao;

import model.Entry;
import util.Database;

import java.sql.*;
import java.util.*;

public class EntryDAO {

  // Pobiera wszystkie wpisy
  public List<Entry> getAllEntries() throws SQLException {
    List<Entry> entries = new ArrayList<>();
    Connection conn = Database.getConnection();
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM entries");

    while (rs.next()) {
      entries.add(new Entry(
        rs.getInt("id"),
        rs.getString("title"),
        rs.getString("content")
      ));
    }
    conn.close();
    return entries;
  }

  // Dodaje nowy wpis
  public void addEntry(Entry entry) throws SQLException {
    Connection conn = Database.getConnection();
    PreparedStatement stmt = conn.prepareStatement(
      "INSERT INTO entries (title, content) VALUES (?, ?)"
    );
    stmt.setString(1, entry.getTitle());
    stmt.setString(2, entry.getContent());
    stmt.executeUpdate();
    conn.close();
  }

  // Usuwa wpis po ID
  public void deleteEntry(int id) throws SQLException {
    Connection conn = Database.getConnection();
    PreparedStatement stmt = conn.prepareStatement(
      "DELETE FROM entries WHERE id = ?"
    );
    stmt.setInt(1, id);
    stmt.executeUpdate();
    conn.close();
  }
}
```

Składniki DAO

W skład DAO wchodzi:

1. **Klasa DAO** – np. EntryDAO, która zawiera metody umożliwiające interakcję z danymi.
2. **Metody CRUD** – implementacja operacji: tworzenie, odczyt, aktualizacja, usuwanie danych w bazie.
3. **Połączenia z bazą danych** – w tym przykładzie jest to zarządzane przez klasę Database.

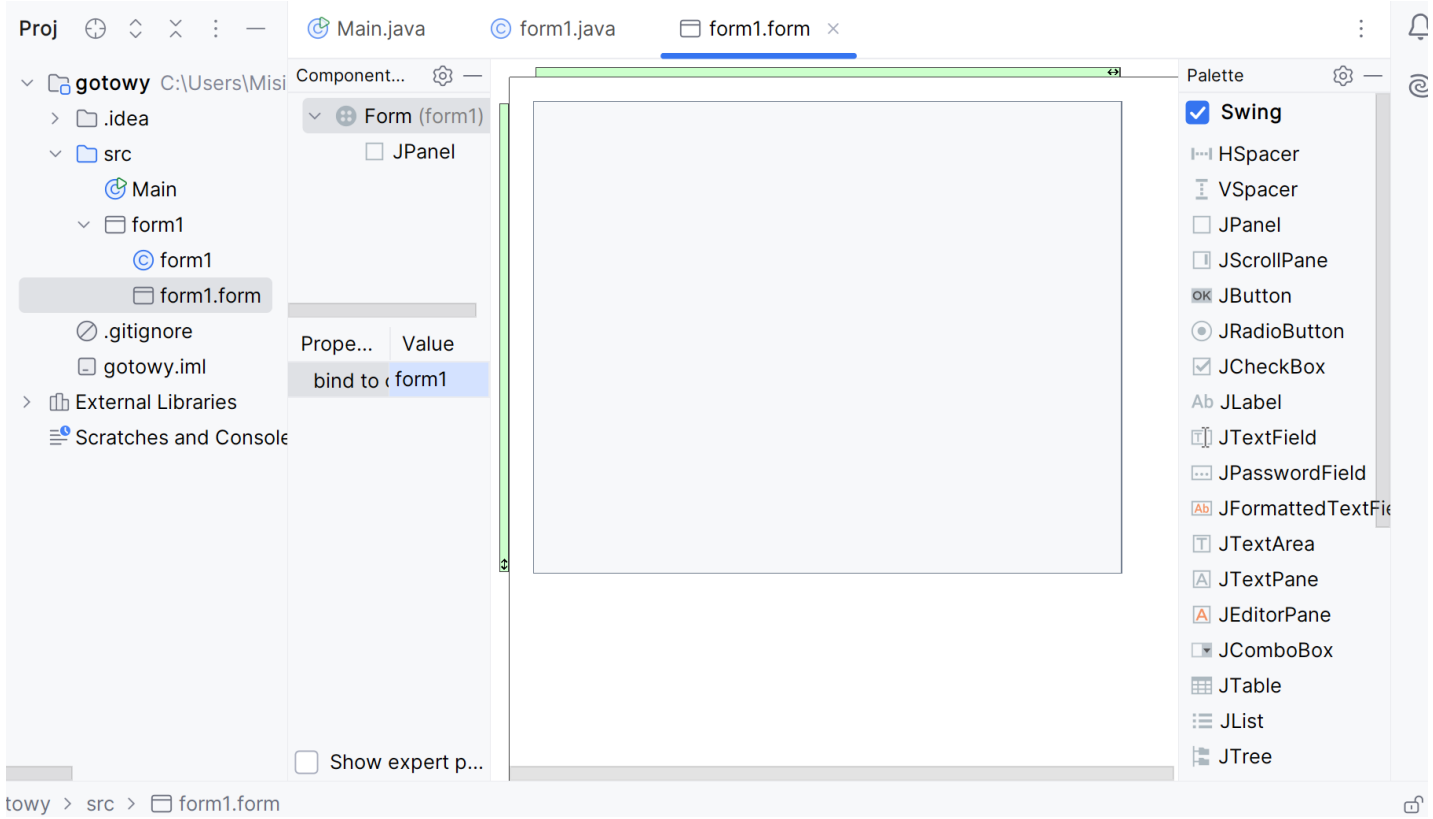
Zalety stosowania DAO:

- **Separation of Concerns**: Logika dostępu do danych jest oddzielona od logiki aplikacji, co ułatwia rozwój i testowanie.
- **Łatwiejsza zmiana bazy danych**: Można zmienić implementację dostępu do danych (np. zmiana bazy SQL na NoSQL) bez zmiany kodu aplikacji.
- **Reusability**: Kod dostępu do danych jest wielokrotnie używalny w różnych częściach aplikacji.
- DAO to bardzo popularny wzorzec projektowy w aplikacjach, które korzystają z baz danych, zwłaszcza w kontekście Javy i innych języków o silnym typowaniu i strukturze.

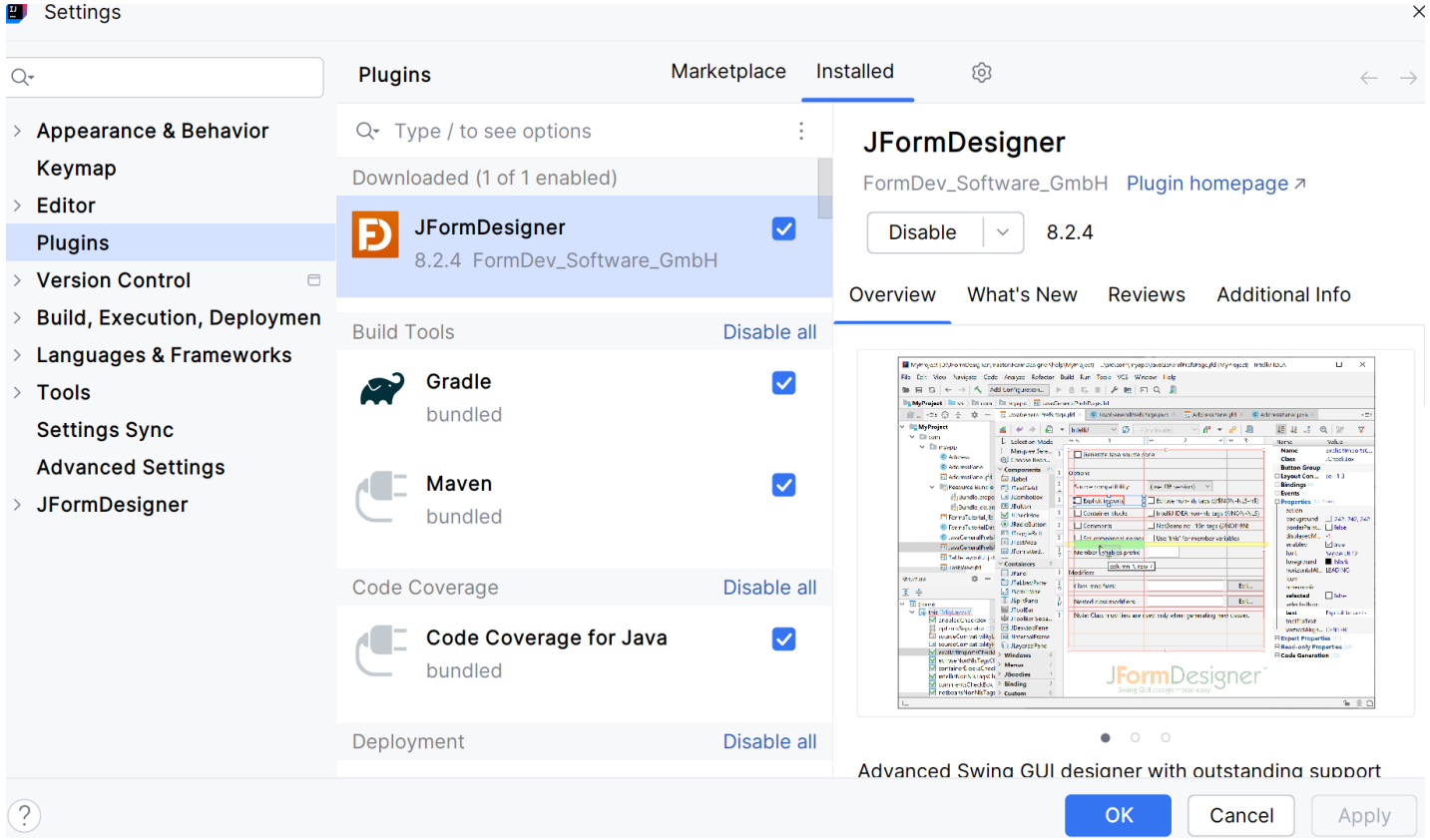
Wtyczki i rozszerzenia do tworzenia IU

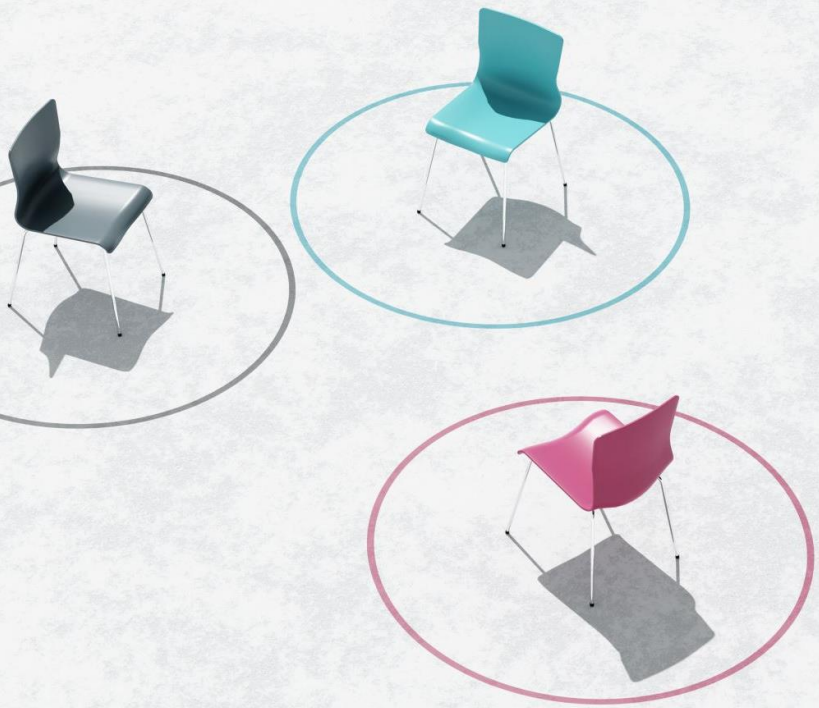
- **JavaFX Scene Builder** - narzędzie do tworzenia interfejsów graficznych dla aplikacji JavaFX. Scene Builder generuje pliki FXML, które mogą być w łatwy sposób zintegrowane z kodem Javy.
- **Swing GUI Builder** (dostępny w IDE, takich jak IntelliJ IDEA lub NetBeans) - pozwala na projektowanie interfejsu użytkownika dla aplikacji Swing.
- **Eclipse WindowBuilder** - wtyczka do Eclipse, która pozwala na tworzenie interfejsów w Swing, SWT, a także GWT.

JForm Designer



Jak zainstalować i włączyć?





Łączenie pliku z „wyglądem” z kodem

Utwórz nowy projekt:

- **Otwórz IntelliJ IDEA.**
- **Wybierz New Project → Java → Skonfiguruj SDK → Kliknij Finish.**

Dodaj plik formularza GUI:

- **Kliknij prawym przyciskiem myszy na folderze src.**
- **Wybierz New → GUI Form.**
- **Nadaj nazwę swojemu plikowi, np. MyForm.**
- **IntelliJ stworzy dwa pliki:**
- **MyForm.form (plik wizualny GUI).**
- **MyForm.java (kod odpowiadający formularzowi).**

Otwórz plik .form w GUI Designer:

- **Otwórz plik MyForm.form.**
- **IntelliJ przełączy się do trybu projektowania.**
- **Możesz przeciągać i upuszczać komponenty (np. JButton, JLabel, JTextField) na okno formularza.**

Dodaj komponenty i właściwości:

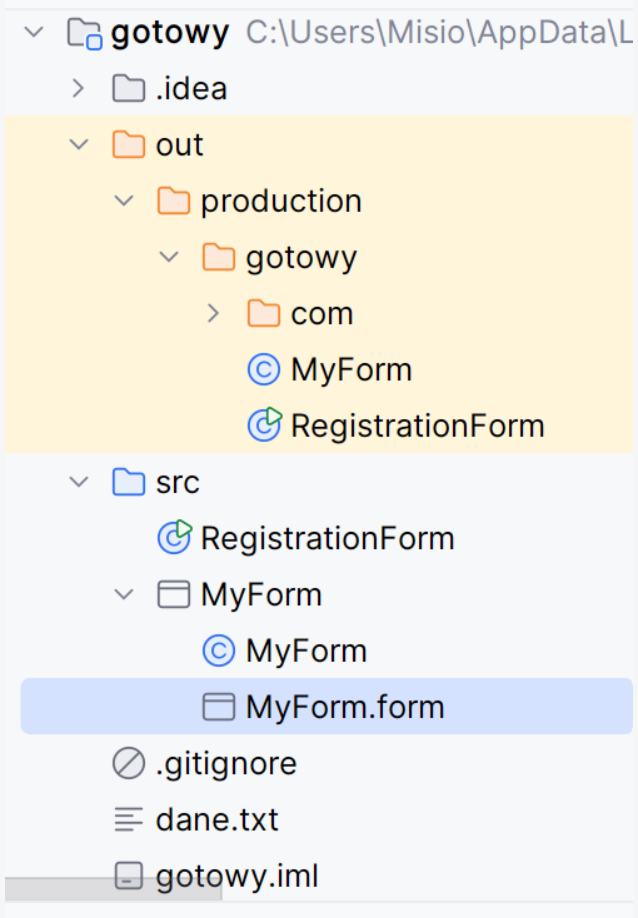
- **Dodaj na przykład:**
- **JLabel (np. "Witaj w aplikacji").**
- **JTextField (pole tekstowe).**
- **JButton (przycisk z napisem "Kliknij mnie").**
- **Ustaw ich właściwości w zakładce Properties.**

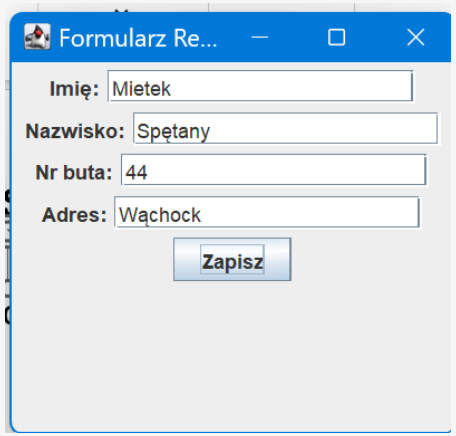
Generowanie kodu przez IntelliJ IDEA:

- **IntelliJ automatycznie wygeneruje kod w pliku MyForm.java. Kod ten będzie zawierał konstruktor, który ładuje projekt GUI.**

Dodaj obsługę zdarzeń w IntelliJ IDEA:

- **Kliknij komponent (np. przycisk) w trybie projektowania.**
- **W zakładce Properties wybierz zakładkę Events.**
- **Wybierz actionPerformed i kliknij dwukrotnie, aby utworzyć metodę obsługi zdarzenia.**





Przykład zastosowania JForm

Prosty formularz zapisujący dane do pliku tekstowego.

Folder „MyForm” zawiera Layout formularza oraz klasę, która automatycznie generuje się po utworzeniu formularza.

Wewnątrz „COM” znajdują się:
MyForm – klasa łącząca Layout z plikiem RegistrationForm
RegistrationForm – klasa, w której znajduje się cała logika aplikacji.

Przykład zastosowania JForm – kody (MyForm – klasa łącząca)

```
import javax.swing.JButton;
import javax.swing.JTextField;

public class MyForm {
    private JTextField textField1;
    private JTextField textField2;
    private JTextField textField3;
    private JTextField textField4;
    private JButton zapiszButton;

    public MyForm() {
        this.$$$setupUI$$$();
    }
}
```

Przykład zastosowania JForm – kody (RegistrationForm)

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class RegistrationForm {
   private JPanel mainPanel = new
JPanel();
   private JTextField textField1 = new
JTextField(20);
   private JTextField textField2 = new
JTextField(20);
   private JTextField textField3 = new
JTextField(20);
   private JTextField textField4 = new
JTextField(20);
   private JButton saveButton = new
JButton("Zapisz");

   public RegistrationForm() {
      this.mainPanel.add(new
JLabel("Imię:"));
      this.mainPanel.add(this.textField1);
      this.mainPanel.add(new
JLabel("Nazwisko:"));
      this.mainPanel.add(this.textField2);
      this.mainPanel.add(new JLabel("Nr
buta:"));
      this.mainPanel.add(this.textField3);
      this.mainPanel.add(new
JLabel("Adres:"));
      this.mainPanel.add(this.textField4);
      this.mainPanel.add(this.saveButton);

this.saveButton.addActionListener(new
ActionListener() {
         public void
actionPerformed(ActionEvent e) {
            String content =
String.format("Imię: %s\nNazwisko:
%s\nNr buta: %s\nAdres: %s\n",
RegistrationForm.this.textField1.getText()
,
RegistrationForm.this.textField2.getText()
,
RegistrationForm.this.textField3.getText()
,
RegistrationForm.this.textField4.getText()
);

            try {
               BufferedWriter writer = new
BufferedWriter(new
FileWriter("dane.txt", true));

               try {
                  writer.write(content);
                  writer.write("----------
-------\n");

JOptionPane.showMessageDialog(Registr
ationForm.this.mainPanel, "Dane
zapisane pomyślnie!", "Sukces", 1);
               } catch (Throwable var7) {
                  try {
                     writer.close();
                  } catch (Throwable var6) {
                     var7.addSuppressed(var6);
                  }

                  throw var7;
               }

               writer.close();
            } catch (IOException var8) {

JOptionPane.showMessageDialog(Registr
ationForm.this.mainPanel, "Błąd zapisu
do pliku!", "Błąd", 0);
            }

         }
      });
   }

   public static void main(String[] args) {
      JFrame frame = new
JFrame("Formularz Rejestracyjny");
      frame.setContentPane((new
RegistrationForm()).mainPanel);
      frame.setDefaultCloseOperation(3);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Klasa MyForm z zawartością Layout

```
import javax.swing.*;

public class MyForm {
    private JTextField textField1;
    private JTextField textField2;
    private JTextField textField3;
    private JTextField textField4;
    private JButton zapiszButton;
}
```

Jak używać qT Designer do Java?

1. Projektowanie UI w Qt Designer

W Qt Designer stwórz interfejs graficzny i zapisz go jako plik .ui.

2. Konwersja .ui do kodu XML

Plik .ui jest w formacie XML, więc możesz go odczytać i przeanalizować w Javie, ale wymaga to dodatkowego kodowania. Nie istnieje natywny sposób konwersji do kodu Java.

3. Własna implementacja interfejsu w Javie

•Przetłumacz strukturę XML na kod Java ręcznie lub napisz parser XML w Javie, który załaduje te elementy interfejsu. Możesz wykorzystać biblioteki XML takie jak javax.xml.parsers lub org.w3c.dom.

4. Tworzenie logiki aplikacji w Javie

•Po załadowaniu interfejsu zaimplementuj logikę aplikacji w Javie, integrując to z bibliotekami graficznymi, takimi jak Swing, JavaFX lub AWT.

5. Alternatywne podejście: Użyj frameworka JNI

Możesz połączyć Javę z C++ używając JNI (Java Native Interface) i załadować komponenty Qt jako natywne widżety. To jednak jest bardzo skomplikowane i rzadko stosowane.

JFreeChart – Excel w JAVA

JFreeChart to popularna, darmowa biblioteka w języku **Java**, która umożliwia tworzenie różnych typów wykresów (np. słupkowych, liniowych, kołowych) w aplikacjach desktopowych oraz webowych. Jest szeroko stosowana w projektach, gdzie wymagane jest wizualne przedstawienie danych.

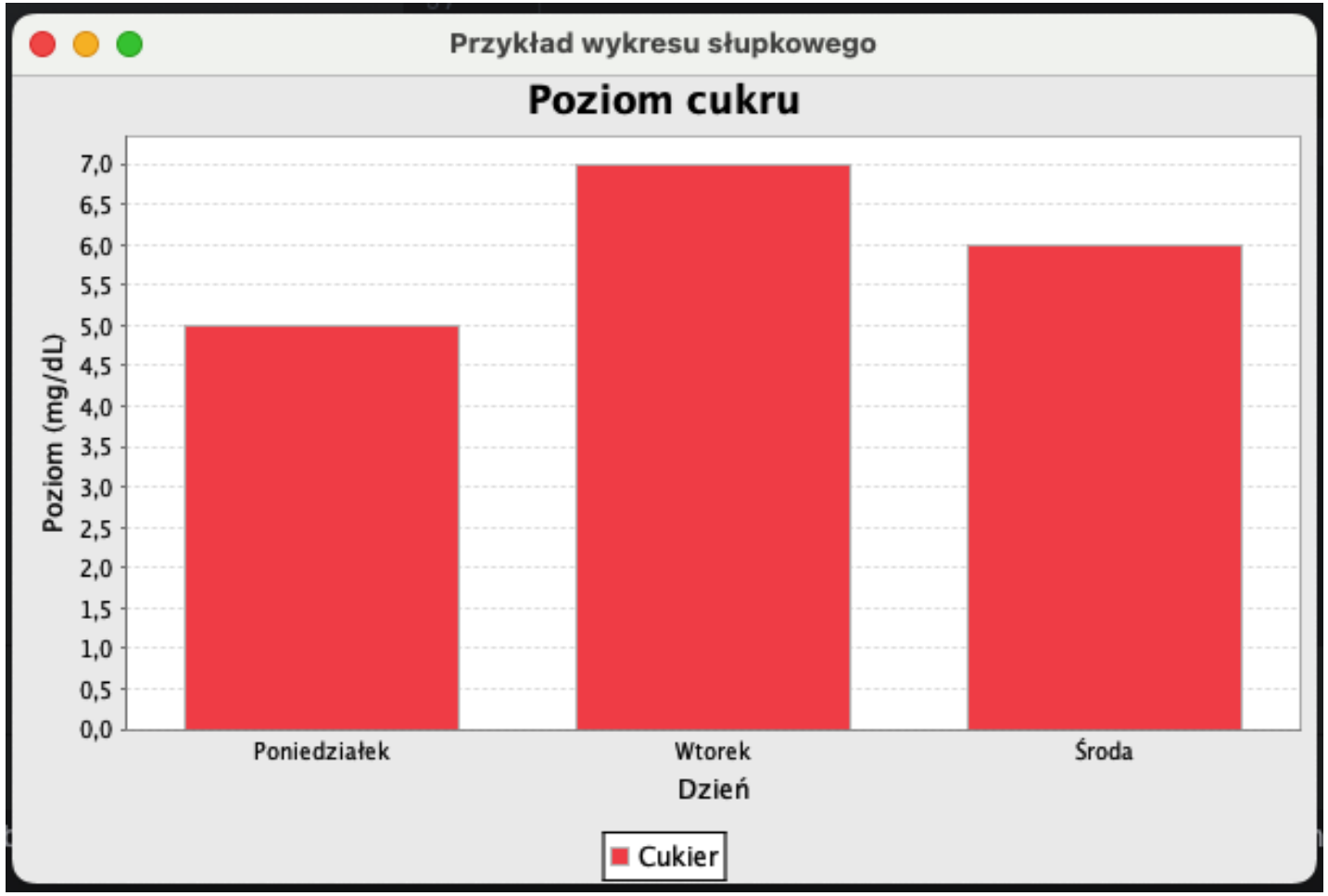
Służy do:

- Tworzenia **wykresów słupkowych, liniowych, kołowych, XY, Gantt**, itp.
- Zapewnia łatwą integrację z **Java Swing** i **JavaFX**.
- Umożliwia **eksport wykresów do PNG, JPEG, PDF**.
- Dostosowania kolorów, legend, tytułów, stylów osi itp.
- Dynamicznej aktualizacji danych (np. do dashboardów).

Przykład – wykres słupkowy

```
import javax.swing.*;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;

public class BarChartExample extends JFrame {

  public BarChartExample() {
    setTitle("Przykład wykresu słupkowego");
    setSize(600, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Tworzymy dane
    DefaultCategoryDataset dataset = new
DefaultCategoryDataset();
    dataset.addValue(5, "Cukier", "Poniedziałek");
    dataset.addValue(7, "Cukier", "Wtorek");
    dataset.addValue(6, "Cukier", "Środa");

    // Tworzymy wykres
    JFreeChart chart = ChartFactory.createBarChart(
      "Poziom cukru",          // Tytuł
      "Dzień",             // Oś X
      "Poziom (mg/dL)",       // Oś Y
      dataset,             // Dane
      PlotOrientation.VERTICAL,  // Orientacja
      true,              // Legenda
      true,              // Tooltips
      false              // URLs
    );

    // Wyświetlamy wykres w panelu
    ChartPanel chartPanel = new ChartPanel(chart);
    setContentPane(chartPanel);
  }

  public static void main(String[] args){
    SwingUtilities.invokeLater(() -> {
      BarChartExample example = new BarChartExample();
      example.setVisible(true);
    });
  }
}
```

Efekt?


Przykład wykresu słupkowego
Poziom cukru
Poziom (mg/dL)
7,0
6,5
6,0
5,5
5,0
4,5
4,0
3,5
3,0
2,5
2,0
1,5
1,0
0,5
0,0
Poniedziałek
Wtorek
Środa
Dzień
Cukier

Formatowanie danych na wykresach

Element	Metoda / Właściwość	Opis działania	Przykład użycia
Tytuł wykresu	chart.setTitle(String)	Ustawia tytuł wykresu	chart.setTitle("Poziom cukru")
	chart.getTitle().setFont(Font)	Czcionka tytułu	new Font("Arial", Font.BOLD, 16)
	chart.getTitle().setPaint(Color)	Kolor tytułu	Color.BLUE
Tło wykresu	chart.setBackgroundPaint(Color)	Tło całego wykresu	Color.LIGHT_GRAY
	plot.setBackgroundPaint(Color)	Tło obszaru danych	Color.WHITE
	plot.setOutlineVisible(boolean)	Obramowanie wykresu	false
Legenda	chart.getLegend().setItemFont(Font)	Czcionka tekstu legendy	new Font("Arial", Font.PLAIN, 12)
	chart.getLegend().setBackgroundPaint(Color)	Tło legendy	Color.WHITE
	chart.getLegend().setFrame(BlockBorder)	Ramka legendy	BlockBorder.NONE

Formatowanie danych na wykresach

Element	Metoda / Właściwość	Opis działania	Przykład użycia
Oś X	plot.getDomainAxis().setLabel(String)	Etykieta osi X	"Dni tygodnia"
	plot.getDomainAxis().setLabelFont(Font)	Czcionka etykiety osi X	Font.BOLD
	plot.getDomainAxis().setTickLabelFont(Font)	Czcionka wartości osi X	Font.PLAIN
Oś Y	plot.getRangeAxis().setLabel(String)	Etykieta osi Y	"Poziom cukru"
	plot.getRangeAxis().setTickLabelPaint(Color)	Kolor liczb osi Y	Color.DARK_GRAY
Siatka	plot.setRangeGridlinesVisible(boolean)	Pokaż/ukryj poziome linie siatki	true
	plot.setRangeGridlinePaint(Color)	Kolor poziomej siatki	Color.GRAY
	plot.setDomainGridlinesVisible(boolean)	Pokaż/ukryj pionowe linie (XYPlot)	true
	plot.setDomainGridlinePaint(Color)	Kolor pionowej siatki	Color.LIGHT_GRAY

Formatowanie danych na wykresach

Element	Metoda / Właściwość	Opis działania	Przykład użycia
Serie / Słupki	renderer.setSeriesPaint(int, Color)	Kolor słupków lub linii	Color.GREEN
	renderer.setDrawBarOutline(boolean)	Pokaż/ukryj obramowanie słupków	false
	renderer.setBarPainter(...)	Styl rysowania słupków	new StandardBarPainter()
Interakcje	chartPanel.setMouseZoomable(boolean)	Zoom myszką	true
	chartPanel.setPopupMenu(null)	Wyłącza menu kontekstowe	null

SPRING

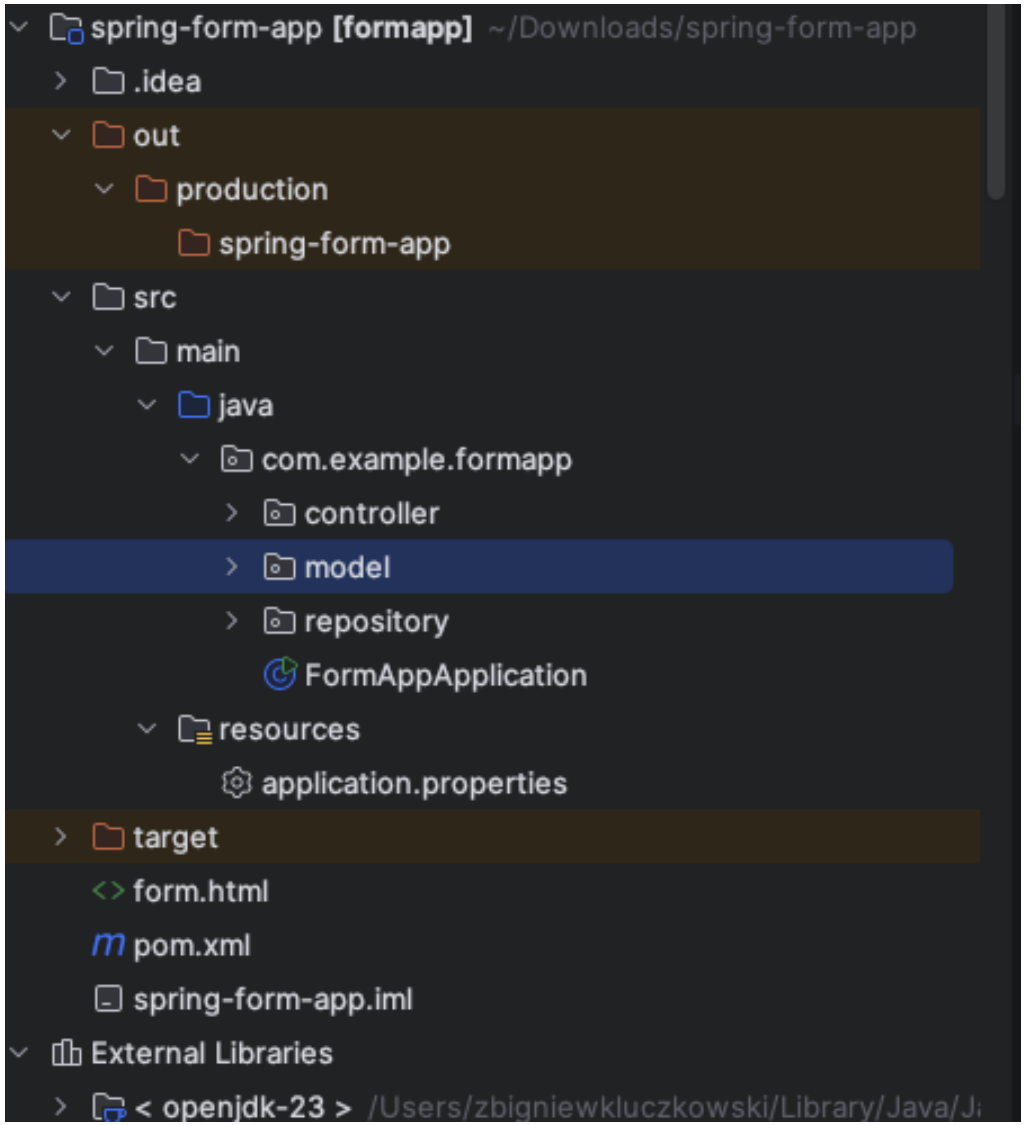
Spring Framework – szkielet tworzenia aplikacji (ang. application framework) w języku Java dla platformy Java Platform, Enterprise Edition (aczkolwiek istnieje też wersja dla środowiska .NET).

Szablon Spring Framework zawiera rozwiązania dla wielu zagadnień technicznych napotykanych przez programistów języka Java oraz organizacji chcących budować aplikację na platformie Java EE. Mnogość komponentów, które zawiera Spring, powoduje, że czasami trudno je rozróżnić. Spring Framework nie jest powiązany tylko i wyłącznie ze środowiskiem Java EE, aczkolwiek jego integracja w tym obszarze jest ważnym powodem jego popularności.

Struktura projektu

- Struktura projektu obejmuje Model, Widok i
- Kontroler:
  - kontroler odpowiada za działanie aplikacji;
  - model przechowuje dane o odpowiedniej strukturze;
  - widok pozwala je wyświetlić.

- W „properties” znajdują się ustawienia projektu a wewnątrz „resources” będą znajdować się pliki graficzne.



W dużym skrócie ...

Połączymy HTML, Javę i XML po to aby wyświetlić dane wprowadzone przez wyszukiwarkę i zapisać je, np. w bazie danych.

Przykład projektu

```
package com.example.demo;


import org.springframework.boot.SpringApplication;

import
org.springframework.boot.autoconfigure.SpringBootA
pplication;


@SpringBootApplication

public class DemoApplication {


   public static void main(String[] args) {

      SpringApplication.run(DemoApplication.class,
args);

   }

}
```

```
package com.example.demo;


import
org.springframework.web.bind.annotation
.GetMapping;

import
org.springframework.web.bind.annotation
.RestController;


@RestController

public class HelloController {


   @GetMapping("/")

   public String hello() {

      return "Witaj w aplikacji Spring Boot!";

   }

}
```

Mapping – ważna część projektu

```
package com.example.demo;


import
org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import
org.springframework.web.bind.annotation.Get
Mapping;


@Controller

public class PageController {


   @GetMapping("/page1")

   public String showPage1(Model model) {

      model.addAttribute("message", "Witaj na
stronie 1!");

      return "page1"; }


   @GetMapping("/page2")

   public String showPage2(Model model) {

      model.addAttribute("message", "Witaj na
stronie 2!");

      return "page2"; }

@GetMapping("/zapping")

   public String
zapping(@org.springframework.web.bind.anno
tation.RequestParam(name = "view",
defaultValue = "page1") String view, Model
model) {

      if ("page2".equals(view)) {

         model.addAttribute("message",
"Przełączono na stronę 2!");

         return "page2";

      } else {

         model.addAttribute("message",
"Przełączono na stronę 1!");

         return "page1";

      }

   }

}
```

Powtórka z HTML

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Formularz</title>
</head>
<body>
  <h1>Formularz wprowadzania danych</h1>

  <p th:text="${message}"></p>

  <!-- Formularz wprowadzania danych -->
  <form action="/submit" method="post">
    <label for="name">Imię:</label>
    <input type="text" id="name" name="name" required><br><br>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required><br><br>

    <button type="submit">Zapisz</button>
  </form>
</body>
</html>
```

Kontroler – najważniejszy dla poprawnego działania aplikacji

```
package com.example.demo;


import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;


import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;


@Controller

public class FormController {


  // Wyświetlanie formularza

  @GetMapping("/form")

  public String showForm() {

    return "form";

  }

  // Obsługa danych z formularza

  @PostMapping("/submit")

  public String submitForm(@RequestParam String name,
@RequestParam String email, Model model) {

    // Zapisywanie danych do pliku

    try (BufferedWriter writer = new BufferedWriter(new
FileWriter("data.txt", true))) {

      writer.write("Name: " + name + ", Email: " + email + "\n");

    } catch (IOException e) {

      e.printStackTrace();

      model.addAttribute("message", "Błąd podczas zapisywania
danych do pliku.");

      return "form";

    }


    model.addAttribute("message", "Dane zostały zapisane.");

    return "form";

  }

}
```

Prosty przykład - formularz

Struktura katalogów i plików:

```
simple-form-spring/
├── src/
│   └── main/
│       ├── java/com/example/simpleform/
│       │   ├── SimpleFormApplication.java
│       │   └── controller/FormController.java
│       └── resources/
│           ├── templates/
│           │   ├── form.html
│           │   └── result.html
│           └── application.properties
├── pom.xml
```


```
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.example</groupId>
 <artifactId>simple-form-spring</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <packaging>jar</packaging>
 <name>Simple Form Spring</name>
 <description>Simple Spring Boot form
example</description>

 <parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.2.5</version>
 </parent>

 <dependencies>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
 </dependencies>

 <build>
  <plugins>
   <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
   </plugin>
  </plugins>
 </build>
</project>
```

Prosty przykład - formularz

Klasa główna:

```
package com.example.simpleform;


import org.springframework.boot.SpringApplication;

import
org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication

public class SimpleFormApplication {

   public static void main(String[] args) {

      SpringApplication.run(SimpleFormApplication.class, args);

   }

}
```


```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <title>Wynik</title>
</head>
<body>
   <h1>Otrzymano dane:</h1>
   <p>Imię: <span th:text="${imie}"></span></p>
   <p>Nazwisko: <span th:text="${nazwisko}"></span></p>
</body>
</html>
```

Kontroler:

```
package com.example.simpleform.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class FormController {

   @GetMapping("/")
   public String showForm() {
      return "form";
   }

   @PostMapping("/submit")
   public String handleForm(@RequestParam String imie, @RequestParam String nazwisko,
Model model) {
      model.addAttribute("imie", imie);
      model.addAttribute("nazwisko", nazwisko);
      return "result";
   }
}
```

Pytania i odpowiedzi ...

Jakie biblioteki potrzebujemy do zapisu do pliku?

Do czego służy JDBC?

Co jest głównym zadaniem klasy HashSET?

Zadaniem specyfikatora jest ...

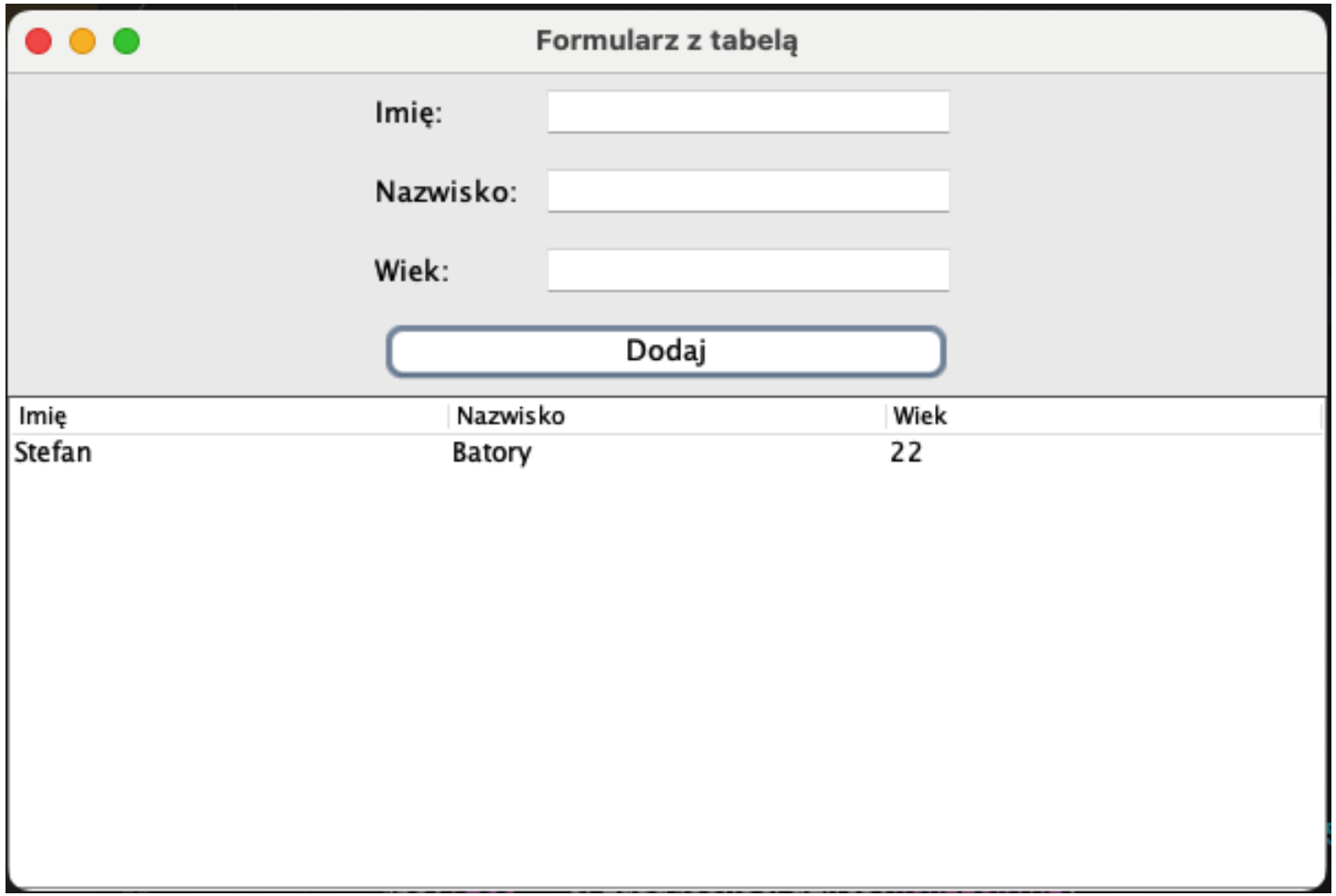
W jakim formacie można zapisywać dane pochodzące z formularza?

Czy zastosowanie MySQL wymaga dodatkowych bibliotek?

Zadania do samodzielnego wykonania

Zadanie 1

Zgodnie ze wzorem umieszczonym obok napisz aplikację wykorzystującą formularz jako element do wprowadzania danych oraz tabelę do ich wyświetlania.

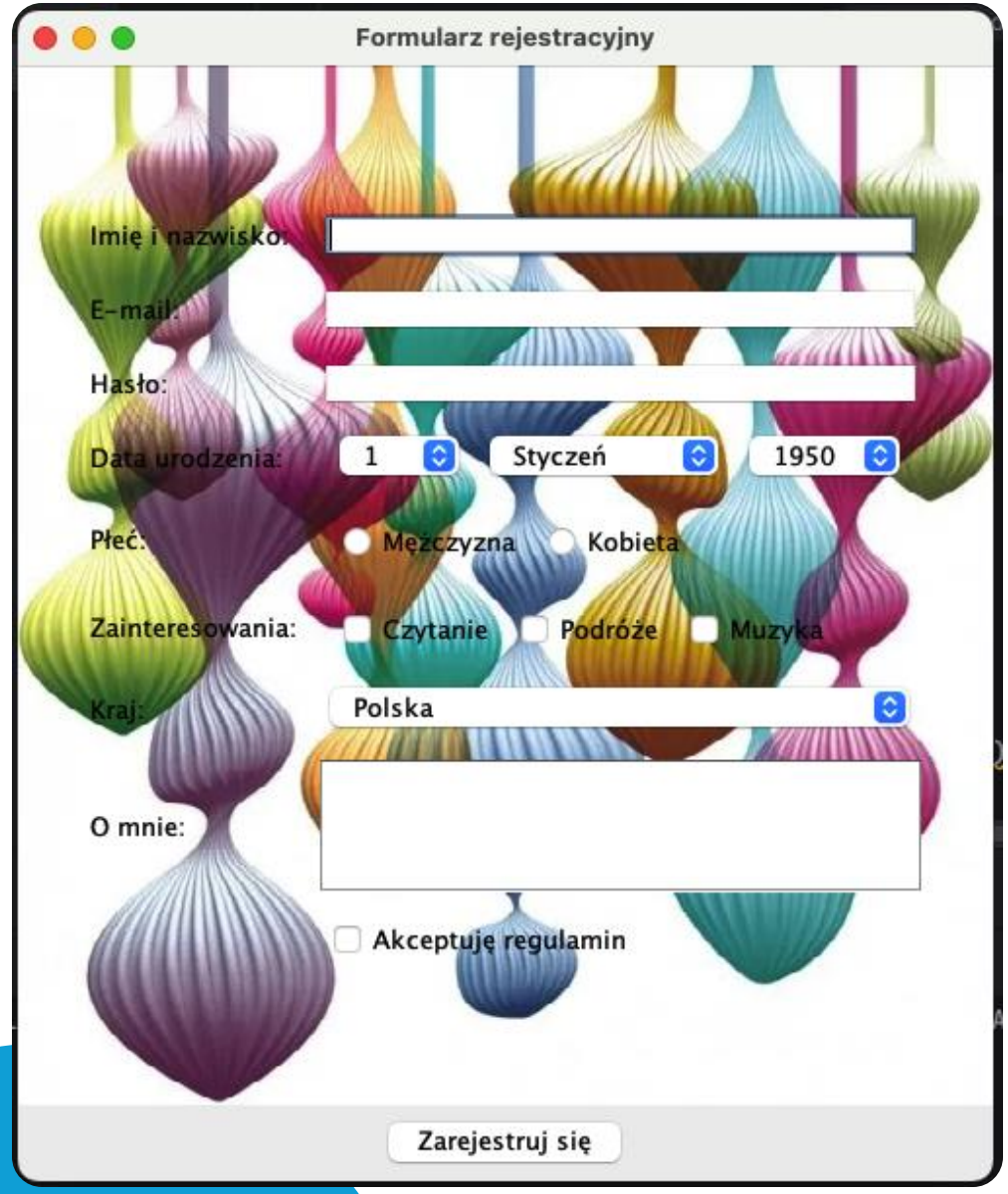


Zadania do samodzielnego wykonania

Zadanie 2

Utwórz kod formularza, który wykorzystuje:

- pola przedstawione na rysunku obok
- zapisuje dane do pliku tekstowego
- wykorzystuje HashSET do struktury danych
- tło w postaci gradientu (w tym pomoże Canvas).



Zadania do samodzielnego wykonania

Zadanie 3

Utwórz formularz według wzoru zamieszczonego obok. Posiada:

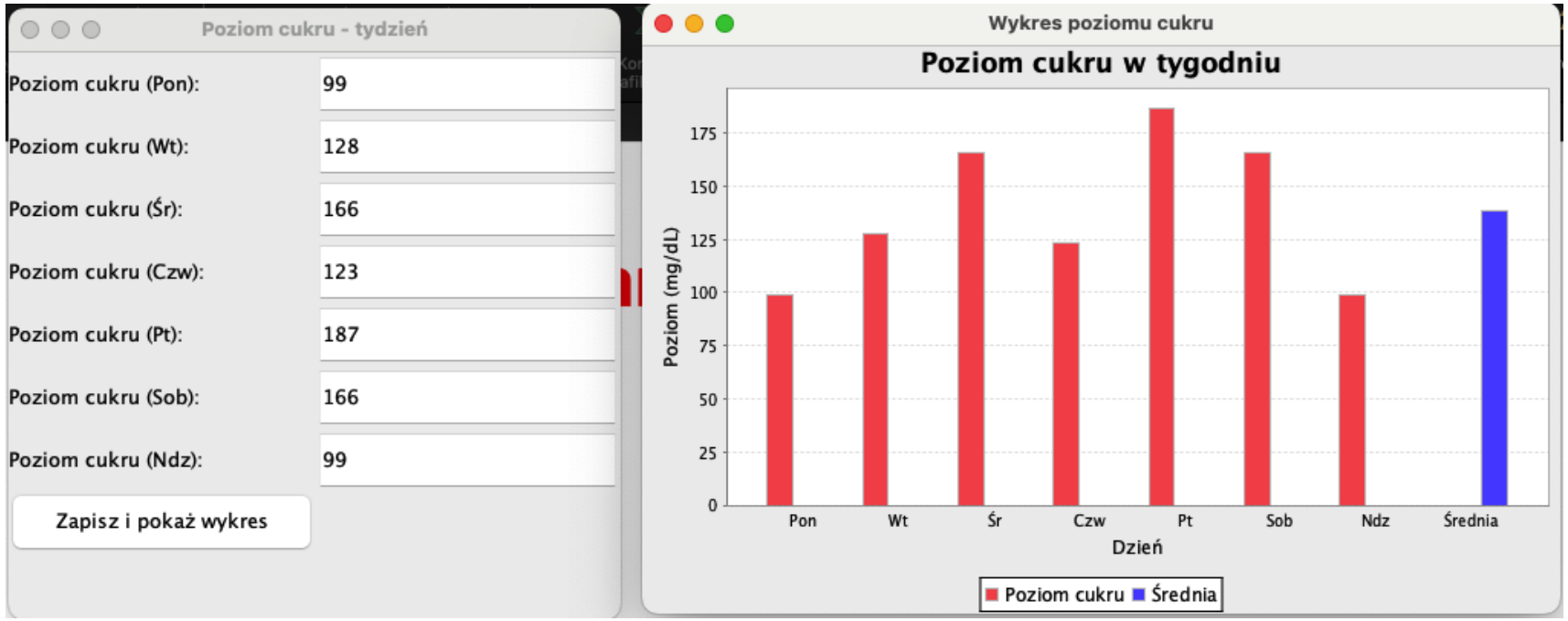
- Walidację pola „E-mail” (sprawdza, czy jest znak „@”)
- Walidację hasła (czy ma minimum 8 znaków)
- Elementy ułożone za pomocą GRID
- Estetyczne tło w postaci obrazka
- Zapis danych w pliku tekstowym

Zadania do samodzielnego wykonania

Zadanie 4 (na ocenę 5)

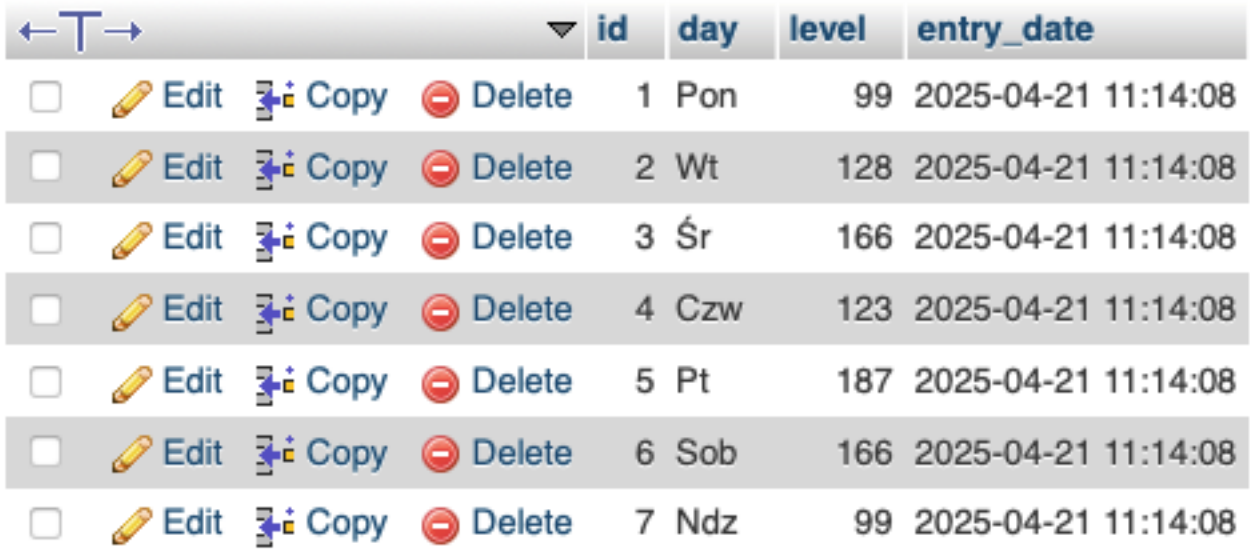
Wykonaj aplikację według wzoru obok. Jej zadaniem jest gromadzenie danych użytkownika oraz wyświetlanie ich na wykresie za pomocą JFreeChart.

Po naciśnięciu przycisku dane zapisywane są w bazie danych SQL.



Zadania do samodzielnego wykonania

Oto tabela potrzebna do ćwiczenia. Potrzebujesz mySQL-connector, utworzenie połączenia oraz zapytanie INSERT INTO do wpisania danych.

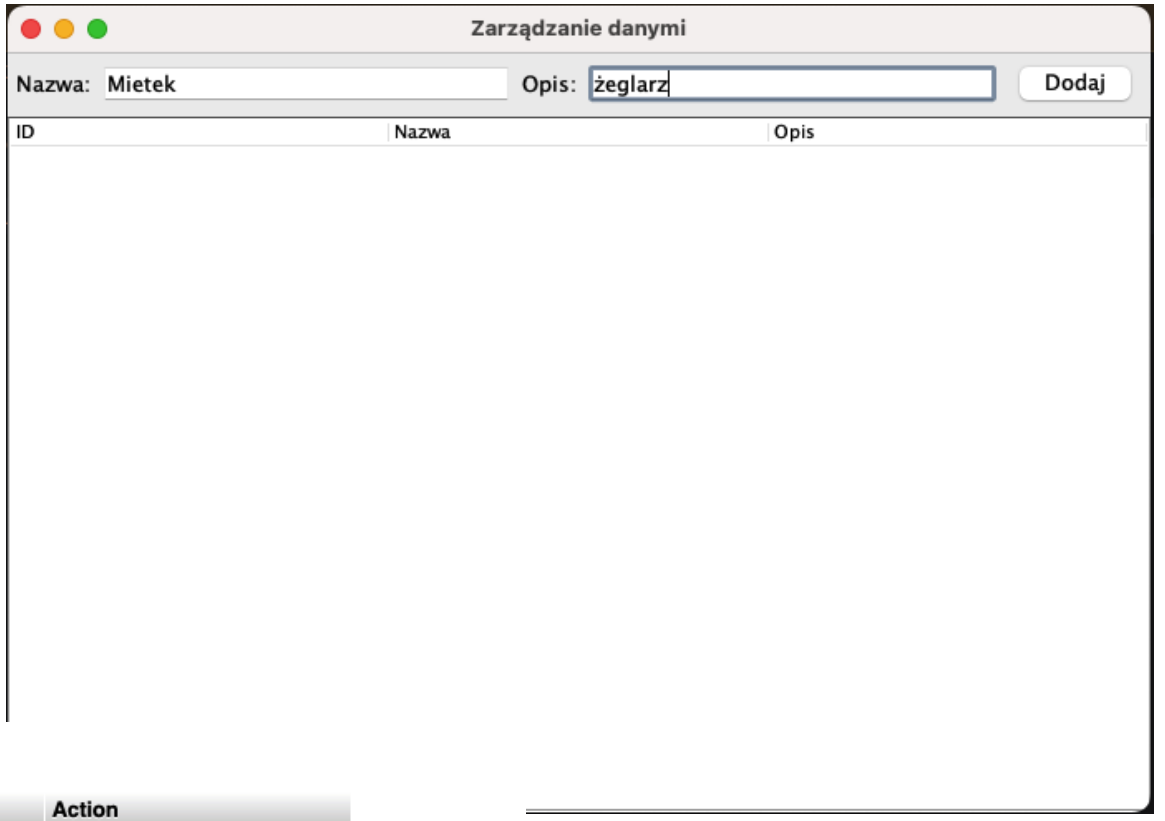


	id	day	level	entry_date
Edit Copy Delete	1	Pon	99	2025-04-21 11:14:08
Edit Copy Delete	2	Wt	128	2025-04-21 11:14:08
Edit Copy Delete	3	Śr	166	2025-04-21 11:14:08
Edit Copy Delete	4	Czw	123	2025-04-21 11:14:08
Edit Copy Delete	5	Pt	187	2025-04-21 11:14:08
Edit Copy Delete	6	Sob	166	2025-04-21 11:14:08
Edit Copy Delete	7	Ndz	99	2025-04-21 11:14:08

Zadania do samodzielnego wykonania

Zadanie 5 (na ocenę 5)

Aplikacja gromadzi dane dane użytkownika
w tabeli i w bazie danych.
Utwórz widok tak jak na rysunku obok.

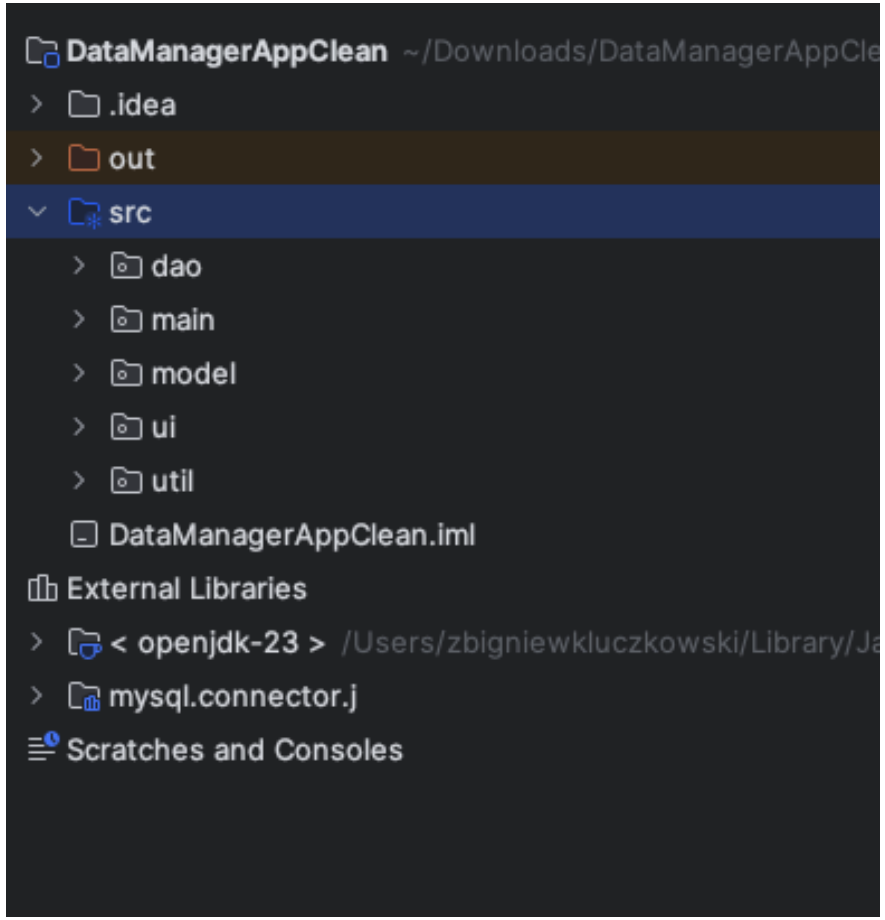


Struktura bazy danych do projektu:

Table structure | Widok relacyjny

#	Nazwa	Typ	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int			Nie	*Brak*		AUTO_INCREMENT	Change Drop Więcej
2	name	varchar(100)	utf8mb4_0900_ai_ci		Nie	*Brak*			Change Drop Więcej
3	date	date			Nie	*Brak*			Change Drop Więcej
4	description	text	utf8mb4_0900_ai_ci		Tak	*NULL*			Change Drop Więcej

Zadania do samodzielnego wykonania



Po utworzeniu bazy danych i imporcie niezbędnych bibliotek utwórz strukturę w oparciu o DAO.

Dlaczego warto tworzyć gry w Java?

Przenośność: Java działa na zasadzie „Write Once, Run Anywhere” (WORA), co oznacza, że raz napisana gra może działać na wielu platformach.

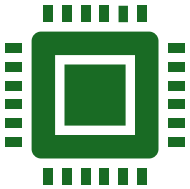
Bogate API: Java oferuje rozbudowane biblioteki, takie jak JavaFX, Swing, czy biblioteki zewnętrzne (np. LWJGL).

Wsparcie społeczności: Duża ilość forów, tutoriali i kursów online.

Podstawowe narzędzia do tworzenia gier w Javie

JDK (Java Development Kit)

Podstawowe narzędzie umożliwiające kompilację i uruchamianie aplikacji Java.



IDE (Integrated Development Environment)

IntelliJ IDEA

Eclipse

NetBeans

Biblioteki i Frameworki

JavaFX: Do tworzenia aplikacji z graficznym interfejsem użytkownika.

LibGDX: Framework przeznaczony do tworzenia gier 2D i 3D.

LWJGL (Lightweight Java Game Library): Narzędzie do tworzenia gier z użyciem OpenGL i OpenAL.

Podstawowe elementy gry

Pętla gry
Sercem każdej gry jest pętla, która aktualizuje stan gry i renderuje obraz na ekranie.

```
while (running) {
update(); // Aktualizacja stanu gry
render(); // Renderowanie grafiki}
```

Obsługa grafiki
W Javie można korzystać z Graphics i Canvas do rysowania obiektów 2D.

```
public void paint(Graphics g) {
 g.setColor(Color.RED);
 g.fillRect(50, 50, 100, 100); // Rysowanie czerwonego prostokąta
}
```

Podstawowe elementy gry

Obsługa zdarzeń
Obsługa klawiatury i myszy za pomocą listenerów.

```
addKeyListener(new KeyAdapter() {
    public void keyPressed(KeyEvent e) {
       if (e.getKeyCode() == KeyEvent.VK_LEFT) { // Ruch w lewo
        }
    }
});
```

W grach w postaci aplikacji desktopowej najczęściej używamy kontrolek w postaci klawiszy. W mobilnych używany zdarzenia onTouch oraz dedykowany do niego Listener.

5 kroków do napisania gry

1. **Przygotowanie projektu**
   - Stworzenie podstawowej klasy Main z metodą main.
2. **Tworzenie okna gry**
   - Wykorzystanie klasy **JFrame**.
3. **Dodanie logiki gry**
   - Implementacja pętli gry, systemu kolizji i ruchu postaci.
4. **Dodanie grafiki i dźwięku**
   - Wczytanie obrazów (np. za pomocą klasy **ImageIO**) i dźwięków (np. za pomocą **Clip**).
5. **Testowanie i optymalizacja**
   - Sprawdzenie działania na różnych urządzeniach i optymalizacja wydajności.

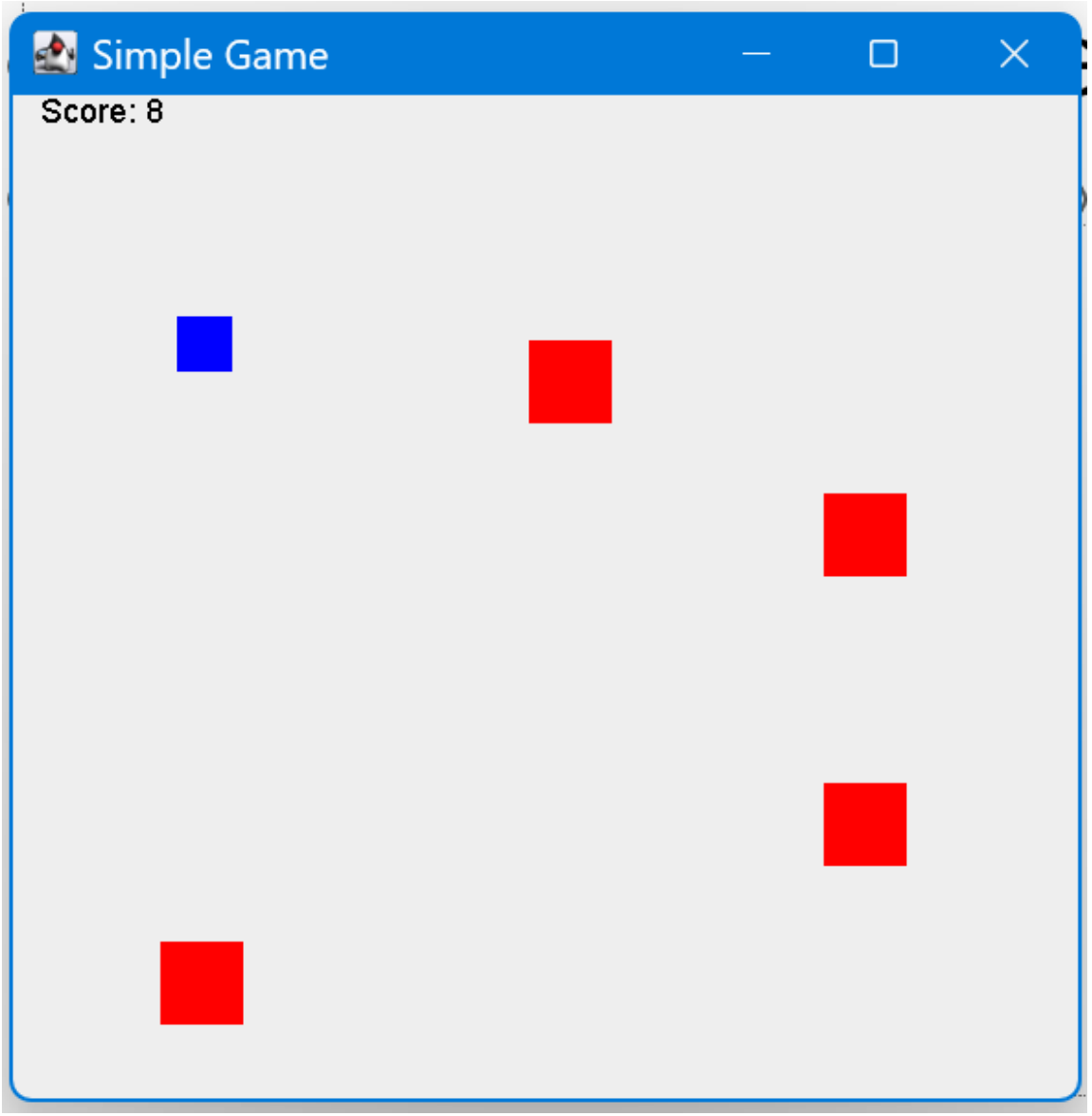
Przykład – gra „Unikaj przeszkód”

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Random;

public class SimpleGame extends JPanel implements
ActionListener {
    private Timer timer;
    private int playerX = 50, playerY = 50;
    private int score = 0;
    private ArrayList<Rectangle> obstacles;
    private Random random;

    public SimpleGame() {
        timer = new Timer(10, this);
        timer.start();
        obstacles = new ArrayList<>();
        random = new Random();
        generateObstacles();

        addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_UP) playerY -= 5;
                if (e.getKeyCode() == KeyEvent.VK_DOWN) playerY += 5;
                if (e.getKeyCode() == KeyEvent.VK_LEFT) playerX -= 5;
                if (e.getKeyCode() == KeyEvent.VK_RIGHT) playerX += 5;
                score++;
            }
        });
        setFocusable(true);
    }

    private void generateObstacles() {
        for (int i = 0; i < 5; i++) {
            int x = random.nextInt(350) + 50;
            int y = random.nextInt(350) + 50;
            obstacles.add(new Rectangle(x, y, 30, 30));
        }
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.BLUE);
        g.fillRect(playerX, playerY, 20, 20);

        g.setColor(Color.RED);
        for (Rectangle obstacle : obstacles) {
            g.fillRect(obstacle.x, obstacle.y, obstacle.width,
obstacle.height);
        }

        g.setColor(Color.BLACK);
        g.drawString("Score: " + score, 10, 10);
    }

    public void actionPerformed(ActionEvent e) {
        for (Rectangle obstacle : obstacles) {
            if (new Rectangle(playerX, playerY, 20,
20).intersects(obstacle)) {
                timer.stop();
                JOptionPane.showMessageDialog(this, "Game Over! Your
score: " + score);
                System.exit(0);
            }
        }
        repaint();
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Simple Game");
        SimpleGame game = new SimpleGame();
        frame.add(game);
        frame.setSize(400, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Działanie kodu ...

Użytkownik za pomocą klawiatury steruje ruchami niebieskiego kwadratu. Musi wykonać slalom pomiędzy czerwonymi kwadratami. Za każdy ruch otrzymuje punkty.

Na tym prostym kodzie można napisać większość prostych gier zręcznościowych.



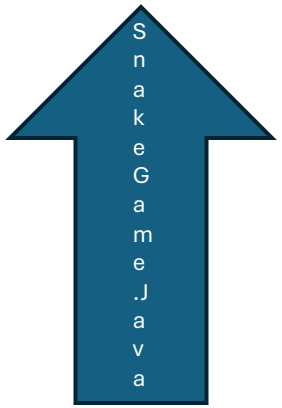
Przykład – gra „Snake”

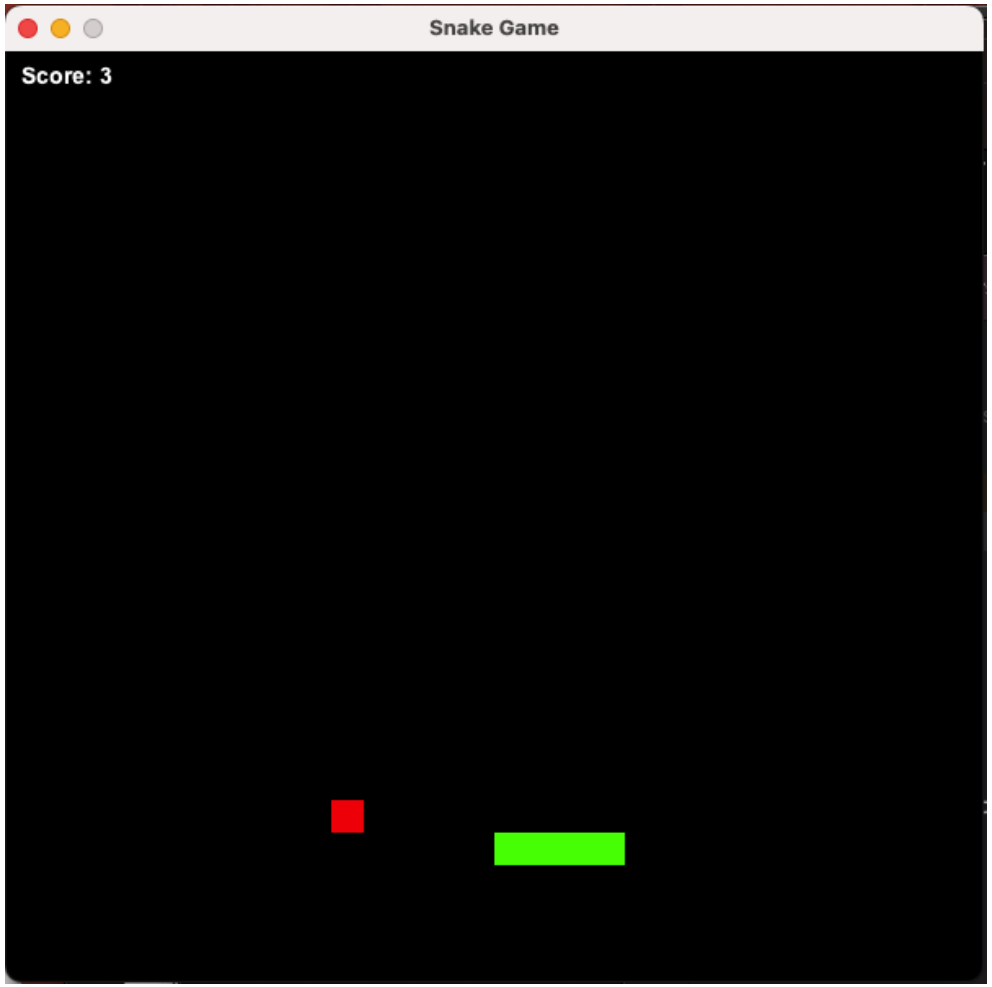
```
import java.awt.*;
import java.awt.event.KeyEvent;
import java.util.*;

public class Snake {
    private LinkedList<Point> body;
    private int direction;

    public static final int UP = KeyEvent.VK_UP;
    public static final int DOWN = KeyEvent.VK_DOWN;
    public static final int LEFT = KeyEvent.VK_LEFT;
    public static final int RIGHT = KeyEvent.VK_RIGHT;

    public Snake(int startX, int startY) {
        body = new LinkedList<>();
        body.add(new Point(startX, startY));
        direction = RIGHT;
    }

    public void move() {
        Point head = getHead();
        Point newHead = new Point(head.x, head.y);

        switch (direction) {
            case UP: newHead.y--; break;
            case DOWN: newHead.y++; break;
            case LEFT: newHead.x--; break;
            case RIGHT: newHead.x++; break;
        }

        body.addFirst(newHead);
        body.removeLast();
    }

    public void grow() {
        Point head = getHead();
        body.addFirst(new Point(head.x, head.y)); // Dodajemy nową część ciała
    }

    public void changeDirection(int keyCode) {
        if (keyCode == UP && direction != DOWN) {
            direction = UP;
        } else if (keyCode == DOWN && direction != UP) {
            direction = DOWN;
        } else if (keyCode == LEFT && direction != RIGHT) {
            direction = LEFT;
        } else if (keyCode == RIGHT && direction != LEFT) {
            direction = RIGHT;
        }
    }

    public boolean collidesWithWall(int width, int height) {
        Point head = getHead();
        return head.x < 0 || head.y < 0 || head.x >= width || head.y >= height;
    }

    public boolean collidesWithSelf() {
        Point head = getHead();
        for (int i = 1; i < body.size(); i++) {
            if (body.get(i).equals(head)) {
                return true;
            }
        }
        return false;
    }

    public Point getHead() {
        return body.getFirst();
    }

    public LinkedList<Point> getBody() {
        return body;
    }
}
```

Snake.java

Przykład – gra „Snake”

```
import javax.swing.*;

public class SnakeGame {
   public static void main(String[] args) {
      JFrame frame = new JFrame("Snake Game");
      GamePanel gamePanel = new GamePanel();

      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(600, 600);
      frame.setResizable(false);
      frame.add(gamePanel);
      frame.setVisible(true);

      gamePanel.startGame();
   }
}
```

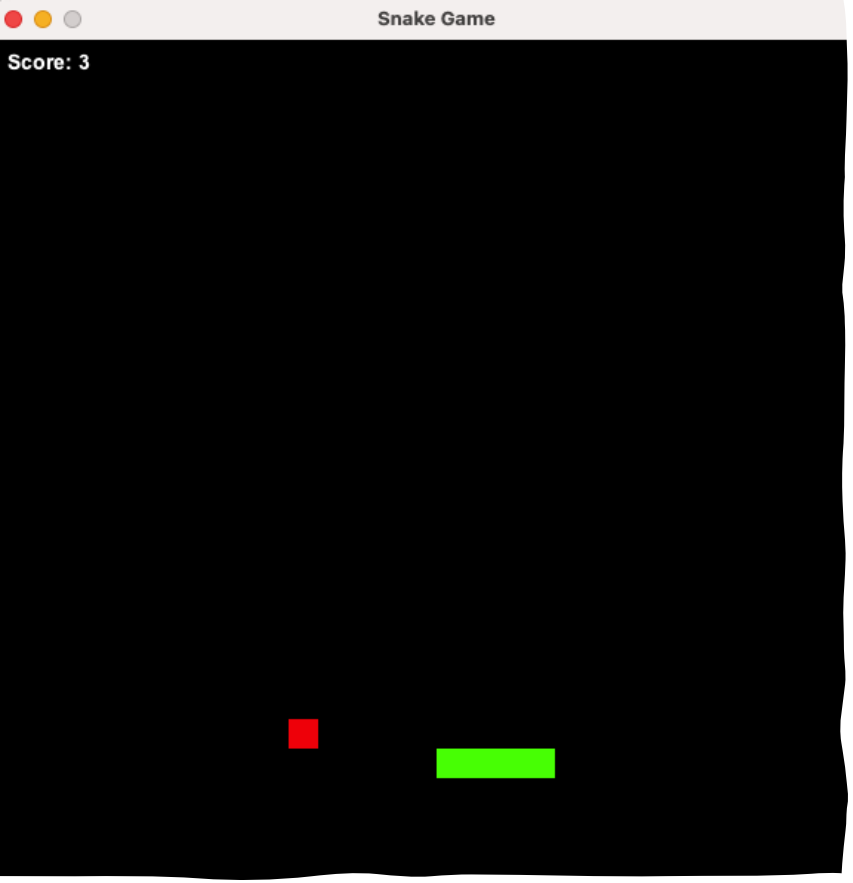


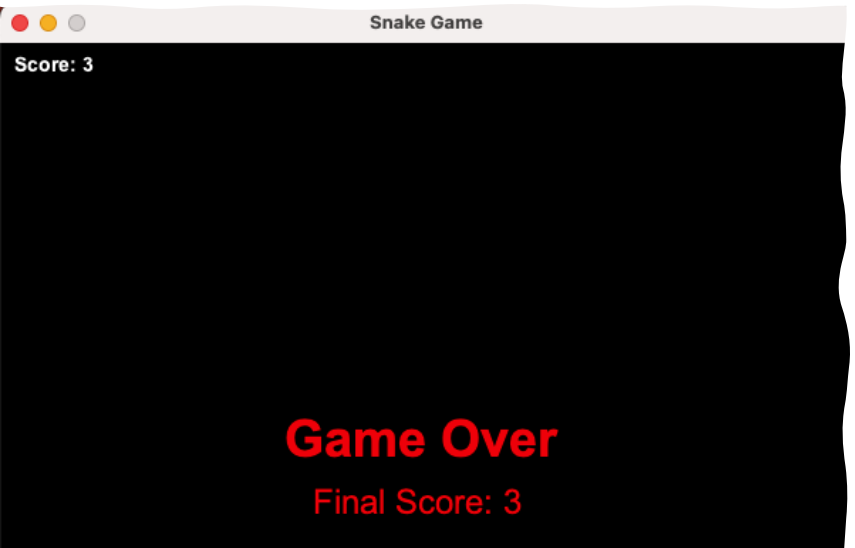


Przykład – gra „Snake”

```
import javax.swing.*;          // <-- javax.swing.Timer
import java.awt.*;
import java.awt.event.*;
import java.util.Random;       // tylko Random z java.util

public class GamePanel extends JPanel implements
ActionListener {
   private final int TILE_SIZE = 20;
   private final int WIDTH = 30;
   private final int HEIGHT = 30;
   private final int GAME_SPEED = 100;

   private Timer timer;          // javax.swing.Timer
   private Snake snake;
   private Point food;
   private boolean gameOver;
   private int score;

   public GamePanel() {
      setBackground(Color.black);
      setFocusable(true);
      addKeyListener(new KeyAdapter() {
         @Override
         public void keyPressed(KeyEvent e) {
            snake.changeDirection(e.getKeyCode());
         }
      });
   }

   public void startGame() {
      snake = new Snake(WIDTH / 2, HEIGHT / 2);
      spawnFood();
      gameOver = false;
      score = 0;

      timer = new Timer(GAME_SPEED, this);
      timer.start();          // <-- uruchamiamy timer
   }

   private void spawnFood() {
      Random rand = new Random();
      food = new Point(rand.nextInt(WIDTH),
rand.nextInt(HEIGHT));
   }

   @Override
   public void actionPerformed(ActionEvent e) {
      if (!gameOver) {
         snake.move();

         if (snake.collidesWithWall(WIDTH, HEIGHT) ||
snake.collidesWithSelf()) {
            gameOver = true;
            timer.stop();
         } else if (snake.getHead().equals(food)) {
            snake.grow();
            score++;
            spawnFood();
         }
         repaint();
      }
   }

   @Override
   protected void paintComponent(Graphics g) {
      super.paintComponent(g);

      // Punkty
      g.setColor(Color.white);
      g.setFont(new Font("Arial", Font.BOLD, 14));
      g.drawString("Score: " + score, 10, 20);

      if (gameOver) {
         g.setColor(Color.red);
         g.setFont(new Font("Arial", Font.BOLD, 36));
         g.drawString("Game Over",
               (WIDTH * TILE_SIZE) / 2 - 100,
               (HEIGHT * TILE_SIZE) / 2 - 10
         );
         g.setFont(new Font("Arial", Font.PLAIN, 24));
         g.drawString("Final Score: " + score,
               (WIDTH * TILE_SIZE) / 2 - 80,
               (HEIGHT * TILE_SIZE) / 2 + 30
         );
      } else {
         // rysujemy węża
         g.setColor(Color.green);
         for (Point p : snake.getBody()) {
            g.fillRect(p.x * TILE_SIZE, p.y * TILE_SIZE,
TILE_SIZE, TILE_SIZE);
         }
         // rysujemy jedzenie
         g.setColor(Color.red);
         g.fillRect(food.x * TILE_SIZE, food.y * TILE_SIZE,
TILE_SIZE, TILE_SIZE);
      }
   }
}
```

GamePanel.java





Działanie kodu ...

Kod składa się z trzech klas. Zawierają kolejno: sterowanie, okno i logikę gry. Sterowanie odbywa się za pomocą klawiszy kierunkowych na klawiaturze.

Za każdy “zjedzony” kwadrat dodawany jest punkt.

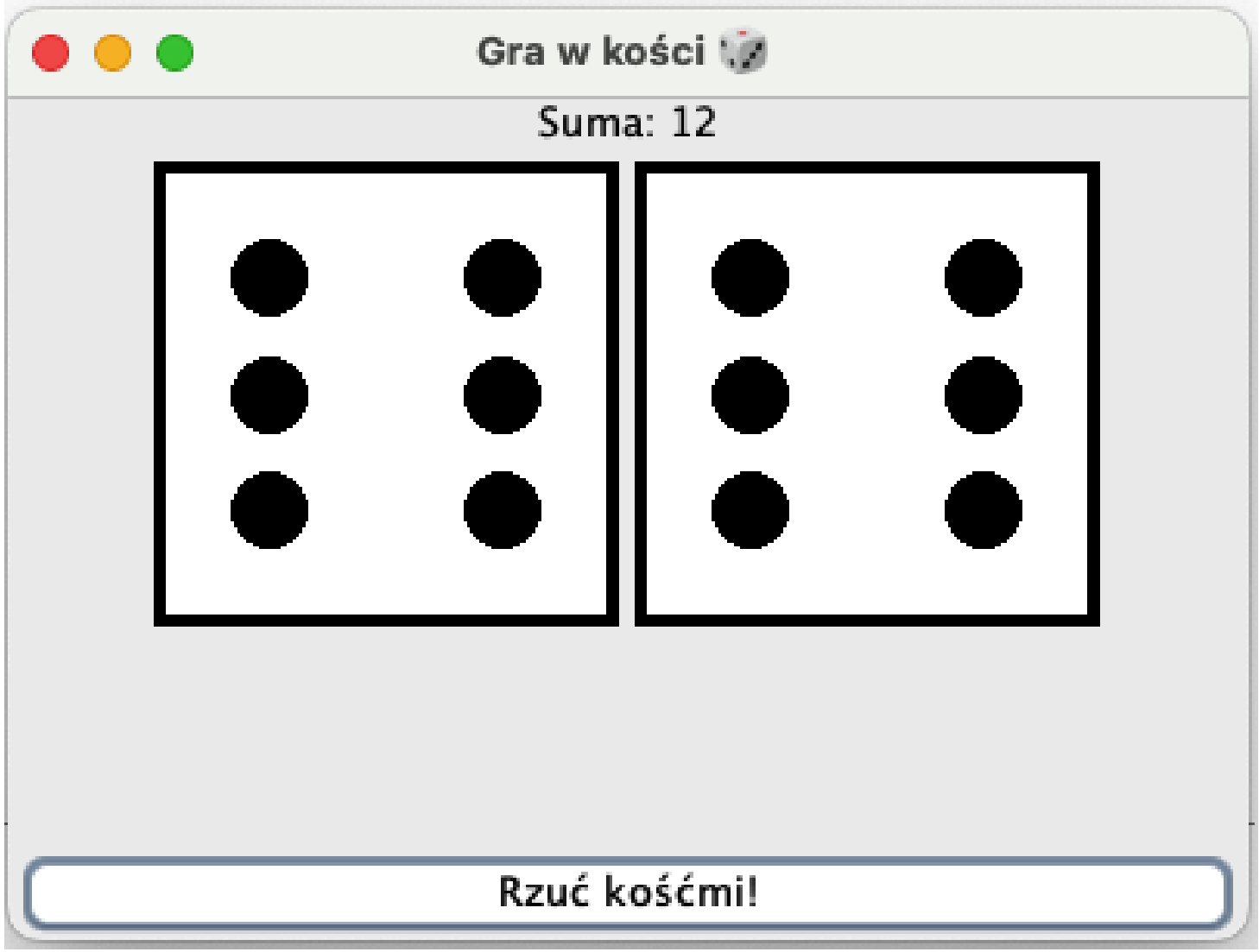
Grę można rozbudować dodając kolejne poziomy, w których można zwiększyć szybkość lub na drodze węża postawić wiele przeszkód.

Przykład? Gra „Kości”

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;


public class DiceGame extends JFrame {


   private JLabel diceLabel1, diceLabel2, sumLabel;
   private JButton rollButton;
   private Random random;


   public DiceGame(){
      super("Gra w kości 🎲");


      random = new Random();
      diceLabel1 = new JLabel();
      diceLabel2 = new JLabel();
      sumLabel = new JLabel("Suma: 0", SwingConstants.CENTER);
      rollButton = new JButton("Rzuć kośćmi!");


diceLabel1.setIcon(new ImageIcon(getClass().getResource("/dice1.png")));
      diceLabel2.setIcon(new ImageIcon(getClass().getResource("/dice1.png")));


      setLayout(new BorderLayout());


      JPanel dicePanel = new JPanel();
      dicePanel.add(diceLabel1);
      dicePanel.add(diceLabel2);


      add(dicePanel, BorderLayout.CENTER);
      add(sumLabel, BorderLayout.NORTH);
      add(rollButton, BorderLayout.SOUTH);


       rollButton.addActionListener(new ActionListener(){
         @Override
         public void actionPerformed(ActionEvent e) {
            rollDice();
         }
      });


 setSize(400, 300);


setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setLocationRelativeTo(null);
      setVisible(true);
   }


   private void rollDice(){
      int roll1 = random.nextInt(6) + 1;
      int roll2 = random.nextInt(6) + 1;


      diceLabel1.setIcon(new ImageIcon(getClass().getResource("/dice" + roll1 + ".png")));
      diceLabel2.setIcon(new ImageIcon(getClass().getResource("/dice" + roll2 + ".png")));


      int sum = roll1 + roll2;
      sumLabel.setText("Suma: " + sum);
   }


   public static void main(String[] args) {
      SwingUtilities.invokeLater(() -> new DiceGame());
   }
}
```

Działanie kodu

Jednocześnie wyrzucamy dwie kostki. Program sumuje ilość wylosowanych „kropek”.
W projekcie należy użyć 6 obrazków, które będą umieszczone w folderze „SRC”. Tzw. „Random na obiektach” jest częstym elementem zadania egzaminacyjnego zarówno z aplikacji desktopowej jak i z mobilnej.



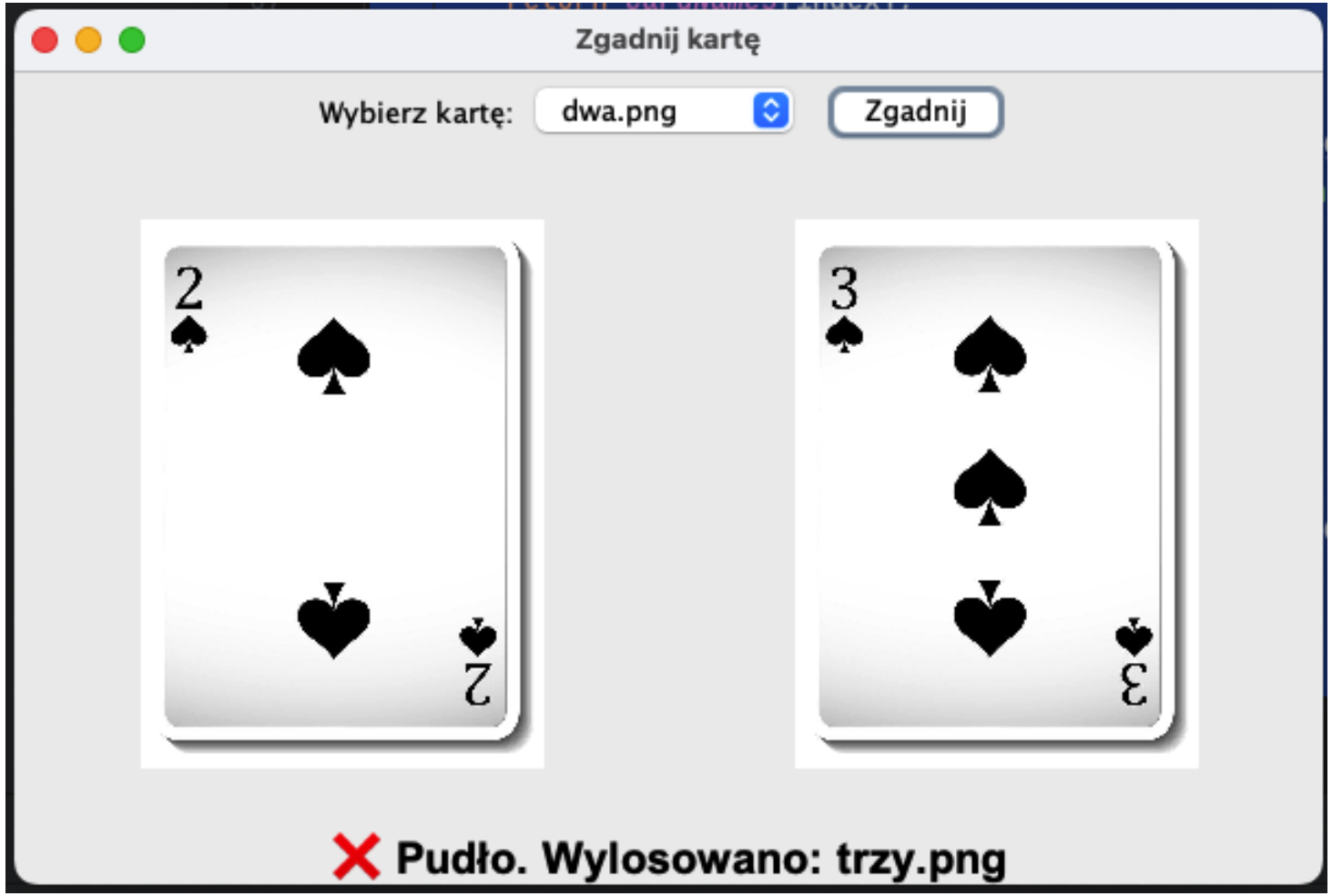
Przykład – gra „Zgadnij Kartę”

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.URL;
import java.util.Random;

public class Main {
    private static final String[] cardNames = {
            "as.png", "cztery.png", "dwa.png",
            "siedem.png", "trzy.png"
    };

    private static JLabel chosenCardLabel;
    private static JLabel drawnCardLabel;
    private static JLabel resultLabel;

    public static void main(String[] args) {
        JFrame frame = new JFrame("Zgadnij kartę");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);

        JPanel topPanel = new JPanel();
        JComboBox<String> cardSelector = new JComboBox<>(cardNames);
        JButton guessButton = new JButton("Zgadnij");
        topPanel.add(new JLabel("Wybierz kartę:"));
        topPanel.add(cardSelector);
        topPanel.add(guessButton);

        JPanel imagePanel = new JPanel(new GridLayout(1, 2));
        chosenCardLabel = new JLabel("Twoja karta", SwingConstants.CENTER);
        drawnCardLabel = new JLabel("Wylosowana karta", SwingConstants.CENTER);
        imagePanel.add(chosenCardLabel);
        imagePanel.add(drawnCardLabel);

        resultLabel = new JLabel("", SwingConstants.CENTER);
        resultLabel.setFont(new Font("Arial", Font.BOLD, 20));


        frame.setLayout(new BorderLayout());
        frame.add(topPanel, BorderLayout.NORTH);
        frame.add(imagePanel, BorderLayout.CENTER);
        frame.add(resultLabel, BorderLayout.SOUTH);

        guessButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String selectedCard = (String) cardSelector.getSelectedItem();
                String drawnCard = drawRandomCard();

                setCardImage(chosenCardLabel, selectedCard);
                setCardImage(drawnCardLabel, drawnCard);

                if (selectedCard.equals(drawnCard)) {
                    resultLabel.setText("🎉 Trafiłeś!");
                } else {
                    resultLabel.setText("❌ Pudło. Wylosowano: " + drawnCard);
                }
            }
        });

        frame.setVisible(true);
    }

    private static String drawRandomCard() {
        Random rand = new Random();
        int index = rand.nextInt(cardNames.length);
        return cardNames[index];
    }

    private static void setCardImage(JLabel label, String cardFileName) {
        URL imageUrl = Main.class.getResource("/cards/" + cardFileName);
        if (imageUrl != null) {
            ImageIcon icon = new ImageIcon(imageUrl);
            label.setIcon(icon);
            label.setText("");
        } else {
            label.setIcon(null);
            label.setText("Brak obrazka: " + cardFileName);
        }
    }
}
```

Działanie kodu

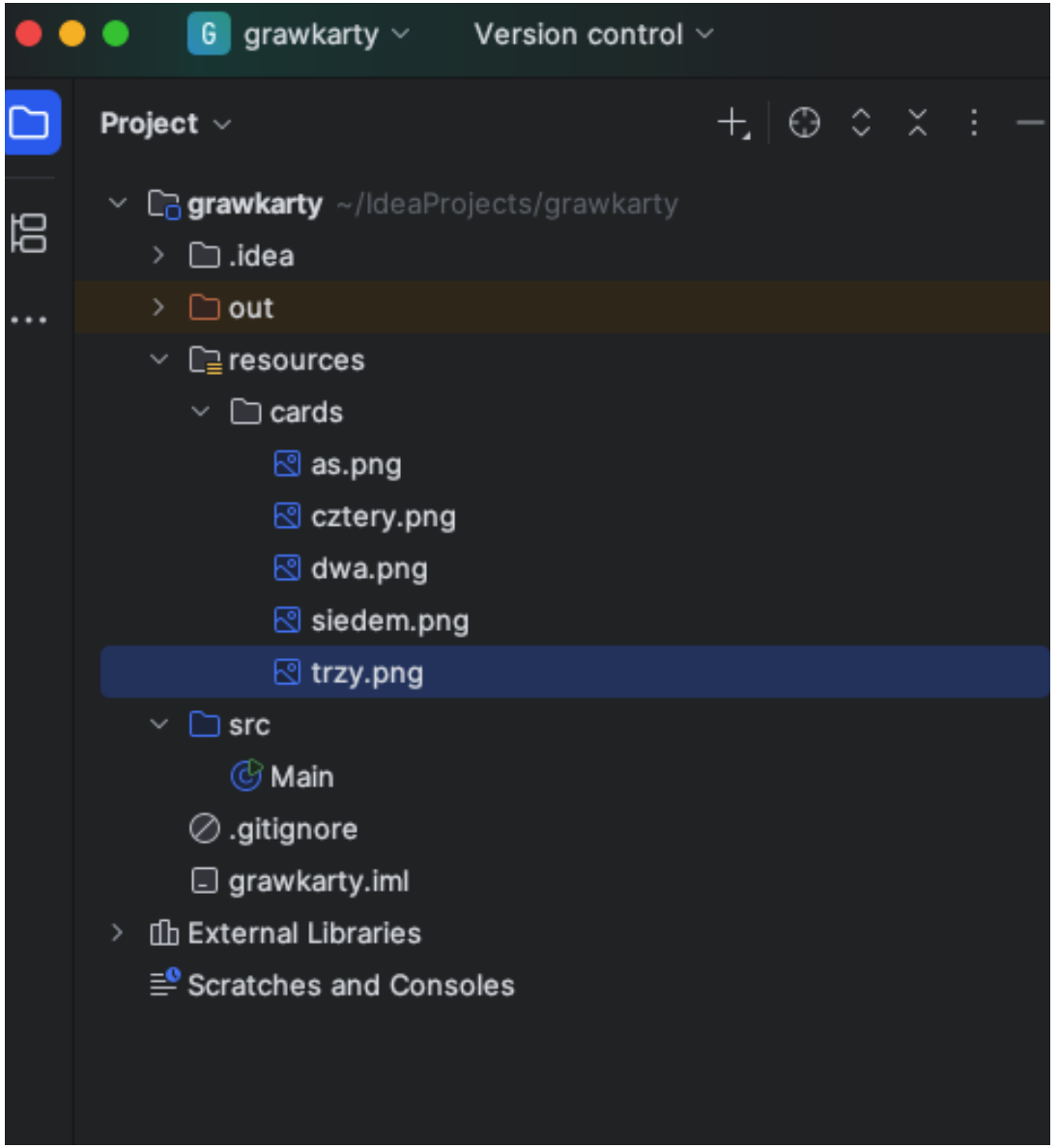
Użytkownik wybiera kartę. Po wyborze potwierdza wybór. Program pokazuje wybraną i wylosowaną kartę i określa czy użytkownik trafił czy nie.

Tzw. „random na obiektach” to częsty element zadania egzaminacyjnego.



Uwaga!

Do projektu potrzebujesz zdjęć oraz poprawnej struktury plików i folderów.



Zapisywanie wyników gry

PLIKI .TXT, .CSV, .JSON DO LOKALNEGO ZAPISU

JDBC + MYSQL DLA PEŁNEJ BAZY DANYCH

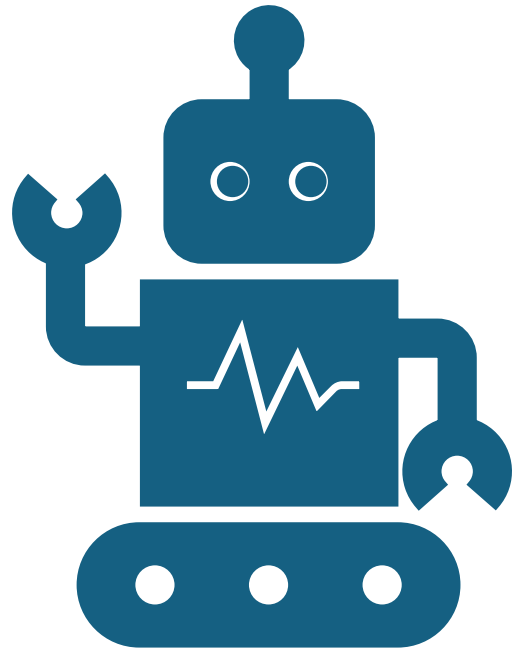
SERIALIZACJA OBIEKTÓW (OBJECTOUTPUTSTREAM) – ZAPIS STANU GRY

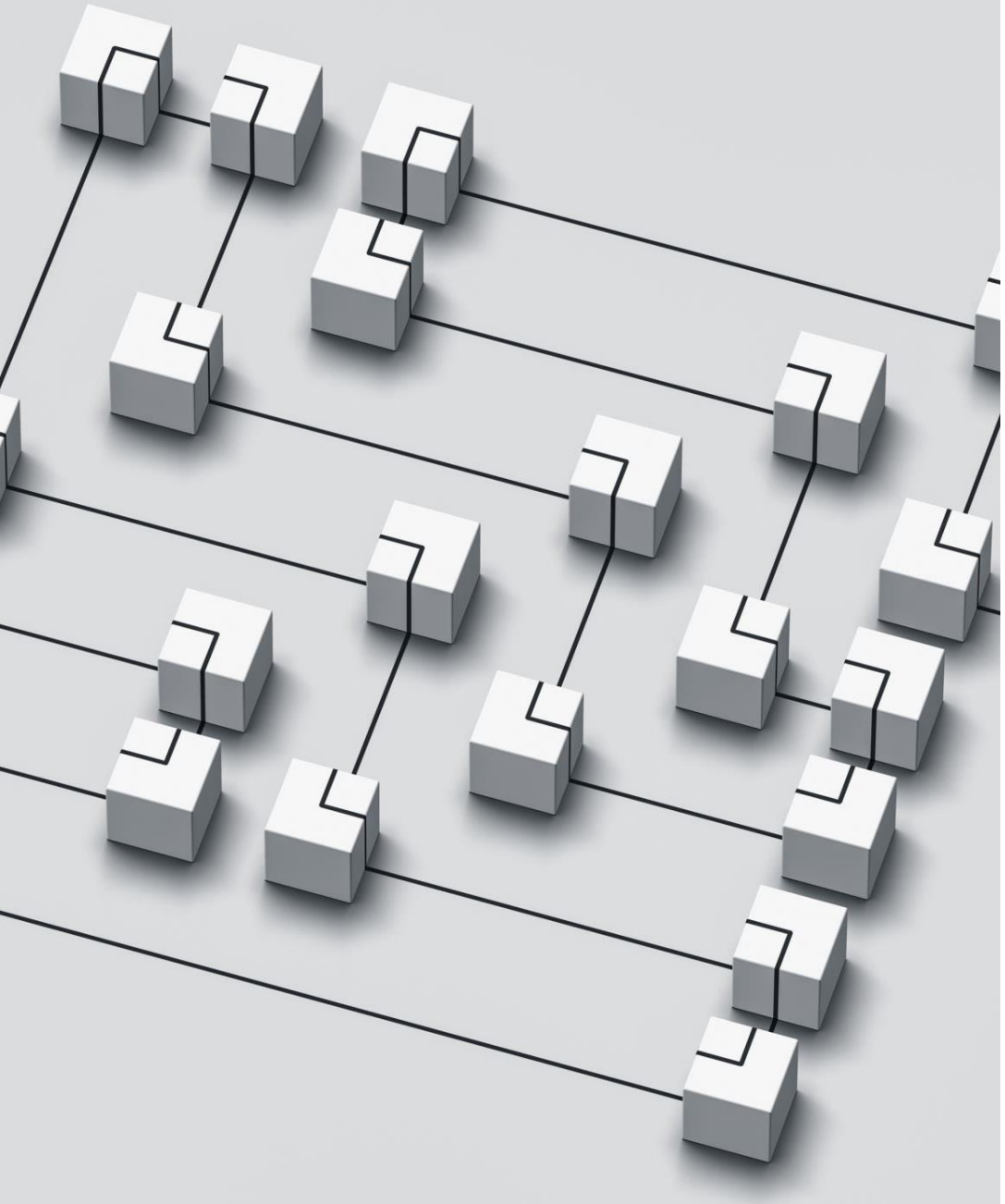
Framework Java do gier

- libGDX – wieloplatformowy silnik gier
- jMonkeyEngine – gry 3D
- Przenoszenie gry na Androida (Java to domyślny język Android)
- Tworzenie multiplayer przez Socket/ServerSocket

Dodatkowe funkcjonalności

- **Fizyka**: Silniki takie jak JBox2D umożliwiają realistyczne symulacje fizyczne.
- **Sieciowość**: Możliwość tworzenia gier multiplayer za pomocą socketów.
- **3D**: Użycie biblioteki LWJGL do tworzenia gier trójwymiarowych.





Wydajność i najlepsze praktyki

- **Wydajność**: Upewnij się, że kod jest zoptymalizowany, aby unikać opóźnień.
- **Modularność**: Podziel projekt na mniejsze moduły (np. grafika, logika gry, obsługa zdarzeń).
- **Testowanie**: Regularne testy zapewniają stabilność gry.

Zastosowanie JAVA w aplikacjach Android

Java jest:

- Obiektowa
- Stabilna i sprawdzona
- Wspierana przez Android API
- Bogata w biblioteki i narzędzia
- Doskonała do nauki programowania mobilnego

Struktura aplikacji Android w Java

Główne komponenty:

- Activity – główne ekrany aplikacji
- Intent – komunikacja między ekranami
- XML – pliki do tworzenia interfejsu graficznego (GUI)

Kod Java odpowiada za logikę działania aplikacji

Prosty przykład

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        TextView tekst = findViewById(R.id.textView);
        tekst.setText("Witaj w aplikacji Java!");
    }
}
```

Projekt z przyciskiem i reakcją

```
<Button
   android:id="@+id/button"
   android:text="Kliknij mnie"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>
```


```
Button btn = findViewById(R.id.button);
btn.setOnClickListener(v -> {
   Toast.makeText(this, "Kliknięto przycisk!",
Toast.LENGTH_SHORT).show();
});
```

XML – opis interfejsu użytkownika

Plik layout: W Android Studio pliki XML służą do określania, jak aplikacja ma wyglądać.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">

   <TextView
      android:id="@+id/textView"
      android:layout_width="wrap_content"

     android:layout_height="wrap_content"
      android:text="Hello, World!" />

</LinearLayout>
```

Elementy XML w Android Studio

View: Podstawowy element UI, np. TextView, Button, ImageView.

ViewGroup: Elementy, które mogą zawierać inne widoki, np. LinearLayout, RelativeLayout, ConstraintLayout.

Atrybuty: W XML atrybuty definiują właściwości widoku, np. android:text, android:layout_width, android:layout_height

Typy Layout w Android Studio

- **LinearLayout**: Rozmieszcza widoki w jednym rzędzie lub kolumnie.

```
<LinearLayout android:orientation="horizontal"
android:layout_width="match_parent"
android:layout_height="wrap_content"> <!--
Elementy wewnętrzne --> </LinearLayout>
```

- **RelativeLayout**: Rozmieszcza elementy względem innych widoków.
- **ConstraintLayout**: Zaawansowany layout, który umożliwia precyzyjne rozmieszczanie elementów UI.

Praca z widokami i atrybutami

```
<TextView
   android:id="@+id/myTextView"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Hello Android!" />


<Button
   android:id="@+id/myButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Click me" />
```

```
<ImageView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/image_name"
/>
```

Właściwości XML - Layout

Właściwość	Opis
android:layout_width	Określa szerokość widoku. Może być ustawiona na match_parent (wypełnia dostępne miejsce) lub wrap_content (dopasowuje szerokość do zawartości).
android:layout_height	Określa wysokość widoku. Może być ustawiona na match_parent (wypełnia dostępne miejsce) lub wrap_content (dopasowuje wysokość do zawartości).
android:orientation	Określa kierunek rozmieszczenia elementów w kontenerze (np. w LinearLayout na horizontal lub vertical).
android:layout_gravity	Określa, jak widok będzie rozmieszczony w swoim rodzicu, np. w LinearLayout lub RelativeLayout (center, left, right, itp.).
android:gravity	Określa wyrównanie zawartości wewnątrz widoku, np. dla TextView (center, top, bottom, itp.).
android:layout_weight	Określa proporcjonalną wagę widoku, szczególnie użyteczne w LinearLayout, aby dzielić przestrzeń.

Właściwości XML – dla Relative Layout

Właściwość	Opis
android:layout_alignParentLeft	Używane w RelativeLayout, ustala, czy widok ma być wyrównany do lewej krawędzi rodzica.
android:layout_alignParentTop	Używane w RelativeLayout, ustala, czy widok ma być wyrównany do górnej krawędzi rodzica.
android:layout_alignParentRight	Używane w RelativeLayout, ustala, czy widok ma być wyrównany do prawej krawędzi rodzica.
android:layout_alignParentBottom	Używane w RelativeLayout, ustala, czy widok ma być wyrównany do dolnej krawędzi rodzica.
android:layout_centerInParent	Używane w RelativeLayout, ustala, czy widok ma być wyśrodkowany w rodzicu.
android:layout_centerHorizontal	Używane w RelativeLayout, ustala, czy widok ma być wyśrodkowany poziomo w rodzicu.
android:layout_centerVertical	Używane w RelativeLayout, ustala, czy widok ma być wyśrodkowany pionowo w rodzicu.

Właściwości specyficzne dla Linear Layout

Właściwość	**Opis**
android:orientation	Określa, w jaki sposób elementy w LinearLayout mają być rozmieszczone: poziomo lub pionowo.
android:layout_weight	Określa proporcjonalną wagę widoku w LinearLayout, by podzielić dostępne miejsce.

Właściwości XML – typy widoków

Właściwość	Opis
android:text	Ustawia tekst wyświetlany w TextView, Button, itp.
android:textSize	Ustawia rozmiar czcionki tekstu. Można podać wartość w jednostkach sp (scale-independent pixels).
android:textColor	Określa kolor tekstu. Może być wartością hex lub odwołaniem do zasobów (@color/color_name).
android:src	Określa źródło obrazu w ImageView. Może być to plik obrazu (@drawable/image_name).
android:hint	Ustawia tekst wskazówki (placeholder) w polach tekstowych (EditText).
android:scaleType	Określa sposób skalowania obrazu w ImageView. Może przyjąć wartości: center, fitCenter, fitXY, itp.

Właściwości estetyczne

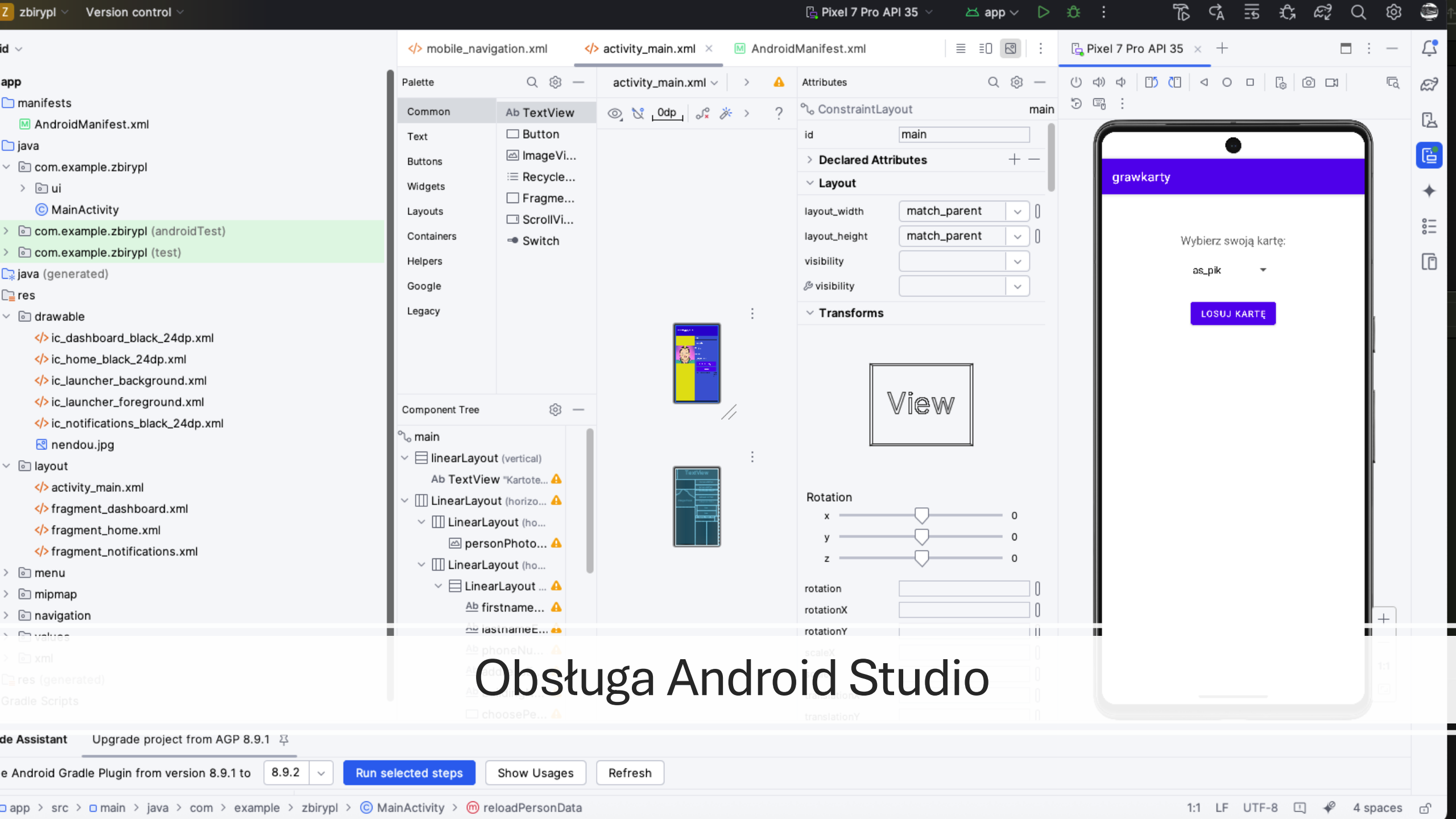
Właściwość	Opis
android:background	Ustawia tło dla widoku. Może być kolorem, gradientem lub obrazem (@drawable/background_name).
android:alpha	Określa przezroczystość widoku (wartość od 0 do 1).
android:clipToOutline	Określa, czy widok ma być przycięty do konturu (np. zaokrąglony przycisk).
android:elevation	Określa wysokość widoku nad jego rodzicem, używane w Material Design do efektów cieni.

Właściwości interakcji i widoczności

Właściwość	Opis
android:visibility	Określa widoczność widoku. Może przyjmować wartości: visible, invisible, gone.
android:focusable	Określa, czy widok może otrzymywać fokus (np. w przypadku przycisków, edytorów tekstu).
android:clickable	Określa, czy widok może reagować na kliknięcia.
android:enabled	Określa, czy widok jest włączony (aktywny) czy wyłączony.

Często używane ...

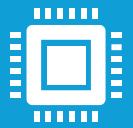
Właściwość	Opis
android:id	Określa unikalny identyfikator widoku, używany do odwoływania się do niego w kodzie (np. @+id/button dla przycisku).
android:layout_margin	Ustala marginesy dla widoku, może być użyte dla wszystkich stron lub indywidualnie (marginTop, marginBottom, itp.).
android:padding	Określa odstęp wewnętrzny dla widoku. Może być ustawiony na wszystkie strony lub każdą z osobna (paddingTop, paddingBottom, itp.).
android:layout_alignWithParentIfMissing	Określa, czy widok ma być wyrównany z rodzicem, jeśli inne właściwości rozmieszczenia są pominięte.


zbirypl
Version control
Pixel 7 Pro API 35
app
mobile_navigation.xml
activity_main.xml
AndroidManifest.xml
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
TextView
Button
ImageVi...
Recycle...
Fragme...
ScrollVi...
Switch
Component Tree
main
linearLayout (vertical)
TextView "Kartote...
LinearLayout (horizo...
LinearLayout (ho...
personPhoto...
LinearLayout (ho...
LinearLayout ...
firstname...
Attributes
ConstraintLayout
main
id
Declared Attributes
Layout
layout_width
match_parent
layout_height
match_parent
visibility
visibility
Transforms
View
Rotation
x
y
z
rotation
rotationX
rotationY
grawkarty
Wybierz swoją kartę:
as_pik
LOSUJ KARTĘ
Obsługa Android Studio
Upgrade project from AGP 8.9.1
Android Gradle Plugin from version 8.9.1 to
8.9.2
Run selected steps
Show Usages
Refresh
app > src > main > java > com > example > zbirypl > MainActivity > reloadPersonData
1:1
LF
UTF-8
4 spaces

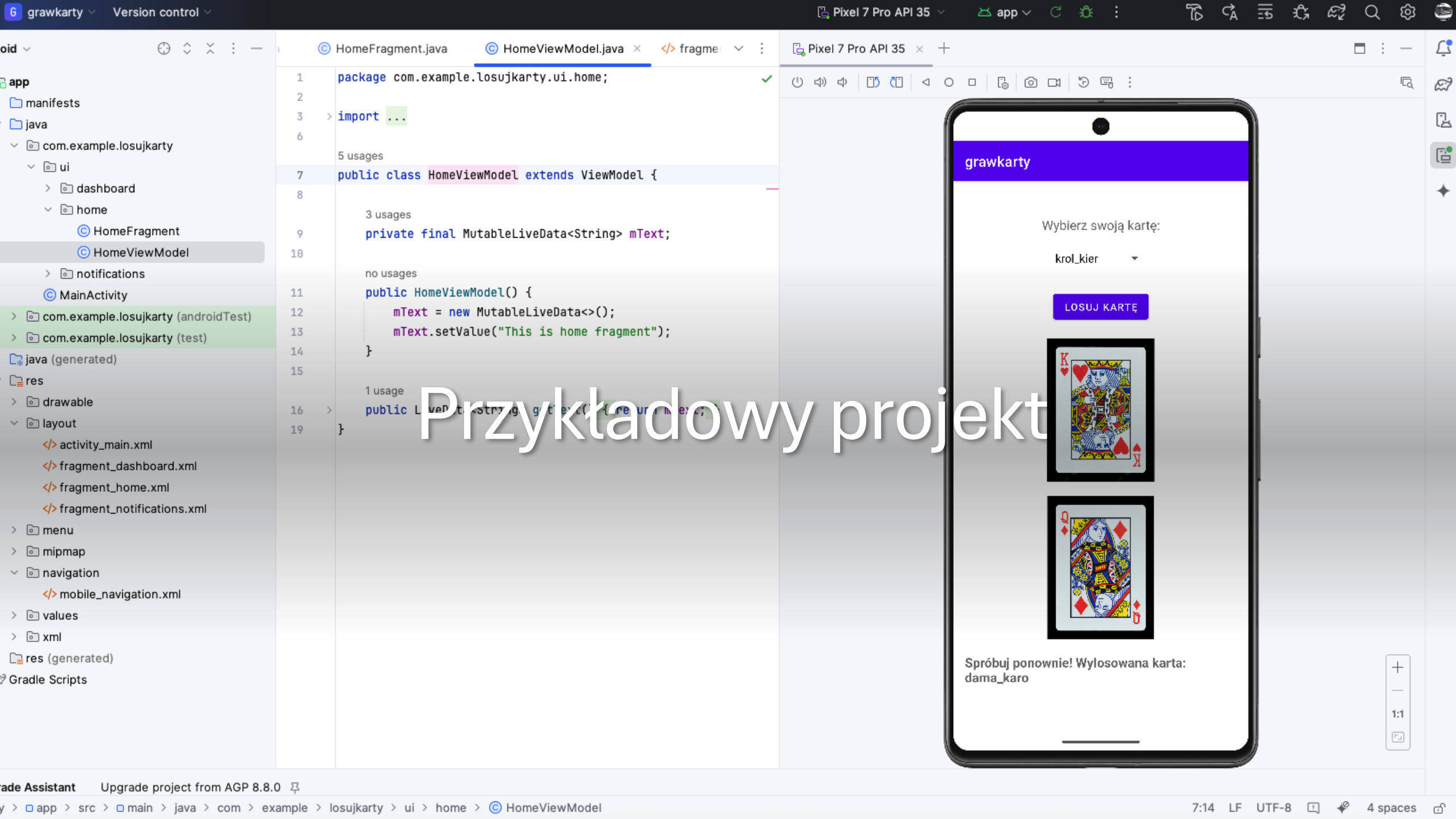
Obsługa Android Studio

Android Studio to środowisko, które zawiera wiele różnego rodzaju ułatwień do tworzenia aplikacji mobilnych. Interfejs aplikacji można w niej wygenerować za pomocą XML jak i za pomocą dostępnego kreatora graficznego.

Ważnym elementem tego środowiska jest emulator urządzenia. Chcąc go skonfigurować potrzebujemy wybrać nazwę urządzenia oraz API (wersję systemu operacyjnego).



Android Studio to środowisko wymagające dużych zasobów – dużej ilości pamięci RAM oraz pobierania dużych „paczek danych”.


grawkarty
Version control
Pixel 7 Pro API 35
app
HomeFragment.java
HomeViewModel.java
app
manifests
java
com.example.losujkarty
ui
dashboard
home
HomeFragment
HomeViewModel
notifications
MainActivity
com.example.losujkarty (androidTest)
com.example.losujkarty (test)
java (generated)
res
drawable
layout
activity_main.xml
fragment_dashboard.xml
fragment_home.xml
fragment_notifications.xml
menu
mipmap
navigation
mobile_navigation.xml
values
xml
res (generated)
Gradle Scripts
package com.example.losujkarty.ui.home;
import ...
5 usages
public class HomeViewModel extends ViewModel {
3 usages
private final MutableLiveData<String> mText;
no usages
public HomeViewModel() {
mText = new MutableLiveData<>();
mText.setValue("This is home fragment");
}
1 usage
}
Przykładowy projekt
Pixel 7 Pro API 35
grawkarty
Wybierz swoją kartę:
krol_kier
LOSUJ KARTĘ
Spróbuj ponownie! Wylosowana karta:
dama_karo
Upgrade project from AGP 8.8.0
app > src > main > java > com > example > losujkarty > ui > home > HomeViewModel
7:14
LF
UTF-8
4 spaces

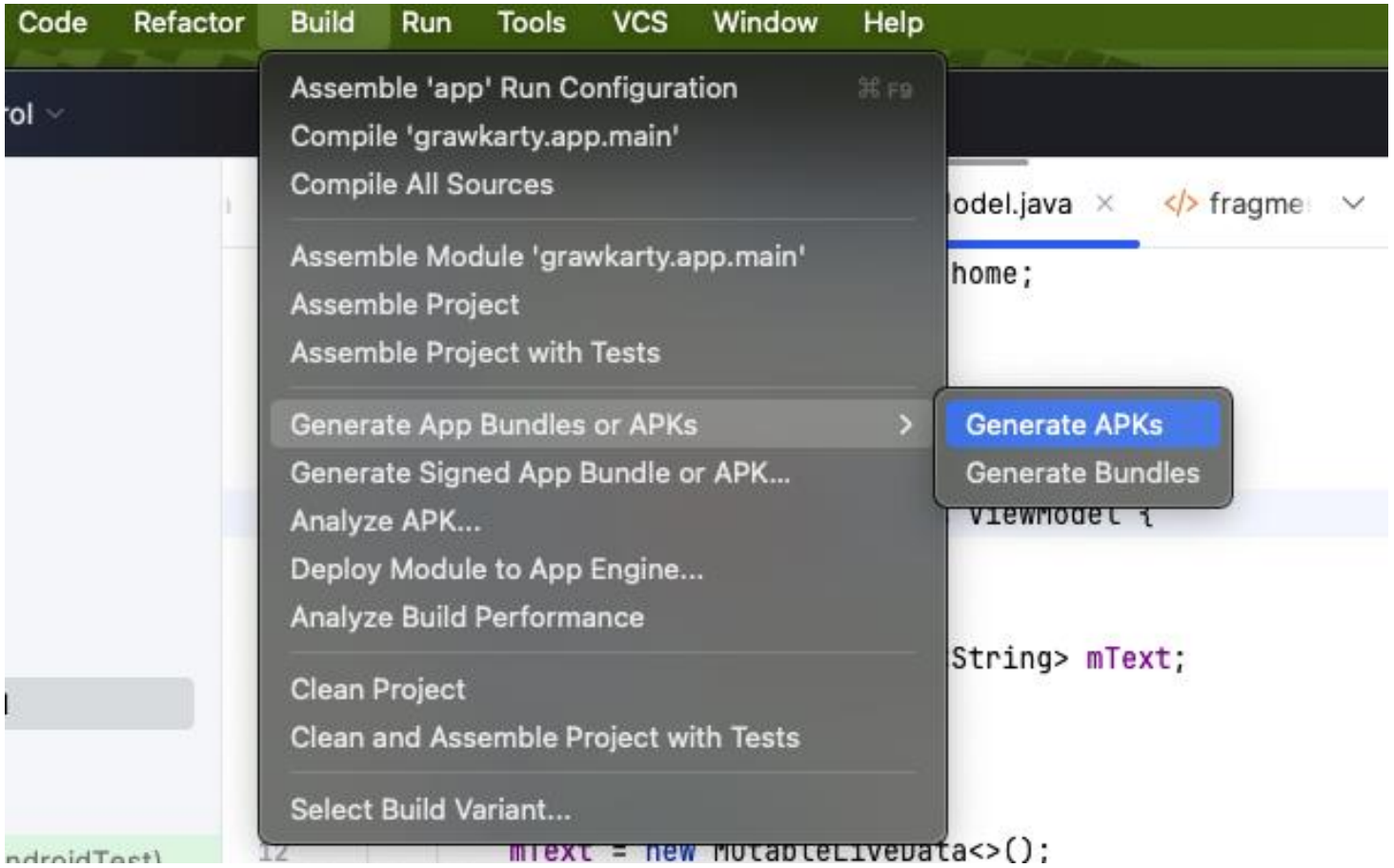
Generowanie APK

Przygotowanie projektu

- Upewnij się, że Twój projekt działa poprawnie i jest gotowy do wygenerowania pliku APK.
- Przed generowaniem APK sprawdź, czy w pliku build.gradle masz odpowiednie ustawienia dla wersji aplikacji (np. versionCode i versionName).

Budowanie APK

- **Otwórz Android Studio i załaduj swój projekt.**
- W menu wybierz **Build** → **Build Bundle(s) / APK(s)** → **Build APK(s)**.



Generowanie APK

Wybór typu kompilacji:

- Możesz wybrać **Debug** lub **Release** w zależności od tego, czy chcesz stworzyć wersję testową (Debug) lub finalną wersję produkcyjną (Release).

Proces kompilacji:

- Android Studio rozpocznie proces kompilacji projektu. Zajmie to kilka minut, w zależności od rozmiaru aplikacji.

Powiadomienie o zakończeniu kompilacji:

- Po zakończeniu kompilacji zobaczysz powiadomienie w dolnej części ekranu (w zakładce **Build**), które poinformuje Cię, że plik APK został wygenerowany.

Otwórz folder z plikiem APK:

- Kliknij w powiadomienie, aby otworzyć folder zawierający plik APK.
- Możesz również przejść do folderu **app/build/outputs/apk/**, gdzie znajdziesz wygenerowany plik APK w odpowiedniej wersji (np. debug lub release).

Generowanie APK w wersji Release

Aby wygenerować APK w wersji **Release** (czyli finalnej, gotowej do opublikowania w Google Play):

1. Skonfiguruj podpis aplikacji:

1. W wersji **Release** aplikacja musi być podpisana. Przejdź do **Build** → **Generate Signed Bundle / APK**.
2. Wybierz **APK**, a następnie kliknij **Next**.

2. Wybierz alias i klucz podpisu:

1. Jeśli jeszcze tego nie zrobiłeś, musisz utworzyć **keystore** (pliku klucza) i dodać odpowiedni alias oraz hasło.
2. Wprowadź ścieżkę do **keystore** i alias, a także hasła wymagane do podpisania aplikacji.

3. Wybierz typ kompilacji:

1. Wybierz **Release** jako typ kompilacji.

4. Kompilacja i generowanie APK:

1. Kliknij **Finish**. Android Studio rozpocznie proces generowania podpisanego pliku APK.

Instalowanie na urządzeniu

Po wygenerowaniu pliku APK możesz go zainstalować na urządzeniu Android za pomocą:

- **Bezpośredniego przesyłania**: Przenieś APK na urządzenie i uruchom go ręcznie.
- **ADB**: Jeśli masz włączoną opcję **Developer options** i **USB Debugging** na urządzeniu, możesz użyć **ADB** (Android Debug Bridge) do zainstalowania APK:

```
adb install path/to/your/app.apk
```

Przykładowe aplikacje - ”Pralki” - Java

```
package com.example.pralka;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

import com.example.pralka.R;

public class MainActivity extends AppCompatActivity {

    private Button zatwierdzButton, wlaczButton;
    private TextView wynikLabel, statusLabel;
    private EditText pranieInput;
    private boolean isOdkurzaczOn = false;  // Stan odkurzacza

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Inicjalizacja widoków
        zatwierdzButton = findViewById(R.id.zatwierdzButton);
        wlaczButton = findViewById(R.id.wlaczButton);
        wynikLabel = findViewById(R.id.wynikLabel);
        statusLabel = findViewById(R.id.statusLabel);
        pranieInput = findViewById(R.id.pranieInput);

        // Ustawienie początkowego stanu przycisku i etykiety
statusu
        wlaczButton.setText("Włącz");
        statusLabel.setText("Odkurzacz wyłączony");
        wynikLabel.setText("");  // Początkowy tekst dla numeru
prania

        // Obsługa kliknięcia przycisku "Zatwierdź" (numer prania)
        zatwierdzButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String inputText = pranieInput.getText().toString();
                if (!inputText.isEmpty()) {
                    int numerPrania = Integer.parseInt(inputText);
                    if (numerPrania >= 1 && numerPrania <= 12) {
                        wynikLabel.setText("Numer prania: " + numerPrania);
                    } else {
                        wynikLabel.setText("Numer prania poza zakresem");
                    }
                } else {
                    wynikLabel.setText("Proszę wprowadzić numer
prania");
                }
            }
        });

        // Obsługa kliknięcia przycisku "Włącz/Wyłącz" (odkurzacz)
        wlaczButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Zmieniamy stan odkurzacza
                if (isOdkurzaczOn) {
                    // Jeśli odkurzacz jest włączony, wyłączamy go
                    wlaczButton.setText("Włącz");
                    statusLabel.setText("Odkurzacz wyłączony");
                } else {
                    // Jeśli odkurzacz jest wyłączony, włączamy go
                    wlaczButton.setText("Wyłącz");
                    statusLabel.setText("Odkurzacz
włączony\nStatus:naładowany");
                }
                // Zmiana stanu
                isOdkurzaczOn = !isOdkurzaczOn;
            }
        });
    }
}
```

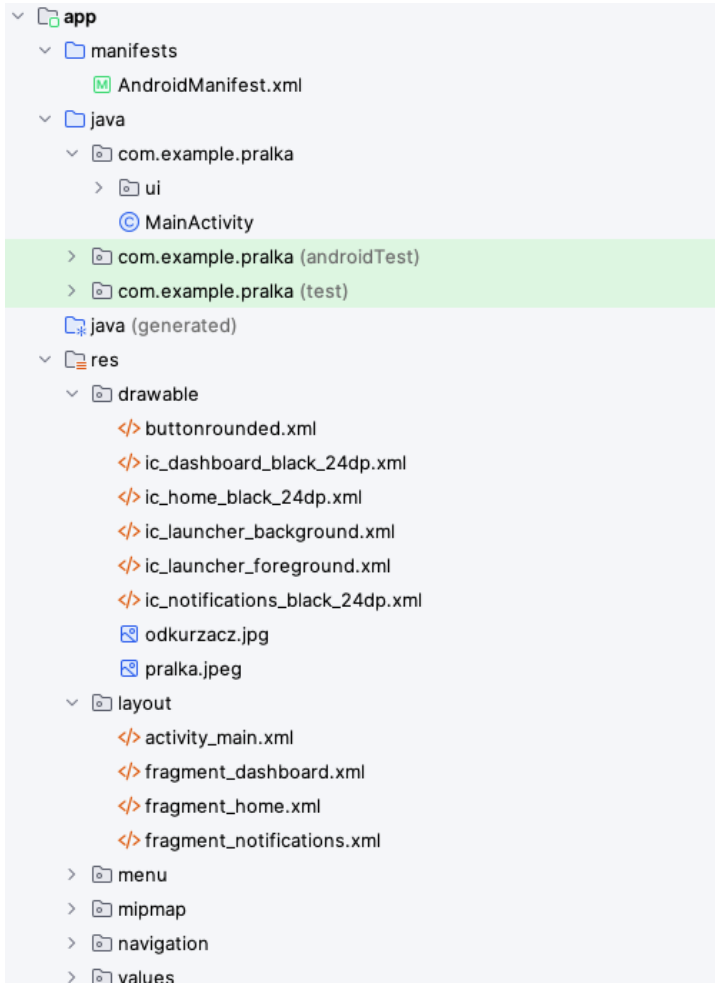
Przykładowe aplikacje - ”Pralki” - XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:padding="20dp"
  android:background="#ADD8E6">

  <TextView
    android:id="@+id/titleLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Urządzenia Domowe"
    android:textSize="24sp"
    android:textStyle="bold"
    android:gravity="center"
    android:layout_marginBottom="20dp"
    android:layout_gravity="center"/>

  <TextView
    android:id="@+id/wykonanieLabel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Wykonał: BENIO"
    android:textSize="18sp"
    android:gravity="center"
    android:layout_marginBottom="20dp"
    android:layout_gravity="center"/>


  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:paddingTop="10dp"
    android:paddingBottom="10dp">

    <LinearLayout
      android:layout_width="0dp"
      android:layout_height="match_parent"
      android:orientation="vertical"
      android:layout_weight="1"
      android:paddingEnd="10dp"
      android:gravity="top">

      <ImageView
        android:id="@+id/pralkaImage"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:src="@drawable/pralka"
        android:layout_marginBottom="80dp"/>

      <ImageView
        android:id="@+id/odkurzaczImage"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:src="@drawable/odkurzacz"
        android:layout_marginBottom="20dp"/>
    </LinearLayout>

    <LinearLayout
      android:layout_width="0dp"
      android:layout_height="match_parent"
      android:orientation="vertical"
      android:layout_weight="1"
      android:paddingStart="10dp"
      android:gravity="start">

      <TextView
        android:id="@+id/pralkaLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pralka"
        android:textSize="20sp"
        android:gravity="start"
        android:layout_marginBottom="20dp"/>

      <EditText
        android:id="@+id/pranieInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Podaj numer prania 1...12"
        android:inputType="number"
        android:background="#87CEEB"
        android:textColor="#000080"
        android:textSize="16sp"
        android:padding="10dp"
        android:layout_marginBottom="20dp"/>

      <Button
        android:id="@+id/zatwierdzButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Zatwierdź"
        android:textColor="#FFFFFF"
        android:layout_marginBottom="20dp"

android:background="@drawable/buttonrounded"
        android:layout_gravity="center_horizontal"/>

      <TextView
        android:id="@+id/wynikLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:textSize="16sp"
        android:gravity="start"
        android:layout_marginBottom="30dp"/>

      <TextView
        android:id="@+id/odkurzaczLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Odkurzacz"
        android:textSize="20sp"
        android:gravity="start"
        android:layout_marginBottom="30dp"/>

      <Button
        android:id="@+id/wlaczButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Włącz"

android:background="@drawable/buttonrounded"
        android:textColor="#FFFFFF"
        android:layout_marginBottom="20dp"
        android:layout_gravity="center"/>

      <TextView
        android:id="@+id/statusLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Odkurzacz wyłączony"
        android:textSize="16sp"
        android:gravity="start"
        android:layout_marginTop="20dp"/>
    </LinearLayout>
  </LinearLayout>

</LinearLayout>
```

Dodatkowe formatowanie dla przycisku

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <solid android:color="#4169E1"/> <!-- Kolor przycisku -->
   <corners android:radius="12dp"/> <!-- Zaokrąglenie rogów -->
   <stroke android:color="#ffffff" android:width="2dp"/> <!-- Obramowanie
-->
</shape>
```

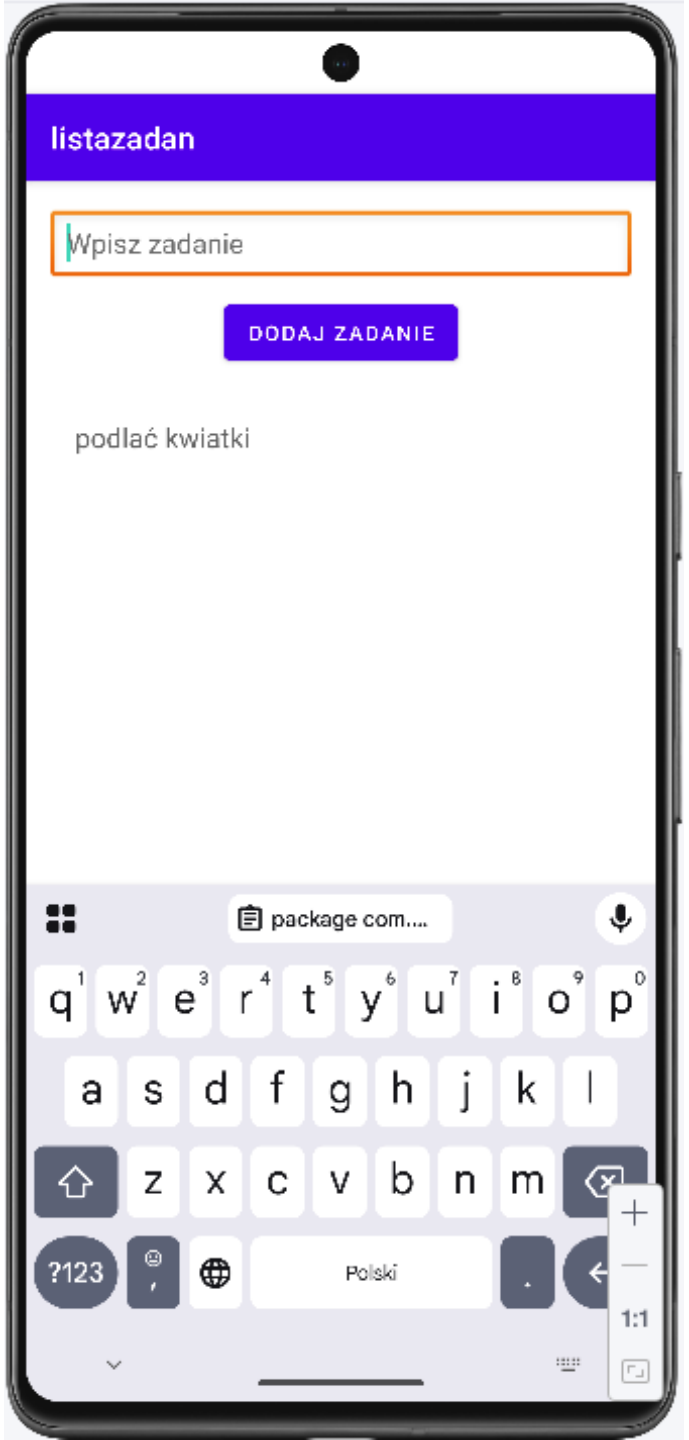
Struktura projektu i działanie programu



Przykładowe aplikacje – „Lista zadań”

MainActivity.java

```
package com.example.listazadan;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;
import java.util.List;
import com.example.listazadan.R;
import com.example.listazadan.Task;
import com.example.listazadan.TaskAdapter;

public class MainActivity extends AppCompatActivity {

    private EditText editTextTask;
    private Button buttonAdd;
    private RecyclerView recyclerViewTasks;
    private TaskAdapter adapter;
    private List<Task> taskList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editTextTask = findViewById(R.id.editTextTask);
        buttonAdd = findViewById(R.id.buttonAdd);
        recyclerViewTasks = findViewById(R.id.recyclerViewTasks);

        taskList = new ArrayList<>();
        adapter = new TaskAdapter(taskList);
        recyclerViewTasks.setLayoutManager(new LinearLayoutManager(this));
        recyclerViewTasks.setAdapter(adapter);

        buttonAdd.setOnClickListener(v -> {
            String taskText = editTextTask.getText().toString().trim();
            if (!taskText.isEmpty()) {
                taskList.add(new Task(taskText));
                adapter.notifyItemInserted(taskList.size() - 1);
                editTextTask.setText("");
            }
        });
    }
}
```



Task – najważniejszy element listy

```
package com.example.listazadan;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.List;
import com.example.listazadan.Task;
public class TaskAdapter extends
RecyclerView.Adapter<TaskAdapter.TaskViewHolder>
{
  private List<Task> tasks;

  public TaskAdapter(List<Task> tasks) {
    this.tasks = tasks;
  }
    public class TaskViewHolder extends
RecyclerView.ViewHolder {
    TextView taskText;

    public TaskViewHolder(View itemView) {
      super(itemView);
      taskText = itemView.findViewById(R.id.taskText);

      itemView.setOnClickListener(v -> {
        int position = getAdapterPosition();
        tasks.remove(position);
        notifyItemRemoved(position);
      });
    }
  }
  @NonNull
  @Override
  public TaskViewHolder
onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
    View view =
LayoutInflater.from(parent.getContext())
        .inflate(R.layout.task_item, parent, false);
    return new TaskViewHolder(view);
  }

  @Override
  public void onBindViewHolder(@NonNull
TaskViewHolder holder, int position) {

holder.taskText.setText(tasks.get(position).getTitle());
  }

  @Override
  public int getItemCount() {
    return tasks.size();
  }
}
```

Co to jest „Task”?

```
public class Task {
    private String title;

    public Task(String title) {
        this.title = title;
    }

    public String getTitle() {
        return title;
    }
}
```

Co robi ta klasa?
Title – to pole przechowujące treść zadania (czyli np. „Napisać raport”).
Konstruktor Task(String title) – tworzy nowe zadanie z podanym tekstem.
Metoda getTitle() – pozwala pobrać tytuł zadania (do wyświetlenia w aplikacji).
W skrócie: Task to obiekt reprezentujący jedno zadanie w Twojej aplikacji „lista zadań”.

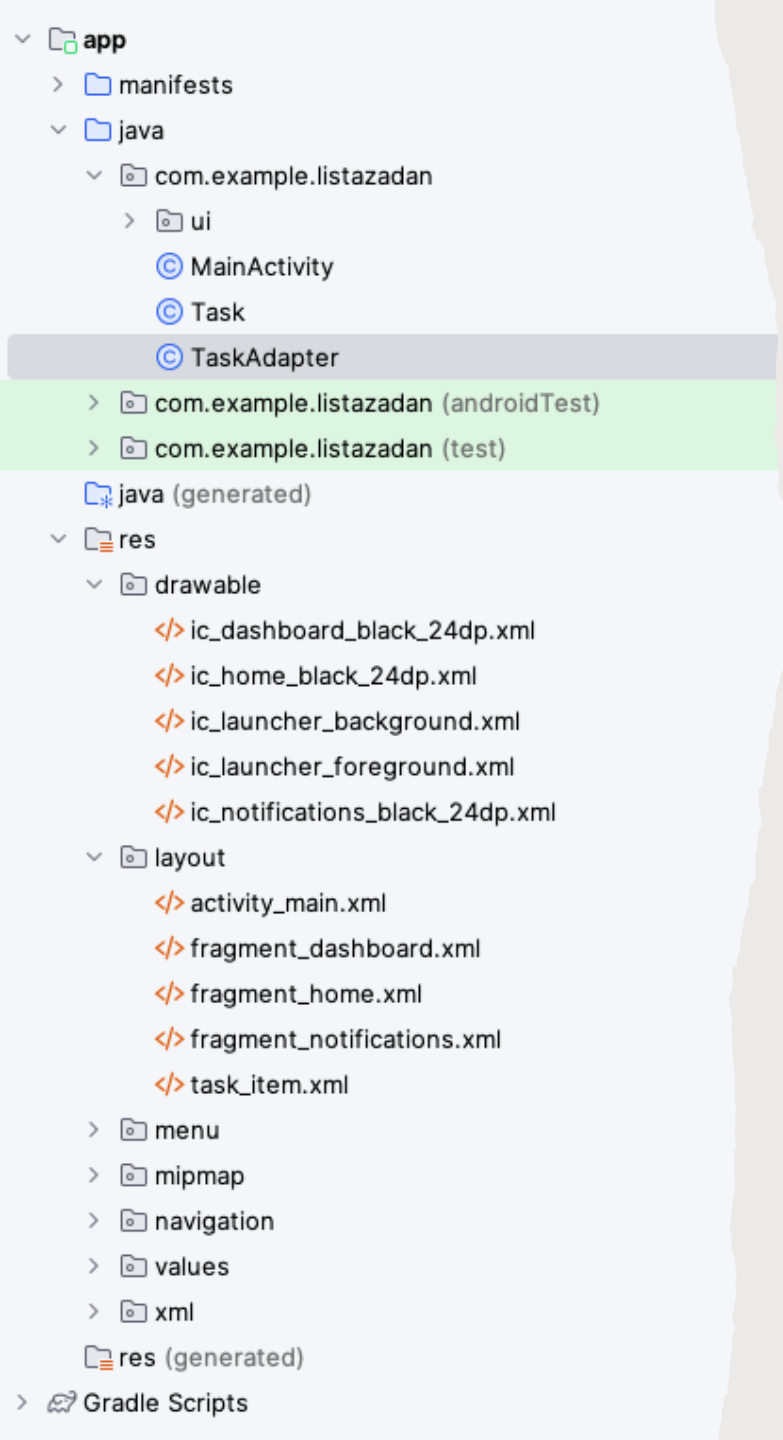
Klasa „Task” może zawierać:

```
public class Task {
    private String title;
    private String description;
    private String date;
    private boolean isDone;

    public Task(String title, String description, String date, boolean isDone) {
        this.title = title;
        this.description = description;
        this.date = date;
        this.isDone = isDone;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public String getDate() {
        return date;
    }

    public boolean isDone() {
        return isDone;
    }

    public void setDone(boolean done) {
        isDone = done;
    }
}
```

Klasa „Task” może zawierać dodatkowe pola, dzięki którym można rozszerzyć funkcjonalność projektu.

XML do projektu:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/
android"
   xmlns:app="http://schemas.android.com/apk/res-
auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/mainLayout"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp"
   tools:context=".MainActivity">

   <!-- EditText: przesunięte o 30dp do dołu -->
   <EditText
      android:id="@+id/editTextTask"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:hint="Wpisz zadanie"
      android:textColor="#000000"
      android:textColorHint="#808080"

android:background="@android:drawable/edit_text"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintTop_toTopOf="parent"

app:layout_constraintBottom_toTopOf="@+id/buttonA
dd"
      app:layout_constraintVertical_bias="0.4"
      android:layout_marginTop="100dp" /> <!--
Marginesa do góry -->

   <!-- Button: pod EditText, wyśrodkuj poziomo -->
   <Button
      android:id="@+id/buttonAdd"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Dodaj zadanie"

app:layout_constraintTop_toBottomOf="@id/editTextTa
sk"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      android:layout_marginTop="8dp" />

   <!-- RecyclerView: wypełnia dół ekranu -->
   <androidx.recyclerview.widget.RecyclerView
      android:id="@+id/recyclerViewTasks"
      android:layout_width="0dp"
      android:layout_height="0dp"
      android:layout_marginTop="16dp"

app:layout_constraintTop_toBottomOf="@id/buttonAd
d"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



Struktura projektu

Pliki Fragments

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".ui.dashboard.DashboardFragment">

  <TextView
    android:id="@+id/text_dashboard"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:textAlignment="center"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Przykładowe aplikacje – „Formularz z zapisem” (JAVA)

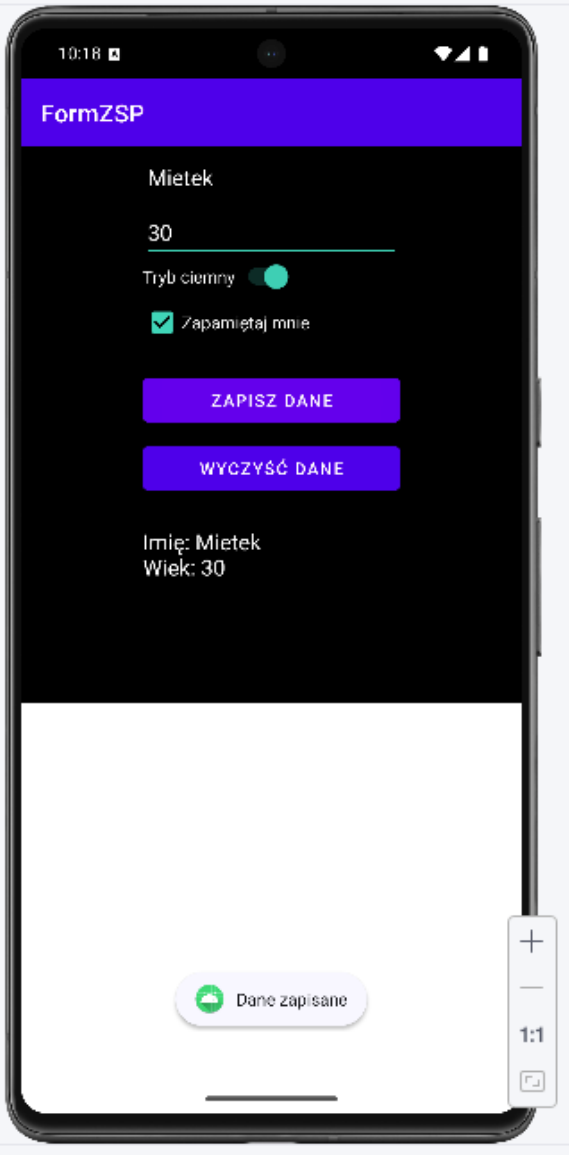
```
package com.example.formzsp;

import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
import android.view.ViewGroup;

public class MainActivity extends AppCompatActivity {

    private EditText nameEditText, ageEditText;
    private Switch darkModeSwitch;
    private CheckBox rememberMeCheckbox;
    private TextView outputText;
    private Button saveButton, clearButton;
    private ViewGroup rootLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Inicjalizacja widoków
        rootLayout = findViewById(R.id.rootLayout);
        nameEditText = findViewById(R.id.nameEditText);
        ageEditText = findViewById(R.id.ageEditText);
        darkModeSwitch = findViewById(R.id.darkModeSwitch);
        rememberMeCheckbox = findViewById(R.id.rememberMeCheckbox);
        outputText = findViewById(R.id.outputText);
        saveButton = findViewById(R.id.saveButton);
        clearButton = findViewById(R.id.clearButton);

        SharedPreferences prefs = getSharedPreferences("myPrefs", MODE_PRIVATE);

        // Przywrócenie danych
        nameEditText.setText(prefs.getString("name", ""));
        ageEditText.setText(String.valueOf(prefs.getInt("age", 0)));
        darkModeSwitch.setChecked(prefs.getBoolean("dark_mode", false));
        rememberMeCheckbox.setChecked(prefs.getBoolean("remember_me", false));

        // Zastosuj tryb ciemny jeśli włączony
        applyDarkMode(darkModeSwitch.isChecked());

        // Obsługa przełącznika
        darkModeSwitch.setOnCheckedChangeListener((buttonView, isChecked) -> {
            applyDarkMode(isChecked);
        });

        saveButton.setOnClickListener(v -> saveData(prefs));
        clearButton.setOnClickListener(v -> clearData(prefs));
    }

    private void applyDarkMode(boolean enabled) {
        int bgColor = enabled ? Color.BLACK : Color.WHITE;
        int textColor = enabled ? Color.WHITE : Color.BLACK;

        rootLayout.setBackgroundColor(bgColor);
        nameEditText.setTextColor(textColor);
        ageEditText.setTextColor(textColor);
        nameEditText.setHintTextColor(textColor);
        ageEditText.setHintTextColor(textColor);
        darkModeSwitch.setTextColor(textColor);
        rememberMeCheckbox.setTextColor(textColor);
        outputText.setTextColor(textColor);
        saveButton.setTextColor(textColor);
        clearButton.setTextColor(textColor);
    }

    private void saveData(SharedPreferences prefs) {
        String name = nameEditText.getText().toString();
        int age = 0;
        try {
            age = Integer.parseInt(ageEditText.getText().toString());
        } catch (NumberFormatException e) {
            Toast.makeText(this, "Wprowadź poprawny wiek",
Toast.LENGTH_SHORT).show();
            return;
        }

        boolean darkMode = darkModeSwitch.isChecked();
        boolean rememberMe = rememberMeCheckbox.isChecked();

        SharedPreferences.Editor editor = prefs.edit();
        editor.putString("name", name);
        editor.putInt("age", age);
        editor.putBoolean("dark_mode", darkMode);
        editor.putBoolean("remember_me", rememberMe);
        editor.apply();

        outputText.setText(getString(R.string.output_text, name, age));
        Toast.makeText(this, getString(R.string.toast_saved),
Toast.LENGTH_SHORT).show();
    }

    private void clearData(SharedPreferences prefs) {
        SharedPreferences.Editor editor = prefs.edit();
        editor.clear();
        editor.apply();
        nameEditText.setText("");
        ageEditText.setText("");
        darkModeSwitch.setChecked(false);
        rememberMeCheckbox.setChecked(false);
        outputText.setText("");
        Toast.makeText(this, getString(R.string.toast_cleared),
Toast.LENGTH_SHORT).show();
    }
}
```

XML – do głównej aktywności i do typu String

```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:id="@+id/rootLayout"
        android:orientation="vertical"
        android:padding="100dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <EditText
            android:id="@+id/nameEditText"
            android:hint="Imię"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <EditText
            android:id="@+id/ageEditText"
            android:hint="Wiek"
            android:inputType="number"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <Switch
            android:id="@+id/darkModeSwitch"
            android:text="Tryb ciemny"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <CheckBox
            android:id="@+id/rememberMeCheckbox"
            android:text="Zapamiętaj mnie"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/saveButton"
            android:text="Zapisz dane"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"/>

        <Button
            android:id="@+id/clearButton"
            android:text="Wyczyść dane"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"/>

        <TextView
            android:id="@+id/outputText"
            android:text=""
            android:layout_marginTop="24dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="18sp" />
    </LinearLayout>
</ScrollView>
```

strings.xml

```xml
<resources>
    <string name="title_home">Home</string>
    <string name="title_dashboard">Dashboard</string>
    <string name="title_notifications">Notifications</string>
    <string name="app_name">FormZSP</string>
    <string name="output_text">Imię: %1$s\nWiek: %2$d</string>
    <string name="toast_saved">Dane zapisane</string>
    <string name="toast_cleared">Dane wyczyszczone</string>
</resources>
```



Działanie kodu

Użytkownik wprowadza dane do formularza. Za pomocą przełącznika może zmienić kolor tła i kolor czcionki na przeciwny (biały na czarny). Po naciśnięciu przycisku dane są zapisywane za pomocą SharedPreferencies.

Dane są zapisywane w pamięci aplikacji, następnie można je usunąć za pomocą przycisku.

SharedPreferencies to ...

prosty mechanizm przechowywania danych w formie **par klucz–wartość**. Służy głównie do **zapisywania ustawień użytkownika** lub **małych danych trwałych** (np. imię, wiek, tryb ciemny), które mają być dostępne nawet po ponownym uruchomieniu aplikacji.

Do czego użyć?

Przykłady danych, które możesz zapisać w SharedPreferences:

- Czy użytkownik jest zalogowany (boolean)
- Nazwa użytkownika (String)
- Ostatnio używany kolor (int)
- Ustawienie trybu ciemnego (boolean).

Gdzie są przechowywane dane?

- W systemie Android

dane SharedPreferences są zapisywane w **pliku XML** wewnątrz katalogu aplikacji (/data/data/<twoja_aplikacja>/shared_prefs/), więc są **trwałe do czasu odinstalowania aplikacji**.

Przykładowe aplikacje – gra ”Zgaduj kartę” (Java)

```
package com.example.losujkarty;

import android.os.Bundle;
import android.view.View;
import android.widget.*;
import androidx.appcompat.app.AppCompatActivity;
import java.util.HashMap;
import java.util.Random;

public class MainActivity extends AppCompatActivity {

    private ImageView imgUserCard, imgDrawnCard;
    private TextView txtResult;
    private Spinner spinnerCards;
    private Button btnDraw;

    private String[] cardNames = {"as_pik", "krol_kier", "dama_karo",
"walet_trefl"};
    private HashMap<String, Integer> cardImages;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        spinnerCards = findViewById(R.id.spinnerCards);
        btnDraw = findViewById(R.id.btnDraw);
        imgUserCard = findViewById(R.id.imgUserCard);
        imgDrawnCard = findViewById(R.id.imgDrawnCard);
        txtResult = findViewById(R.id.txtResult);

        // Mapa z obrazami kart
        cardImages = new HashMap<>();
        cardImages.put("as_pik", R.drawable.as_pik);
        cardImages.put("krol_kier", R.drawable.krol_kier);
        cardImages.put("dama_karo", R.drawable.dama_karo);
        cardImages.put("walet_trefl", R.drawable.walet_trefl);

        // Spinner - wybór kart
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_dropdown_item, cardNames);
        spinnerCards.setAdapter(adapter);

        btnDraw.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                drawCard();
            }
        });
    }

    private void drawCard() {
        String selectedCard =
spinnerCards.getSelectedItem().toString();

imgUserCard.setImageResource(cardImages.get(selectedCard));

        // Losowanie karty
        Random random = new Random();
        String drawnCard =
cardNames[random.nextInt(cardNames.length)];

imgDrawnCard.setImageResource(cardImages.get(drawnCard));

        // Porównanie kart
        if (selectedCard.equals(drawnCard)) {
            txtResult.setText("Gratulacje! Trafiłeś swoją kartę!");
        } else {
            txtResult.setText("Spróbuj ponownie! Wylosowana karta: " +
drawnCard);
        }
    }
}
```

Dodatkowe klasy (Home Fragment i View Model)

```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;

import com.example.losujkarty.databinding.FragmentHomeBinding;

public class HomeFragment extends Fragment {

    private FragmentHomeBinding binding;

    private HomeFragment() {
    }

    public static HomeFragment createHomeFragment() {
        return new HomeFragment();
    }

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        HomeViewModel homeViewModel =
                new ViewModelProvider(this).get(HomeViewModel.class);

        binding = FragmentHomeBinding.inflate(inflater, container, false);
        View root = binding.getRoot();

        final TextView textView = binding.textHome;
        homeViewModel.getText().observe(getViewLifecycleOwner(), textView::setText);
        return root;
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}
```

```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;

import com.example.losujkarty.databinding.FragmentHomeBinding;

public class HomeFragment extends Fragment {

    private FragmentHomeBinding binding;

    private HomeFragment() {
    }

    public static HomeFragment createHomeFragment() {
        return new HomeFragment();
    }

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        HomeViewModel homeViewModel =
                new ViewModelProvider(this).get(HomeViewModel.class);

        binding = FragmentHomeBinding.inflate(inflater, container, false);
        View root = binding.getRoot();

        final TextView textView = binding.textHome;
        homeViewModel.getText().observe(getViewLifecycleOwner(), textView::setText);
        return root;
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}
```

Fragments – element wielu projektów

Fragment to:

- Część interfejsu użytkownika (UI), która może zawierać własny **layout**, **logikę** i **cykl życia**.
- Osadzany **wewnątrz aktywności (Activity)**.
- Może być **wielokrotnie wykorzystywany** w różnych miejscach aplikacji.

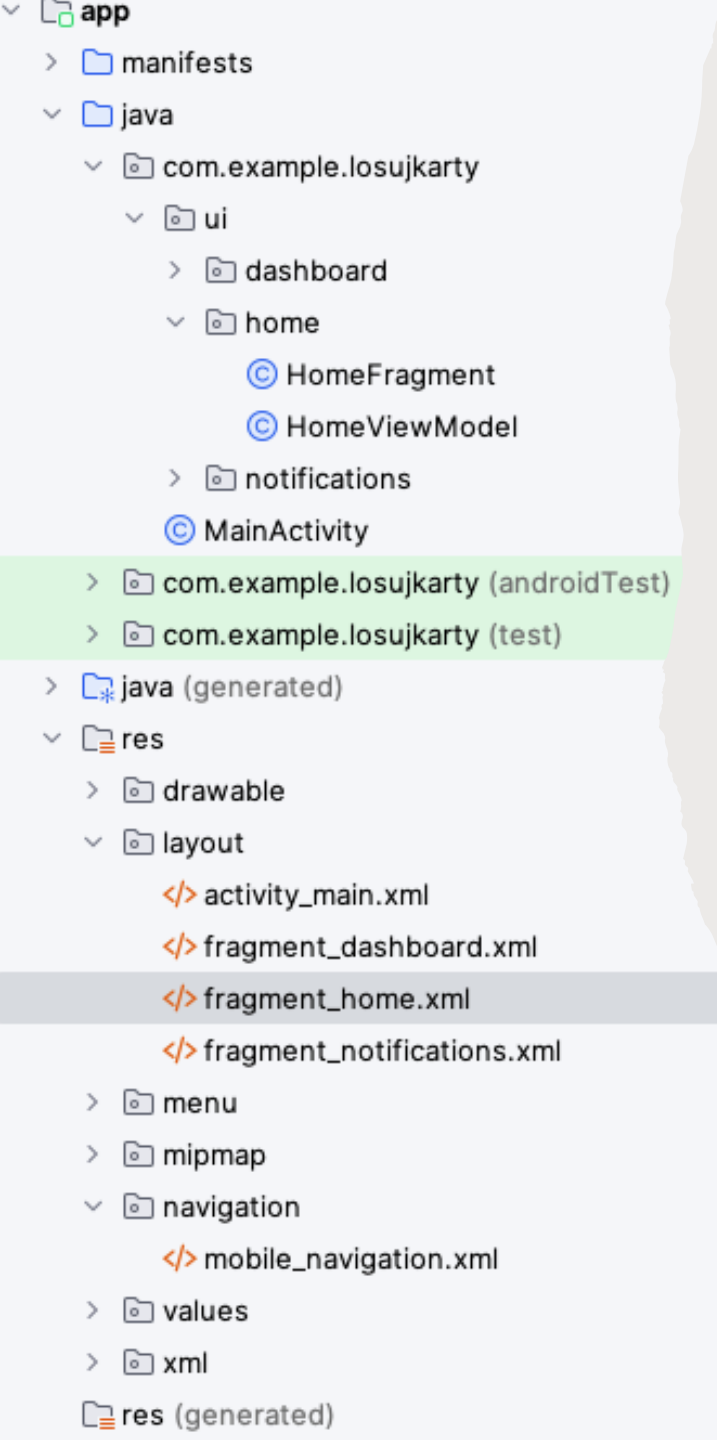
Activity	Fragment
Samodzielna jednostka UI	Część UI w Activity
Posiada własny cykl życia	Cykl życia zależny od Activity
Trudniejsze do dzielenia UI	Łatwiejsze w dzieleniu i ponownym użyciu
Idealna dla "głównych ekranów"	Idealne dla części interfejsu

XML do projektu

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:gravity="center"
    android:background="@android:color/white">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Wybierz swoją kartę:"
        android:textSize="18sp"
        android:layout_marginBottom="10dp"/>

    <Spinner
        android:id="@+id/spinnerCards"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/btnDraw"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Losuj kartę"
        android:layout_marginTop="20dp"/>

    <ImageView
        android:id="@+id/imgUserCard"
        android:layout_width="150dp"
        android:layout_height="200dp"
        android:layout_marginTop="20dp"/>

    <ImageView
        android:id="@+id/imgDrawnCard"
        android:layout_width="150dp"
        android:layout_height="200dp"
        android:layout_marginTop="20dp"/>

    <TextView
        android:id="@+id/txtResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:textStyle="bold"
        android:layout_marginTop="20dp"/>
</LinearLayout>
```

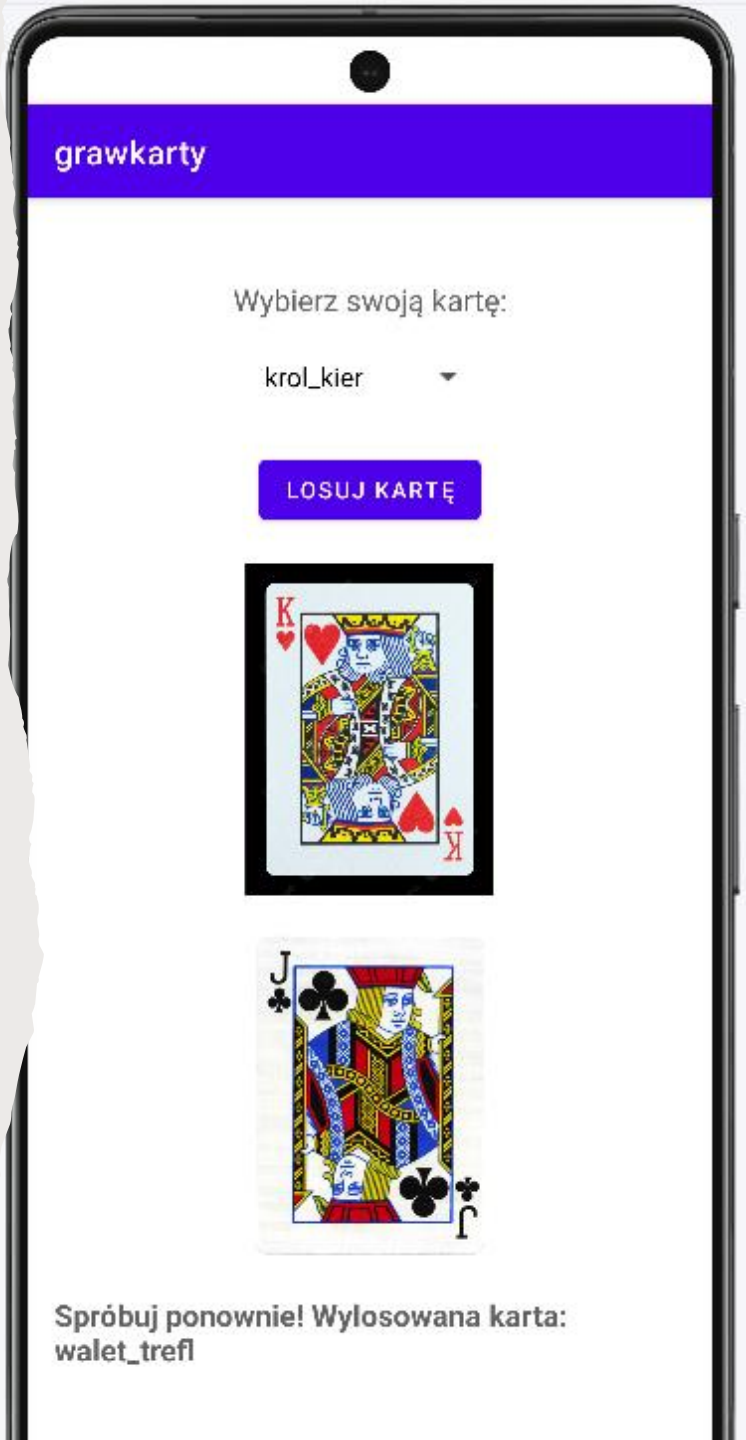
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.dashboard.DashboardFragment">

    <TextView
        android:id="@+id/text_dashboard"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:textAlignment="center"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```


```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

tools:context=".ui.dashboard.DashboardFragment">

    <TextView
        android:id="@+id/text_dashboard"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:textAlignment="center"
        android:textSize="20sp"

app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayou
t>
```



Struktura projektu i działanie

Użytkownik wybiera kartę a następnie program losuje kolejną kartę. Następuje porównanie i wyświetlenie wyniku.



Pytania i polecenia:

Co oznacza „zasada WORA”?

Wymień 5 kroków napisania gry

Na czym polega „RANDOM na obiektach”?

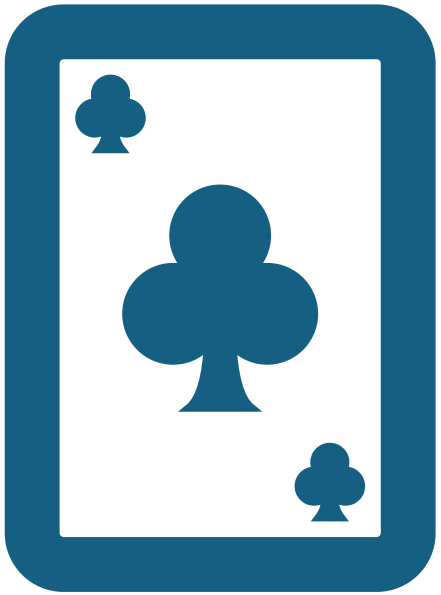
Co czego służy XML w aplikacjach Android?

Co czego służy „@overide”?

Do czego służy ConstraintLayout?

Wymień przynajmniej 5 właściwości specyficznych dla dowolnego layout

Do czego służą Fragments?

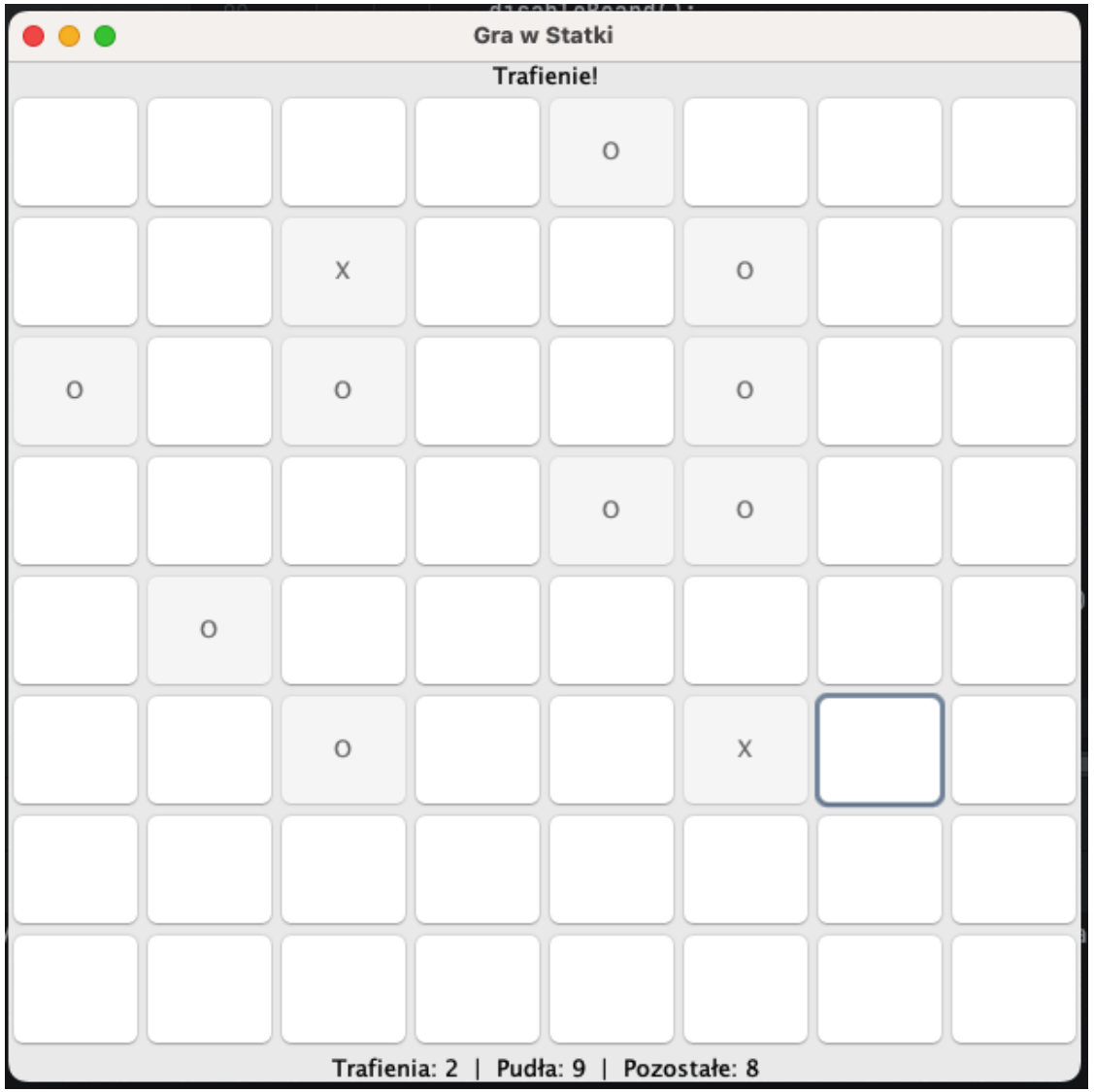


Zadania do samodzielnego wykonania

Zadanie 1

Za pomocą Swing i AWT napisz kod gry w statki, w której użytkownik typuje pola niewidzialnego statku.

Użytkownik wskazuje na pole. Po każdym naciśnięciu przycisku program zlicza trafienia, pudła i pokazuje pozostałe statki do zatopienia.

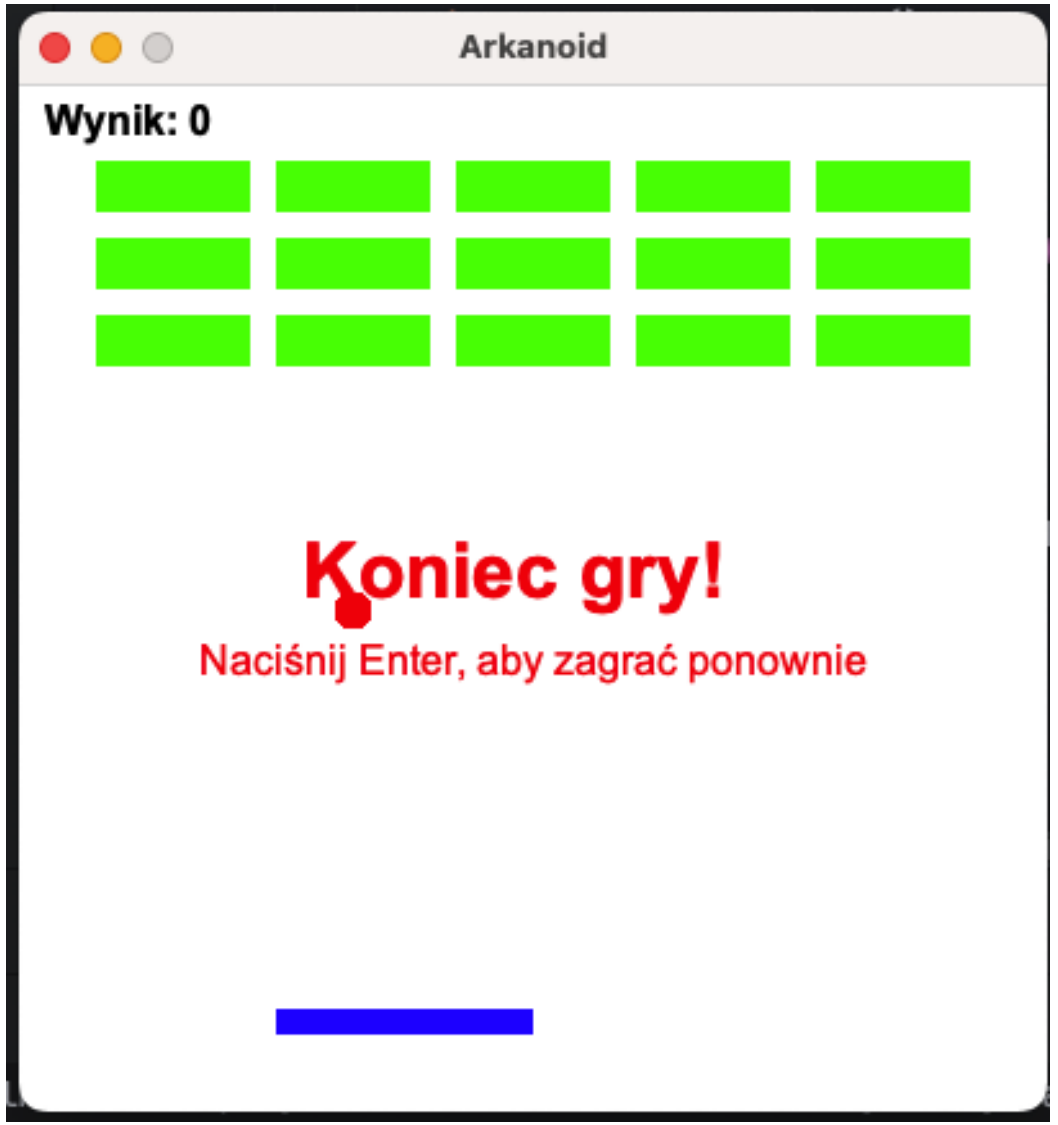


Zadania do samodzielnego wykonania

Zadanie 2

Za pomocą Swing., AWT i Canvas napisz kod gry, w której użytkownik porusza paletką za pomocą klawiatury. Za każdy klocek, którego dotyka piłka użytkownik otrzymuje 10 punktów.

Sterowanie odbywa się klawiszami kierunkowymi, główna pętla gry pozwala na powtórzenie rozgrywki.

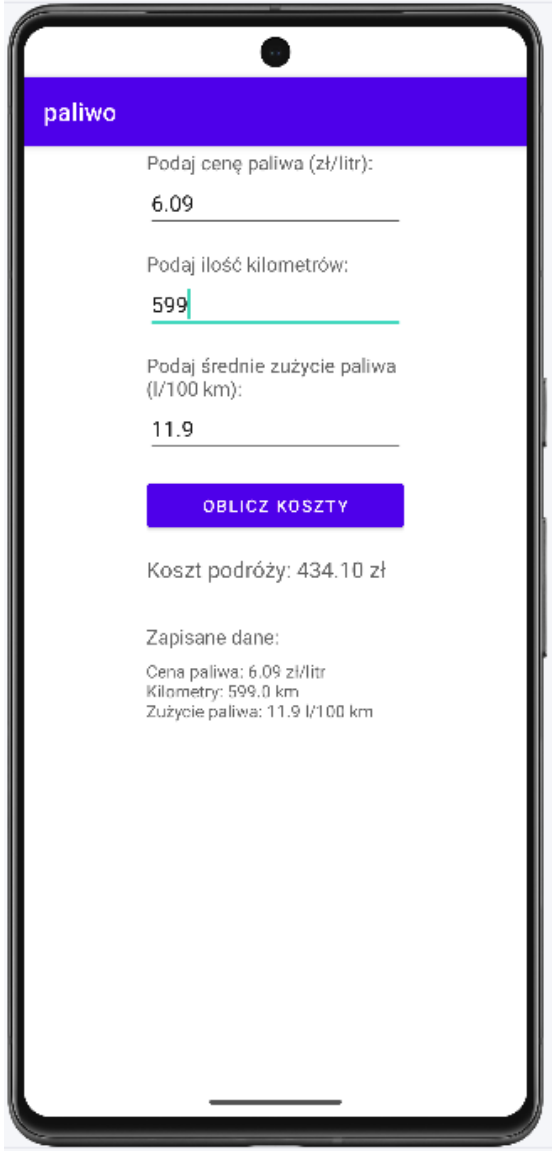


Zadania do samodzielnego wykonania

Zadanie 3

Napisz aplikację, która po wpisaniu danych przedstawionych na zdjęciu wyliczy koszt podróży.

Wszystkie dane pochodzą od użytkownika. Aplikacja wykorzystuje SharedPreferencies do zapisu danych. Dane są nadpisywane.

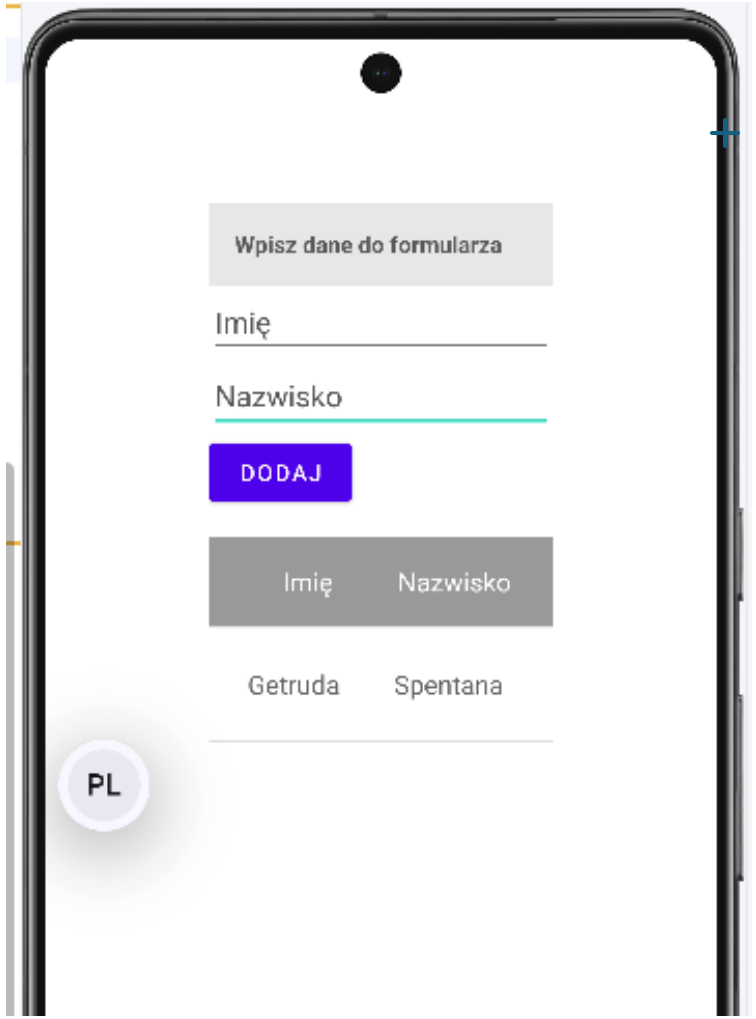


Zadania do samodzielnego wykonania

Zadanie 4 (na 5)

Napisz aplikację według wzoru. Projekt zawiera formularz, etykiety oraz tabelę z nagłówkiem, w której zapisywane są dane.

Projekt zawiera Adapter, kilka plików XML oraz Recycler View. Na następnej stronie znajdziesz szczegółową instrukcję.

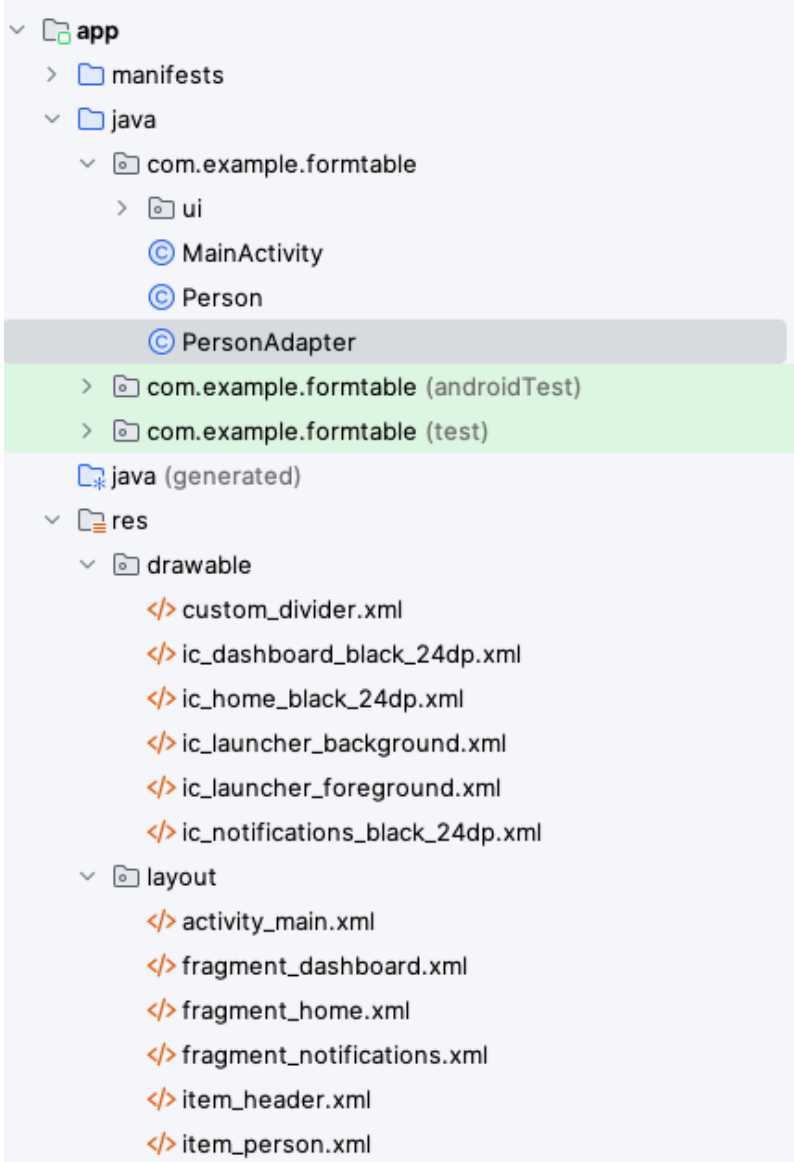


Zadania do samodzielnego wykonania

Zadanie 4 (na 5)

Główne zadania adaptera:

1. **Połączenie danych z widokiem**: Adapter przekształca dane (np. obiekty w liście) na widoki (np. elementy listy, wiersze tabeli).
2. **Zarządzanie widokami**: Adapter jest odpowiedzialny za tworzenie, aktualizowanie i usuwanie widoków w odpowiedzi na zmiany danych.
3. **Interakcja z elementami**: Może obsługiwać interakcje, takie jak kliknięcia na elementy listy, przewijanie itp.



Podpowiedź – dodatkowe pliki XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="horizontal"
  android:padding="16dp"
  android:background="@android:color/darker_gray">

  <TextView
    android:id="@+id/first_name_header"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Imię"
    android:textSize="16sp"
    android:textColor="@android:color/white"
    android:gravity="center"/>

  <TextView
    android:id="@+id/last_name_header"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Nazwisko"
    android:textSize="16sp"
    android:textColor="@android:color/white"
    android:gravity="center"/>
</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="horizontal"
  android:padding="16dp"
  android:background="@android:color/darker_gray">

  <TextView
    android:id="@+id/first_name_header"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Imię"
    android:textSize="16sp"
    android:textColor="@android:color/white"
    android:gravity="center"/>

  <TextView
    android:id="@+id/last_name_header"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Nazwisko"
    android:textSize="16sp"
    android:textColor="@android:color/white"
    android:gravity="center"/>
</LinearLayout>
```

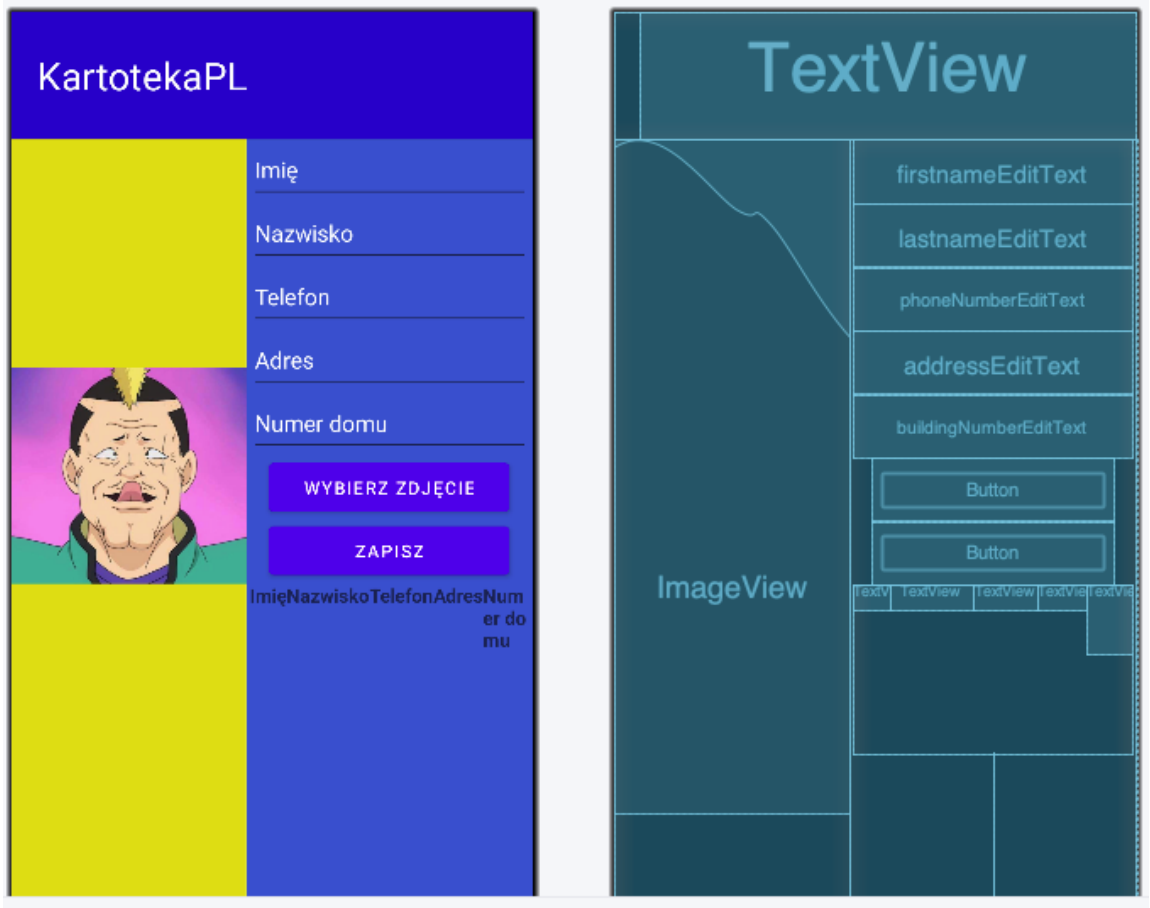
Zadania do samodzielnego wykonania

Zadanie 5

Za pomocą kreatora utwórz widok aplikacji według załączonego wzoru. Następnie utwórz kod aplikacji.

Do projektu wykorzystaj uprawnienia do przeglądania i załadowania zdjęcia do aplikacji.

Znajdziesz je na następnej stronie.



Dodawanie uprawnień w Manifeście

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">

    <!-- Uprawnienia aplikacji -->
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.MyApp">

        <!-- Inne elementy aplikacji -->

    </application>
</manifest>
```

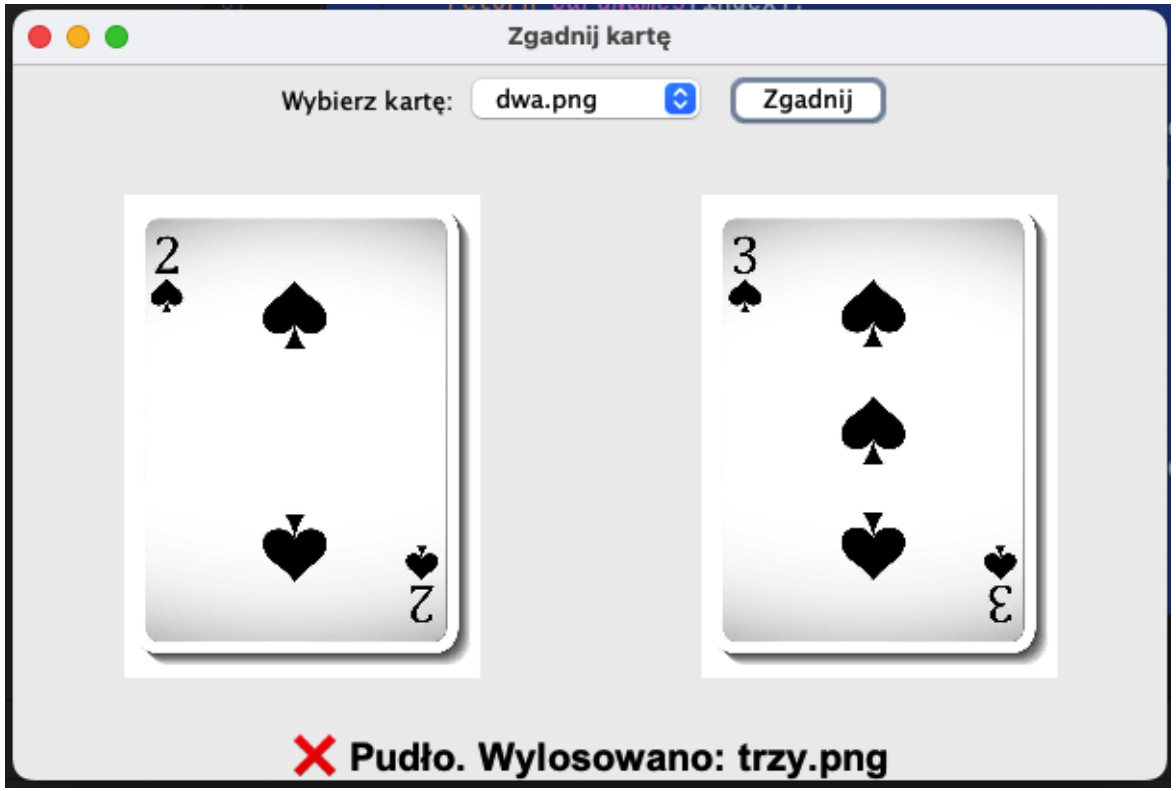
Uprawnienie	Opis	Deklaracja w AndroidManifest.xml
READ_EXTERNAL_STORAGE	Pozwala na odczyt plików z pamięci zewnętrznej urządzenia.	<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
WRITE_EXTERNAL_STORAGE	Pozwala na zapis plików w pamięci zewnętrznej urządzenia.	<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

Każde z tych uprawnień należy dodać w pliku AndroidManifest.xml, aby aplikacja miała odpowiednią możliwość działania w zależności od jej wymagań. Pamiętaj, że od Androida 6.0 (API 23) musisz także poprosić użytkowników o przyznanie tych uprawnień w czasie działania aplikacji, jeśli są to uprawnienia, które wymagają potwierdzenia przez użytkownika.

Zadania do samodzielnego wykonania

Zadanie 6

Tak, to nie pomyłka. Za pomocą Android Studio zaproponuj layout do znanej gry oraz napisz kod aplikacji. Skorzystaj ze zdjęć zapisanych w „RES” oraz z Fragments lub ze Spinnera.



Zadania do samodzielnego wykonania

Zadanie 7

Przekształć znaną aplikację w Aplikację w Android Studio. Do projektu dodaj więcej kostek a następnie wykaż czy została wyrzucona parzysta czy nieparzysta ilość oczek. Dodaj do projektu stosowną etykietę.

Android Studio na egzaminie INF.04

1. **Tworzenie aplikacji mobilnej** w Android Studio (Java lub Kotlin).
2. **Projektowanie layoutu** z przyciskami, polami tekstowymi, listami.
3. **Obsługa zdarzeń** (kliknięcia, zmiana tekstu).
4. **Zapis i odczyt danych** (SharedPreferences lub lokalna baza SQLite).
5. **Wyświetlanie danych** w tabeli / liście (RecyclerView).
6. **Prosta walidacja** danych.
7. **Stylizacja aplikacji** (kolory, ikony, marginesy).
8. **Tworzenie własnej klasy lub adaptera**.

Element	Uwaga
Gradle	Nie zmieniaj wersji Android Studio ani Gradle — użyj tego, co podano.
Internet	Może być wyłączony – upewnij się, że wszystkie biblioteki masz lokalnie.
Emulator	Nie musi działać – aplikacja może być sprawdzana tylko na kodzie.
Logcat / Debug	Przydaje się, ale nie wymagane.
Pliki projektu	Sprawdź, czy są poprawnie zapisane i uruchomione przez Sync.

Co musisz umieć przed egzaminem?

Podstawy:

- Struktura projektu Android Studio (Manifest, MainActivity, res/layout, java/, gradle/)
- Tworzenie GUI za pomocą activity_main.xml
- Obsługa przycisków w MainActivity.java
- Ustawianie TextView, EditText, Button, RecyclerView.

Pomocne narzędzia

Pomocne narzędzia:

- **SharedPreferences** – do zapisu danych użytkownika.
- **RecyclerView** – do wyświetlania listy danych (jak todolist).
- **Gson** – do zapisu obiektów jako JSON (jeśli wymagane).

```
OilChangeApp/
├── java/com/example/oilchangeapp/
│   ├── MainActivity.java
│   ├── OilChangeEntry.java
│   └── OilChangeAdapter.java
├── res/
│   ├── layout/
│   │   └── activity_main.xml
│   └── values/
│       └── strings.xml
├── AndroidManifest.xml
└── build.gradle
```

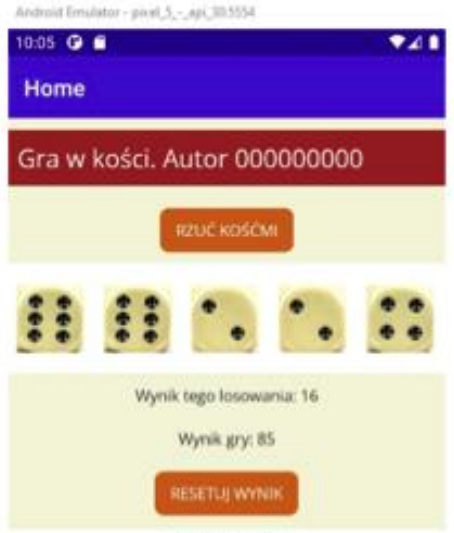
Przykładowy projekt egzaminacyjny

Część II. Aplikacja mobilna

Za pomocą środowiska programistycznego dostępnego na stanowisku egzaminacyjnym wykonaj aplikację mobilną do gry w kości. Do zbudowania aplikacji zastosuj obrazy z archiwum *pliki1.zip* zabezpieczonego hasłem **Gra^Kosci5**

Obraz 3.
Środowisko .NET MAUI, stan początkowy. Emulacja Pixel 5

Obraz 4.
Środowisko Android Studio, stan początkowy. Emulacja Pixel 5



Obraz 5.
Po wciśnięciu przycisku RZUĆ KOŚĆMI

Obraz 6.
Po wciśnięciu przycisku RESETUJ WYNIK

Wykorzystanie .NET MAUI

Czym jest .NET MAUI?
(.NET Multi-platform App UI) – framework firmy Microsoft umożliwiający tworzenie aplikacji na wiele platform z jednego kodu źródłowego (Android, iOS, macOS, Windows).

- **Dlaczego warto?**
- Wspólna baza kodu
- Nowoczesny interfejs użytkownika
- Integracja z .NET i C#

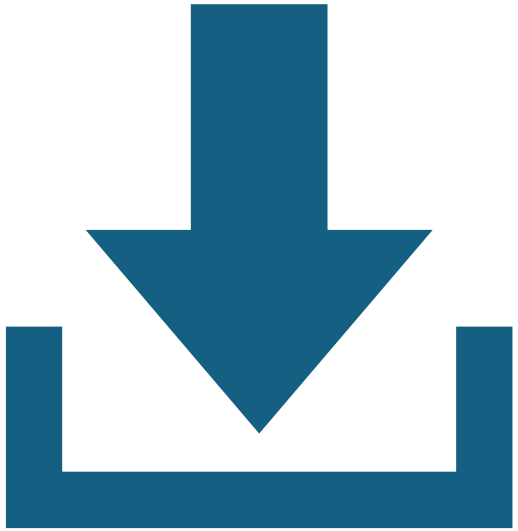
Środowisko i technologie:

- **Język programowania**: C#
- **Framework**: .NET 8 + .NET MAUI
- **IDE**: Visual Studio 2022/2023
- **Inne narzędzia**:
    - Android Emulator / fizyczne urządzenie
    - SQLite dla lokalnej bazy danych.

Architektura aplikacji w MAUI

Warstwy aplikacji:

- **UI (XAML)** – definiowanie wyglądu
- **Code-behind (C#)** – logika widoku
- **Model** – struktura danych (np. klasa Note)
- **Data Access** – zapisywanie i odczyt danych (SQLite)



Instalacja MAUI wewnątrz Visual Studio

1. Zainstaluj Visual Studio 2022/2023

Pobierz z: https://visualstudio.microsoft.com/pl/

Wybierz wersję **Community** (darmowa)

2. Podczas instalacji zaznacz:

.NET Multi-platform App UI development

(Opcjonalnie) **Mobile development with .NET**

Android SDK/Emulator – jeśli chcesz testować aplikacje na Androida

Jeśli już masz Visual Studio – uruchom **Instalator Visual Studio** → "Zmień" → dodaj powyższe komponenty.

3. Zaktualizuj .NET SDK

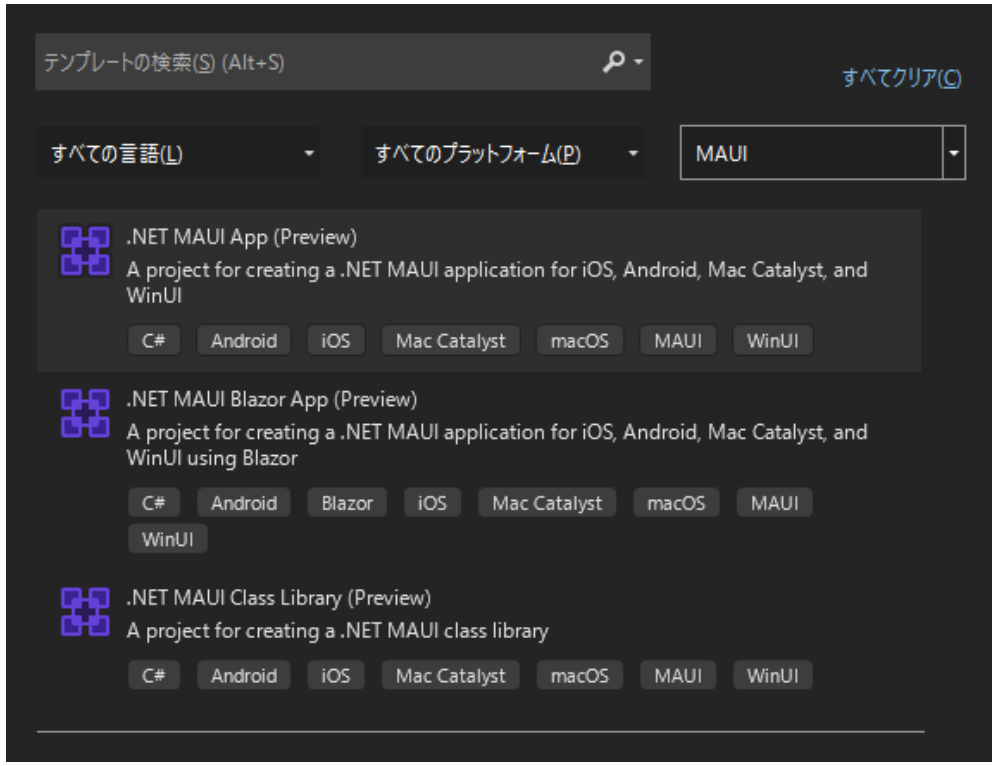
Pobierz najnowsze .NET SDK:
https://dotnet.microsoft.com/en-us/download

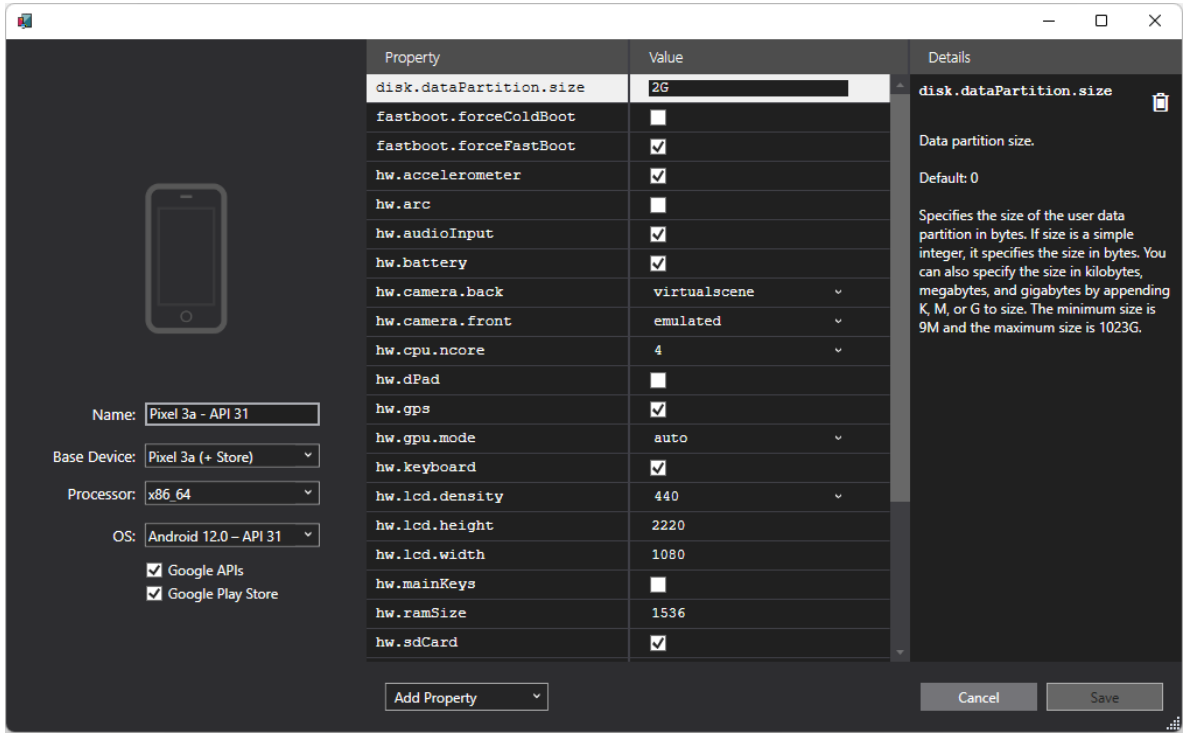
Wersja **.NET 8 lub nowsza** (MAUI działa najlepiej z .NET 8)

4. Sprawdź, czy MAUI działa

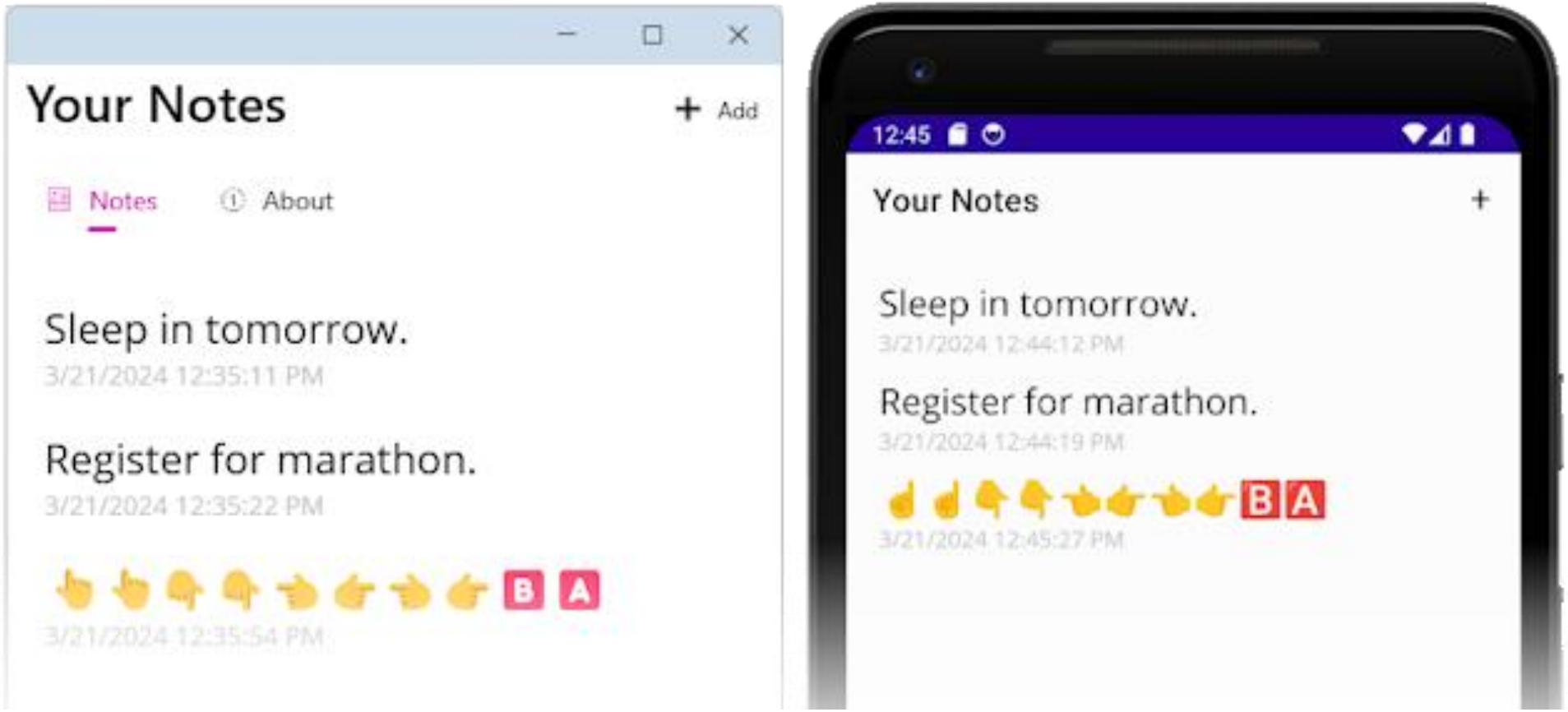
- Otwórz terminal (CMD / PowerShell) i wpisz:
- bash
- KopiujEdytuj
- dotnet workload install maui dotnet workload restore

Okno programu





Okno programu



Koniec?

W prezentacji znajdują się najważniejsze elementy podstawy programowej oraz rozszerzenia pozwalające tworzyć różnorodne projekty.

Pozostałe elementy podstawy programowej znajdziecie w dedykowanych zadaniach do przedmiotów zawodowych oraz oczywiście poznacie na zajęciach.