
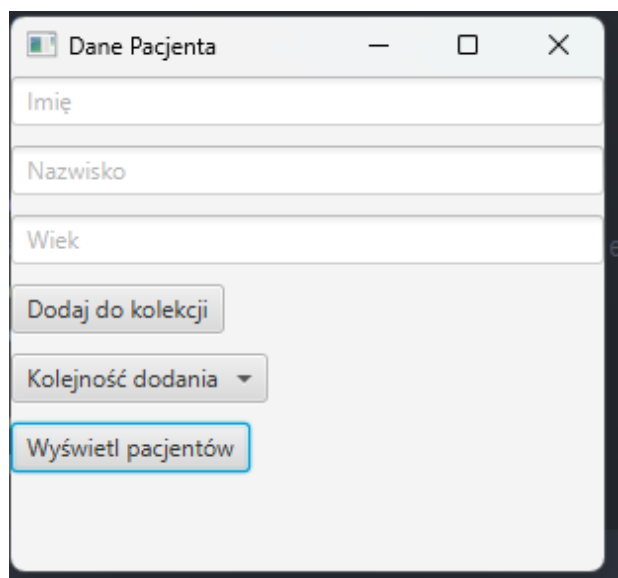


SPRAWOZDANIE NR 7			
Nazwa ćwiczenia	Lab07 - pacjent		 POLITECHNIKA BYDGOSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki
Przedmiot	Programowanie obiektowe		
Student grupa	Paweł Jońca gr 7		
Data ćwiczeń	19.01.2025	21.01.2025	Data oddania sprawozdania

Program



```

Dodano pacjenta: [
  "id": 1,
  "imie": "aneta",
  "nazwisko": "krak",
  "wiek": 30
]
Dodano pacjenta: [
  "id": 2,
  "imie": "karol",
  "nazwisko": "bąk",
  "wiek": 31
]
Dodano pacjenta: [
  "id": 3,
  "imie": "irena",
  "nazwisko": "toral",
  "wiek": 19
]

```

Sortowanie

```
=====
Sortowanie według nazwisk:
[
  "id": 2,
  "imie": "karol",
  "nazwisko": "bąk",
  "wiek": 31
]
[
  "id": 1,
  "imie": "aneta",
  "nazwisko": "krak",
  "wiek": 30
]
[
  "id": 3,
  "imie": "irena",
  "nazwisko": "toral",
  "wiek": 19
]
]
=====

Sortowanie według kolejności dodania:
[
  "id": 1,
  "imie": "aneta",
  "nazwisko": "krak",
  "wiek": 30
]
[
  "id": 2,
  "imie": "karol",
  "nazwisko": "bąk",
  "wiek": 31
]
[
  "id": 3,
  "imie": "irena",
  "nazwisko": "toral",
  "wiek": 19
]
]
=====
```

```
=====
Sortowanie według imion:
[
  "id": 1,
  "imie": "aneta",
  "nazwisko": "krak",
  "wiek": 30
]
[
  "id": 3,
  "imie": "irena",
  "nazwisko": "toral",
  "wiek": 19
]
[
  "id": 2,
  "imie": "karol",
  "nazwisko": "bąk",
  "wiek": 31
]
]
=====

Sortowanie według wieku:
[
  "id": 3,
  "imie": "irena",
  "nazwisko": "toral",
  "wiek": 19
]
[
  "id": 1,
  "imie": "aneta",
  "nazwisko": "krak",
  "wiek": 30
]
[
  "id": 2,
  "imie": "karol",
  "nazwisko": "bąk",
  "wiek": 31
]
]
=====
```

```

public class PacjentApp extends Application {

    // Kolekcja pacjentów
    private final List<Pacjent> pacjenci = new ArrayList<>(); 2 usages

    // Zmienna do generowania unikalnych ID dla pacjentów
    private int currentId = 1; 1 usage

    public static void main(String[] args) {
        launch(args);
    }
}

```

```

@Override
public void start(Stage primaryStage) {
    // Pola tekstowe do wprowadzania danych pacjenta
    TextField imieField = new TextField();
    imieField.setPromptText("Imię");

    TextField nazwiskoField = new TextField();
    nazwiskoField.setPromptText("Nazwisko");

    TextField wiekField = new TextField();
    wiekField.setPromptText("Wiek");

    // Przycisk dodawania pacjenta
    Button dodajButton = new Button(s: "Dodaj do kolekcji");
    dodajButton.setOnAction(e -> {
        try {
            String imie = imieField.getText();
            String nazwisko = nazwiskoField.getText();
            int wiek = Integer.parseInt(wiekField.getText());

            Pacjent pacjent = new Pacjent(currentId++, imie, nazwisko, wiek);
            pacjenci.add(pacjent);

            // Czyścimy pola tekstowe
            imieField.clear();
            nazwiskoField.clear();
            wiekField.clear();

            System.out.println("Dodano pacjenta: " + pacjent.toJson());
        } catch (NumberFormatException ex) {
            System.err.println("Wiek musi być liczbą całkowitą!");
        }
    });
}

```

```

// ChoiceBox do wyboru kryterium sortowania
ChoiceBox<String> choiceBox = new ChoiceBox<>();
choiceBox.getItems().addAll(...es: "Nazwiska", "Imienia", "Wiek", "Kolejność dodania");
choiceBox.setValue("Nazwiska");

// Przycisk wyświetlania pacjentów
Button wyswietlButton = new Button(s: "Wyświetl pacjentów");
wyswietlButton.setOnAction(e -> {
    List<Pacjent> sortedList = new ArrayList<>(pacjenci);

    String sortingCriteria = choiceBox.getValue();
    switch (sortingCriteria) {
        case "Nazwiska":
            sortedList.sort(Comparator.comparing(Pacjent::getNazwisko));
            printPatients( sortingCriteria: "Nazwiska", sortedList);
            break;
        case "Imienia":
            sortedList.sort(Comparator.comparing(Pacjent::getImie));
            printPatients( sortingCriteria: "Imienia", sortedList);
            break;
        case "Wiek":
            sortedList.sort(Comparator.comparingInt(Pacjent::getWiek));
            printPatients( sortingCriteria: "Wiek", sortedList);
            break;
        case "Kolejność dodania":
            sortedList.sort(Comparator.comparingInt(Pacjent::getId));
            printPatients( sortingCriteria: "Kolejność dodania", sortedList);
            break;
    }
});

```

```

// Układ GUI
VBox root = new VBox(v: 10, imieField, nazwiskoField, wiekField, dodajButton, choiceBox, wyswietlButton);
Scene scene = new Scene(root, v: 300, v1: 250);

primaryStage.setTitle("Dane Pacjenta");
primaryStage.setScene(scene);
primaryStage.show();
}

```

```

// Metoda do wyświetlania pacjentów z tytułem sortowania
private void printPatients(String sortingCriteria, List<Pacjent> patients) { 4 usages
    System.out.println("=====");
    switch (sortingCriteria) {
        case "Nazwiska":
            System.out.println("Sortowanie według nazwisk:");
            break;
        case "Imienia":
            System.out.println("Sortowanie według imion:");
            break;
        case "Wiek":
            System.out.println("Sortowanie według wieku:");
            break;
        case "Kolejność dodania":
            System.out.println("Sortowanie według kolejności dodania:");
            break;
        default:
            System.out.println("Nieznane kryterium sortowania:");
            break;
    }

    for (Pacjent pacjent : patients) {
        System.out.println(pacjent.toJson());
    }
    System.out.println("=====");
}
}

```

```

class Pacjent { 10 usages
    private final int id; // Id pacjenta 3 usages
    private String imie; 4 usages
    private String nazwisko; 4 usages
    private int wiek; 4 usages

    public Pacjent(int id, String imie, String nazwisko, int wiek) { 1 usage
        this.id = id;
        this.imie = imie;
        this.nazwisko = nazwisko;
        this.wiek = wiek;
    }

    public int getId() { 1 usage
        return id;
    }

    public String getImie() { 1 usage
        return imie;
    }

    public void setImie(String imie) { no usages
        this.imie = imie;
    }

    public String getNazwisko() { 1 usage
        return nazwisko;
    }

    public void setNazwisko(String nazwisko) { no usages
        this.nazwisko = nazwisko;
    }

    public int getWiek() { 1 usage
        return wiek;
    }

    public void setWiek(int wiek) { no usages
        this.wiek = wiek;
    }

    // Ręczne formatowanie JSON
    public String toJson() { 2 usages
        return String.format("[\n \"id\": %d,\n \"imie\": \"%s\", \n \"nazwisko\": \"%s\", \n \"wiek\": %d\n]2123", id, imie, nazwisko, wiek);
    }
}

```

Wnioski:

Program spełnia wymagania podane w treści zadania. Program umożliwia wprowadzanie danych pacjentów i dodawanie ich do wykorzystywania kolekcji. Spełnia wymagania sortowania. Do kolejności dodawania dodałem id, które pozwala zachować kolejność dodawania pacjentów co zwiększa funkcjonalność