SPRAWOZDANIE NR 4				
Nazwa ćwiczenia	MECHANIZM RE	MECHANIZM REFLEKSJI W JAVIE		
Przedmiot	Zaawansowane progr	amowanie obiektowe	POLITECHNIKA BYDGOSKA	
Student Grupa Nr indeksu	Paweł Jońca gr 7 122348		BY JUGUSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki	
Data ćwiczeń	19.05	23.06	Data oddania sprawozdania	

# Spis treści

Treść zadania	1
Rozwiązanie problemu	2
Kod do ObjectEditorApp	
Kod do SampleBean	
Wygląd aplikacji w działaniu	5
Wnioski	6

### Treść zadania

Napisać aplikację okienkową (z wykorzystaniem bilbioteki Swing lub JavaFX), która za pomocą mechanizmu reflekcji pozwoli automatycznie wykrywać, edytować i zmieniać właściwości obiektu dowolnej klasy napisanej w konwencji JavaBean. Właściwości to prywatne pola obiektu, do których dostęp jest kontrolowany przez metody dostępowe tzw. gettery i settery. Aplikacją powinna spełniać następujące wymagania:

Po uruchomieniu aplikacji należy podać nazwę klasy wraz z nazwą pakietu (1). Przycisk Create Object (2) pozwala utworzyć obiekt wybranej klasy (przy użyciu refleksji) i przejść do edycji wszystkich właściwości tego obiektu (3). Przycisk Save Changes (4) ma umożliwiać zapisanie wprowadzonych zmian, które powinny być prezentowanie w prostej konsoli aplikacji (5).

W przypadku wprowadzenia niepoprawnej wartości (6), w konsoli aplikacji powinien pojawić się stosowny komunikat o braku możliwości zapisania zmiany dla tej konkretnej właściwości (7). Sytuacja ta jednak nie powinna wpływać na przypisanie wartości (o ile są poprawne) pozostałym właściwościom.

Jeśli w nazwie pola wykorzystywanej klasy zawarte będzie słowo "text", do edycji należy wykorzystać komponent TextArea. W pozostałych przypadkach wykorzystamy TextField, także dla typu logicznego (zadanie dodatkowe dla chętnych, można wykorzystać komponent Checkbox dla pól typu logicznego) Aplikacja powinna prawidłowo obsługiwać właściwości typu Boolean, Byte, Short, Integer, Long, Float, Double, String, Character, jak również typy proste odpowiadające wymienionym klasom (boolean, byte, short, int etc.).

Settery i Gettery dla poszczególnych pól należy również wywoływać za pomocą refleksji, pozyskując nazwy tych metod z nazw pól danej klasy i dodając odpowiedni przedrostek set lub get. Przyjmijmy dla ułatwienia, że getter dla pola xxx typu logicznego zamiast isXxx() będzie miał postać getXxx().

## Rozwiązanie problemu

#### Kod do ObjectEditorApp

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.inse
import javafx.scene.Scene;
import javafx.scene.control
import javafx.scene.layout.
import javafx.stage.Stage;
import java.lang.reflect.*;
import java.util.*;
public class ObjectEditorApp extends Application {
     @Override
           logArea.setEditable(false);
           logArea.setPrefRowCount(8);
          VBox layout = new VBox(10, classInputField, createInstanceButton,
           layout.setPadding(new Insets(10));
           window.setScene(new Scene(layout, 600, 600));
```

```
private void showAttributes(Class<?> clazz) {
                Label attributeLabel = new Label(attributeName + ":");
                    TextArea textArea = new TextArea(fieldValue != null ?
field.getType() == Boolean.class) {
                    checkBox.setSelected(fieldValue != null &&
                    TextField textField = new TextField(fieldValue != null
       Class<?> objectClass = activeObject.getClass();
        for (Field field : objectClass.getDeclaredFields()) {
            String attributeName = field.getName();
            String capitalizedName =
attributeName.substring(1);
                Method setterMethod =
objectClass.getMethod(setterMethodName, field.getType());
```

```
logArea.appendText("Pole " + attributeName + " zmienione na
            if (targetType == int.class || targetType == Integer.class)
return Integer.parseInt(inputValue);
            if (targetType == byte.class || targetType == Byte.class)
return Byte.parseByte(inputValue);
    public static void main(String[] args) {
```

#### Kod do SampleBean

```
package pl.edu.reflectioneditor;

public class SampleBean {
    private String name;
    private int age;
```

```
private boolean active;
private String descriptionText;

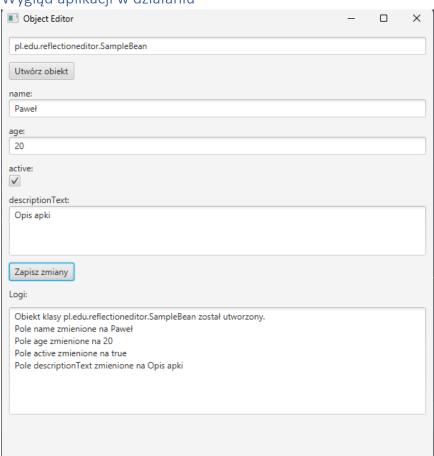
public SampleBean() {
    this.name = "Paweł";
    this.age = 20;
    this.active = true;
    this.descriptionText = "Opis apki";
}

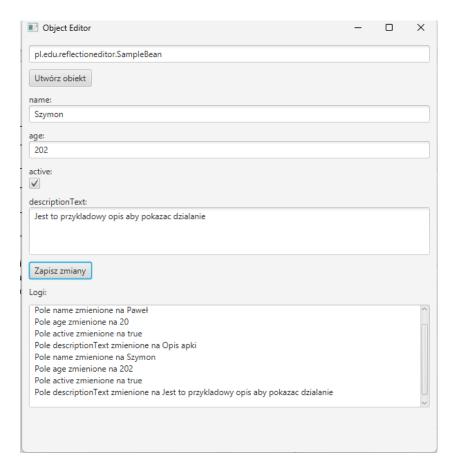
public String getName() { return name; }
    public void setName(String name) { this.name = name; }

public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }

public boolean getActive() { return active; }
    public String getDescriptionText() { return descriptionText; }
    public void setDescriptionText() { return descriptionText; }
    public void setDescriptionText(String descriptionText) {
    this.descriptionText = descriptionText; }
}
```

#### Wygląd aplikacji w działaniu





### Wnioski

Zadanie nauczyło mnie jak korzystać z refleksji w Javie i jak za jej pomocą dynamicznie odczytywać pola, wywoływać metody oraz tworzyć obiekty. Dodatkowo nauczyłem się jak korzystać z JavaBean. Dzięki ćwiczeniu poznałem stosowanie konwencji JavaBean pola prywatne, gettery i setery w projektowaniu klas. Dzięki temu możliwe było automatyczne wykrywanie właściwości obiektu i ich dynamiczna edycja.