


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

Spis treści

Treść zadania	1
Rozwiązanie problemu	2
Kod do ObjectEditorApp.....	2
Kod do SampleBean.....	4
Wygląd aplikacji w działaniu.....	5
Wnioski	6

Treść zadania

Napisać aplikację okienkową (z wykorzystaniem biblioteki Swing lub JavaFX), która za pomocą mechanizmu refleksji 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 dostępne tzw. gettery i settery. Aplikację 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 prezentowane 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

```
package pl.edu.reflectioneditor;

import javafx.application.Application;
import javafx.geometry.Insets;
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 {
    private VBox attributesBox = new VBox(10);
    private Object activeObject;
    private Map<String, Control> inputFields = new HashMap<>();
    private TextArea logArea = new TextArea();

    @Override
    public void start(Stage window) {
        TextField classInputField = new
        TextField("pl.edu.reflectioneditor.SampleBean");
        classInputField.setPromptText("Podaj pełną nazwę klasy");

        Button createInstanceButton = new Button("Utwórz obiekt");
        createInstanceButton.setOnAction(event -> {
            String className = classInputField.getText();
            try {
                Class<?> loadedClass = Class.forName(className);
                activeObject =
                loadedClass.getDeclaredConstructor().newInstance();
                showAttributes(loadedClass);
                logArea.appendText("Obiekt klasy " + className + " został
                utworzony.\n");
            } catch (Exception ex) {
                logArea.appendText("Błąd przy tworzeniu obiektu: " +
                ex.getMessage() + "\n");
            }
        });

        Button saveButton = new Button("Zapisz zmiany");
        saveButton.setOnAction(event -> saveUpdates());

        logArea.setEditable(false);
        logArea.setPrefRowCount(8);

        VBox layout = new VBox(10, classInputField, createInstanceButton,
        attributesBox, saveButton,
        new Label("Logi:"), logArea);
        layout.setPadding(new Insets(10));

        window.setScene(new Scene(layout, 600, 600));
    }
}
```

```

        window.setTitle("Object Editor");
        window.show();
    }

    private void showAttributes(Class<?> clazz) {
        attributesBox.getChildren().clear();
        inputFields.clear();
        for (Field field : clazz.getDeclaredFields()) {
            String attributeName = field.getName();
            String capitalizedName =
Character.toUpperCase(attributeName.charAt(0)) +
attributeName.substring(1);
            String getterMethodName = "get" + capitalizedName;

            try {
                Method getterMethod = clazz.getMethod(getterMethodName);
                Object fieldValue = getterMethod.invoke(activeObject);

                Label attributeLabel = new Label(attributeName + ":");
                Control inputControl;
                if (attributeName.toLowerCase().contains("text")) {
                    TextArea textArea = new TextArea(fieldValue != null ?
fieldValue.toString() : "");
                    textArea.setPrefRowCount(3);
                    inputControl = textArea;
                } else if (field.getType() == boolean.class ||
field.getType() == Boolean.class) {
                    CheckBox checkBox = new CheckBox();
                    checkBox.setSelected(fieldValue != null &&
Boolean.parseBoolean(fieldValue.toString()));
                    inputControl = checkBox;
                } else {
                    TextField textField = new TextField(fieldValue != null
? fieldValue.toString() : "");
                    inputControl = textField;
                }
                inputFields.put(attributeName, inputControl);
                attributesBox.getChildren().add(new VBox(attributeLabel,
inputControl));
            } catch (Exception e) {
                logArea.appendText("Błąd odczytu pola " + attributeName +
": " + e.getMessage() + "\n");
            }
        }
    }

    private void saveUpdates() {
        Class<?> objectClass = activeObject.getClass();
        for (Field field : objectClass.getDeclaredFields()) {
            String attributeName = field.getName();
            String capitalizedName =
Character.toUpperCase(attributeName.charAt(0)) +
attributeName.substring(1);
            String setterMethodName = "set" + capitalizedName;
            Control inputControl = inputFields.get(attributeName);
            if (inputControl == null) continue;

            try {
                Method setterMethod =
objectClass.getMethod(setterMethodName, field.getType());
                Object parsedValue = convertInput(inputControl,

```

```

field.getType());
        setterMethod.invoke(activeObject, parsedValue);
        logArea.appendText("Pole " + attributeName + " zmienione na " + parsedValue + "\n");
    } catch (Exception e) {
        logArea.appendText("Błąd zapisu pola " + attributeName + ": " + e.getMessage() + "\n");
    }
}

private Object convertInput(Control control, Class<?> targetType) {
    try {
        String inputValue;
        if (control instanceof TextArea) {
            inputValue = ((TextArea) control).getText();
        } else if (control instanceof TextField) {
            inputValue = ((TextField) control).getText();
        } else if (control instanceof CheckBox) {
            return ((CheckBox) control).isSelected();
        } else {
            return null;
        }

        if (targetType == String.class) return inputValue;
        if (targetType == int.class || targetType == Integer.class)
            return Integer.parseInt(inputValue);
        if (targetType == float.class || targetType == Float.class)
            return Float.parseFloat(inputValue);
        if (targetType == double.class || targetType == Double.class)
            return Double.parseDouble(inputValue);
        if (targetType == long.class || targetType == Long.class)
            return Long.parseLong(inputValue);
        if (targetType == short.class || targetType == Short.class)
            return Short.parseShort(inputValue);
        if (targetType == byte.class || targetType == Byte.class)
            return Byte.parseByte(inputValue);
        if (targetType == char.class || targetType == Character.class)
            return inputValue.charAt(0);
        if (targetType == boolean.class || targetType == Boolean.class)
            return Boolean.parseBoolean(inputValue);
    } catch (Exception e) {
        logArea.appendText("Błąd konwersji: " + e.getMessage() + "\n");
    }
    return null;
}

public static void main(String[] args) {
    launch(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 void setActive(boolean active) { this.active = active; }

public String getDescriptionText() { return descriptionText; }
public void setDescriptionText(String descriptionText) {
    this.descriptionText = descriptionText; }
}

```

Wygląd aplikacji w działaniu

The screenshot shows a Java Swing window titled "Object Editor". Inside the window, there is a text field containing the class name "pl.edu.reflectioneditor.SampleBean". Below it is a button labeled "Utwórz obiekt".

Underneath the button, there are four labeled input fields:

- "name:" with a text field containing "Paweł".
- "age:" with a text field containing "20".
- "active:" with a checked checkbox.
- "descriptionText:" with a text area containing "Opis apki".

Below these fields is a button labeled "Zapisz zmiany".

At the bottom of the window, there is a section labeled "Logi:" containing a text area with the following log messages:

- "Obiekt klasy pl.edu.reflectioneditor.SampleBean został utworzony."
- "Pole name zmienione na Paweł"
- "Pole age zmienione na 20"
- "Pole active zmienione na true"
- "Pole descriptionText zmienione na Opis apki"

The screenshot shows a Java Swing window titled "Object Editor". At the top, there is a text field containing the class name "pl.edu.reflectioneditor.SampleBean". Below it is a button labeled "Utwórz obiekt". The form then contains several fields: "name:" with the value "Szymon", "age:" with the value "202", "active:" with a checked checkbox, and "descriptionText:" with the value "Jest to przykładowy opis aby pokazac dzialanie". Below these fields are two buttons: "Zapisz zmiany" (highlighted with a blue border) and "Utwórz obiekt". At the bottom, there is a "Logi:" section with a scrollable text area containing a list of log messages, such as "Pole name zmienione na Pawel", "Pole age zmienione na 20", "Pole active zmienione na true", and "Pole descriptionText zmienione na Opis apki".

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.