|  |  |  |  |
| --- | --- | --- | --- |
| ***SPRAWOZDANIE NR 4*** | | | |
| Nazwa ćwiczenia | **Walidacja z wykorzystaniem własnej adnotacji** | | Obraz zawierający tekst  Opis wygenerowany automatycznie |
| Przedmiot | Zaawansowane programowanie obiektowe | |
| Student  Grupa  Nr indeksu | Paweł Jońca gr 7 122348 | |
| Data ćwiczeń |  | 28.06 | Data oddania sprawozdania |

Spis treści

[Treść zadania 1](#_Toc202025698)

[Rozwiązanie problemu 1](#_Toc202025699)

[Kod do UserFormApp 1](#_Toc202025700)

[Kod do RegexValidator 3](#_Toc202025701)

[Kod ValidatedTextField 3](#_Toc202025702)

[Kod UserData 5](#_Toc202025703)

[Kod FieldValidator 6](#_Toc202025704)

[Kod do ValidationPattern 6](#_Toc202025705)

[Wygląd aplikacji w działaniu 6](#_Toc202025706)

[Wnioski 7](#_Toc202025707)

# Treść zadania

Napisać aplikację okienkową (z wykorzystaniem bilbioteki Swing lub JavaFX), która umożliwi walidację wybranego pola dowolnej klasy w konwencji JavaBean, oznaczonego własną adnotacją. Zadania do wykonania i wymagania:

Utworzyć nową adnotację do walidacji oznaczonych pól klasowych. Adnotacja powinna bazować na wyrażeniach regularnych, które posłużą do sprawdzania poprawności wprowadzanych danych. ⎫ Dla dowolnej klasy utworzonej w konwencji JavaBean wybrać i oznaczyć za pomocą stworzonej adnotacji

# Rozwiązanie problemu

## Kod do UserFormApp

package lab06.validationapp;  
  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
import java.lang.reflect.Field;  
  
public class UserFormApp extends Application {  
 private final UserData userData = new UserData();  
 private Button submitButton;  
  
 @Override  
 public void start(Stage primaryStage) {  
 Label headerLabel = new Label("Rejestracja użytkownika");  
 headerLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");  
  
 Label nameLabel = new Label("Podaj imię:");  
  
 ValidatedTextField nameField = new ValidatedTextField(new TextField());  
 nameField.setMaxWidth(250);  
  
 submitButton = new Button("Zapisz");  
 submitButton.setDisable(true);  
 submitButton.setOnAction(e -> {  
 userData.setUsername(nameField.getText());  
 System.out.println("Użytkownik zapisany: " + userData.getUsername());  
 });  
  
 initializeValidation(nameField, "username");  
  
 nameField.textProperty().addListener((obs, oldVal, newVal) -> {  
 boolean isInputEmpty = newVal == null || newVal.isEmpty();  
 submitButton.setDisable(isInputEmpty || !nameField.isValid());  
 });  
  
 VBox layout = new VBox(15);  
 layout.setPadding(new Insets(25));  
 layout.setAlignment(Pos.CENTER);  
 layout.getChildren().addAll(headerLabel, nameLabel, nameField, submitButton);  
  
 Scene scene = new Scene(layout, 400, 250);  
 primaryStage.setScene(scene);  
 primaryStage.setTitle("Walidacja danych użytkownika");  
 primaryStage.setMinWidth(350);  
 primaryStage.setMinHeight(200);  
 primaryStage.centerOnScreen();  
 primaryStage.show();  
 }  
  
 private void initializeValidation(ValidatedTextField inputField, String fieldName) {  
 try {  
 Field field = UserData.class.getDeclaredField(fieldName);  
 if (field.isAnnotationPresent(ValidationPattern.class)) {  
 ValidationPattern annotation = field.getAnnotation(ValidationPattern.class);  
 FieldValidator validator = new RegexValidator(annotation.regex(), annotation.errorMessage());  
 inputField.addValidator(validator);  
 }  
 } catch (NoSuchFieldException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args) {  
 launch(args);  
 }  
}

## Kod do RegexValidator

package lab06.validationapp;  
  
import java.util.regex.Pattern;  
  
public class RegexValidator implements FieldValidator {  
 private final Pattern regexPattern; // Skompilowane wyrażenie regularne  
 private final String errorMessage; // Komunikat błędu  
 private boolean valid; // Status walidacji  
  
 public RegexValidator(String regex, String errorMessage) {  
 this.regexPattern = Pattern.compile(regex); // Kompilacja wyrażenia regularnego  
 this.errorMessage = errorMessage;  
 this.valid = false;  
 }  
  
 @Override  
 public void check(String value) {  
 // Sprawdzenie zgodności wartości z wzorcem  
 valid = regexPattern.matcher(value).matches();  
 }  
  
 @Override  
 public boolean isCorrect() {  
 return valid;  
 }  
  
 @Override  
 public String getErrorMessage() {  
 return errorMessage;  
 }  
}

## Kod ValidatedTextField

package lab06.validationapp;  
  
import javafx.beans.property.StringProperty;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextInputControl;  
import javafx.scene.control.Tooltip;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.layout.HBox;  
  
public class ValidatedTextField extends HBox {  
 private final TextInputControl inputControl;  
 private final Label validationIndicator;  
 private FieldValidator validator;  
 private boolean valid;  
 private boolean edited = false;  
  
 public ValidatedTextField(TextInputControl inputControl) {  
 this.inputControl = inputControl;  
 this.valid = false;  
  
 validationIndicator = new Label();  
 validationIndicator.setPrefSize(15, 15);  
 validationIndicator.setStyle("-fx-cursor: hand;");  
 validationIndicator.setVisible(false);  
 updateIndicator();  
  
 setSpacing(5);  
 getChildren().addAll(inputControl, validationIndicator);  
  
 inputControl.textProperty().addListener((observable, oldValue, newValue) -> {  
 if (!edited && !newValue.isEmpty()) {  
 edited = true;  
 validationIndicator.setVisible(true);  
 }  
 validateInput();  
 });  
 }  
  
 public StringProperty textProperty() {  
 return inputControl.textProperty();  
 }  
  
 public void addValidator(FieldValidator validator) {  
 this.validator = validator;  
 validateInput();  
 }  
  
 private void validateInput() {  
 if (validator != null) {  
 validator.check(inputControl.getText());  
 valid = validator.isCorrect();  
 Tooltip.uninstall(validationIndicator, null);  
 }  
 updateIndicator();  
 }  
  
 private void updateIndicator() {  
 if (!edited) {  
 validationIndicator.setVisible(false);  
 return;  
 }  
  
 String iconPath = valid ? "/success.png" : "/error.png";  
 try {  
 Image icon = new Image(getClass().getResourceAsStream(iconPath));  
 ImageView iconView = new ImageView(icon);  
 iconView.setFitWidth(15);  
 iconView.setFitHeight(15);  
 validationIndicator.setGraphic(iconView);  
  
 if (valid) {  
 Tooltip validTooltip = new Tooltip("Dane poprawne");  
 validTooltip.setStyle("-fx-font-size: 12px; -fx-text-fill: green;");  
 Tooltip.install(validationIndicator, validTooltip);  
 } else {  
 String errorMsg = validator != null ? validator.getErrorMessage() : "Nieprawidłowe dane";  
 Tooltip invalidTooltip = new Tooltip(errorMsg);  
 invalidTooltip.setStyle("-fx-font-size: 12px; -fx-text-fill: red;");  
 Tooltip.install(validationIndicator, invalidTooltip);  
 }  
 } catch (Exception e) {  
 validationIndicator.setText(valid ? "✓" : "✗");  
 }  
 }  
  
 public boolean isValid() {  
 return valid;  
 }  
  
 public boolean wasEdited() {  
 return edited;  
 }  
  
 public String getText() {  
 return inputControl.getText();  
 }  
}

## Kod UserData

package lab06.validationapp;  
  
public class UserData {  
 @ValidationPattern(  
 regex = "^[A-Za-z]+$",  
 errorMessage = "Imię może zawierać tylko litery (A-Z, a-z)"  
 )  
 private String username;  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
}

## Kod FieldValidator

package lab06.validationapp;  
  
public interface FieldValidator {  
 void check(String value); // Metoda walidująca wartość  
 boolean isCorrect(); // Sprawdza poprawność wartości  
 String getErrorMessage(); // Zwraca komunikat błędu  
}

## Kod do ValidationPattern

package lab06.validationapp;  
  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Retention(RetentionPolicy.RUNTIME) // Dostępność w czasie działania programu  
@Target(ElementType.FIELD) // Stosowana tylko do pól klas  
public @interface ValidationPattern {  
 String regex(); // Wyrażenie regularne dla walidacji  
 String errorMessage(); // Komunikat błędu  
}

## Wygląd aplikacji w działaniu

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Zawartość wygenerowana przez AI może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, multimedia

Zawartość wygenerowana przez AI może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

# Wnioski

To zadanie pozwoliło mi nauczyć się jak implementować walidację danych użytkownika w aplikacjach graficznych z wykorzystaniem JavaFX. Również dowiedziałem się jak oddzielić logikę walidacji od interfejsu użytkownika, stosując adnotacje i dedykowanie klasy walidatorów co sprzyja czytelności i łatwej rozbudowie kodu.