|  |  |  |  |
| --- | --- | --- | --- |
| ***SPRAWOZDANIE NR*** | | | |
| Nazwa ćwiczenia | Rodziny\_wyjątków | | Obraz zawierający tekst  Opis wygenerowany automatycznie |
| Przedmiot | Programowanie obiektowe | |
| Student  grupa | Paweł Jońca gr 7 | |
| Data ćwiczeń | 10.12.2024 | 15.12.2024r | Data oddania sprawozdania |

Klasa Order

* Status – status zamówienia (fulfilled lub unfulfilled)
* timeStatus – status czasu realizacji (on time lub exceeded)

konstruktor przyjmuje dwa parametry (status I czas realizacji )

Metoda checkOrder():

* Sprawdza warunki:
  1. Jeśli status nie jest równy fulfilled, rzucany jest wyjątek UnfulfilledOrderException.
  2. Jeśli timeStatus jest równy exceeded, rzucany jest wyjątek OrderTimeExceededException

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

OrderApp definiuje interfejs graficzny użytkownika i logikę aplikacji

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający elektronika, tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Obraz zawierający tekst, elektronika, zrzut ekranu, wyświetlacz

Opis wygenerowany automatycznie

Dodałem kilka bloków catch, każdy dedykowany jednemu rodzajowi wyjątku

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, elektronika, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Wnioski :

W trakcie realizacji zadania nauczyłem się tworzyć hierarchię wyjątków w Javie. Dzięki zastosowaniu własnych klas wyjątków mogłem skutecznie obsłużyć różne scenariusze błędów, zapewniając użytkownikowi czytelne komunikaty w formie okien dialogowych.

package com.example.rodziny\_wyjatkow;  
  
public class Order {  
 private String status; // Status zamówienia (np. "fulfilled", "unfulfilled")  
 private String timeStatus; // Status czasu realizacji zamówienia (np. "on time","exceeded")  
  
 public Order(String status, String timeStatus) {  
 this.status = status;  
 this.timeStatus = timeStatus;  
 }  
  
 public void checkOrder() throws OrderProcessingException {  
 if (!"fulfilled".equalsIgnoreCase(status)) {  
 throw new UnfulfilledOrderException("Zamówienie nie zostało zrealizowane!");  
 }  
 if ("exceeded".equalsIgnoreCase(timeStatus)) {  
 throw new OrderTimeExceededException("Przekroczono czas realizacji zamówienia!");  
 }  
 }  
}

package com.example.rodziny\_wyjatkow;  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
public class OrderApp extends Application {  
 @Override  
 public void start(Stage primaryStage) {  
 primaryStage.setTitle("Sprawdzanie statusu zamówienia");  
  
 // Tworzenie elementów interfejsu  
 //status fulfilled unfulfilled  
 Label label = new Label("Wprowadź status zamówienia('fulfilled') i status czasu (np on time, exceeded):");  
 TextField statusField = new TextField();  
 TextField timeField = new TextField();  
 Button checkButton = new Button("Sprawdź zamówienie");  
 Label resultLabel = new Label();  
  
 // Obsługa przycisku  
 checkButton.setOnAction(e -> {  
 String status = statusField.getText().trim();  
 String timeStatus = timeField.getText().trim();  
 Order order = new Order(status, timeStatus);  
  
 try {  
 order.checkOrder(); //wywołanie metody  
 resultLabel.setText("Zamówienie zostało pomyślnie zrealizowane.");  
 } catch (OrderProcessingException ex) {  
 // Pokazanie alertu z błędem  
 Alert alert = new Alert(Alert.AlertType.ERROR);  
 alert.setTitle("Błąd zamówienia");  
 alert.setHeaderText("Wystąpił wyjątek");  
 alert.setContentText(ex.getMessage());  
 alert.showAndWait();  
 resultLabel.setText("Błąd: " + ex.getMessage());  
 }  
 });  
 // Układ interfejsu  
 VBox layout = new VBox(10);  
 layout.setPadding(new Insets(20));  
 layout.getChildren().addAll(label, statusField, timeField, checkButton, resultLabel);  
  
 // Scena i pokazanie okna  
 Scene scene = new Scene(layout, 500, 350);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
 public static void main(String[] args) {  
 launch(args);  
 }  
}

package com.example.rodziny\_wyjatkow;  
  
public class OrderProcessingException extends Exception {  
 public OrderProcessingException(String message) {  
 super(message);  
 }  
}

package com.example.rodziny\_wyjatkow;  
  
public class OrderTimeExceededException extends OrderProcessingException{  
 public OrderTimeExceededException(String message){  
 super(message);  
 }  
}

package com.example.rodziny\_wyjatkow;  
  
public class UnfulfilledOrderException extends OrderProcessingException {  
 public UnfulfilledOrderException(String message) {  
 super(message);  
 }  
}