


SPRAWOZDANIE NR			
Nazwa ćwiczenia	Rodziny_wyjatków		 POLITECHNIKA BYDGOSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki
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, rzuca wyjątek UnfulfilledOrderException.
 2. Jeśli timeStatus jest równy exceeded, rzuca wyjątek OrderTimeExceededException

```

1 package com.example.rodziny_wyjatkow;
2
3 public class Order {
4     private String status; // Status zamówienia (np. "fulfilled", "unfulfilled") 2 usages
5     private String timeStatus; // Status czasu realizacji zamówienia (np. "on time", "exceeded") 2 usages
6
7     public Order(String status, String timeStatus) { 1 usage
8         this.status = status;
9         this.timeStatus = timeStatus;
10    }
11
12    public void checkOrder() throws OrderProcessingException { 1 usage
13        if (!"fulfilled".equalsIgnoreCase(status)) {
14            throw new UnfulfilledOrderException("Zamówienie nie zostało zrealizowane!");
15        }
16        if ("exceeded".equalsIgnoreCase(timeStatus)) {
17            throw new OrderTimeExceededException("Przekroczono czas realizacji zamówienia!");
18        }
19    }
20 }

```

OrderApp definiuje interfejs graficzny użytkownika i logikę aplikacji

```
package com.example.rodziny_wyjatkow;
import ...

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(s: "Wprowadź status zamówienia('fulfilled') i status czasu (np on time, exceeded):");
        TextField statusField = new TextField();
        TextField timeField = new TextField();
        Button checkButton = new Button(s: "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(v: 10);
layout.setPadding(new Insets(v: 20));
layout.getChildren().addAll(label, statusField, timeField, checkButton, resultLabel);

// Scena i pokazanie okna
Scene scene = new Scene(layout, v: 500, v1: 350);
primaryStage.setScene(scene);
primaryStage.show();
}

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

```

```

1 package com.example.rodziny_wyjatkow;
2
3 @ public class OrderProcessingException extends Exception { 2 inheritors
4     public OrderProcessingException(String message) { 2 usages
5         super(message);
6     }
7 }
8

```

```

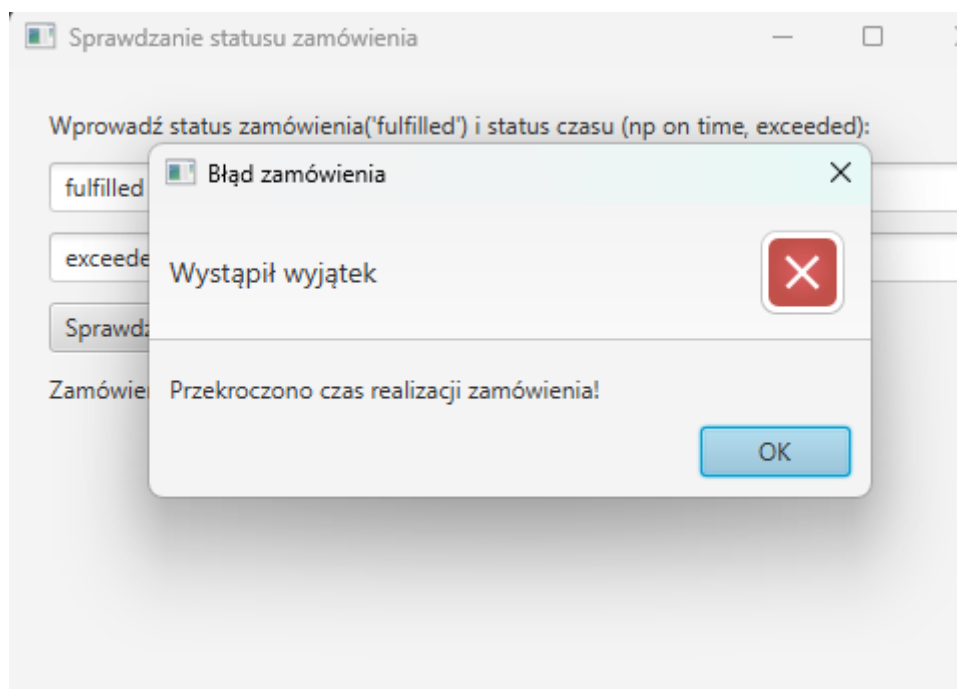
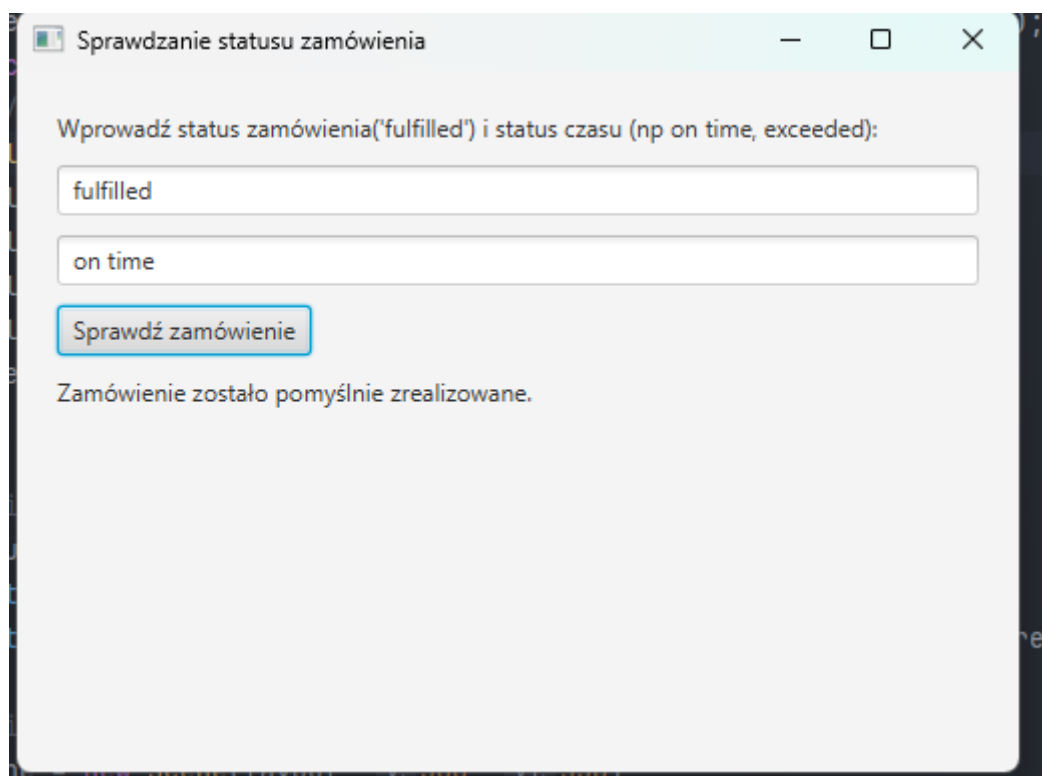
1 package com.example.rodziny_wyjatkow;
2
3 public class OrderTimeExceededException extends OrderProcessingException{
4     public OrderTimeExceededException(String message){ 1 usage
5         super(message);
6     }
7 }
8

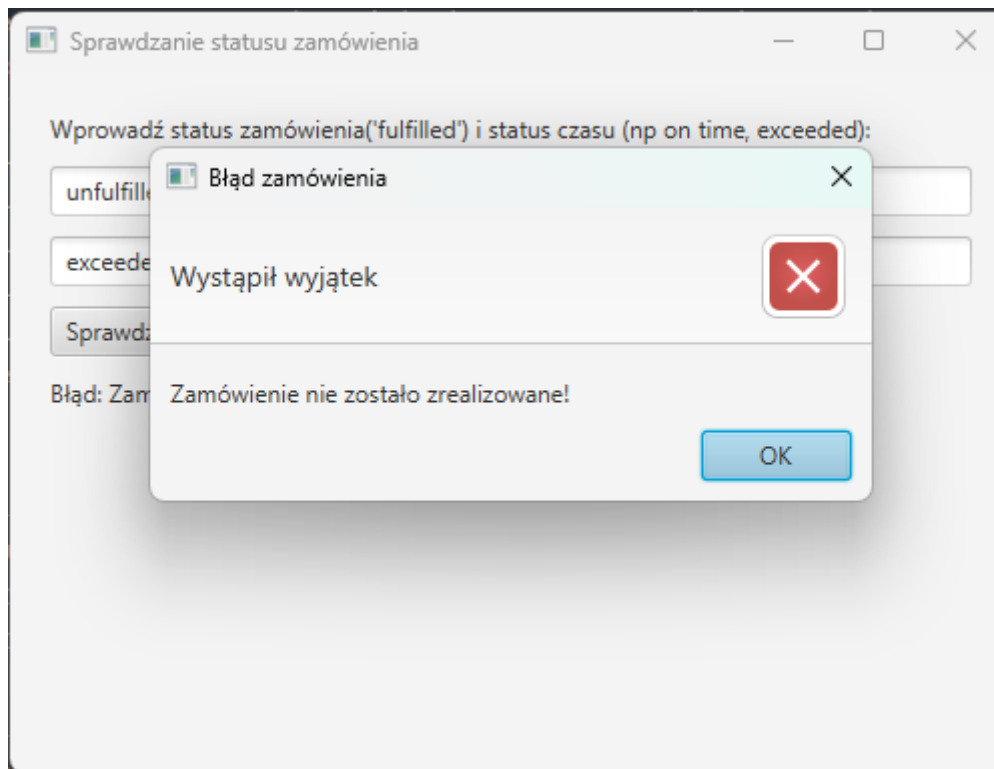
```

```

1 package com.example.rodziny_wyjatkow;
2
3 public class UnfulfilledOrderException extends OrderProcessingException {
4     public UnfulfilledOrderException(String message) { 1 usage
5         super(message);
6     }
7 }
8

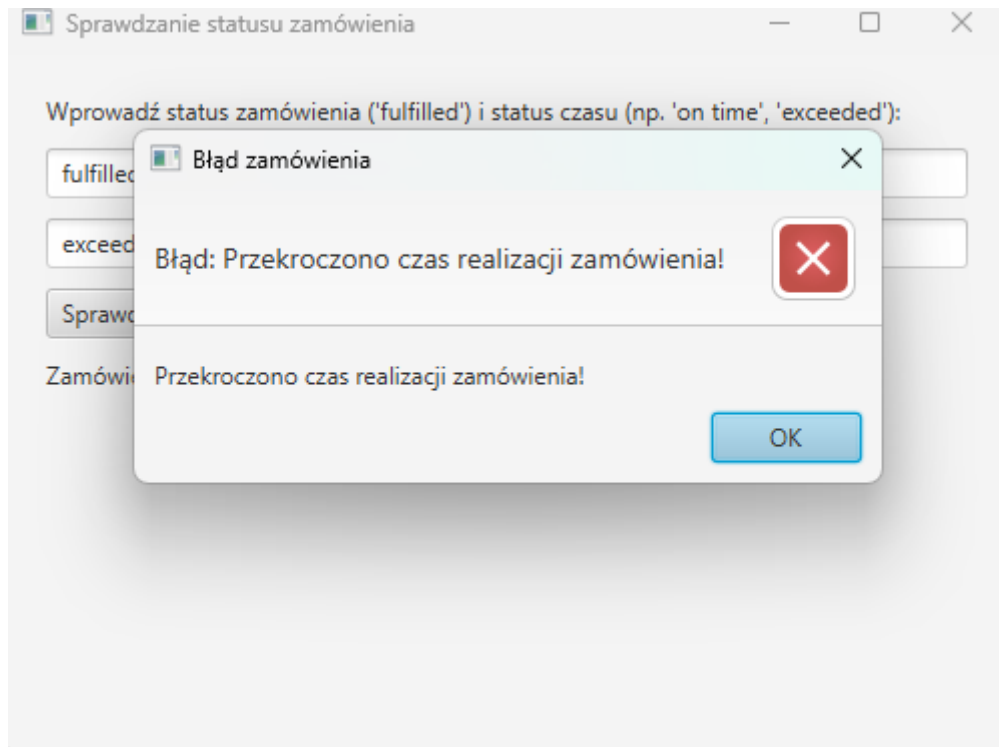
```





Dodałem kilka bloków catch, każdy dedykowany jednemu rodzajowi wyjątku

```
try {
    order.checkOrder(); // Wywołanie metody
    resultLabel.setText("Zamówienie zostało pomyślnie zrealizowane.");
} catch (UnfulfilledOrderException ex) {
    showAlert( header: "Błąd: Zamówienie nie zostało zrealizowane!", ex.getMessage());
    resultLabel.setText("Błąd: " + ex.getMessage());
} catch (OrderTimeExceededException ex) {
    showAlert( header: "Błąd: Przekroczono czas realizacji zamówienia!", ex.getMessage());
    resultLabel.setText("Błąd: " + ex.getMessage());
} catch (OrderProcessingException ex) {
    showAlert( header: "Ogólny błąd przetwarzania zamówienia!", ex.getMessage());
    resultLabel.setText("Błąd: " + ex.getMessage());
}
});
```



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);
    }
}

```