

JAVA : lab03 – UI

Zadania

Głównym zadaniem jest stworzenie UI dla kodu z lab2. Inne funkcje będą dla zwykłego użytkownika, a inne dla admina. Proszę o rozróżnienie tych dwóch opcji! Admin może np. wprowadzać zmiany w danych (dodawanie, usuwanie, edycja), natomiast użytkownik może te dane przeglądać, adoptować zwierzęta, prosić o kontakt ze schroniskiem itp.

1. **Zaprojektuj interfejs graficzny** do obsługi schronisk – 3 moduły: do logowania się, dla admina i dla klienta.
2. **Zaimplementuj obsługę interfejsu graficznego** z uwzględnieniem następujących reguł:
 - Interfejs składa się z dwóch list: schronisk (_list of shelters) oraz zwierząt (_list of animals in selected shelter).
 - Stwórz generyczny komponent na bazie `[JTable]`
(<https://docs.oracle.com/javase/7/docs/api/javax/swing/JTable.html>)
 - Oraz `[AbstractTableModel]` który można wykorzystać zarówno dla schronisk jak i dla zwierząt.
(<https://docs.oracle.com/javase/7/docs/api/javax/swing/table/AbstractTableModel.html>), Więcej informacji nt. AbstractTableModel na [StackOverflow] (<https://stackoverflow.com/questions/9845800/abstracttablemodel-tutorial>).
 - Po **wybraniu** schroniska (zaznaczeniu go na liście schronisk) wyświetla się dostępna w nim lista zwierząt.
 - Zaznaczone schronisko lub zwierzę ma zostać usunięte/zmodyfikowane z listy po naciśnięciu odpowiedniego guzika.
 - Po naciśnięciu przycisków dodaj schronisko lub dodaj zwierzę, wprowadź odpowiednie parametry za pomocą komponentu [JOptionPane] (<https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html>).
Alternatywnie, przycisk może dodać wiersz do tabeli a wprowadzenie wartości odbywa się poprzez edycję w tabeli.
 - Po naciśnięciu przycisku sort shelters ma zostać wykonane sortowanie obiektów (np. względem maksymalnego obciążenia). ****Uwaga****: przekazuj referencje w odpowiedni sposób, wówczas taka operacja to wywołanie jednej metody.
 - Pole filter textbox służy do filtrowania list. Po naciśnięciu klawisza enter, w tabeli mają zostać wyświetlone zwierzęta/schroniska zgodne z wprowadzonym tekstem.
 - Pole state combobox ma zostawić w tabeli produkty, zgodne z wybraną wartością. W tym celu skorzystaj z komponentu [JComboBox](<https://docs.oracle.com/javase/7/docs/api/javax/swing/JComboBox.html>).
3. **Staraj się nie mieszać modelu systemu z warstwą użytkownika.** Ogranicz interakcję pomiędzy nimi do minimum.
4. Dla każdej metody stwórz **testy jednostkowe** przy użyciu biblioteki JUnit (<https://github.com/junit-team/junit5/wiki>). Testy zamknij w odpowiednich TestSuite. Wykorzystaj mechanizmy wbudowane w IDE do uruchomienia testów i prezentacji wyników
5. Zaimplementuj **własne klasy wyjątków** i wykorzystaj je w najbardziej narażonych na błędy miejscach. Nie wykorzystuj wyjątków typu Runtime!

Przykładowe pytania teoretyczne

UI

1. Swing vs AWT
2. Layouty w Swing – po co je używać, jakie problemy rozwiązują? Jakie znasz layouty i do czego służą.
3. Obsługa zdarzeń Komponentów w Swing – ActionListener
4. SwingUtilities.invokeLater
5. Po co używamy Modeli w Swingu? Na przykładzie ListModel.
6. Idea architektury MVC. Jak wygląda MVC w Swing?
7. Czym jest MVC? Rola i sposób implementacji w JavaFX.
8. Różnice między JavaFX i Swing.
9. Zmiany w JavaFX od JDK11.
10. Jak możesz uniezależnić implementację UI od logiki aplikacji? Podaj przykład.

Wyjątki

11. Własne typy wyjątków, hierarchia dziedziczenia wyjątków
12. Do czego służy klasa Object i jakie ma zastosowanie ?

Testy

13. Jakie dwa rodzaje testów wyróżniamy i czym się one cechują (testy automatyczne i manualne)
14. Czym są testy jednostkowe i do czego służą?
15. Czym jest TestSuite w JUnit? [Przykład](https://howtodoinjava.com/junit5/junit5-test-suites-examples/)
16. Czym jest Mock Objects? Gdzie się go stosuje?
17. Na czym polega Test fixture w JUnit. Jakich adnotacji możemy użyć.
18. Na czym polega Test Driven Development (TDD)?
19. Czym jest pokrycie kodu (ang. Code coverage)?
20. Jak zbudowane są testy JUnit? (adnotacje i nazwy metod)

Uwagi i wskazówki do UI

1. Aby oddzielić interfejs użytkownika od modelu systemu, możesz wykorzystać wzorec projektowy Fasada .

2. Gdy operacje są nieprawidłowe lub niezgodne z modelem systemu, zaimplementuj odpowiednie wyjątki, które wyrzucane są na warstwie modelu. **Nie wykorzystuj wyjątków typu Runtime!**
Więcej info na ten temat
[tutaj](https://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html).

3. Wyjątki można obsługiwać przy pomocy komponentu JOptionPane. Przeglądnij dokumentację lub odpowiednie źródła.

4. Aby komponenty dostosowywały się do rozmiaru okna i nie rozeżdżały się , wykorzystaj [Layouty](https://examples.javacodegeeks.com/desktop-java/swing/java-swing-layout-example/).

Godnymi uwagi na pewno będą [GridLayout](https://examples.javacodegeeks.com/desktop-java/awt/gridlayout/java-gridlayout-example/)

oraz [GridBagLayout](https://examples.javacodegeeks.com/desktop-java/swing/java-swing-gridbaglayout-example/).

5. Wykorzystaj IDE do projektowania interfejsu użytkownika. Pozwala układać komponenty przy pomocy myszki oraz udostępnia interfejs do zarządzania Layoutami! Jest to bardzo przydatne, szczególnie dla GridBagLayout'u.
6. Stwórz klasę DataGenerator, w której uzupełnione zostaną dane, niezbędne do weryfikacji poprawności oprogramowania. Możesz wykorzystać w tym miejscu wzorzec projektowy Singleton.

Wskazówki do testów:

1. [Testowanie kodu w IntelliJ Idea](<https://www.jetbrains.com/help/idea/configuring-testing-libraries.html>)
2. [Analiza pokrycia testami](<https://www.jetbrains.com/help/idea/code-coverage.html>)
3. [Pisanie testów jednostkowych](<https://junit.org/junit5/docs/current/user-guide/#writing-tests>)
4. [Wykonywanie testów jednostkowych](<https://junit.org/junit5/docs/current/user-guide/#running-tests>)

Materiały SWING

1. [Layouty w Swingu 1](<https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>)
2. [Layouty w Swingu 2](<https://examples.javacodegeeks.com/desktop-java/swing/java-swing-layout-example/>)
3. [Lista eventów w Swingu](<https://docs.oracle.com/javase/7/docs/api/javax/swing/event/package-summary.html>)
4. [Przykład obsługi eventów w Swingu 1](https://www.tutorialspoint.com/swing/swing_event_handling.htm)
5. [Przykład obsługi eventów w Swingu 2](<https://stackoverflow.com/questions/19122514/handling-events-in-java-swing>)
6. Jak można stworzyć generyczny komponent na bazie AbstractTableModel:
7. [Jak korzystać z AbstractTableModel?](<https://stackoverflow.com/questions/9845800/abstracttablemodel-tutorial>)
8. [Wykorzystanie refleksji do pobrania properties](<https://stackoverflow.com/questions/8524011/java-reflection-how-can-i-get-the-all-getter-methods-of-a-java-class-and-invoke>)

Materiały JavaFX

6. [JavaFX in IntelliJ](<https://www.jetbrains.com/help/idea/javafx.html>)
7. JavaFX Scene Builder:
 - [Scene Builder in IntelliJ](<https://www.jetbrains.com/help/idea/opening-fxml-files-in-javafx-scene-builder.html>)
 - [FXML and SceneBuilder](<https://www.vojtechruzicka.com/javafx-fxml-scene-builder/>)
8. [JavaFX Tutorial]
 - a. (<http://tutorials.jenkov.com/javafx/index.html>)
9. [Tutorialspoint JavaFX]

- b. (<https://www.tutorialspoint.com/javafx/index.htm>)
- 10. [Zetcode blog]
 - c. (<https://www.tutorialspoint.com/javafx/index.htm>)
- 11. Events in JavaFX:
 - https://www.tutorialspoint.com/javafx/javafx_event_handling.htm
 - http://www.java2s.com/Tutorials/Java/JavaFX/1100_JavaFX_Events.htm
- 12. [JavaFX UI controls]
 - d. (https://www.tutorialspoint.com/javafx/javafx_ui_controls.htm)
- 13. Controllers:
 - <https://www.educba.com/javafx-controller/>
 - [Bind controller with app]
 - a. (<https://stackoverflow.com/questions/33881046/how-to-connect-fx-controller-with-main-app>)
 - [FXML controller example]
 - a. (<https://examples.javacodegeeks.com/desktop-java/javafx/fxml/javafx-fxml-controller-example/>)