Java: lab07 - IO, serializacja, refleksje

Wykorzystanie wbudowanych w JDK mechanizmów do serializacji, obsługi wejścia/wyjścia oraz refleksji do stworzenia namiastki bazy danych

Materiały

1. Serializacja
   - [Serialization - GeeksforGeeks] (https://www.geeksforgeeks.org/serialization-in-java/)
2. [Java Serialization - Baeldung] (https://www.baeldung.com/java-serialization)
3. Refleksje
   - [Reflections - GeeksforGeeks] (https://www.geeksforgeeks.org/reflection-in-java/)
   - [Retrieve Fields from a Java Class] (https://www.baeldung.com/java-reflection-class-fields)
4. Adnotacje:
   - [Annotations in Java - GeeksforGeeks] (https://www.geeksforgeeks.org/annotations-in-java/)
   - [Tutorials Jenkov] (http://tutorials.jenkov.com/java/annotations.html)
   - [Retention policies] (https://stackoverflow.com/questions/3107970/how-do-different-retention-policies-affect-my-annotations)

Zadania

Zadania w niniejszym laboratorium wymagają zaimplementowania API do jednego z dwóch poprzednich laboratoriów - Swing lub JavaFX. Dużo prościej będzie wykonać tą instrukcję posiadając również UI.

1. Wykorzystując serializację, napisz funkcjonalność pozwalającą na zapisanie i odczytanie stanu salonów i zakupów (tutaj może być „koszyk” bądź lista ulubionych). W przypadku obiektów typu `Item` nie serializuj pola typu `ItemCondition`. Przy imporcie oznacz produkty jako nowe.
2. Wykorzystując wbudowany pakiet wejścia/wyjścia ([`java.io`](https://docs.oracle.com/javase/7/docs/api/java/io/package-summary.html)), napisz funkcjonalność eksportującą oraz importującą stan wybranego salonu do/z pliku CSV. Wykonaj to samo w przypadku zakupów.
3. Skorzystaj z refleksji i adnotacji do dodania nazewnictwa kolumn i ich kolejności w tabeli (`JTable`, `TableView`, plik CSV). Stworzone adnotacje mają znajdować się w modelu przy odpowiednich polach. Wykorzystaj je do wskazania pól do eksportu, np. nie eksportuj `ItemCondition` i ustaw przy imporcie stan produktu jako nowy.

Przykładowo: jeżeli eksportowany plik ma zawierać kolumny, eksportuj tylko te posiadające adnotację z nazwą kolumny, pozostałe pomiń. W przeciwnym przypadku - eksportuj wszystko, bez nazw kolumn.

4. Zmodyfikuj dotychczasowy UI:
   - Dodaj kontrolki, obsługujące nowe funkcjonalności.
   - Potraktuj CSV/serializację jako data provider do swojej aplikacji. Przy uruchomieniu aplikacja ma próbować wczytać automatycznie dane. W przypadku, gdy jest to niemożliwe, wystąpi wyjątek lub plik nie istnieje, wyrzuć wyjątek i obsłuż w odpowiedni sposób na poziomie UI, np. wypisz komunikat, że salony są puste - uzupełnij dane.
   - Przy zamykaniu aplikacji ma pojawić się komunikat, czy zapisać wprowadzone dane do pliku (serializowanego lub CSV, obojętne).
   - \[_Opcjonalnie_\] Polecenie 4 wykonaj w obydwu aplikacjach. Dobrze napisane API powinno być bezpośrednio przenaszalne pomiędzy nimi. Różnić powinna się wyłącznie eksportowana klasa.

Uwagi i wskazówki

1. Funkcjonalność serializacji i zapisu do pliku jest _niezależna_ od UI, gdyż dotyczy wyłącznie obiektów modelu i działania logiki biznesowej. Wasza implementacja powinna być więc przenaszalna.

2. Proszę zwrócić uwagę na standard CSV (ang. _comma separated values_). Więcej informacji na [Wiki](https://en.wikipedia.org/wiki/Comma-separated_values).

Przykładowe pytania teoretyczne

1. Rola adnotacji w językach obiektowych oraz działanie _Retention policies_ na przykładzie Javy.

2. W jaki sposób wykonywana jest serializacja? Jak ją personalizować? Gdzie można ją wykorzystać?

3. Refleksje a performance. Omów na przykładzie.