

## Java: lab06 - JavaFX

implementacja aplikacji klienckiej do obsługi salonu

### Materiały

1. [JavaFX in IntelliJ]  
(<https://www.jetbrains.com/help/idea/javafx.html>)
2. 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/>)
3. [JavaFX Tutorial]
  - a. (<http://tutorials.jenkov.com/javafx/index.html>)
4. [Tutorialspoint JavaFX]
  - a. (<https://www.tutorialspoint.com/javafx/index.htm>)
5. [Zetcode blog]
  - a. (<https://www.tutorialspoint.com/javafx/index.htm>)
6. Events in JavaFX:
  - [https://www.tutorialspoint.com/javafx/javafx\\_event\\_handling.htm](https://www.tutorialspoint.com/javafx/javafx_event_handling.htm)
  - [http://www.java2s.com/Tutorials/Java/JavaFX/1100\\_JavaFX\\_Events.htm](http://www.java2s.com/Tutorials/Java/JavaFX/1100_JavaFX_Events.htm)
7. [JavaFX UI controls]
  - a. ([https://www.tutorialspoint.com/javafx/javafx\\_ui\\_controls.htm](https://www.tutorialspoint.com/javafx/javafx_ui_controls.htm))
8. 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/>)

### Zadania

1. Do niniejszego projektu wykorzystaj API salonów samochodowych z poprzednich zajęć. Rozbuduj je o dodatkowe pola, metody i klasy. Zainspiruj się stronami internetowymi tj. Otomoto, aaaauto itd.
2. Zaimplementuj funkcjonalności aplikacji klienckiej do obsługi zakupów w salonie:
  1. Użytkownik może wybierać salon (bądź miasto\*) przy użyciu listy lub pola rozwijalnego (ComboBox). Lista powinna zawierać opcję \_dowolny\_, która pozwoli na wyszukiwanie produktów we wszystkich salonach.

2. Domyślnym widokiem powinna być lista wszystkich pojazdów z wszystkich salonów. Do tego celu można wykorzystać dowolny komponent mający strukturę tabelaryczną. Powinna ona być sortowalna względem przynajmniej: nazwy (alfabetycznie), ceny, roku produkcji. Każdy rekord powinien zawierać co najmniej: nazwę produktu, cenę, roku produkcji oraz w jakim salonie (mieście) się on znajduje.
3. Istnieje możliwość wyszukiwania produktów po ich nazwie. Wystarczy wyszukiwanie wprost.
4. Lista dostępnych produktów, spełniających kryteria wyszukiwania ma być wyświetlona w formie tabeli z punktu 2.
5. Po najejchaniu kursorem na samochód ma wyświetlać się `_Tooltip_` z informacjami szczegółowymi, np. opis salonu, w którym się produkt znajduje, przebieg, silnik.
6. Aplikacja powinna umożliwiać rezerwację/zakup samochodu, ewentualnie kontakt ze sprzedającym. Należy dokonać weryfikacji poprawności danych, np. samochód nie jest zarezerwowany/kupiony.

3. Funkcjonalność zakupu/rezerwacji samochodu z salonu może być zaimplementowana na dwa sposoby:

\* Rezerwacja produktu. Wówczas samochód nadal będzie dostępny w salonie, natomiast klient będzie widział zarezerwowane pojazdy.

\* Kupno samochodu skutkujące usunięciem pojazdu z salonu. Nie ma możliwości przeglądnięcia historii zakupów.

### **Uwagi i wskazówki**

1. Wybór sposobu implementacji zakupów nie wpływa na ocenę, jednak może przyczynić się do uproszczenia pracy z kolejnymi instrukcjami, gdyż temat salonów będzie kontynuowany.
2. Na obecnym etapie nie jest istotne, kim jest użytkownik, więc funkcjonalność użytkowników jest tutaj zbędna.

### **Przykładowe pytania teoretyczne**

1. Podobnie jak w Swingu - pytania odnośnie budowy UI.
2. Czym jest MVC? Rola i sposób implementacji w JavaFX.
3. Różnice między JavaFX i Swing.
4. Zmiany w JavaFX od JDK11.