JDBC - tworzenie połączenia do bazy danych

java.sql.Connection

- ➔ **połączenie -** między aplikacją Java i bazą danych
- ➔ **interfejs** - nie można go wprost utworzyć
- ➔ **close()** - zawsze zamykamy pod koniec pracy
- ➔ **try-with-resources** - zamiast close()

javax.sql.DataSource

- ➔ **fabryka** - do tworzenia obiektów typu Connection
- ➔ **stwórz raz** - dla każdej bazy danych do użycia w aplikacji
- ➔ **wykorzystuj wielokrotnie** - pobieraj obiekty Connection kiedy chcesz
- ➔ **serverName, databaseName, user, password** - parametry bazy danych
- ➔ ***.properties** - parametry w zewnętrznym pliku (np.: **database.properties**)

java.sql.Driver

- ➔ **sterownik** - do łączenia aplikacji z bazą danych
- ➔ **interfejs** - nie można go wprost utworzyć
- ➔ **core** - wykorzystywany przez DataSource i DriverManager

java.sql.DriverManager

- ➔ **manager** - zarządzanie sterownikami (rejestracja, szukanie)
- ➔ **fabryka** - do tworzenia obiektów typu Connection
- ➔ **legacy code** - rzadko wykorzystywane, można spotkać w starym kodzie

Zadanie nr 1:

1. Otwórz klasę ***ConnectionViaDataSource*** (w module **jdbc-starter**) i uzupełnij konfigurację bazy danych tak żeby połączyć się ze swoją lokalną bazą MySQL (może to być baza stworzona na wcześniejszych zajęciach). Należy ustawić odpowiednio pola: **DB_SERVER_NAME**, **DB_NAME**, **DB_USER**, **DB_PASSWORD** - są to parametry konfiguracyjne bazy danych. Uruchom metodę main i sprawdź czy połączenie do bazy danych powiodło się.

2. Użyj konstrukcji **try-with-resources** do automatycznego zamknięcia obiektu ***Connection***:
    a. Zamień 3 linie kodu w której znajduje się rozpoczęcie bloku try i utworzenie połączenia czyli:
    ```
    Connection connection = null;
    try {
    connection = dataSource.getConnection();
    ```
    na kod umieszczony niżej:
    ```
    try(Connection connection = dataSource.getConnection()){
    ```
    b. Usuń z kodu cały blok `finally{..}`
    c. Uruchom i sprawdź czy program działa
    d. Odtąd przy tworzeniu połączeń będziemy się posługiwać **try-with-resources**

3. Wzorując się na klasie ***ConnectionViaDataSource*** uzupełnij kod w klasie ***ConnectionFactory.*** Klasa ta ma stać się naszą fabryką do tworzenia połączeń do wybranej przez nas bazy danych. Wewnątrz klasy ***ConnectionFactory*** należy stworzyć obiekt DataSource (pamiętaj: tworzymy raz, wykorzystujemy wielokrotnie) i następnie użyć go w metodzie `getConnection()` do stworzenia nowego połączenia z bazą danych (czyli obiektu Connection).

4. Przenieś parametry bazy danych do zewnętrznego pliku:
    a. Utwórz plik **database.properties** w katalogu **resources**
    b. Skopiuj zawartość pliku **remote-database.properties** (z katalogu **resources**) do pliku który przed chwilą utworzyłeś
    c. Zmień wartości (po prawej stronie znaku '=') na parametry twojej lokalnej bazy
    d. Utwórz konstruktory:
    ```
    ConnectionFactory(String filename)
    ConnectionFactory()
    ```
    e. Drugi konstruktor ma wywołać pierwszy konstruktor z argumentem filename="database.properties**"**
    f. Pobierz parametry bazy danych z pliku za pomocą metody `getDataBaseProperties(String fileName)`
    g. Użyj parametrów bazy danych pobranych z pliku przy pomocy metody: `Properties.getProperty(String key)` np.:
    ```
    properties.getProperty("pl.sda.jdbc.db.server")
    ```
    h. W metodzie main stwórz obiekt ***ConnectionFactory*** pobierz połączenie i sprawdź czy działa - możesz użyć kodu z ***ConnectionViaDataSource***
    i. Tym samym sposobem spróbuj połączyć się z bazą danych, której parametry zapisane są w pliku **remote-database.properties**

5. **(dla chętnych)** Otwórz klasę ***ConnectionViaDriverManager*** i uzupełnij konfigurację bazy danych (parametry: **DB_URL, DB_USER, DB_PASSWORD**) tak żeby połączyć się ze swoją lokalną bazą MySQL. Uruchom metodę main i sprawdź czy połączenie do bazy danych się powiodło.