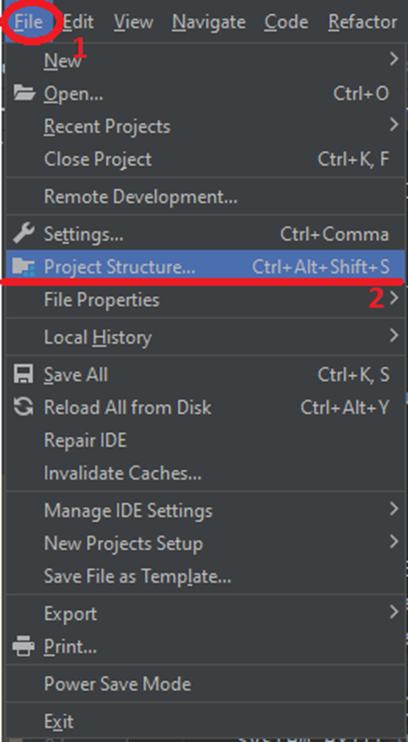
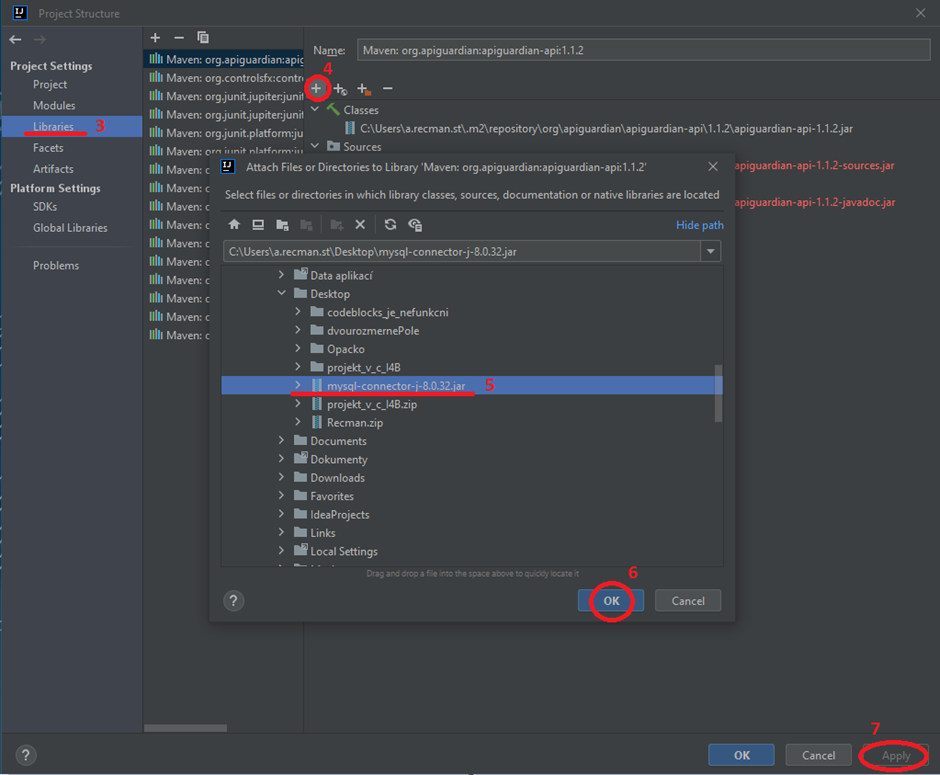
23. Java a databáze – DDL, DML

Práce s SQL databází v jazyce Java obvykle probíhá pomocí JDBC (Java Database Connectivity). JDBC poskytuje standardní rozhraní pro připojení k databázím, odesílání dotazů a zpracování výsledků.

**Vytvoření spojení**

**1. Přidání modulu mysql connector**

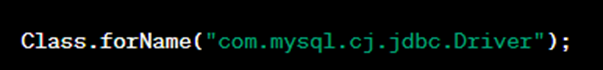




**2. Načtení ovladače (Driver Loading)**

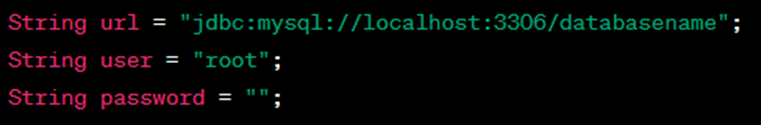
Nejprve musíte načíst ovladač pro konkrétní databázi, se kterou chcete pracovat. Každá databáze má svůj vlastní JDBC ovladač. (pozn. autora: od verze JDBC 4.0 není vždy nutné explicitně načítat ovladač. JDBC 4.0 a novější mají schopnost automaticky načíst ovladač, pokud je dostupný na classpath.)

Př. (pro MySQL)



**3. Vytvoření připojení (Creating a Connection)**

Poté vytvořte připojení k databázi pomocí URL, uživatelského jména a hesla.





**Statement vs. PreparedStatement**

Rozhraní **PreparedStatement** rozšiřuje rozhraní **Statement**

**Statement:**

* *Výhody:*
  + Jednoduché použití pro jednorázové dotazy.
  + Snadné vytváření pro jednoduché SQL dotazy.
* *Nevýhody:*
  + Méně efektivní pro opakované vykonávání dotazů.
  + Bezpečnostní riziko při manipulaci s parametry (SQL injection).

**PreparedStatement:**

* *Výhody:*
  + Předkompilovaný dotaz, což zvyšuje výkon.
  + Umožňuje použití parametrů
  + Bezpečné vložení parametrů pomocí placeholderů, snižuje riziko SQL injection.
* *Nevýhody:*
  + Trochu složitější syntaxe pro vytváření dotazů.
  + Může být příliš zdrcující pro jednorázové a jednoduché dotazy.

**Přehled metod pro provádění SELECT dotazů:**

**Rozhraní Statement**

* **boolean execute (String sql,…)**

Provede jakýkoli SQL dotaz (DDL, DML, DQL) a vrací true/false**. true** **vrací** **pro výsledky dotazu, které vrací ResultSet.** Ten poté lze získat pomocí metody **getResultSet().**

* **ResultSet executeQuery (String sql)**

Určena pro provádění dotazů, které vrací jeden ResultSet. Převážně se používá pro SELECT. Jednodušší práce než s metodou **execute()**, protože nám přímo vrátí výsledek.

* int[ ] executeBatch() - **Nepoužívá se**, i když teoreticky se použít dá

Je vhodnější pro hromadné operace, jako jsou aktualizace nebo vkládání, a není běžně používáno pro provádění SELECT dotazů.

**Rozšiřující rozhraní PreparedStatement**

**PreparedStatemtent** přidává hlavně metody pro práci s parametry.

Metody pro nastavení parametrů, slouží k nastavení hodnoty parametru na určitém indexu.

* **setInt(int parameterIndex, int x)**
* **setString(int parameterIndex, String x)**
* **setDouble(int parameterIndex, double x)**

….

Metoda pro vyprázdnění parametrů

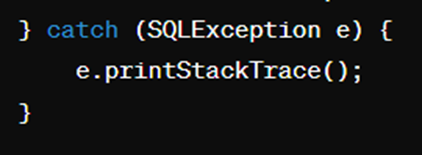
* **clearParameters()**

**Př. Rozhraní Statement a metoda execute()**

1. try-catch

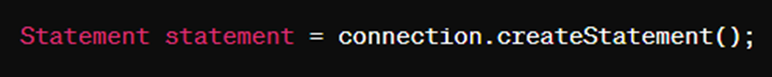
Celý kód musí být obalen v try-catch k zachycení případných výjimek, které mohou vzniknout při vykonání SQL dotazu





2. Vytvoření objektu statement

Objekt statement umožňuje vykonávat SQL dotazy na připojené databázi.



3. Vytvoření SELECT dotazu

Vytvoření jednoduchého dotazu SELECT na tabulku moje\_tabulka.



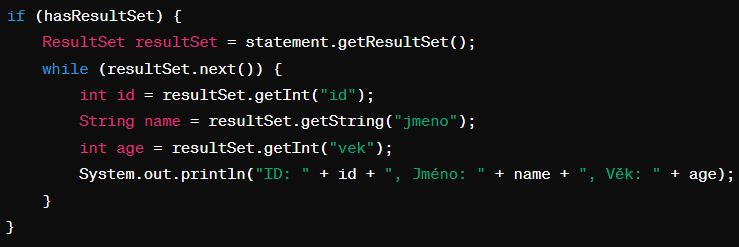
4. Vykonání dotazu

Pro tento příklad používáme obecnou metodu **execute()**, použití metody **executeQuery()** by bylo jednodušší, protože ta vrací rovnou **ResultSet**.



5. Zpracování výsledků

Jelikož používáme metodu **execute()** musíme první ověřit jestli dotaz vrátil ResultSet, pokud ano použijme metodu **getResultSet()** k získání výsledku. Poté projdeme výsledek po řádcích, získáme hodnotu jednotlivých sloupců pomocí metod třídy **ResultSet** a na příklad ho můžeme vypsat pro uživatele.



6. Uvolnění objektu

Po použití by objekt statement a resultSet měl být uzavřeny, aby byly uvolněny zdroje a bylo zabráněno zbytečnému zatěžování databáze.



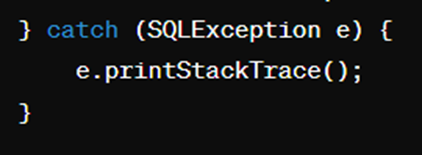


**Př. Rozhraní PreparedStatement a metoda executeQuery()**

1. try-catch

Celý kód musí být obalen v try-catch k zachycení případných výjimek, které mohou vzniknout při vykonání SQL dotazu





2. Vytvoření objektu preparedStatement

Objekt preparedStatement umožňuje vykonávat SQL dotazy na připojené databázi.



3. Vytvoření SELECT dotazu

Vytvoření jednoduchého dotazu SELECT s parametrem na tabulku moje\_tabulka.



4. Nastavení parametru dotazu

Vytvoří proměnou vekLimit a nastaví hodnotu parametru na 1. pozici na vekLimit.



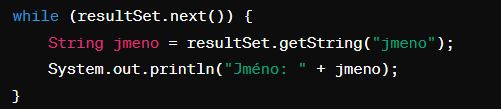
5. Vykonání dotazu

Pro tento příklad používáme metodu **executeQuery()**, která nám rovnou vrátí ResultSet, je možné použít i obecnou metodu **execute()**.



6. Zpracování výsledků

Projdeme výsledek po řádcích, získáme hodnotu jednotlivých sloupců pomocí metody **getString()** třídy **ResultSet** a na příklad ho můžeme vypsat pro uživatele.



7. Uvolnění objektu

Po použití by objekt preparedStatement a resultSet měl být uzavřeny, aby byly uvolněny zdroje a bylo zabráněno zbytečnému zatěžování databáze.

