JDBC

Connecting to a Database

Simple JDBC url

- JDBC url has this form:

```
jdbc:postgresql://localhost:5432/phonebook_db
```

protocol subprotocol subname

- instead of `localhost` you can use IP address:

```
jdbc:postgresql://127.0.0.1:5432/phonebook_db
```

- if using default port with localhost you can omit it altogether:

```
jdbc:postgresql://localhost/phonebook_db
```

- when connecting to remote host you can use hostname or IP, but with port:

```
jdbc:postgresql://192.168.1.170:5432/phonebook_db
```

Advanced JDBC url

- JDBC url can contain other features, like username, password, enable ssl encryption, etc.

```
jdbc:postgresql://localhost/phonebook_db?user=luka?password=luka123
```

```
jdbc:postgresql://localhost/phonebook_db?ssl=true
```

- in order to use JDBC features package `java.sql.*` must be imported

```
// let's connect to the database, inside main() method


String url =
   "jdbc:postgresql://localhost/phonebook_db?user=luka&password=luka123";
try (Connection conn = DriverManager.getConnection(url)) {
   if (conn != null) System.out.println("Connected to the database!");
      else System.out.println("Failed to make connection!");
} catch (SQLException e) {
    System.err.println(e.getMessage());
}
```

it's always recommended to use try-with-resources when connecting to db,
to make sure that the connection is properly closed after our operations are done

```
// there is an overriden version of getConnection() method


String url = "jdbc:postgresql://localhost/phonebook_db";
try (Connection conn = DriverManager.getConnection(url, "luka", "luka123")) {
    // make some operations
} catch (SQLException e) {
     System.err.println(e.getMessage());
}
```