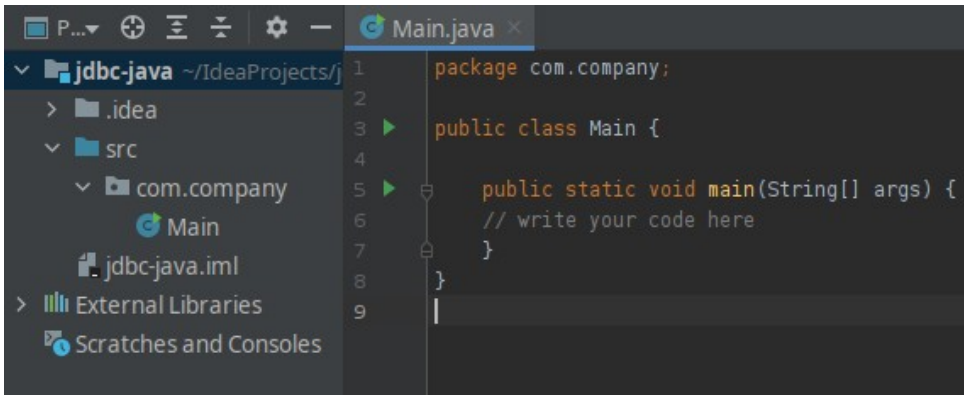
Поключение Jdbc к java.

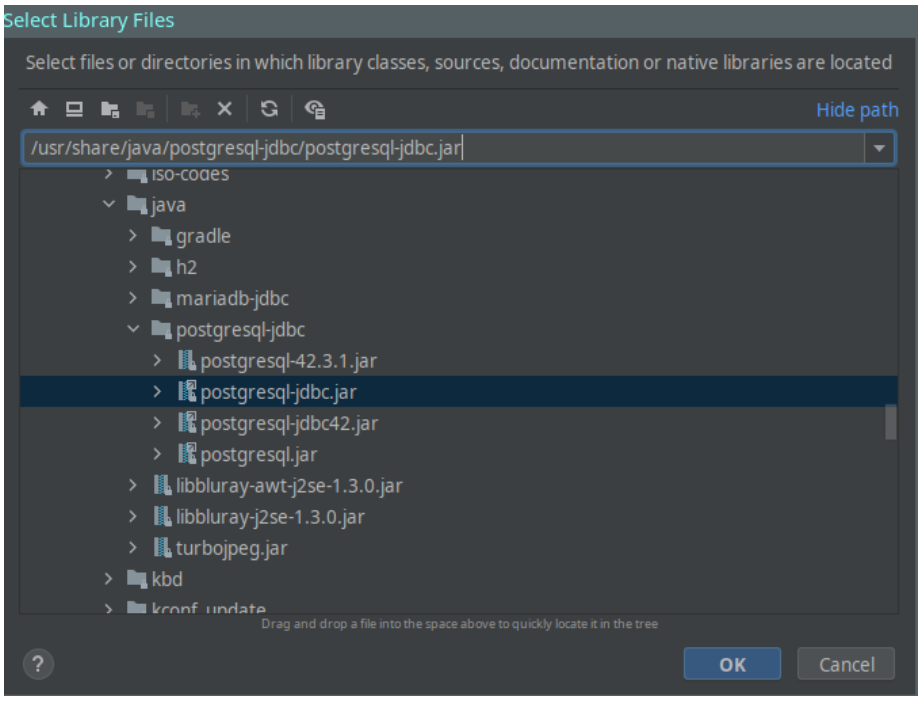
JDBC (Java DataBase Connectivity — соединение с базами данных на Java) предназначен для взаимодействия Java-приложения с различными системами управления базами данных (СУБД). Всё движение в JDBC основано на драйверах которые указываются специально описанным URL.

1. Создаём «чистый» проект на java.



2. Добавляем как зависимость jar файл модуля jdbc

Я использую СУБД : postgresql (*Пост-Грэс-Кью-Эл*)
и соответствующий jdbc : postgresql jdbc
Для этого, мы открываем вкладку file → project structure → libraries и в верхнем левом углу нажимаем на «+». Откройтся проводник, где мы должны найти драйвер в файловой структуре.
Путь на моём компьютере под управлением операционныой системы manjaro: /usr/share/java/postgresql-jdbc/postgresql-jdbc.jar



3. Создадим базу данных в системе управления базами данных:

```
[denis@denis-hpelitebookfolio9470m ~]$ sudo -iu postgres
[sudo] пароль для denis:
[postgres@denis-hpelitebookfolio9470m ~]$ createdb template
[postgres@denis-hpelitebookfolio9470m ~]$ psql -d template
psql (13.6)
Введите "help", чтобы получить справку.

template=#
```

4. Создадим таблицу employee:

```
template=# create table employee (
id_emp integer,
id_branch integer,
lastname varchar(30),
firstname varchar(30),
date_emp date,
salary integer,
commision numeric(4,2)
);
CREATE TABLE
```

5. Внесём в неё данные:

```
template=#
insert into employee
values (1, 1, 'Igor', 'Stezea', to_date('01/02/2012','dd/mm/yyyy'), 5600, 2.3);
INSERT 0 1
template=#
insert into employee
values (2, 1, 'Andrei', 'Cartea', to_date('12/04/2013','dd/mm/yyyy'), 5300, 1.9);
INSERT 0 1
```

6. Переходим к java. Нам нужно ввести этот код:

```
package com.company;

import java.sql.*;

public class Main {
    public static void main(String[] args) {
        Connection connection = null;
        try {
            connection =
DriverManager.getConnection("jdbc:postgresql://192.168.43.184/template", "postgres",
"1961");
            System.out.println("\nConnected to the PostgreSQL server successfully.\n");
```

```
            Statement statement = connection.createStatement();
            String selectEmployee = "SELECT " +
                    "id_emp, " +
                    "id_branch, " +
                    "rpad(lastname, 15), " +
                    "rpad(firstname, 15), " +
                    "date_emp, " +
                    "salary, " +
                    "commision " +
                    "FROM employee;";
            ResultSet resultSet = statement.executeQuery(selectEmployee);
            while(resultSet.next()) {
                int id_emp = resultSet.getInt(1);
                int id_branch = resultSet.getInt(2);
                String firstname = resultSet.getString(3);
                String lastname = resultSet.getString(4);
                Date date = resultSet.getDate(5);
                int salary = resultSet.getInt(6);
                double commisiom = resultSet.getDouble(7);
                System.out.println(id_emp + "\t|  " + id_branch + "\t|  " + firstname + "  |
" + lastname + "  |  " + date + "  |  " + salary + "  |  " + commisiom);
            }
            connection.close();
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Поясняю:

Для подключения к базе данных необходимо создать объект java.sql.Connection. Для его создания применяется метод: connection = DriverManager.getConnection(url, username, password)

```
connection = DriverManager.getConnection("jdbc:postgresql://192.168.43.184/template",
"postgres", "1961");
```

Метод DriverManager.getConnection в качестве параметров принимает адрес источника данных, логин и пароль. В качестве логина и пароля передаются логин и пароль от сервера postgresql. Адрес локальной базы данных postgresql указывается в следующем формате: jdbc:postgresql://localhost/название_базы данных, но так как я содал удалённую базу данных, в маём случае это ip address моего устройства jdbc:postgresql://192.168.43.184/

Для взаимодействия с базой данных приложение отправляет серверу MySQL команды на языке SQL. Чтобы выполнить команду, вначале необходимо создаеть объект Statement.

Для его создания у объекта Connection вызывается метод createStatement():

Statement statement = connection.createStatement();

```
Statement statement = connection.createStatement();
```

Для выполнения команд SQL в классе Statement определено три метода:

- executeUpdate: выполняет такие команды, как INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE. В качестве результата возвращает количество строк, затронутых операцией (например, количество добавленных, измененных или удаленных строк), или 0, если ни одна строка не затронута операцией или если команда не изменяет содержимое таблицы (например, команда создания новой таблицы)
- executeQuery: выполняет команду SELECT. Возвращает объект ResultSet, который содержит результаты запроса.
- execute(): выполняет любые команды и возвращает значение boolean: true - если команда возвращает набор строк (SELECT), иначе возвращается false.

Для выборки данных с помощью команды SELECT применяется метод executeQuery:

```
String selectEmployee = "SELECT " +
                    "id_emp, " +
                    "id_branch, " +
                    "rpad(lastname, 15), " +
                    "rpad(firstname, 15), " +
                    "date_emp, " +
                    "salary, " +
                    "commision " +
                    "FROM employee;";
            ResultSet resultSet = statement.executeQuery(selectEmployee);
```

В объекте ResultSet итератор устаналивается на позиции перед первой строкой. И чтобы переместиться к первой строке (и ко всем последующим) необходимо вызвать метод next(). Пока в наборе ResultSet есть доступные строки, метод next будет возвращать true.

```
ResultSet resultSet = statement.executeQuery(selectEmployee);
            while(resultSet.next()) {
                int id_emp = resultSet.getInt(1);
                int id_branch = resultSet.getInt(2);
                String firstname = resultSet.getString(3);
                String lastname = resultSet.getString(4);
                Date date = resultSet.getDate(5);
                int salary = resultSet.getInt(6);
                double commisiom = resultSet.getDouble(7);
                System.out.println(id_emp + "\t|   " + id_branch + "\t|   " + firstname + "   |
" + lastname + "   |   " + date + "   |   " + salary + "   |   " + commisiom);
            }
```

После перехода к строке мы можем получить ее содержимое. Для этого у ResultSet определен ряд методов. Некоторые из них:

- getInt() возвращает значение int
- getString() возвращает значение String
- getDate() возвращает значение Date, объект определённый в пакете java.sql.*
- getDouble() возвращает значение double

Результат:

```
Connected to the PostgreSQL server successfully.

1   |  1    |  Igor            |  Stezea          |  2012-02-01  |  5600  |  2.3
2   |  1    |  Andrei          |  Cartea          |  2013-04-12  |  5300  |  1.9

Process finished with exit code 0
```

Для добавления, редактирования и удаления данных мы можем ипользовать рассмотренный метод executeUpdate. С помощью результата метода мы можем проконтроллировать, сколько строк было добавлено, изменено или удалено.

Добавим эти строки до создания объекта rezultSet:

```
String insert = "INSERT INTO employee " +
        "VALUES(20, 7, 'Irina', 'Verlan', to_date('11.11.2019', 'dd.mm.yyyy'), 6100);";
int rows = statement.executeUpdate(insert);
System.out.println("\nБыло добавленно " + rows + " строк.");
```

Должно выглядеть так:

```
String insert = "INSERT INTO employee " +
        "VALUES(20, 7, 'Irina', 'Verlan', to_date('11.11.2019', 'dd.mm.yyyy'), 6100);";
int rows = statement.executeUpdate(insert);
System.out.println("\nБыло добавленно " + rows + " строк.");
ResultSet resultSet = statement.executeQuery(selectEmployee);
while(resultSet.next()) {
```

Запустим программу:

```
Connected to the PostgreSQL server successfully.


Было добавленно 1 строк.
1   |  1    |  Igor            |  Stezea          |  2012-02-01  |  5600  |  2.3
2   |  1    |  Andrei          |  Cartea          |  2013-04-12  |  5300  |  1.9
20  |  7    |  Irina           |  Verlan          |  2019-11-11  |  6100  |  0.0

Process finished with exit code 0
```