Tutorial Database and File

Designed by ZHU Yueming , PAN Chao and Wwy.

Experimental objective

- Analyze that, compare with file, what is the advantage and disadvantage of database .
- Learn how to import data from .sql files.
- Learn how to build the jdbc program according to the source code.
- In this lab, you do not need to understand meaning of those sql statements.

Before your work thinking some questions

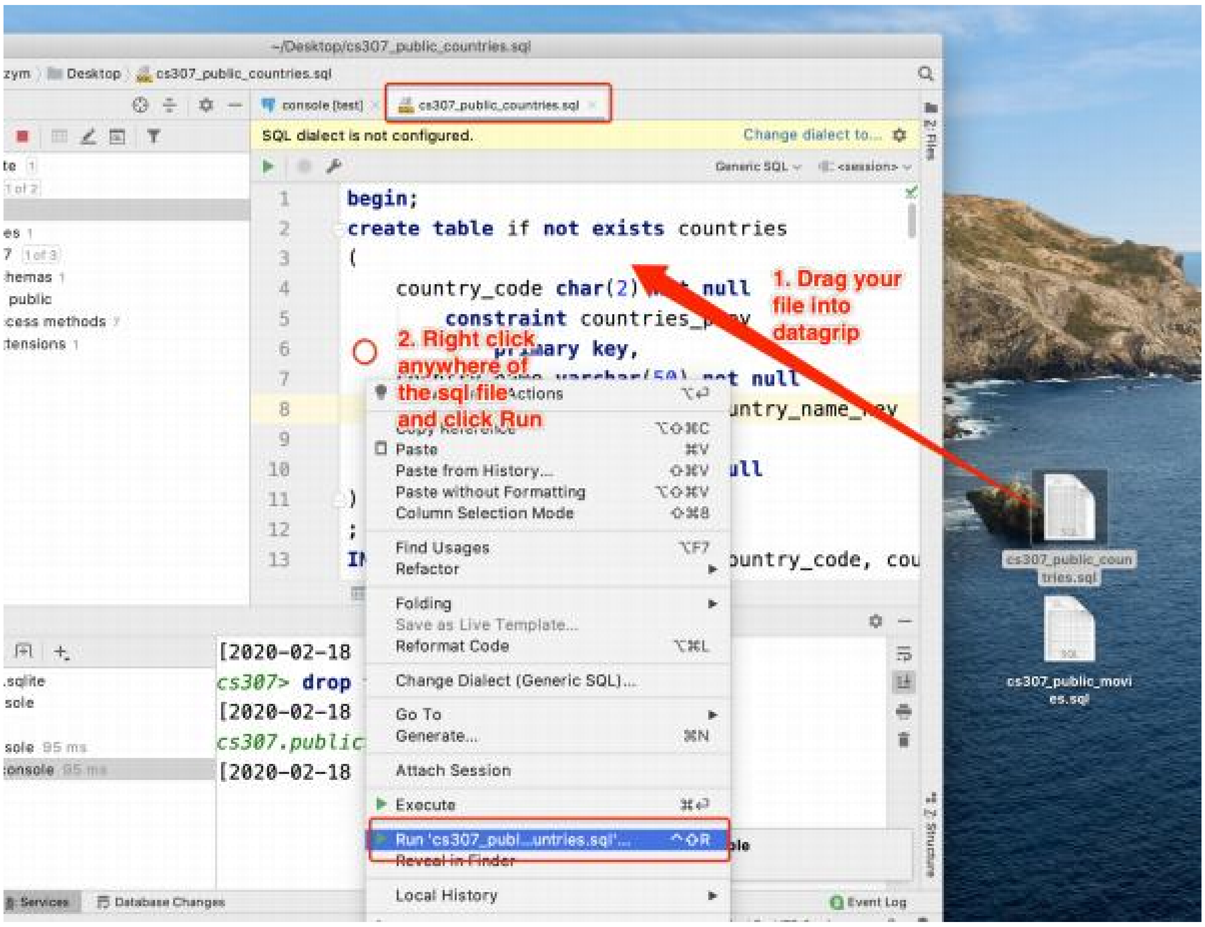
The following questions are based on processing data by using file system. In this project we only have two .txt files including movies.txt and countries.txt.

1. If we want to add a new movie into movies.txt, what should we do?
2. If we want to get all continent names from countries.txt, what should we do?
3. Suppose that only get the continent names is not enough, and we need continent names with how many countries in each continent, what should we do?
4. More requirements, if we need the information including title of movie, the country name, the continent and the runtime of all movies according to a specific range of runtime, in this case, we have to fetch data from two txt files. What should we do?
5. If we have other requirements, how can we accomplish them?

Import Data

1. By Datagrip

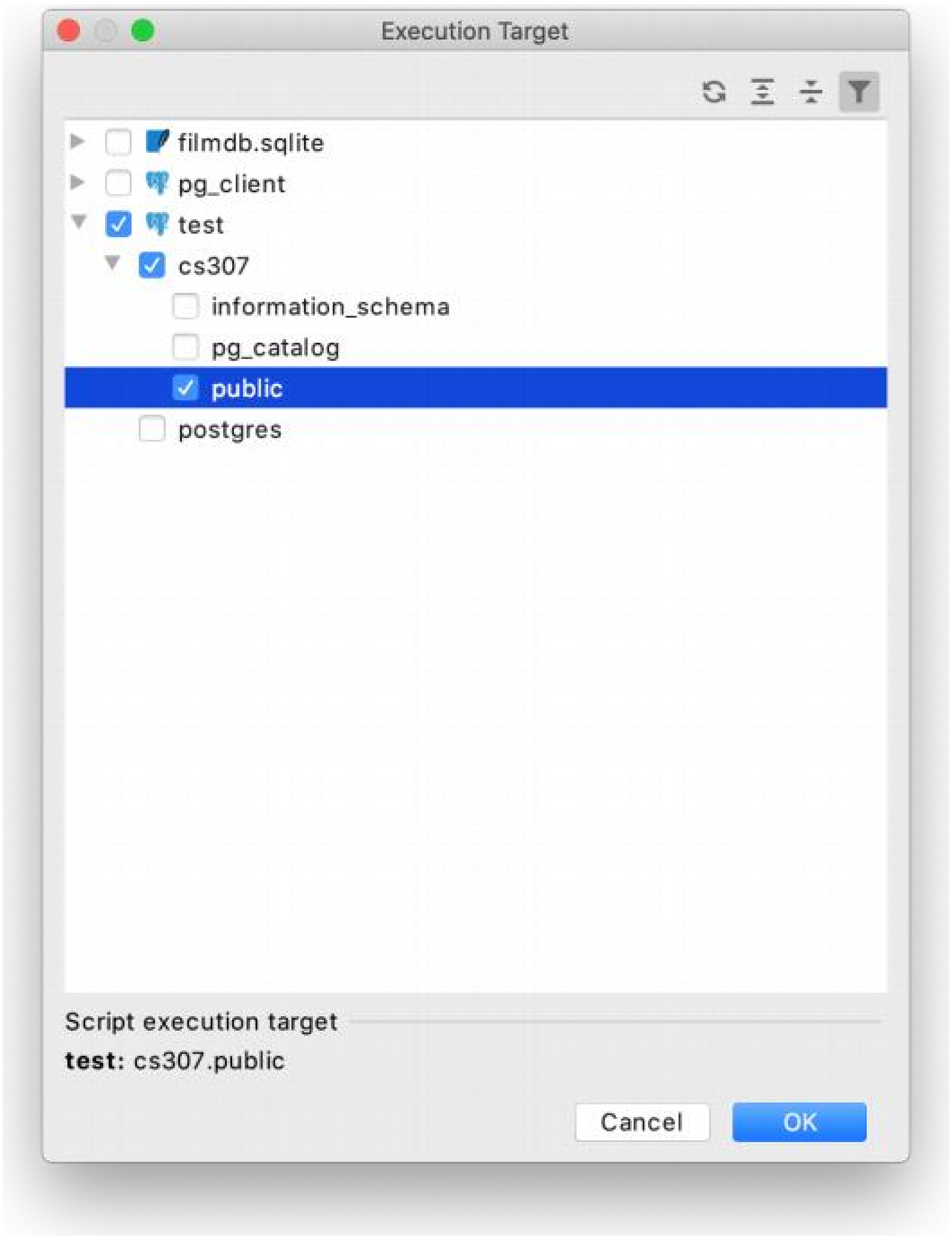
- Drag your cs307_public_countries.sql into datagrip



```
begin;
create table if not exists countries
(
    country_code char(2) not null
        constraint countries_pkey
            primary key,
    country_name varchar(50) not null
                                  untry_name_key

                                  ull
)
;
IN                                ountry_code, cou
```

```
[2020-02-18
cs307> drop
[2020-02-18
cs307.public
[2020-02-18
```

- Execute .sql file in target database.



Script execution target

test: cs307.public

- Similar way to import cs307_public_movies.sql

2. By CLI

- Connect database

```
psql -U checker -d cs307 -h 127.0.0.1 -p 5432
```

- Execute cs307_public_countries.sql

```
\i your_path_of_cs307_public_countries.sql
```

- Execute cs307_public_movies.sql

  ```
  \i your_path_of_cs307_public_movies.sql
  ```

Test yout import

After execute:

```
select count(*) from movies;
```

result is 9536

```
select count(*) from countries;
```

result is 185

Build your project

1. By IDE

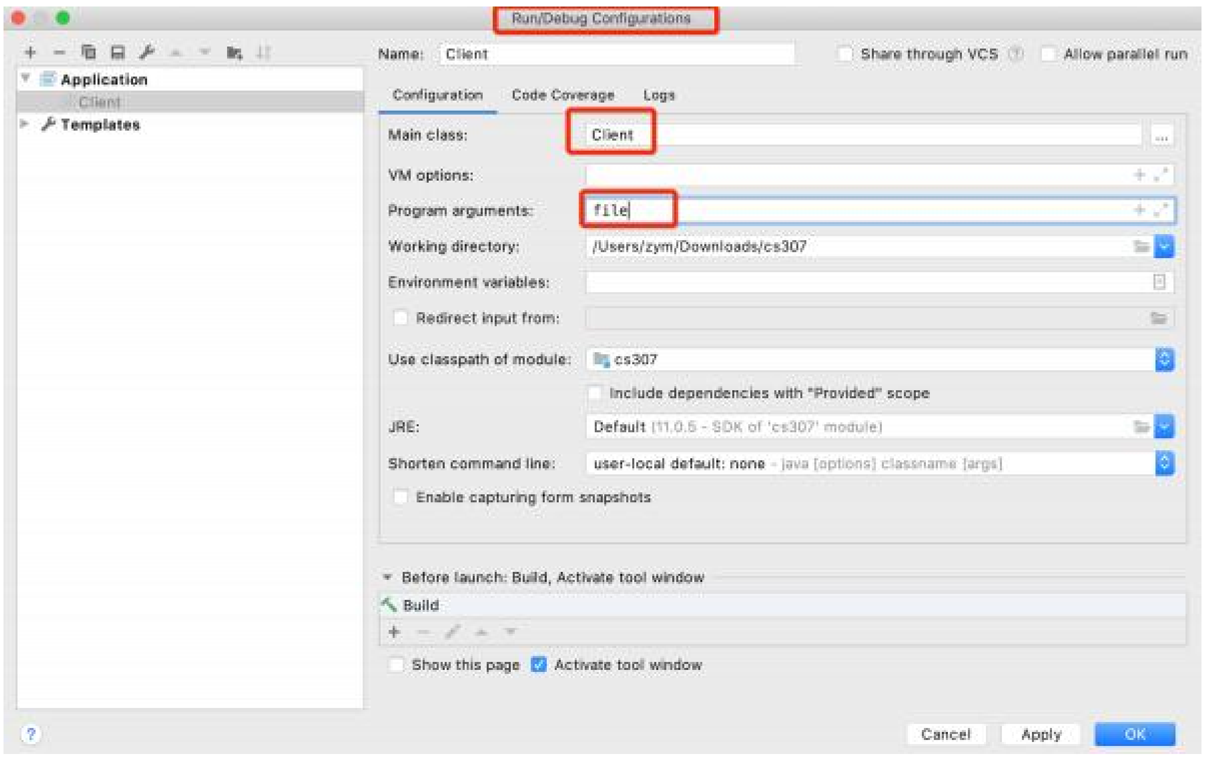
In this tutorial, we use **intellij idea** as IDE, and you can also do it by other way.

- Drag four .java files into src (generated sources root) folder.
- Drag two .txt files into your project folder.
- Run Client.java

  It will meet an exception as follows.

  Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:……
- Setting Configurations

  Run -> Edit Configurations -> Configuration -> Program arguments


Run/Debug Configurations

Application
Client
Templates

Name: Client
Share through VCS
Allow parallel run

Configuration | Code Coverage | Logs

Main class: Client
VM options:
Program arguments: file
Working directory: /Users/zym/Downloads/cs307
Environment variables:
Redirect input from:
Use classpath of module: cs307
Include dependencies with "Provided" scope
JRE: Default (11.0.5 - SDK of 'cs307' module)
Shorten command line: user-local default: none - java [options] classname [args]
Enable capturing form snapshots

Before launch: Build, Activate tool window
Build

Show this page
Activate tool window

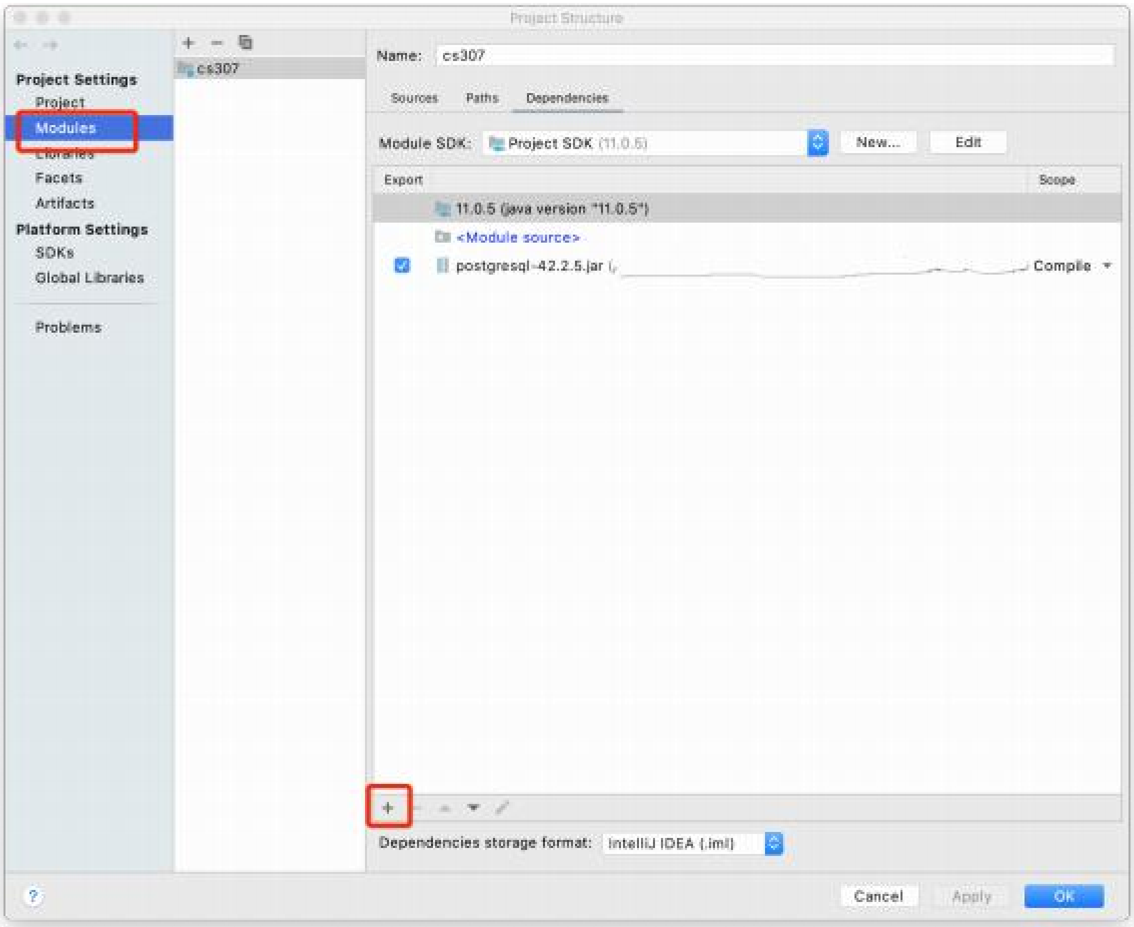
Cancel | Apply | OK

- Run again the result would be

```
ASIA
EUROPE
AFRICA
AMERICA
OCEANIA

EUROPE   40
AFRICA   52
......
```

- Import postgres Driver into your project.

  File -> Project Structure -> Modules -> '+' -> Jar or directories -> browse your driver path



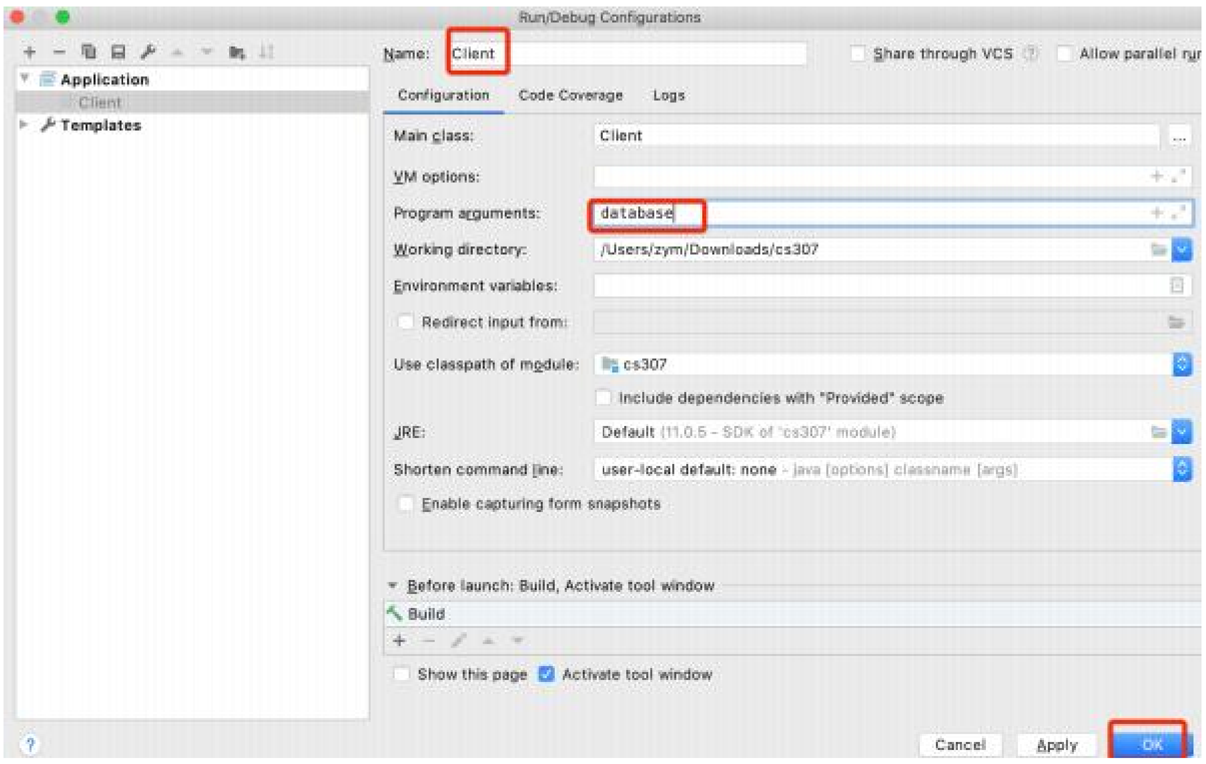
Name: cs307

Module SDK: Project SDK (11.0.5)

11.0.5 (java version "11.0.5")
<Module source>
✓ postgresql-42.2.5.jar — Compile

Dependencies storage format: IntelliJ IDEA (.iml)

- Change Program argument to **database**



Name: Client

Main class: Client

VM options:

Program arguments: database

Working directory: /Users/zym/Downloads/cs307

Environment variables:

Redirect input from:

Use classpath of module: cs307

Include dependencies with "Provided" scope

JRE: Default (11.0.5 - SDK of 'cs307' module)

Shorten command line: user-local default: none - java [options] classname [args]

Enable capturing form snapshots

Before launch: Build, Activate tool window

Show this page ✓ Activate tool window

2. By CLI

- Make sure your computer has java and javac environment.
- Put all your .java files, .txt files and postgreSQL driver into one folder
- Compile:

```
javac Client.java
javac FileManipulation.java
javac DatabaseManipulation.java
```

- Run file manipulation

  ```
  java Client file
  ```

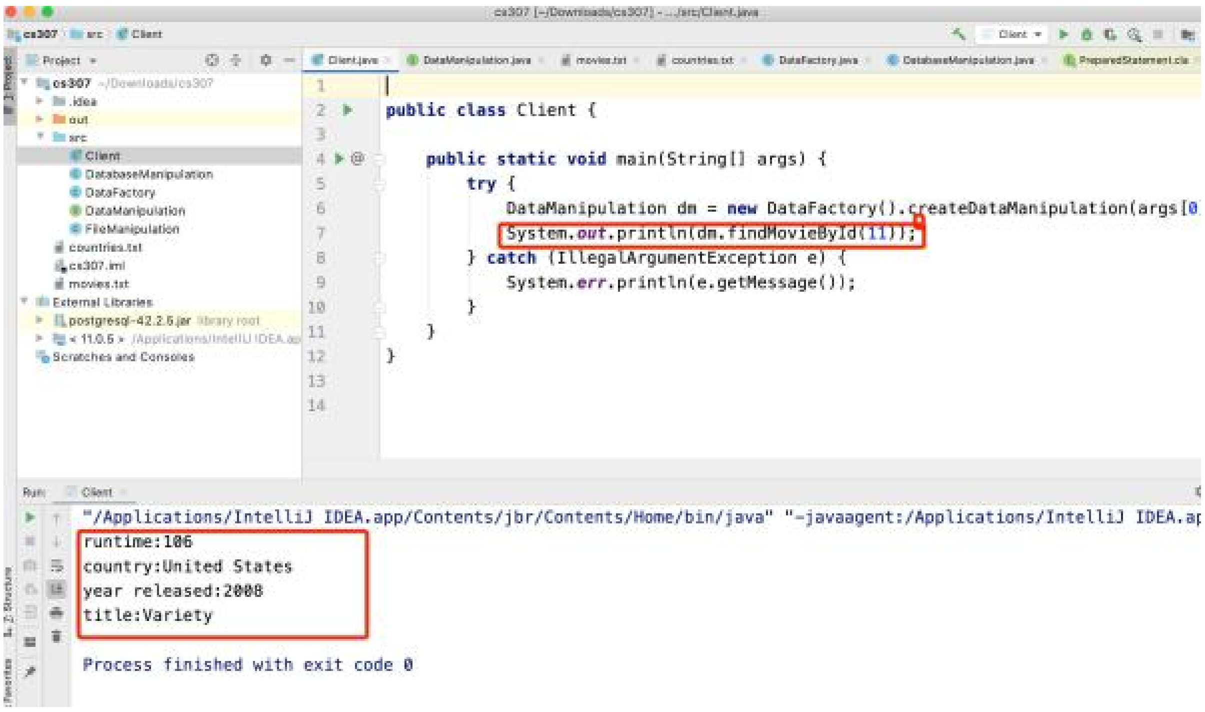
- Run database manipulation

  Linux or Mac

  ```
  java -cp .:postgresql-42.2.5.jar Client database
  ```

  Windows

  ```
  java -cp .;postgresql-42.2.5.jar Client database
  ```



```java

public class Client {

    public static void main(String[] args) {
        try {
            DataManipulation dm = new DataFactory().createDataManipulation(args[0
            System.out.println(dm.findMovieById(11));
        } catch (IllegalArgumentException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

```
"/Applications/IntelliJ IDEA.app/Contents/jbr/Contents/Home/bin/java" "-javaagent:/Applications/IntelliJ IDEA.ap
runtime:106
country:United States
year released:2008
title:Variety

Process finished with exit code 0
```

Tips:

You can use executeQuery() method to execute the query in prepare statement, and the result would stored in resultSet

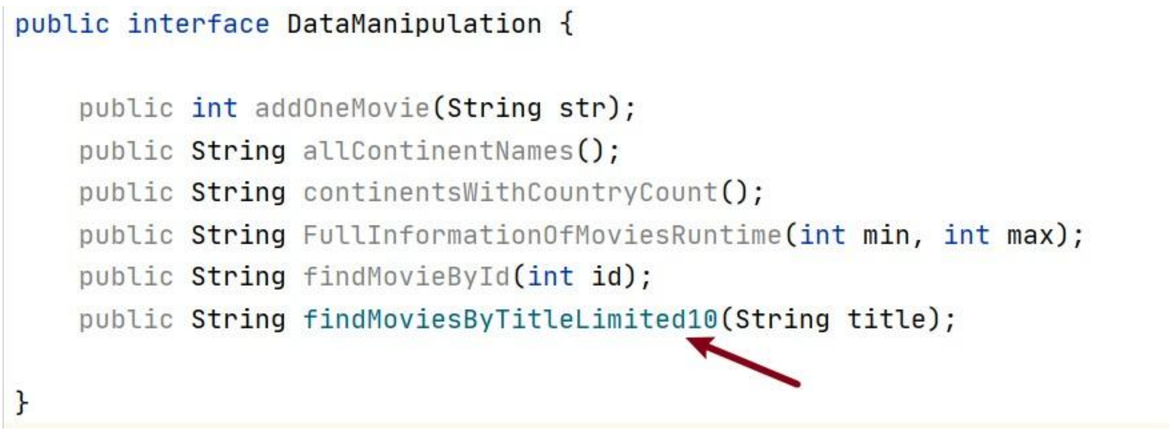
```
resultSet = preparedStatement.executeQuery();
```

If you want to find the first selecting result in resultSet, you can do:

```
resultSet.next();
```

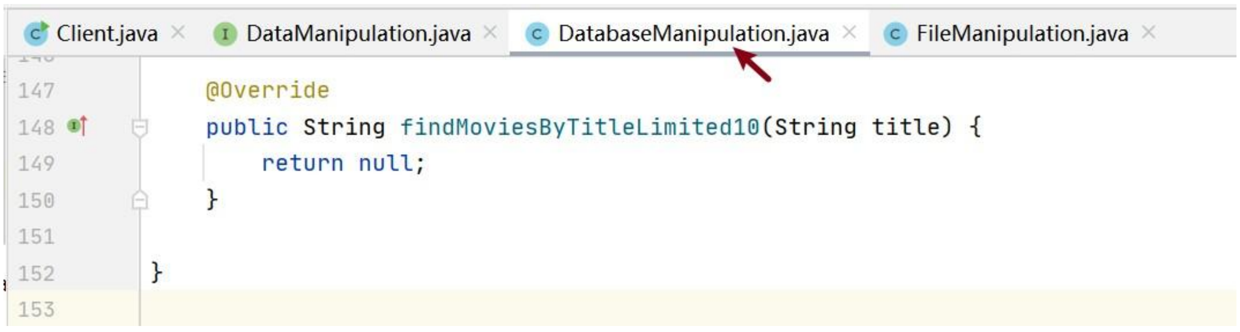
Experiment:
Add a new function 'findMoviesByTitleLimited10' in interface file

Step1: add function in interface

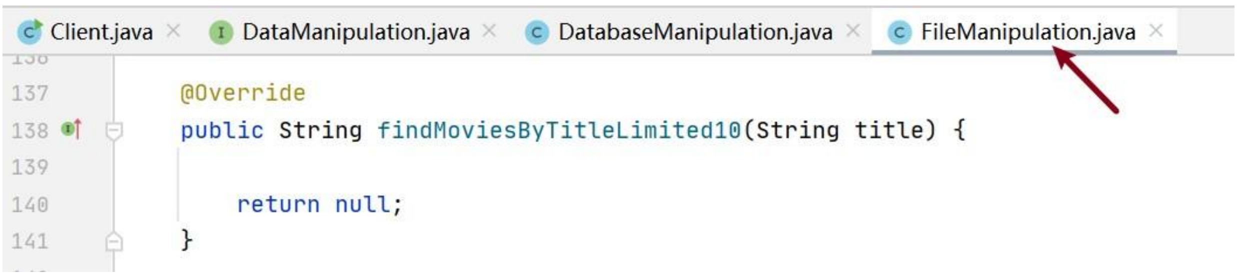


```
public interface DataManipulation {

    public int addOneMovie(String str);
    public String allContinentNames();
    public String continentsWithCountryCount();
    public String FullInformationOfMoviesRuntime(int min, int max);
    public String findMovieById(int id);
    public String findMoviesByTitleLimited10(String title);

}
```

Step2: add realization functions in DatabaseManipulation/FileManipulation class.



```
    @Override
    public String findMoviesByTitleLimited10(String title) {
        return null;
    }

}
```



```
    @Override
    public String findMoviesByTitleLimited10(String title) {

        return null;
    }
```

Step3:Create a SQL to achieve

```
select m.title, c.country_name country, m.runtime,m.year_released
from movies m join countries c on m.country = c.country_code
where m.title like '%'||'ah'||'%'limit 10;
```

Step4:Complete the fuction(here use statement)

```
@Override
public String findMoviesByLimited10(String title) {
     getConnection(); // start connection
     String sql = "select m.title, c.country_name country,
m.runtime,m.year_released\n"+
"from movies m join countries c on m.country = c.country_code\n"+
"where m.title like '%'||"+title+"||'%'limit 10;";// string combination
     try {
          Statement statment = con.createStatement();
          resultSet = statment.executeQuery(sql);
          /*
          // U can also use preparedStatement
          // change here!
          PreparedStatement preparedStatement = con.prepareStatement(sql);
          // change here!
          preparedStatement.setString(1, title);
          resultSet = preparedStatement.executeQuery();// and here!
          */
          StringBuilder strb=new StringBuilder(); //combine multi-strings
          while (resultSet.next()){
               strb.append(String.format("%-20s\t",
               resultSet.getString("country")));
               strb.append(resultSet.getInt("year_released")).append("\t");
               strb.append(resultSet.getInt("runtime")).append("\t");
               strb.append(resultSet.getString("title")).append("\n");
          }
          return strb.toString();
     } catch (SQLException throwables) {
          throwables.printStackTrace();
     }
     finally {
          closeConnection(); // close connection
     }
     return null;
}
```

Step5:Add output in client file , run client, and get the result.

```
System.out.println(dm.findMoviesByLimited10("'aba'"));
```

Tips: Statement vs PreparedStatement
Both Statement and PreparedStatement can be used to execute SQL queries. These interfaces look very similar. However, they differ significantly from one another in features and performance:

- Statement – Used to execute string-based SQL queries
- PreparedStatement – Used to execute parameterized SQL queries

Advantages/Disadvantages

- In Statement, JDBC passes the query with inline values to the database
- the Statement interface is suitable for DDL queries like CREATE, ALTER, and DROP
- the PreparedStatementprotects against SQL injection
- PreparedStatement uses pre-compilation.This feature speeds up the communication between the database and the JVM
- the PreparedStatement provides a batch execution during a single database connection
- the Statement interface is suitable for DDL queries like CREATE, ALTER, and DROPit is more suitable to use Statement for DDL queries and for DML queries