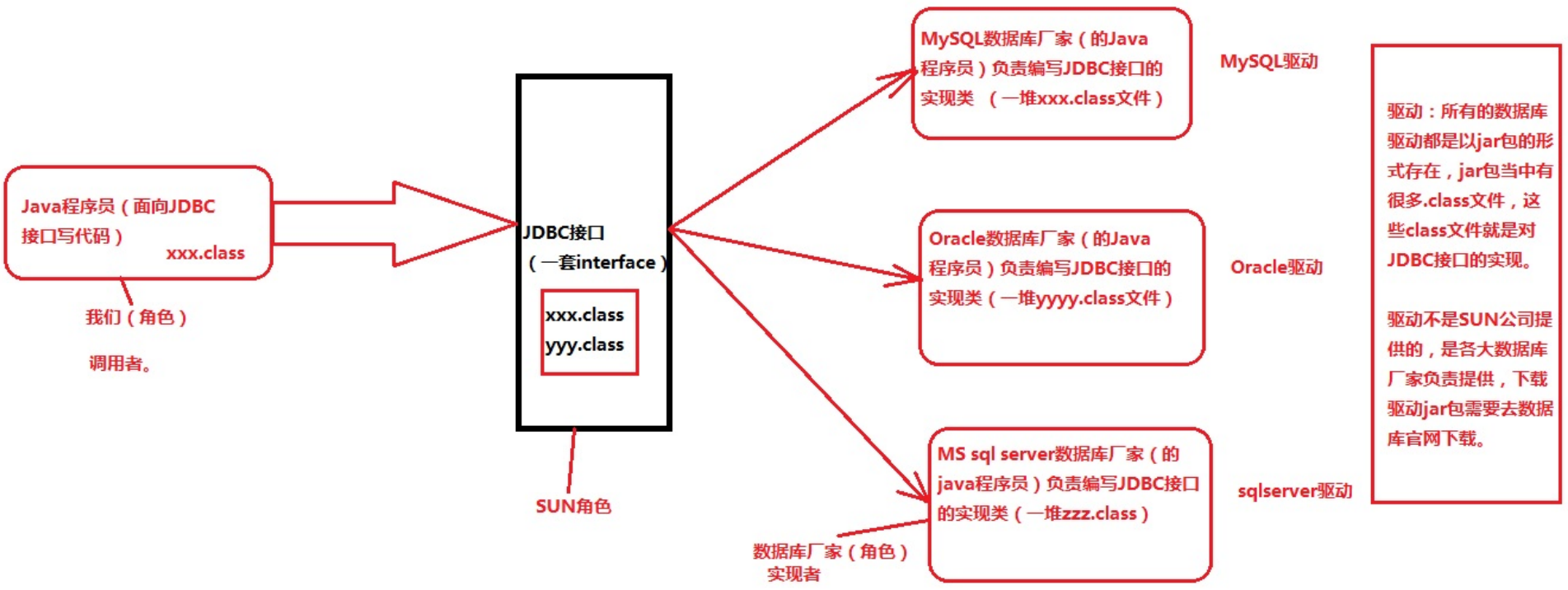
JDBC

1、JDBC是什么?

Java DataBase Connectivity（Java语言连接数据库）

2、JDBC的本质是什么?

```
JDBC是SUN公司制定的一套接口（interface）
    java.sql.*;（这个软件包下有很多接口。）

接口都有调用者和实现者。
面向接口调用、面向接口写实现类，这都属于面向接口编程。

为什么要面向接口编程?
    解耦合：降低程序的耦合度，提高程序的扩展力。
    多态机制就是非常典型的：面向抽象编程。（不要面向具体编程）
        建议：
            Animal a = new Cat();
            Animal a = new Dog();
            // 喂养的方法
            public void feed(Animal a){ // 面向父类型编程。
            }
        不建议：
            Dog d = new Dog();
            Cat c = new Cat();

思考：为什么SUN制定一套JDBC接口呢?
    因为每一个数据库的底层实现原理都不一样。
    Oracle数据库有自己的原理。
    MySQL数据库也有自己的原理。
    MS SqlServer数据库也有自己的原理。
    ....
    每一个数据库产品都有自己独特的实现原理。

JDBC的本质到底是什么?
    一套接口。
```



3、模拟实现JDBC本质

```
/**
 * SUN公司负责制定这套JDBC接口
 */
public interface JDBC {
    /**
     * 连接数据库的方法
     */
    void getConnection();
}
```

```
/**
 * MySQL数据库厂家负责编写JDBC接口的实现类
 * 实现类被称为驱动 （MySQL驱动）
 */
```

```
public class MySql implements JDBC {
    @Override
    public void getConnection() {
        /**
         * 具体这里的代码怎么写，对于Java程序员来说没关系
         * 这段代码涉及到mysql底层数据库的实现原理
         */
        System.out.println("连接MYSQL数据库成功！");
    }
}

/**
 * Oracle数据库厂家负责编写JDBC接口的实现类
 * 实现类被称为驱动  （Oracle驱动）
 */
public class Oracle implements JDBC {
    @Override
    public void getConnection() {
        /**
         * 这段代码涉及到Oracle底层数据库的实现原理
         */
        System.out.println("连接Oracle数据库成功！");
    }
}

/**
 * SqlServer数据库厂家负责编写JDBC接口的实现类
 * 实现类被称为驱动  （SqlServer驱动）
 */
public class SqlServer implements JDBC {
    @Override
    public void getConnection() {
        /**
         * 这段代码涉及到SqlServer底层数据库的实现原理
         */
        System.out.println("连接SqlServer数据库成功！");
    }
}
```

```
/**
 *  Java程序员角色
 *  不需要关心具体是哪个品牌的数据库，只需要面向JDBC接口写代码。
 *  面向接口编程，面向抽象编程，不要面向具体编程。
 */
public class JavaProgrammer {
    public static void main(String[] args) throws Exception {
        //JDBC jdbc = new MySql();
        //JDBC jdbc = new SqlServer();
        //JDBC jdbc = new Oracle();

        // 创建对象可以通过反射机制
        //Class c = Class.forName("com.sql.database_Manufacturer.Oracle");
        //Class c = Class.forName("com.sql.database_Manufacturer.MySql");
        //Class c = Class.forName("com.sql.database_Manufacturer.SqlServer");

        //资源绑定器
        ResourceBundle bundle = ResourceBundle.getBundle("JDBC");
        String className = bundle.getString("Oracle");
        Class c = Class.forName(className);

        JDBC jdbc = (JDBC) c.newInstance();

        // 以下代码都是面向接口调用方法，不需要修改
        jdbc.getConnection();
    }
}
```

```
#属性配置文件  JDBC.properties
Oracle=Oracle
MySql=MySql
SqlServer=SqlServer
```

4、JDBC开发前的准备工作

```
3、JDBC开发前的准备工作，先从官网下载对应的驱动jar包，然后将其配置到环境变量classpath当中。
classpath=.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;X:\newTool\MySql Connector Java 5.1.23\mysql-connector-java-5.1.23-bin.jar

以上的配置是针对于文本编辑器的方式开发，使用IDEA工具的时候，不需要配置以上的环境变量。
IDEA有自己的配置方式。
```

5、JDBC编程六步

```
4、JDBC编程六步（需要背会）
第一步：注册驱动（作用：告诉Java程序，即将要连接的是哪个品牌的数据库）

第二步：获取连接（表示JVM的进程和数据库进程之间的通道打开了，这属于进程之间的通信，重量级的，使用完之后一定要关闭通道。）

第三步：获取数据库操作对象（专门执行sql语句的对象）

第四步：执行SQL语句（DQL DML....）

第五步：处理查询结果集（只有当第四步执行的是select语句的时候，才有这第五步处理查询结果集。）

第六步：释放资源（使用完资源之后一定要关闭资源。Java和数据库属于进程间的通信，开启之后一定要关闭。）
```

```
IDEA:
1.导入驱动jar包
    复制mysql-connection-5.1.37-bin.jar到创建好的libs项目目录下面
    右键libs---->Add As library正真的把jar包加载进来
2.注册驱动
3.获取数据库连接对象 Connection
4.定义sql语句
5.获取执行sql语句的对象 Statement
6.执行sql,接受返回结果
7.处理结果
8.释放资源
```

6、JDBC代码

1.编程六步

```java
package com.java.jdbc;
import com.mysql.jdbc.Driver;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
/**
 * JDBC 编程六步
 */
public class JDBCTest01 {
    public static void main(String[] args) {
        Connection connection = null;
        Statement statement = null;
        try {
            //第一步：注册驱动
            Driver driver = new com.mysql.jdbc.Driver();//Mysql的驱动
            //Driver driver = new oracle.jdbc.driver.OracleDriver();//oracle的驱动
            DriverManager.registerDriver(driver);

            //第二步：获取连接
            /*
                url:统一资源定位符（网络中某个资源的绝对路径）
                https://www.baidu.com/  这就是URL
                URL包括哪几部分？
                    协议  IP  Port  资源名

                https://182.61.200.7:80/index.html
                    https:// 通信协议
                    182.61.200.7 服务器IP地址
                    80 服务器上软件的端口
                    index.html 是服务器上某个资源名

                jdbc:mysql://127.0.0.1:3306/bjpowernode
                    jdbc:mysql:// 协议
                    127.0.0.1 IP地址
                    3306 mysql数据库端口号
                    bjpowernode 具体的数据库实例名

                说明：localhost 和 127.0.0.1 都是本地IP地址

                jdbc:mysql://192.168.151.27:3306/bjpowernode

                什么是通信协议？有什么用？
                    通信协议是通信之前就提前定好的数据传送格式。
                    数据包具体怎么传数据，格式提前定好的。

                oracle的URL: jdbc:oracle:thin:@localhost:1521:orcl
             */
            String url = "jdbc:mysql://127.0.0.1:3306/bjpowernode";
            String user = "root";
            String password = "111";
            connection = DriverManager.getConnection(url, user, password);
            //数据库连接对象: com.mysql.jdbc.JDBC4Connection@14514713
            System.out.println("数据库连接对象: " + connection);

```

```
            //第三步：获取数据库操作对象(Statement专门执行sql语句的)
            statement = connection.createStatement();

            //第四步：执行SQL语句
            String sql = "insert into dept(deptno, dname, loc) values(50, '人事部', '北京')";
            //专门执行DML语句的（insert delete update）
            //返回值是“影响数据库中的记录条数”
            int count = statement.executeUpdate(sql);
            System.out.println(count == 1 ? "Success!" : "Fail");

            //第五步：处理查询结果集
        } catch (SQLException e) {
            e.printStackTrace();
        }finally {
            //第六步：释放资源
            // 为了保证资源一定释放，在finally语句块中关闭资源
            // 并且要遵循从小到大依次关闭
            // 分别对其try...catch
            if(statement != null){
                try {
                    statement.close();
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
            if(connection != null){
                try {
                    connection.close();
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
        }
    }
}
```

2.JDBC完成delete update

```
package com.java.jdbc;
import java.sql.*;
/**
 * JDBC完成delete update
 */
public class JDBCTest02 {
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //1、注册驱动
            DriverManager.registerDriver(new com.mysql.jdbc.Driver());

            //2、获取连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");

            //3、获取数据库操作对象
            stmt = conn.createStatement();

            //4、执行Sql语句
            //String sql = "delete from dept where deptno = 40";
            String sql = "update dept set dname = '销售部', loc = '盐城' where deptno = 20";
            int count = stmt.executeUpdate(sql);
            System.out.println(count == 1 ? "Success" : "Fail");
        }catch (SQLException e){
            e.printStackTrace();
        }finally {
            //6、释放资源
            if(stmt != null){
                try {
                    stmt.close();
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
            if (conn != null){
                try {
                    conn.close();
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
        }
    }
}
```

3.注册驱动的另一种方式

```java
/**
 * 注册驱动的另一种方式
 */
public class JDBCTest03 {
    public static void main(String[] args) {
        try {
            /*
            //注册驱动
            //这是注册驱动的第一种写法
            DriverManager.registerDriver(new com.mysql.jdbc.Driver());
            //获取连接
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
            System.out.println(conn);//com.mysql.jdbc.JDBC4Connection@14514713
            */

            //注册驱动
            //这是注册驱动的第二种写法:常用写法
            //为什么这种方式常用？因为参数是一个字符串，字符串可以写到xx.properties文件中
            //以下方法不需要接收返回值，因为我们只想用它的类加载动作。
            Class.forName("com.mysql.jdbc.Driver");
            //获取连接
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
            System.out.println(conn);//com.mysql.jdbc.JDBC4Connection@443b7951

        }catch (SQLException e){
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}

package com.mysql.jdbc;
import java.sql.DriverManager;
import java.sql.SQLException;
public class Driver extends NonRegisteringDriver implements java.sql.Driver {
    public Driver() throws SQLException {
    }

    static {
        try {
            DriverManager.registerDriver(new Driver());
        } catch (SQLException var1) {
            throw new RuntimeException("Can't register driver!");
        }
    }
}
```

4.将连接数据库的所有信息配置到配置文件中

```properties
#属性配置文件 jdbc.properties
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/bjpowernode
user=root
password=111
```

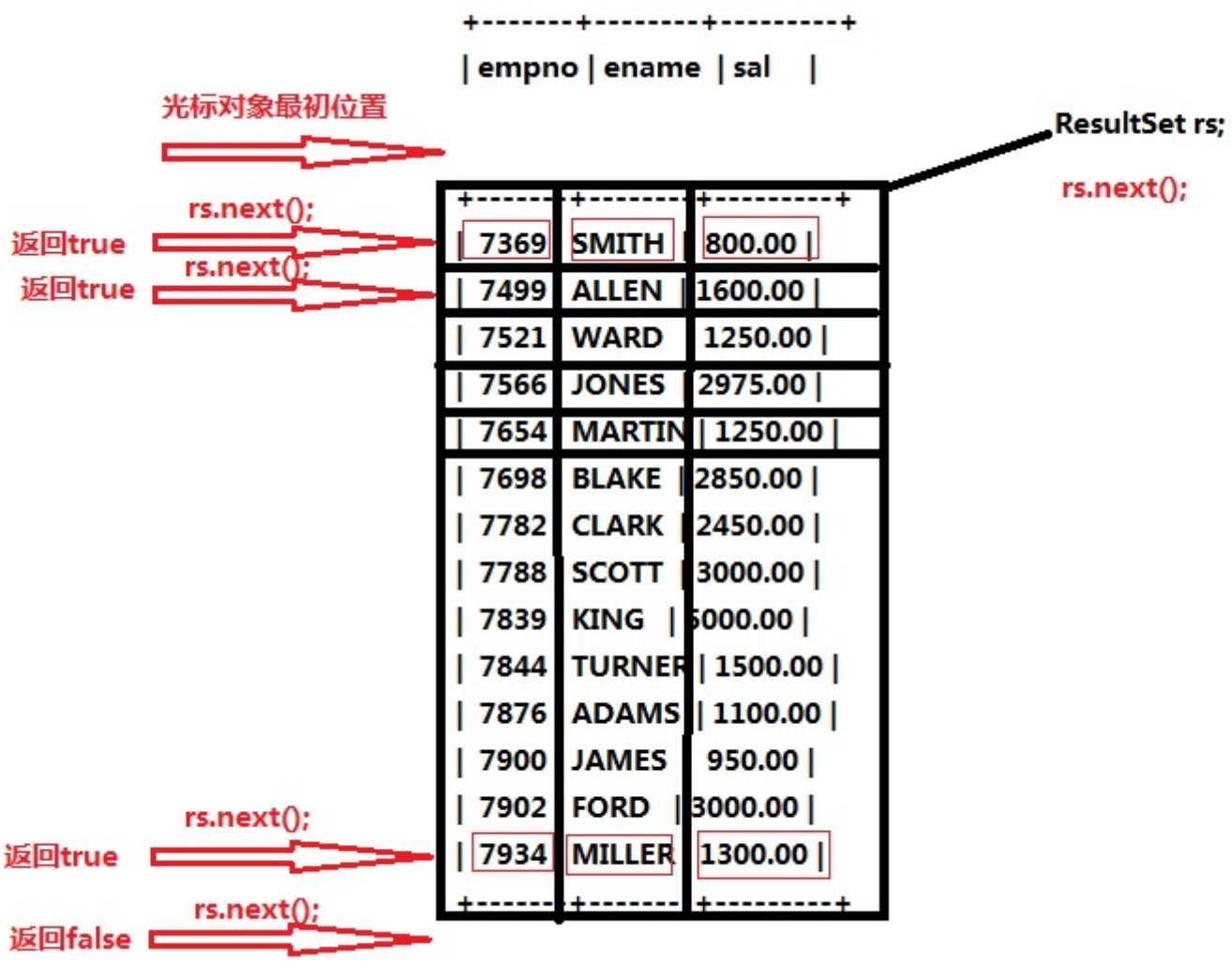
```java
/**
 * 将连接数据库的所有信息配置到配置文件中
 */
public class JDBCTest04 {
    public static void main(String[] args) {
        //使用资源绑定器绑定属性配置文件
        ResourceBundle bundle = ResourceBundle.getBundle("jdbc");
        String driver = bundle.getString("driver");
        String url = bundle.getString("url");
        String user = bundle.getString("user");
        String password = bundle.getString("password");

        Connection conn = null;
        Statement stmt = null;
        try {
            //1、注册驱动
            Class.forName(driver);
            //2、获取连接
            conn = DriverManager.getConnection(url, user, password);
            //3、获取数据库操作对象
            stmt = conn.createStatement();
            //4、执行Sql语句
            String sql = "update dept set dname = '云和增值部', loc = '无锡' where deptno = 50";
            int count = stmt.executeUpdate(sql);
            System.out.println(count == 1 ? "Success" : "Fail");
        }catch (SQLException e){
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
```

```
            e.printStackTrace();
        } finally {
            //6、释放资源
            if(stmt != null){
                try {
                    stmt.close();
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
            if (conn != null){
                try {
                    conn.close();
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
        }
    }
}
```

5.处理查询结果集

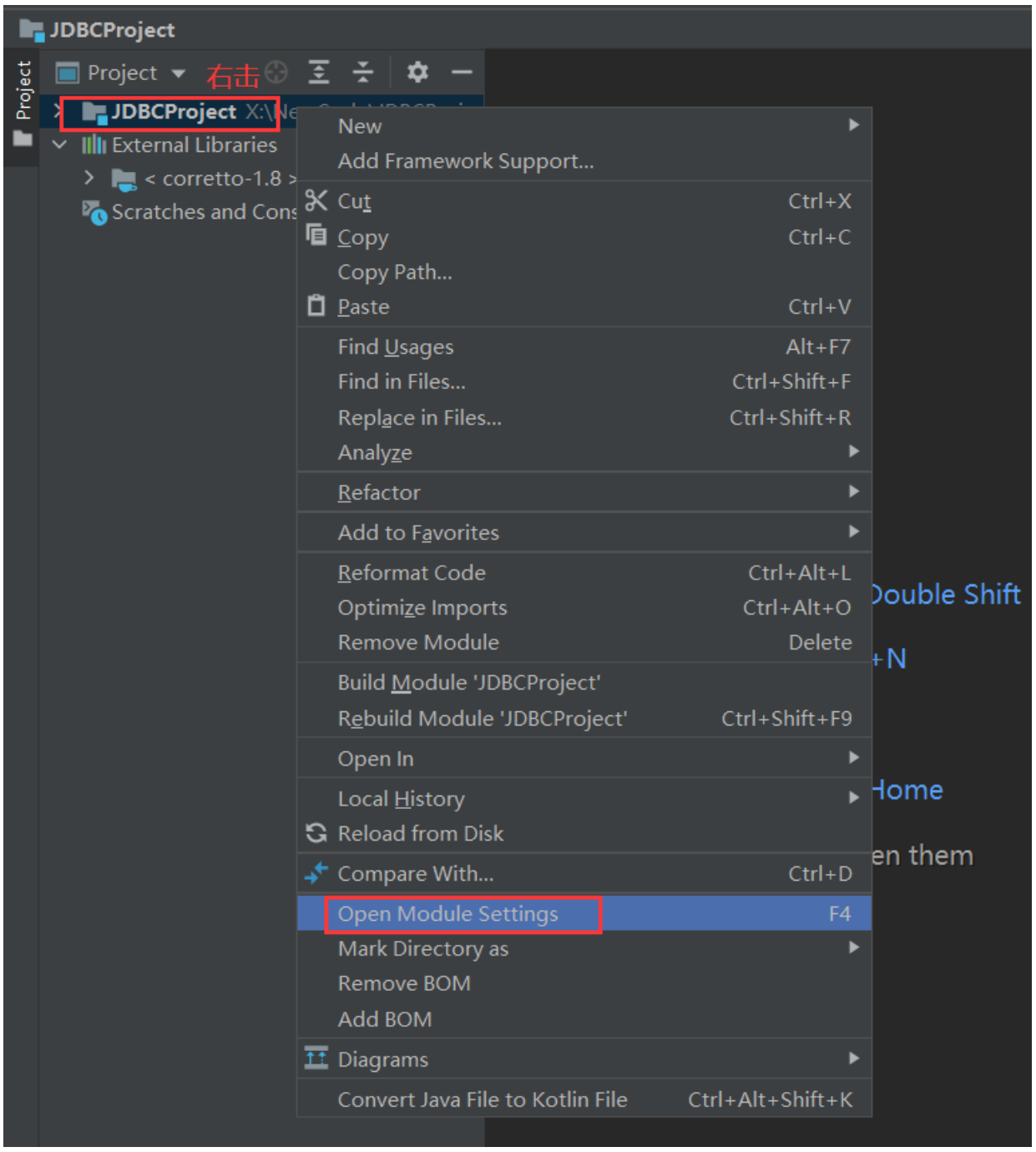
```
/**
* 处理查询结果集（遍历结果集）
*
* int executeUpdate(insert/update/delete)
* ResultSet executeQuery(select)
*/
public class JDBCTest05 {
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet resultSet = null;
        try {
            //1、注册驱动
            Class.forName("com.mysql.jdbc.Driver");
            //2、获取连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
            //3、获取数据库操作对象
            stmt = conn.createStatement();
            //4、执行SQL
            String sql = "select empno as a, ename, sal from emp";
            resultSet = stmt.executeQuery(sql);//专门执行DQL语句的方法
            //5、处理查询结果集
            /*while(resultSet.next()){
                //光标指向的行有数据 --> 取数据
                //getString()方法的特点是：不管数据库中的数据类型是什么，都以String的形式取出
                String empno = resultSet.getString(1);//JDBC中所有下标从1开始。不是从0开始
                String ename = resultSet.getString(2);
                String sal = resultSet.getString(3);
                System.out.println(empno + "," + ename + "," + sal);
            }*/
            /*while(resultSet.next()){
                //光标指向的行有数据 --> 取数据
                //这个不是以列的下标获取，以列的名字获取
                String empno = resultSet.getString("a");//注意：列名称不是表中的列名称，而是查询结果集的列名称
                String ename = resultSet.getString("ename");
                String sal = resultSet.getString("sal");
                System.out.println(empno + "," + ename + "," + sal);
            }*/
            /*while(resultSet.next()){
                //光标指向的行有数据 --> 取数据
                //以列的下标获取指定类型的数据
                int empno = resultSet.getInt(1);
                String ename = resultSet.getString(2);
                double sal = resultSet.getDouble(3);
                System.out.println(empno + "," + ename + "," + (sal + 100));
            }*/
            while(resultSet.next()){
                //光标指向的行有数据 --> 取数据
                //这个不是以列的下标获取，以列的名字获取指定类型的数据
                int empno = resultSet.getInt("a");
                String ename = resultSet.getString("ename");
                double sal = resultSet.getDouble("sal");
                System.out.println(empno + "," + ename + "," + sal);
            }
        }catch (SQLException e){
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } finally {
            //6、释放资源
            if(resultSet != null){
                try {
                    resultSet.close();
                } catch (SQLException e) {
                    e.printStackTrace();
```

```
66                  }
67              }
68              if(stmt != null){
69                  try {
70                      stmt.close();
71                  } catch (SQLException e) {
72                      e.printStackTrace();
73                  }
74              }
75              if(conn != null){
76                  try {
77                      conn.close();
78                  } catch (SQLException e) {
79                      e.printStackTrace();
80                  }
81              }
82          }
83      }
84  }
```

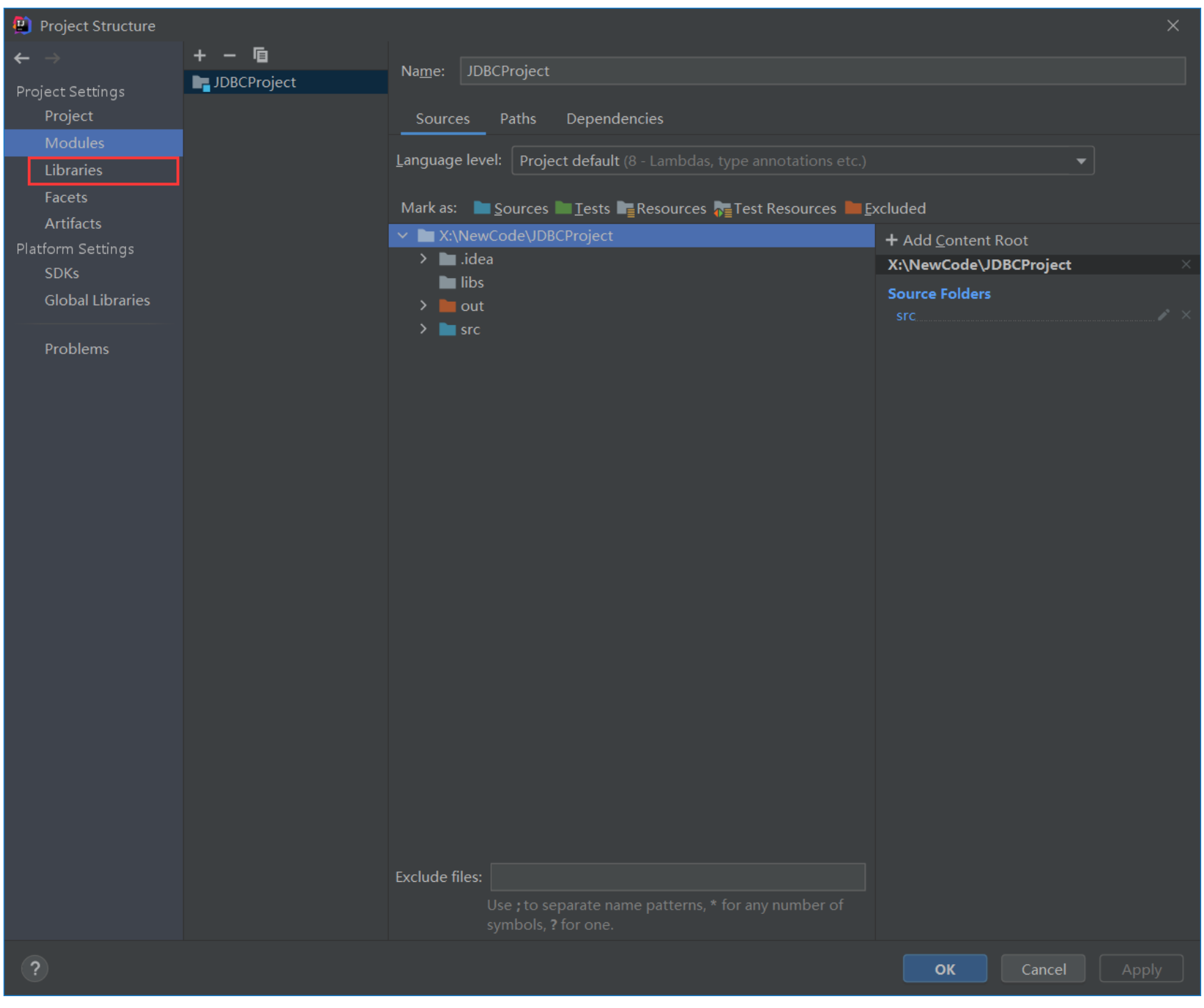


6.使用IDEA开发JDBC代码配置驱动

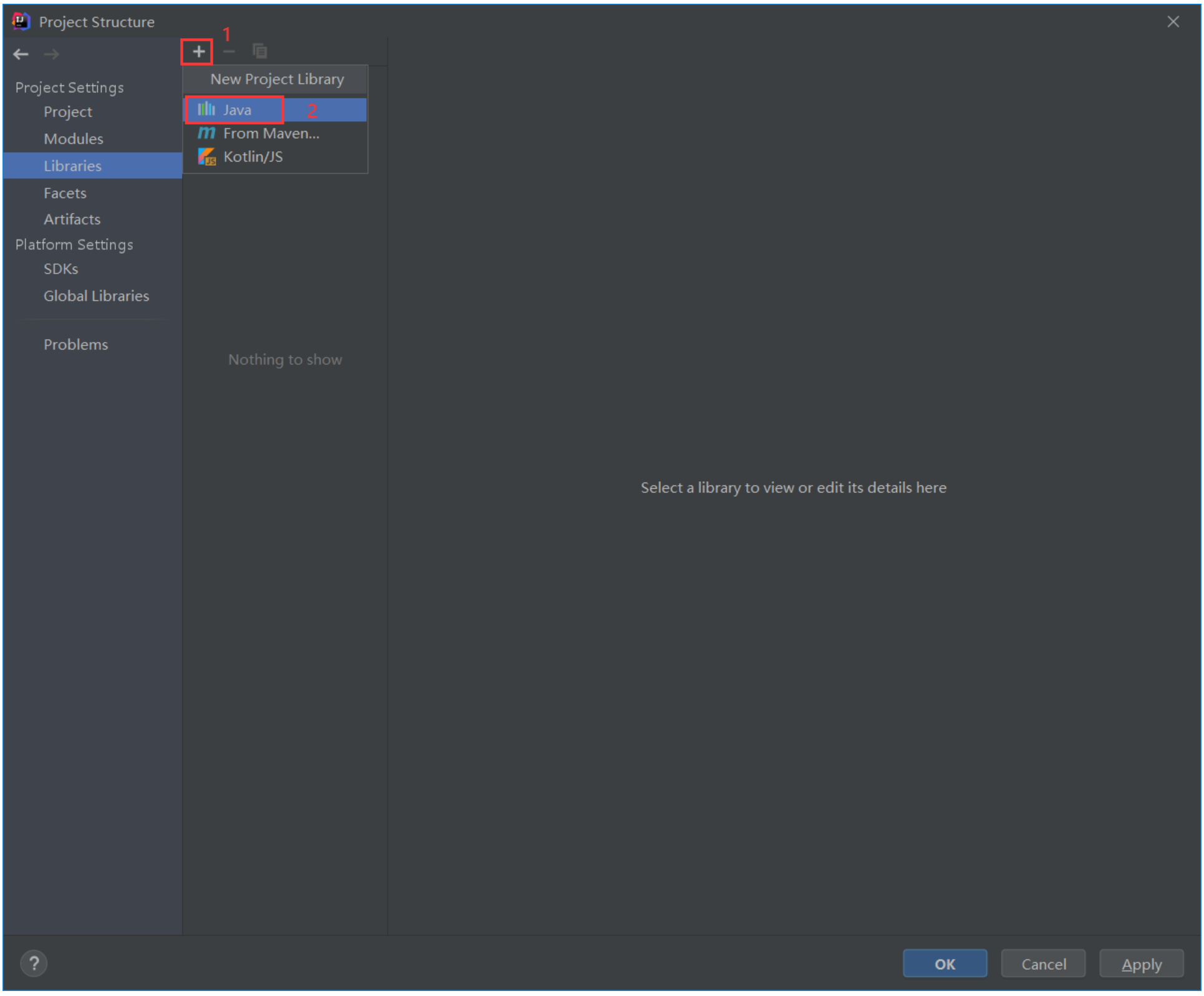
a


JDBCProject
Project
右击
JDBCProject X:\Ne
External Libraries
< corretto-1.8 >
Scratches and Cons
New
Add Framework Support...
Cut Ctrl+X
Copy Ctrl+C
Copy Path...
Paste Ctrl+V
Find Usages Alt+F7
Find in Files... Ctrl+Shift+F
Replace in Files... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Remove Module Delete
Build Module 'JDBCProject'
Rebuild Module 'JDBCProject' Ctrl+Shift+F9
Open In
Local History
Reload from Disk
Compare With... Ctrl+D
Open Module Settings F4
Mark Directory as
Remove BOM
Add BOM
Diagrams
Convert Java File to Kotlin File Ctrl+Alt+Shift+K
Double Shift
+N
Home
en them

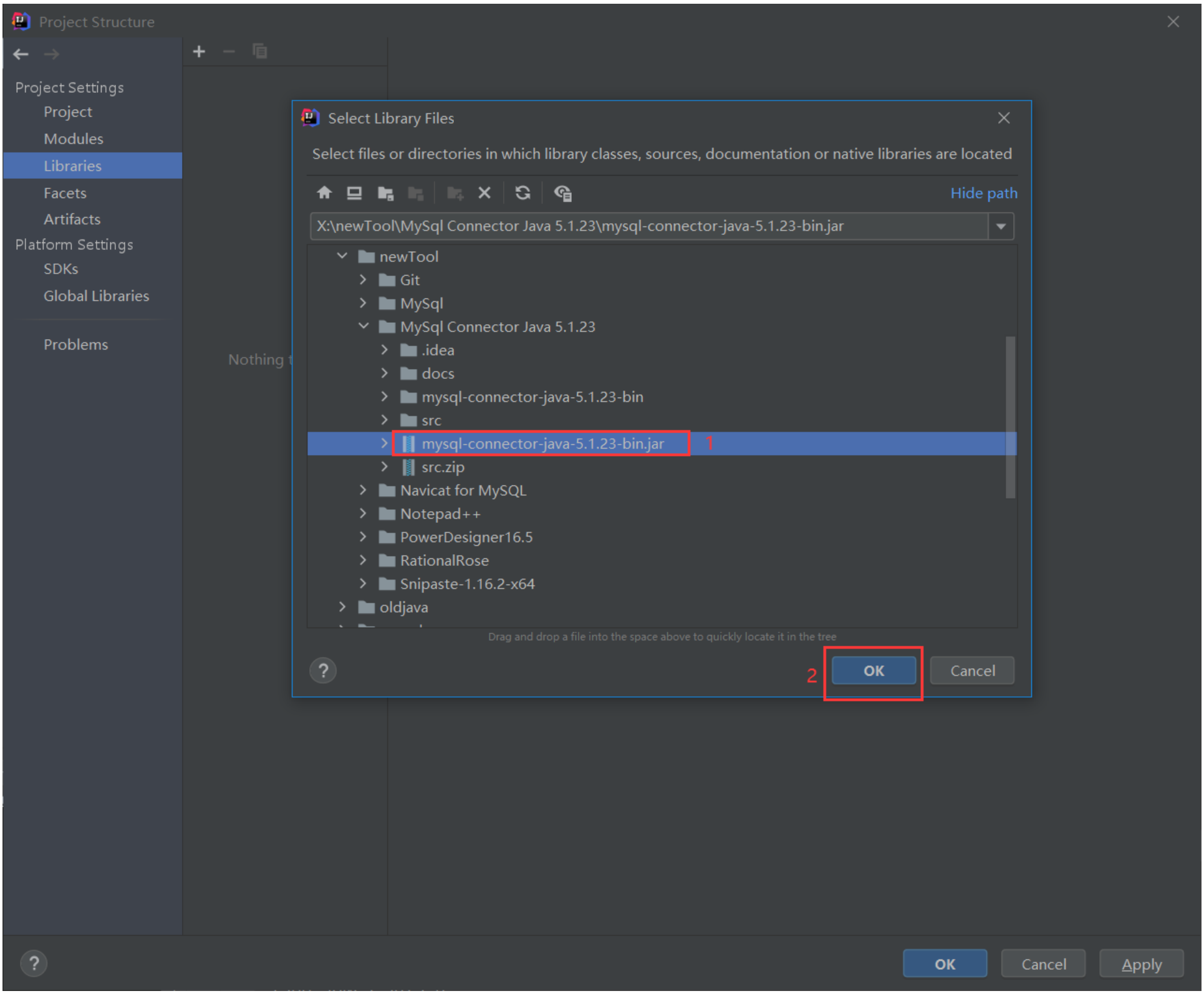
b


Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
JDBCProject
Name: JDBCProject
Sources
Paths
Dependencies
Language level: Project default (8 - Lambdas, type annotations etc.)
Mark as: Sources Tests Resources Test Resources Excluded
X:\NewCode\JDBCProject
.idea
libs
out
src
Add Content Root
X:\NewCode\JDBCProject
Source Folders
src
Exclude files:
Use ; to separate name patterns, * for any number of symbols, ? for one.
OK
Cancel
Apply

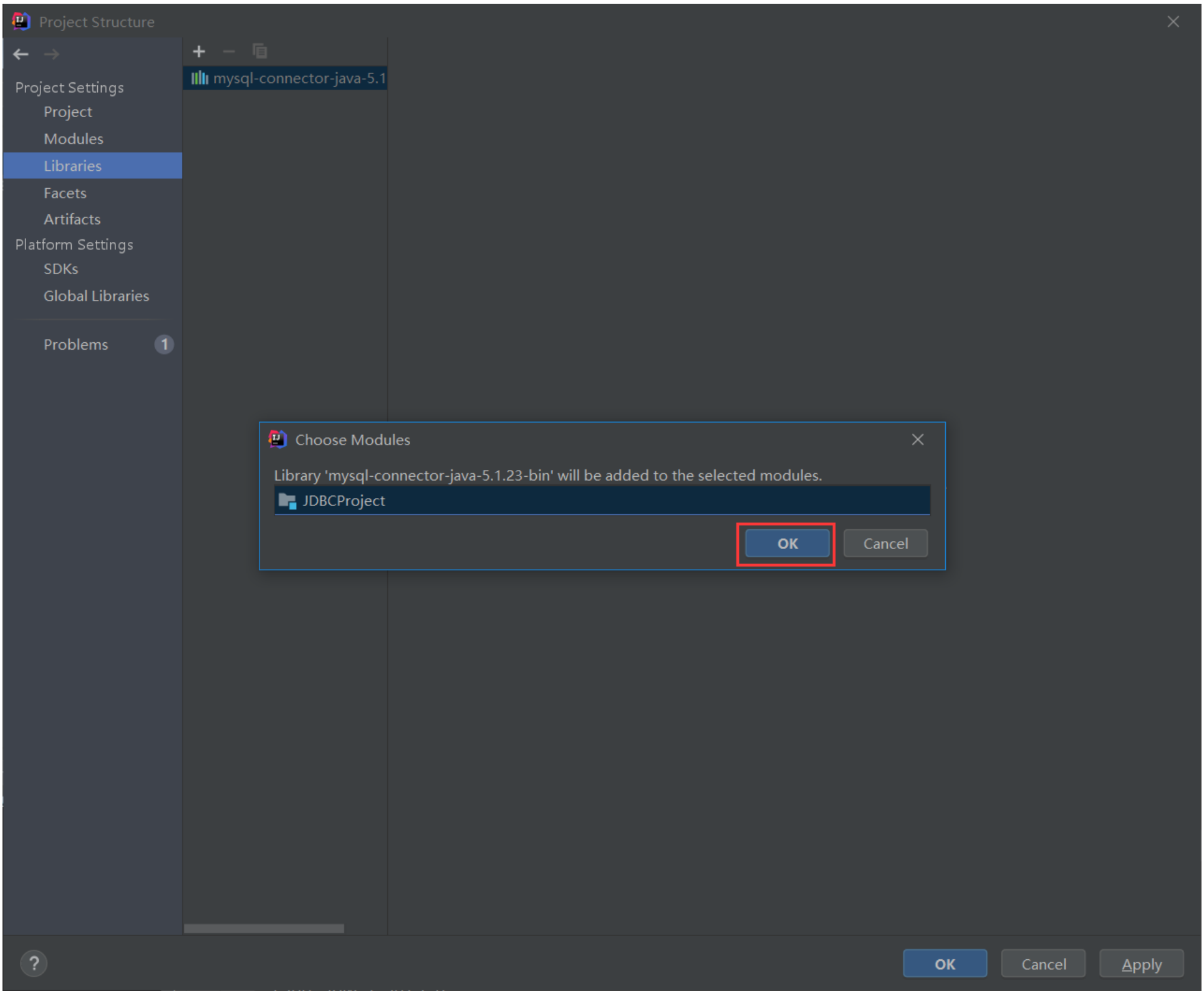
c


Project Structure
1
New Project Library
Java
2
From Maven...
Kotlin/JS
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
Nothing to show
Select a library to view or edit its details here
OK
Cancel
Apply

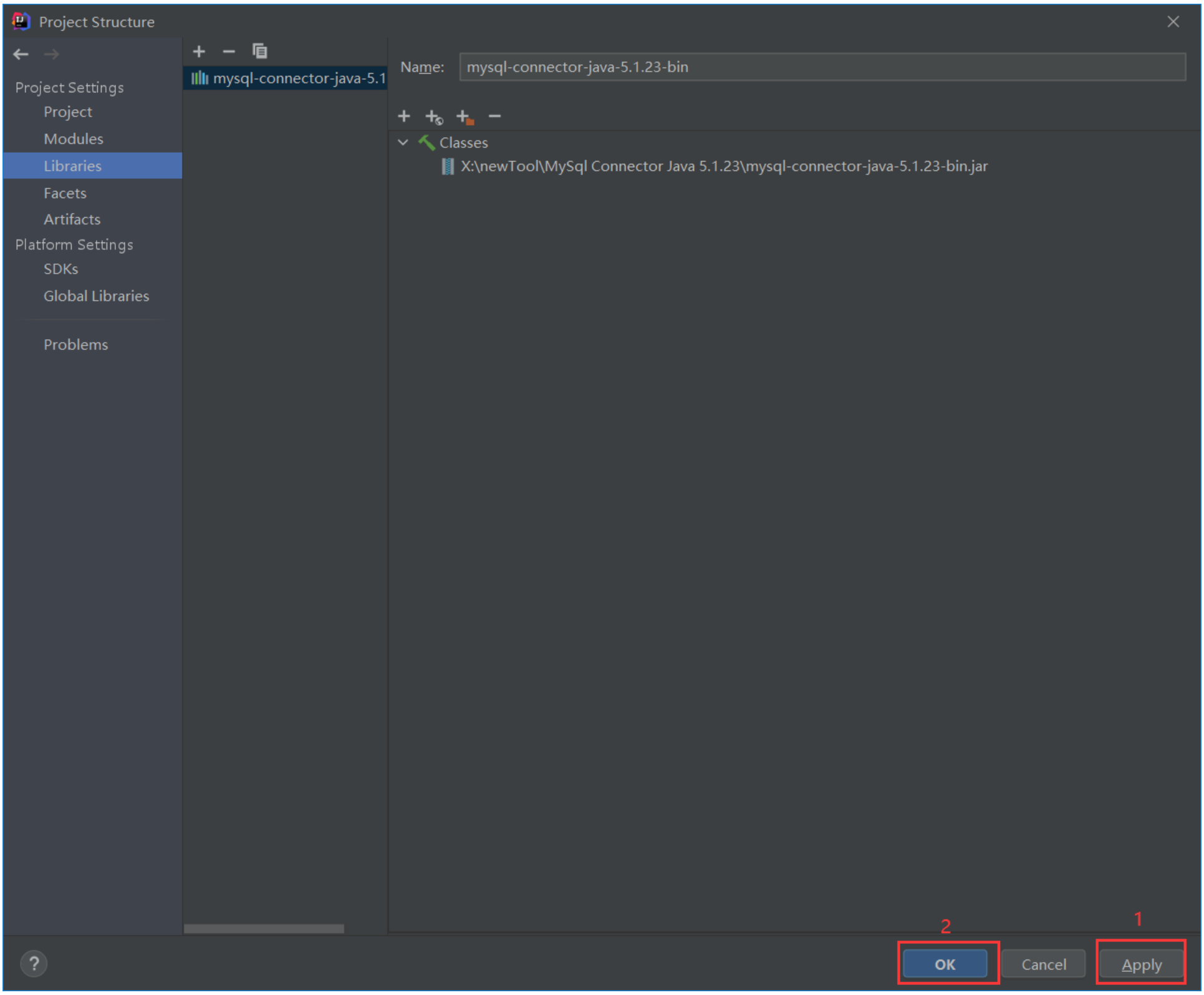
d


Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
Nothing t
Select Library Files
Select files or directories in which library classes, sources, documentation or native libraries are located
Hide path
X:\newTool\MySql Connector Java 5.1.23\mysql-connector-java-5.1.23-bin.jar
newTool
Git
MySql
MySql Connector Java 5.1.23
.idea
docs
mysql-connector-java-5.1.23-bin
src
mysql-connector-java-5.1.23-bin.jar
1
src.zip
Navicat for MySQL
Notepad++
PowerDesigner16.5
RationalRose
Snipaste-1.16.2-x64
oldjava
Drag and drop a file into the space above to quickly locate it in the tree
2
OK
Cancel
OK
Cancel
Apply

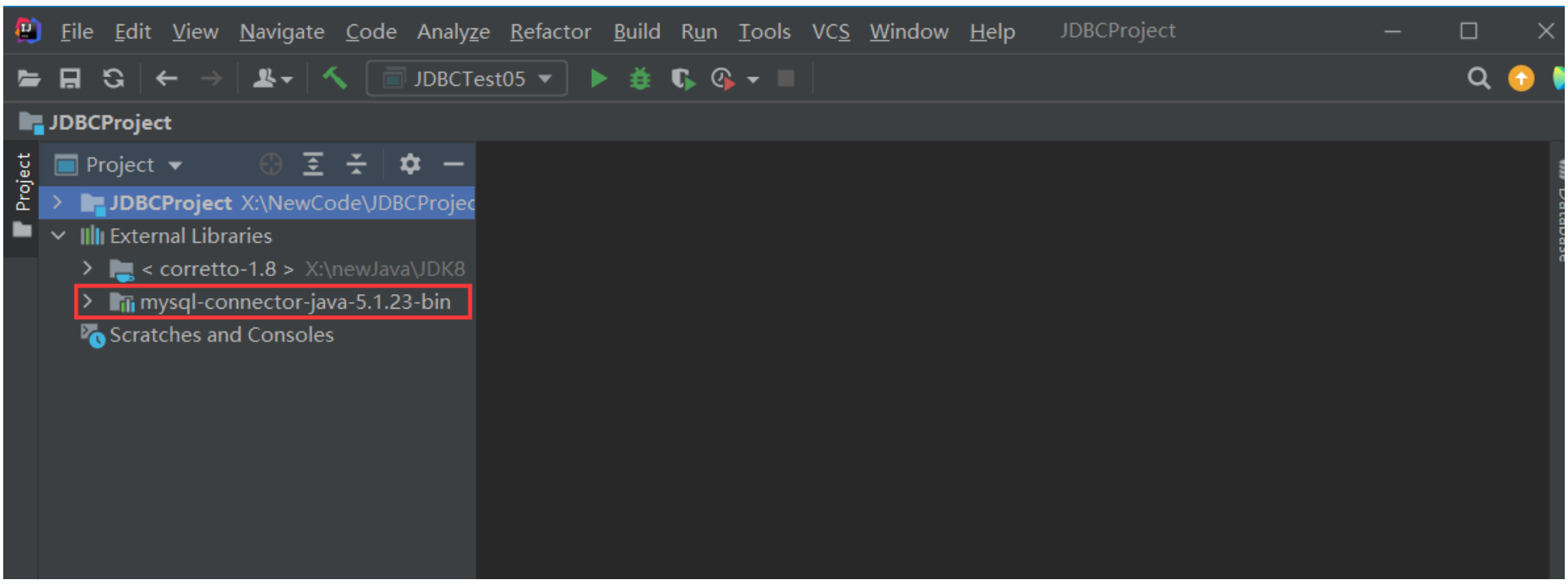
e


Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
1
mysql-connector-java-5.1
Choose Modules
Library 'mysql-connector-java-5.1.23-bin' will be added to the selected modules.
JDBCProject
OK
Cancel
OK
Cancel
Apply

f



g



7.用户登录业务

```sql
#sql脚本
drop table if exists t_user;
/*==============================================================*/
/* Table: t_user                                                */
/*==============================================================*/
create table t_user
(
   id                   bigint auto_increment,
   loginName            varchar(255),
   loginPwd             varchar(255),
   realName             varchar(255),
   primary key (id)
);

insert into t_user(loginName,loginPwd,realName) values('zhangsan', '123', '张三');
```

```
insert into t_user(loginName,loginPwd,realName) values('jack', '456', '杰克');
commit;
select * from t_user;
+----+-----------+----------+----------+
| id | loginName | loginPwd | realName |
+----+-----------+----------+----------+
|  1 | zhangsan  | 123      | 张三     |
|  2 | jack      | 456      | 杰克     |
+----+-----------+----------+----------+
```

```
/**
 * 实现功能
 *      1、需求：模拟用户登录功能的实现
 *      2、业务描述
 *          程序运行的时候，提供一个输入的入口，可以让用户输入用户名和密码
 *          用户输入用户名和密码之后，提交信息，java程序收集到用户信息
 *          Java程序连接数据库验证用户名和密码是否合法
 *          合法：显示登录成功
 *          不合法：显示登录失败
 *      3、数据的准备
 *          在实际开发中，表的设计会使用专业的建模工具，这里安装一个建模工具：PowerDesigner
 *          使用PD工具来进行数据库表的设计。
 *      4、这种代码存在的问题
 *          用户名：aaa
 *          密码：aaa' or '1'='1
 *          登录成功
 *          这种现象称为SQl注入(安全隐患)。黑客经常使用
 *      5、导致SQL注入的原因是什么？
 *          用户输入的信息中含有sql语句的关键字，并且这些关键字参与sql语句的编译过程
 *          导致sql语句的原意被扭曲，从而达到sqlzhuru
 *          select * from t_user where loginName = 'aaa' and loginPwd = 'aaa' or '1'='1';
 */
public class JDBCTest06 {
    public static void main(String[] args) {
        //初始化一个界面
        Map<String,String> userLoginInfo = initUI();
        //验证用户名和密码
        boolean loginSuccess = login(userLoginInfo);
        //输出最后结果
        System.out.println(loginSuccess ? "登录成功" : "登录失败");
    }

    /**
     * 用户登录
     * @param userLoginInfo 用户登录信息
     * @return false表示失败，true表示成功
     */
    private static boolean login(Map<String, String> userLoginInfo) {
        //打标记的意识
        boolean loginSuccess = false;

        //单独定义变量
        String loginName = userLoginInfo.get("loginName");
        String loginPwd = userLoginInfo.get("loginPwd");

        //JDBC代码
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            //1、注册驱动
            Class.forName("com.mysql.jdbc.Driver");
            //2、获取连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
            //3、获取数据库操作对象
            stmt = conn.createStatement();
            //4、执行SQL
            String sql = "select * from t_user where " +
                    "loginName = '" + loginName + "' and " +
                    "loginPwd = '" + loginPwd + "';" ;
            rs = stmt.executeQuery(sql);
            //5、处理结果集
            if(rs.next()){
                // 登录成功
                loginSuccess = true;
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            //6、释放资源
            if(rs != null){
                try {
                    rs.close();
                } catch (SQLException e) {
                    e.printStackTrace();
```

```
            }
        }
        if(stmt != null){
            try {
                stmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(conn != null){
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    return loginSuccess;
}

/**
 * 初始化用户界面
 * @return 用户输入的用户名和密码等登录信息
 */
private static Map<String, String> initUI() {
    Scanner scanner = new Scanner(System.in);

    System.out.print("用户名: ");
    String loginName = scanner.nextLine();

    System.out.print("密码: ");
    String loginPwd = scanner.nextLine();

    Map<String,String> userLoginInfo = new HashMap<>();
    userLoginInfo.put("loginName", loginName);
    userLoginInfo.put("loginPwd", loginPwd);

    return userLoginInfo;
}
}
```

8.PreparedStatement (预编译数据库操作对象)

```
/**
 * 1、解决SQL注入问题
 *      只要用户提供的信息不参与SQL语句的编译过程，问题就解决了。
 *      即使用户提供的信息中含有SQL语句的关键字，但是没有参与编译，不起作用。
 *      要想用户信息不参与SQL语句的编译，那么必须使用 java.sql.PreparedStatement
 *      PreparedStatement接口继承了java.sql.Statement
 *      PreparedStatement是属于预编译的数据库操作对象
 *      PreparedStatement的原理是：预先对SQL语句的框架进行编译，然后再给SQL语句传"值"
 * 2、测试结果
 *      用户名: aaa
 *      密码: aaa' or '1'='1
 *      登录失败
 * 3、解决SQL注入的关键是什么？
 *      用户提供的信息中即使含有SQL语句的关键字，但是这些关键字并没有参与编译。不起作用。
 * 4、对比一下 Statement 和 PreparedStatement？
 *      - Statement存在SQL注入问题，PreparedStatement解决了SQL注入问题
 *      - Statement是编译一次执行一次，PreparedStatement是编译一次可执行N次。PreparedStatement效率较高一些
 *      - PreparedStatement会在编译阶段做类型的安全检查
 *
 *      综上所述：PreparedStatement使用较多。只有极少数的情况下需要使用Statement
 * 5、什么情况下必须使用Statement呢？
 *      业务方面要求必须支持SQL注入的时候。
 *      Statement支持SQL注入，凡是业务方面要求是需要进行SQL语句拼接的，必须使用Statement
 */
public class JDBCTest07 {
    public static void main(String[] args) {
        //初始化一个界面
        Map<String,String> userLoginInfo = initUI();
        //验证用户名和密码
        boolean loginSuccess = login(userLoginInfo);
        //输出最后结果
        System.out.println(loginSuccess ? "登录成功" : "登录失败");
    }

    /**
     * 用户登录
     * @param userLoginInfo 用户登录信息
     * @return false表示失败，true表示成功
     */
    private static boolean login(Map<String, String> userLoginInfo) {
        //打标记的意识
        boolean loginSuccess = false;

        //单独定义变量
        String loginName = userLoginInfo.get("loginName");
```

```
        String loginPwd = userLoginInfo.get("loginPwd");

        //JDBC代码
        Connection conn = null;
        PreparedStatement ps = null;//这里使用PreparedStatement （预编译的数据库操作对象）
        ResultSet rs = null;
        try {
            //1、注册驱动
            Class.forName("com.mysql.jdbc.Driver");

            //2、获取连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");

            //3、获取预编译的数据库操作对象
            //SQL语句的框子。其中一个?，表示一个占位符；一个？将接收一个"值"。注意：占位符?不能使用单引号括起来
            String sql = "select * from t_user where loginName = ? and loginPwd = ?";
            //程序执行到此处，会发送sql语句框子给DBMS，然后DBMS进行sql语句的预先编译
            ps = conn.prepareStatement(sql);
            //给占位符?传值（第一个?下标是1，第二个?下标是2，JDBC中所有下标从1开始。）
            ps.setString(1, loginName);
            ps.setString(2, loginPwd);

            //4、执行SQL
            rs = ps.executeQuery();

            //5、处理结果集
            if(rs.next()){
                // 登录成功
                loginSuccess = true;
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            //6、释放资源
            if(rs != null){
                try {
                    rs.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(ps != null){
                try {
                    ps.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(conn != null){
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
        return loginSuccess;
    }

    /**
     * 初始化用户界面
     * @return 用户输入的用户名和密码等登录信息
     */
    private static Map<String, String> initUI() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("用户名: ");
        String loginName = scanner.nextLine();

        System.out.print("密码: ");
        String loginPwd = scanner.nextLine();

        Map<String,String> userLoginInfo = new HashMap<>();
        userLoginInfo.put("loginName", loginName);
        userLoginInfo.put("loginPwd", loginPwd);

        return userLoginInfo;
    }
}
```

9.演示Statement的用途

```java
/**
 * 演示Statement的用途
 */
public class JDBCTest08 {
    public static void main(String[] args) {
        // 用户在控制台输入desc就是降序，输入asc就是升序
        Scanner s = new Scanner(System.in);
        System.out.println("请输入desc或asc，desc表示降序，asc表示升序");
        System.out.print("请输入：");
        String keyWords = s.nextLine();

        // 执行SQL
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            // 1、注册驱动
            Class.forName("com.mysql.jdbc.Driver");
            // 2、获取连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
            // 3、获取数据库操作对象
            stmt = conn.createStatement();
            // 4、执行SQL语句
            String sql = "select ename from emp order by ename " + keyWords;
            rs = stmt.executeQuery(sql);
            // 5、遍历结果集
            while (rs.next()){
                System.out.println(rs.getString("ename"));
            }
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // 6、释放资源
            if(rs != null){
                try {
                    rs.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(stmt != null){
                try {
                    stmt.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(conn != null){
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

10.PreparedStatement完成增删改

```java
/**
 * PreparedStatement完成insert delete update
 */
public class JDBCTest09 {
    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement ps = null;
        try {
            //1、注册驱动
            Class.forName("com.mysql.jdbc.Driver");
            //2、获取连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
            //3、获取预编译的数据库操作对象
            /*
            // insert
            String sql = "insert into t_user(loginName,loginPwd,realName) values(?,?,?)";
            ps = conn.prepareStatement(sql);
            ps.setString(1, "xukang");
            ps.setString(2, "123456");
            ps.setString(3, "徐康");
            */

            /*
```

```
            // update
            String sql = "update t_user set loginPwd = ?,realName = ? where id = ?";
            ps = conn.prepareStatement(sql);
            ps.setString(1, "789789");
            ps.setString(2, "虚空");
            ps.setInt(3, 3);
            */

            // delete
            String sql = "delete from t_user where loginName = ? and loginPwd = ?";
            ps = conn.prepareStatement(sql);
            ps.setString(1, "xukang");
            ps.setString(2, "789789");
            //4、执行SQL
            int count = ps.executeUpdate();
            System.out.println(count);
            //5、处理结果集
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            //6、释放资源
            if(ps != null){
                try {
                    ps.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(conn != null){
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

11.JDBC事务机制

```
/**
 * JDBC事务机制
 *      1、JDBC中的事务是自动提交的，什么是自动提交？
 *           只要执行任意一条DML语句，则自动提交一次，这是JDBC默认的事务行为。
 *           但是在实际的业务当中，通常都是N条DML语句共同联合才能完成的，必须
 *           保证他们这些DML语句在同一个事务中同时完成或者同时失败。
 *      2、以下程序先来验证一下JDBC的事务是否是自动提交机制！
 *           测试结果：JDBC中只要执行任意一条DML语句，就提交一次（断点调试）
 */
public class JDBCTest10 {
    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement ps = null;
        try {
            //1、注册驱动
            Class.forName("com.mysql.jdbc.Driver");
            //2、获取连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
            //3、获取预编译的数据库操作对象
            String sql1 = "update dept set dname = ? where deptno = ?";
            ps = conn.prepareStatement(sql1);

            // 第一次给占位符传值
            ps.setString(1, "X部门");
            ps.setInt(2, 30);
            int count = ps.executeUpdate();// 执行第一条UPDATE语句
            System.out.println(count);

            // 重新给占位符传值
            ps.setString(1, "Y部门");
            ps.setInt(2, 20);
            count = ps.executeUpdate();// 执行第二条UPDATE语句
            System.out.println(count);

        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            //6、释放资源
            if(ps != null){
                try {
                    ps.close();
                } catch (SQLException e) {
                    e.printStackTrace();
```

```
                }
            }
            if(conn != null){
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

```
/**
 * sql脚本：
 *      drop table if exists t_act;
 *      create table t_act(
 *          actno     bigint,
 *          balance   double(7,2) //7表示有效数字的个数，2表示小数位的个数
 *      );
 *      insert into t_act(actno,balance) values(111,20000);
 *      insert into t_act(actno,balance) values(222,0);
 *      commit;
 *      select * from t_act;
 *
 * 重点：三行代码
 *      conn.setAutoCommit(false);//开启事务
 *      conn.commit();//手动提交事务
 *      conn.rollback();//手动回滚事务
 */
public class JDBCTest11 {
    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement ps = null;
        try {
            //1、注册驱动
            Class.forName("com.mysql.jdbc.Driver");
            //2、获取连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");

            //将自动提交机制改为手动提交
            conn.setAutoCommit(false);//开启事务

            //3、获取预编译的数据库操作对象
            String sql = "update t_act set balance = ? where actno = ?";
            ps = conn.prepareStatement(sql);
            //给?传值
            ps.setDouble(1, 10000);
            ps.setInt(2, 111);
            int count = ps.executeUpdate();

            //String s = null;
            //s.toString();// 空指针异常 会导致111用户平白无故损失10000元

            ps.setDouble(1, 10000);
            ps.setInt(2, 222);
            count += ps.executeUpdate();

            System.out.println(count == 2 ? "转账成功" : "转账失败");

            //程序能够走到这里说明以上程序没有异常，事务结束，手动提交数据
            conn.commit();//提交事务
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            if(conn != null){
                try {
                    conn.rollback();//回滚事务
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
            e.printStackTrace();
        } finally {
            //6、释放资源
            if(ps != null){
                try {
                    ps.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
            if(conn != null){
                try {
                    conn.close();
                } catch (SQLException e) {
                    e.printStackTrace();
```

```
                }
            }
        }
    }
}
```

12.JDBC工具类的封装

```
/**
 * JDBC工具类，简化JDBC编程
 */
public class JDBCUtil {
    /**
     * 工具类的构造方法都是私有的
     * 因为工具类当中的方法都是静态的，不需要new对象，直接采用类名调用
     */
    private JDBCUtil(){}

    // 静态代码块在类加载时执行，并且只执行一次
    // 注册驱动
    static {
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    /**
     * 获取数据库连接对象
     * @return 连接对象
     * @throws SQLException
     */
    public static Connection getConnection() throws SQLException {
        Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
        return conn;
    }

    /**
     * 关闭资源
     * @param conn 连接对象
     * @param ps   数据库操作对象
     * @param rs   结果集
     */
    public static void close(Connection conn, Statement ps, ResultSet rs){
        if(rs != null){
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(ps != null){
            try {
                ps.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if(conn != null){
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
/**
 * 两个任务：
 *      1、测试JDBCUtil工具类是否好用
 *      2、模糊查询怎么写？
 */
public class JDBCTest12 {
    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            //获取连接
            conn = JDBCUtil.getConnection();
            //获取预编译的数据库操作对象
            //错误的写法
            /*
            String sql = "select ename from emp where ename like '_?%'";
            ps = conn.prepareStatement(sql);
```

```
            ps.setString(1, "A");
            */
            //正确的写法
            String sql = "select ename from emp where ename like ?";
            ps = conn.prepareStatement(sql);
            ps.setString(1, "_A%");//第二个字母为A的
            //执行SQL
            rs = ps.executeQuery();
            //遍历结果集
            while (rs.next()){
                System.out.println(rs.getString("ename"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // 释放资源
            JDBCUtil.close(conn, ps, rs);
        }
    }
}
```

13.悲观锁和乐观锁的概念

```
ename       job          sal                 version
-----------------------------------------------------------------
| BLAKE | MANAGER | 2850.00 |                1.1
```

事务1-->读取到版本号1.1

事务2--->读取到版本号1.1

其中事务1先修改了，修改之后看了版本号是1.1，于是提交修改的数据，将版本号修改为1.2

其中事务2后修改的，修改之后准备提交的时候，发现版本号是1.2，和它最初读的版本号不一致。回滚。

悲观锁：事务必须排队执行。数据锁住了，不允许并发。（行级锁：select后面添加for update）

乐观锁：支持并发，事务也不需要排队，只不过需要一个版本号。

```
/**
 * 这个程序开启一个事务，这个事务专门进行查询，并且使用行级锁（悲观锁）,锁住相关的记录
 */
public class JDBCTest13 {
    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            conn = JDBCUtil.getConnection();
            //开启事务
            conn.setAutoCommit(false);

            String sql = "select ename, job, sal from emp where job = ? for update";
            ps = conn.prepareStatement(sql);
            ps.setString(1, "MANAGER");

            rs = ps.executeQuery();
            while (rs.next()){
                System.out.println(rs.getString("ename") + ", " + rs.getString("job") + ", " + rs.getDouble("sal"));
            }
            //提交事务（事务结束）
            conn.commit();
        } catch (SQLException e) {
            if(conn != null){
                try {
                    //回滚事务（事务结束）
                    conn.rollback();
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
            e.printStackTrace();
        } finally {
            JDBCUtil.close(conn, ps, rs);
        }
    }
}
```

```
/**
 * 这个程序负责修改被锁定的记录
 */
```

```java
public class JDBCTest14 {
    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement ps = null;
        try {
            conn = JDBCUtil.getConnection();
            conn.setAutoCommit(false);

            String sql = "update emp set sal = sal * 1.1 where job = ?";
            ps = conn.prepareStatement(sql);
            ps.setString(1, "MANAGER");

            int count = ps.executeUpdate();
            System.out.println(count);

            conn.commit();
        } catch (SQLException e) {
            if(conn != null){
                try {
                    conn.rollback();
                } catch (SQLException throwables) {
                    throwables.printStackTrace();
                }
            }
            e.printStackTrace();
        }finally {
            JDBCUtil.close(conn, ps, null);
        }
    }
}
```