

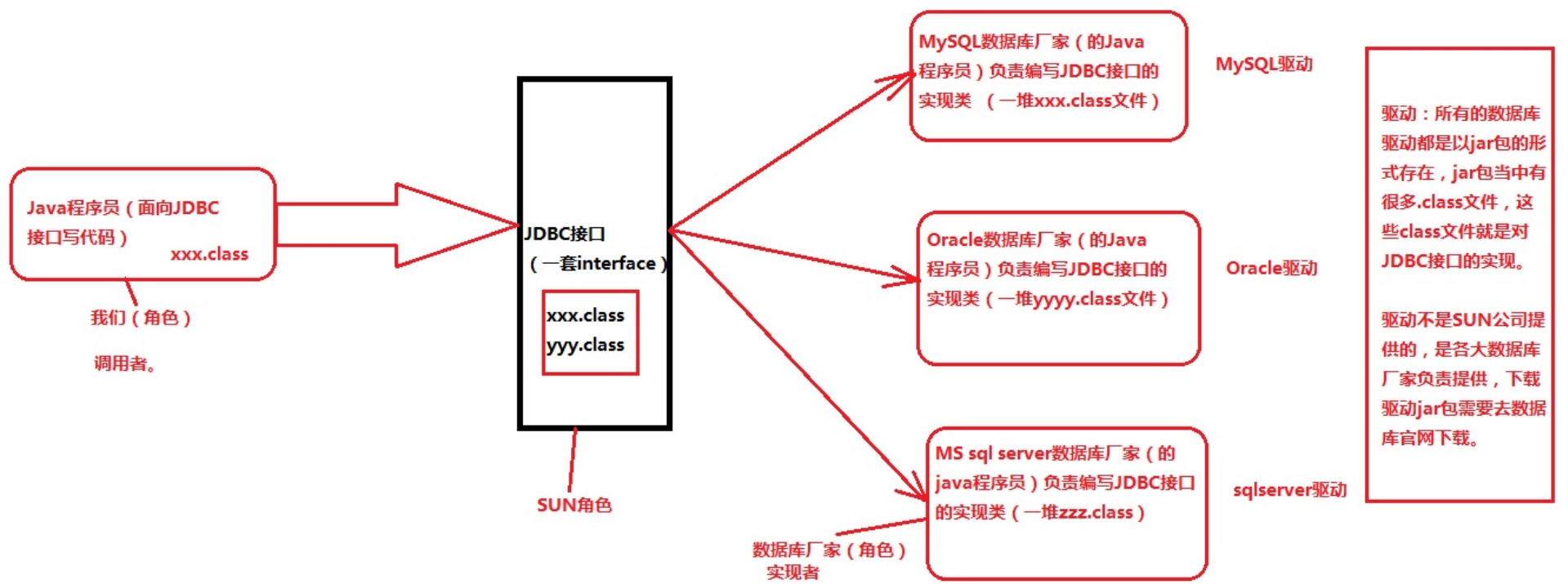
JDBC

1、JDBC是什么？

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

2、JDBC的本质是什么？

1 JDBC是SUN公司制定的一套接口（interface）
2 java.sql.*; (这个软件包下有很多接口。)
3
4 接口都有调用者和实现者。
5 面向接口调用、面向接口写实现类，这都属于面向接口编程。
6
7 为什么要面向接口编程？
8 解耦合：降低程序的耦合度，提高程序的扩展力。
9 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）
10 建议：
11 Animal a = new Cat();
12 Animal a = new Dog();
13 // 喂养的方法
14 public void feed(Animal a){ // 面向父类型编程。
15 }
16 不建议：
17 Dog d = new Dog();
18 Cat c = new Cat();
19
20 思考：为什么SUN制定一套JDBC接口呢？
21 因为每一个数据库的底层实现原理都不一样。
22 Oracle数据库有自己的原理。
23 MySQL数据库也有自己的原理。
24 MS SqlServer数据库也有自己的原理。
25
26 每一个数据库产品都有自己独特的实现原理。
27
28 JDBC的本质到底是什么？
29 一套接口。



3、模拟实现JDBC本质

1 /**
2 * SUN公司负责制定这套JDBC接口
3 */
4 public interface JDBC {
5 /**
6 * 连接数据库的方法
7 */
8 void getConnection();
9 }

1 /**
2 * MySQL数据库厂家负责编写JDBC接口的实现类
3 * 实现类被称为驱动 （MySQL驱动）
4 */

```
5 public class MySql implements JDBC {
6     @Override
7     public void getConnection() {
8         /**
9          * 具体这里的代码怎么写，对于Java程序员来说没关系
10         * 这段代码涉及到mysql底层数据库的实现原理
11         */
12         System.out.println("连接MYSQL数据库成功！");
13     }
14 }
15
16 /**
17  * Oracle数据库厂家负责编写JDBC接口的实现类
18  * 实现类被称为驱动 （Oracle驱动）
19  */
20 public class Oracle implements JDBC {
21     @Override
22     public void getConnection() {
23         /**
24          * 这段代码涉及到Oracle底层数据库的实现原理
25          */
26         System.out.println("连接Oracle数据库成功！");
27     }
28 }
29
30 /**
31  * SqlServer数据库厂家负责编写JDBC接口的实现类
32  * 实现类被称为驱动 （SqlServer驱动）
33  */
34 public class SqlServer implements JDBC {
35     @Override
36     public void getConnection() {
37         /**
38          * 这段代码涉及到SqlServer底层数据库的实现原理
39          */
40         System.out.println("连接SqlServer数据库成功！");
41     }
42 }
```

```
1 /**
2  * Java程序员角色
3  * 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。
4  * 面向接口编程，面向抽象编程，不要面向具体编程。
5  */
6 public class JavaProgrammer {
7     public static void main(String[] args) throws Exception {
8         //JDBC jdbc = new MySql();
9         //JDBC jdbc = new SqlServer();
10        //JDBC jdbc = new Oracle();
11
12        // 创建对象可以通过反射机制
13        //Class c = Class.forName("com.sql.database_Manufacturer.Oracle");
14        //Class c = Class.forName("com.sql.database_Manufacturer.MySql");
15        //Class c = Class.forName("com.sql.database_Manufacturer.SqlServer");
16
17        //资源绑定器
18        ResourceBundle bundle = ResourceBundle.getBundle("JDBC");
19        String className = bundle.getString("Oracle");
20        Class c = Class.forName(className);
21
22        JDBC jdbc = (JDBC) c.newInstance();
23
24        // 以下代码都是面向接口调用方法，不需要修改
25        jdbc.getConnection();
26    }
27 }
```

```
1 #属性配置文件 JDBC.properties
2 Oracle=Oracle
3 MySql=MySql
4 SqlServer=SqlServer
```

4、JDBC开发前的准备工作

```
1 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。
2 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
3
4 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。
5 IDEA有自己的配置方式。
```

5、JDBC编程六步

1	4、JDBC编程六步（需要背会）
2	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）
3	
4	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。）
5	
6	第三步：获取数据库操作对象（专门执行sql语句的对象）
7	
8	第四步：执行SQL语句（DQL DML....）
9	
10	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。）
11	
12	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。）

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

6、JDBC代码

1.编程六步

1	package com.java.jdbc;
2	import com.mysql.jdbc.Driver;
3	import java.sql.Connection;
4	import java.sql.DriverManager;
5	import java.sql.SQLException;
6	import java.sql.Statement;
7	/**
8	* JDBC 编程六步
9	*/
10	public class JDBCTest01 {
11	public static void main(String[] args) {
12	Connection connection = null;
13	Statement statement = null;
14	try {
15	//第一步：注册驱动
16	Driver driver = new com.mysql.jdbc.Driver();//Mysql的驱动
17	//Driver driver = new oracle.jdbc.driver.OracleDriver();//oracle的驱动
18	DriverManager.registerDriver(driver);
19	
20	//第二步：获取连接
21	/*
22	url:统一资源定位符（网络中某个资源的绝对路径）
23	https://www.baidu.com/ 这就是URL
24	URL包括哪几部分？
25	协议 IP Port 资源名
26	
27	https://182.61.200.7:80/index.html
28	https:// 通信协议
29	182.61.200.7 服务器IP地址
30	80 服务器上软件的端口
31	index.html 是服务器上某个资源名
32	
33	jdbc:mysql://127.0.0.1:3306/bjpowernode
34	jdbc:mysql:// 协议
35	127.0.0.1 IP地址
36	3306 mysql数据库端口号
37	bjpowernode 具体的数据库实例名
38	
39	说明：localhost 和 127.0.0.1 都是本地IP地址
40	
41	jdbc:mysql://192.168.151.27:3306/bjpowernode
42	
43	什么是通信协议？有什么用？
44	通信协议是通信之前就提前定好的数据传送格式。
45	数据包具体怎么传数据，格式提前定好的。
46	
47	oracle的URL： jdbc:oracle:thin:@localhost:1521:orcl
48	*/
49	String url = "jdbc:mysql://127.0.0.1:3306/bjpowernode";
50	String user = "root";
51	String password = "111";
52	connection = DriverManager.getConnection(url, user, password);
53	//数据库连接对象： com.mysql.jdbc.JDBC4Connection@14514713
54	System.out.println("数据库连接对象： " + connection);
55	

```

56         //第三步：获取数据库操作对象(Statement专门执行sql语句的)
57         statement = connection.createStatement();
58
59         //第四步：执行SQL语句
60         String sql = "insert into dept(deptno, dname, loc) values(50, '人事部', '北京')";
61         //专门执行DML语句的(insert delete update)
62         //返回值是“影响数据库中的记录条数”
63         int count = statement.executeUpdate(sql);
64         System.out.println(count == 1 ? "Success!" : "Fail");
65
66         //第五步：处理查询结果集
67     } catch (SQLException e) {
68         e.printStackTrace();
69     }finally {
70         //第六步：释放资源
71         // 为了保证资源一定释放，在finally语句块中关闭资源
72         // 并且要遵循从小到大依次关闭
73         // 分别对其try...catch
74         if(statement != null){
75             try {
76                 statement.close();
77             } catch (SQLException throwables) {
78                 throwables.printStackTrace();
79             }
80         }
81         if(connection != null){
82             try {
83                 connection.close();
84             } catch (SQLException throwables) {
85                 throwables.printStackTrace();
86             }
87         }
88     }
89 }
90 }

```

2.JDBC完成delete update

```

1  package com.java.jdbc;
2  import java.sql.*;
3  /**
4   * JDBC完成delete update
5   */
6  public class JDBCTest02 {
7      public static void main(String[] args) {
8          Connection conn = null;
9          Statement stmt = null;
10         try {
11             //1、注册驱动
12             DriverManager.registerDriver(new com.mysql.jdbc.Driver());
13
14             //2、获取连接
15             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
16
17             //3、获取数据库操作对象
18             stmt = conn.createStatement();
19
20             //4、执行Sql语句
21             //String sql = "delete from dept where deptno = 40";
22             String sql = "update dept set dname = '销售部', loc = '盐城' where deptno = 20";
23             int count = stmt.executeUpdate(sql);
24             System.out.println(count == 1 ? "Success" : "Fail");
25         }catch (SQLException e){
26             e.printStackTrace();
27         }finally {
28             //6、释放资源
29             if(stmt != null){
30                 try {
31                     stmt.close();
32                 } catch (SQLException throwables) {
33                     throwables.printStackTrace();
34                 }
35             }
36             if (conn != null){
37                 try {
38                     conn.close();
39                 } catch (SQLException throwables) {
40                     throwables.printStackTrace();
41                 }
42             }
43         }
44     }
45 }

```

3.注册驱动的另一方式

```
1  /**
2   * 注册驱动的另一方式
3   */
4  public class JDBCTest03 {
5      public static void main(String[] args) {
6          try {
7              /**
8               //注册驱动
9               //这是注册驱动的第一种写法
10              DriverManager.registerDriver(new com.mysql.jdbc.Driver());
11              //获取连接
12              Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
13              System.out.println(conn);//com.mysql.jdbc.JDBC4Connection@14514713
14              */
15
16              //注册驱动
17              //这是注册驱动的第二种写法:常用写法
18              //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
19              //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。
20              Class.forName("com.mysql.jdbc.Driver");
21              //获取连接
22              Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
23              System.out.println(conn);//com.mysql.jdbc.JDBC4Connection@443b7951
24
25          }catch (SQLException e){
26              e.printStackTrace();
27          } catch (ClassNotFoundException e) {
28              e.printStackTrace();
29          }
30      }
31  }
32
33  package com.mysql.jdbc;
34  import java.sql.DriverManager;
35  import java.sql.SQLException;
36  public class Driver extends NonRegisteringDriver implements java.sql.Driver {
37      public Driver() throws SQLException {
38      }
39
40      static {
41          try {
42              DriverManager.registerDriver(new Driver());
43          } catch (SQLException var1) {
44              throw new RuntimeException("Can't register driver!");
45          }
46      }
47  }
```

4.将连接数据库的所有信息配置到配置文件中

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

```
1  /**
2   * 将连接数据库的所有信息配置到配置文件中
3   */
4  public class JDBCTest04 {
5      public static void main(String[] args) {
6          //使用资源绑定器绑定属性配置文件
7          ResourceBundle bundle = ResourceBundle.getBundle("jdbc");
8          String driver = bundle.getString("driver");
9          String url = bundle.getString("url");
10         String user = bundle.getString("user");
11         String password = bundle.getString("password");
12
13         Connection conn = null;
14         Statement stmt = null;
15         try {
16             //1、注册驱动
17             Class.forName(driver);
18             //2、获取连接
19             conn = DriverManager.getConnection(url, user, password);
20             //3、获取数据库操作对象
21             stmt = conn.createStatement();
22             //4、执行Sql语句
23             String sql = "update dept set dname = '云和增值部', loc = '无锡' where deptno = 50";
24             int count = stmt.executeUpdate(sql);
25             System.out.println(count == 1 ? "Success" : "Fail");
26         }catch (SQLException e){
27             e.printStackTrace();
28         } catch (ClassNotFoundException e) {
```

```

29         e.printStackTrace();
30     } finally {
31         //6、释放资源
32         if(stmt != null){
33             try {
34                 stmt.close();
35             } catch (SQLException throwables) {
36                 throwables.printStackTrace();
37             }
38         }
39         if (conn != null){
40             try {
41                 conn.close();
42             } catch (SQLException throwables) {
43                 throwables.printStackTrace();
44             }
45         }
46     }
47 }
48 }

```

5.处理查询结果集

```

1  /**
2  * 处理查询结果集（遍历结果集）
3  *
4  * int executeUpdate(insert/update/delete)
5  * ResultSet executeQuery(select)
6  */
7  public class JDBCTest05 {
8      public static void main(String[] args) {
9          Connection conn = null;
10         Statement stmt = null;
11         ResultSet resultSet = null;
12         try {
13             //1、注册驱动
14             Class.forName("com.mysql.jdbc.Driver");
15             //2、获取连接
16             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
17             //3、获取数据库操作对象
18             stmt = conn.createStatement();
19             //4、执行SQL
20             String sql = "select empno as a, ename, sal from emp";
21             resultSet = stmt.executeQuery(sql);//专门执行DQL语句的方法
22             //5、处理查询结果集
23             /*while(resultSet.next()){
24                 //光标指向的行有数据 --> 取数据
25                 //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
26                 String empno = resultSet.getString(1);//JDBC中所有下标从1开始。不是从0开始
27                 String ename = resultSet.getString(2);
28                 String sal = resultSet.getString(3);
29                 System.out.println(empno + "," + ename + "," + sal);
30             }*/
31             /*while(resultSet.next()){
32                 //光标指向的行有数据 --> 取数据
33                 //这个不是以列的下标获取，以列的名字获取
34                 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
35                 String ename = resultSet.getString("ename");
36                 String sal = resultSet.getString("sal");
37                 System.out.println(empno + "," + ename + "," + sal);
38             }*/
39             /*while(resultSet.next()){
40                 //光标指向的行有数据 --> 取数据
41                 //以列的下标获取指定类型的数据
42                 int empno = resultSet.getInt(1);
43                 String ename = resultSet.getString(2);
44                 double sal = resultSet.getDouble(3);
45                 System.out.println(empno + "," + ename + "," + (sal + 100));
46             }*/
47             while(resultSet.next()){
48                 //光标指向的行有数据 --> 取数据
49                 //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
50                 int empno = resultSet.getInt("a");
51                 String ename = resultSet.getString("ename");
52                 double sal = resultSet.getDouble("sal");
53                 System.out.println(empno + "," + ename + "," + sal);
54             }
55         } catch (SQLException e) {
56             e.printStackTrace();
57         } catch (ClassNotFoundException e) {
58             e.printStackTrace();
59         } finally {
60             //6、释放资源
61             if(resultSet != null){
62                 try {
63                     resultSet.close();
64                 } catch (SQLException e) {
65                     e.printStackTrace();

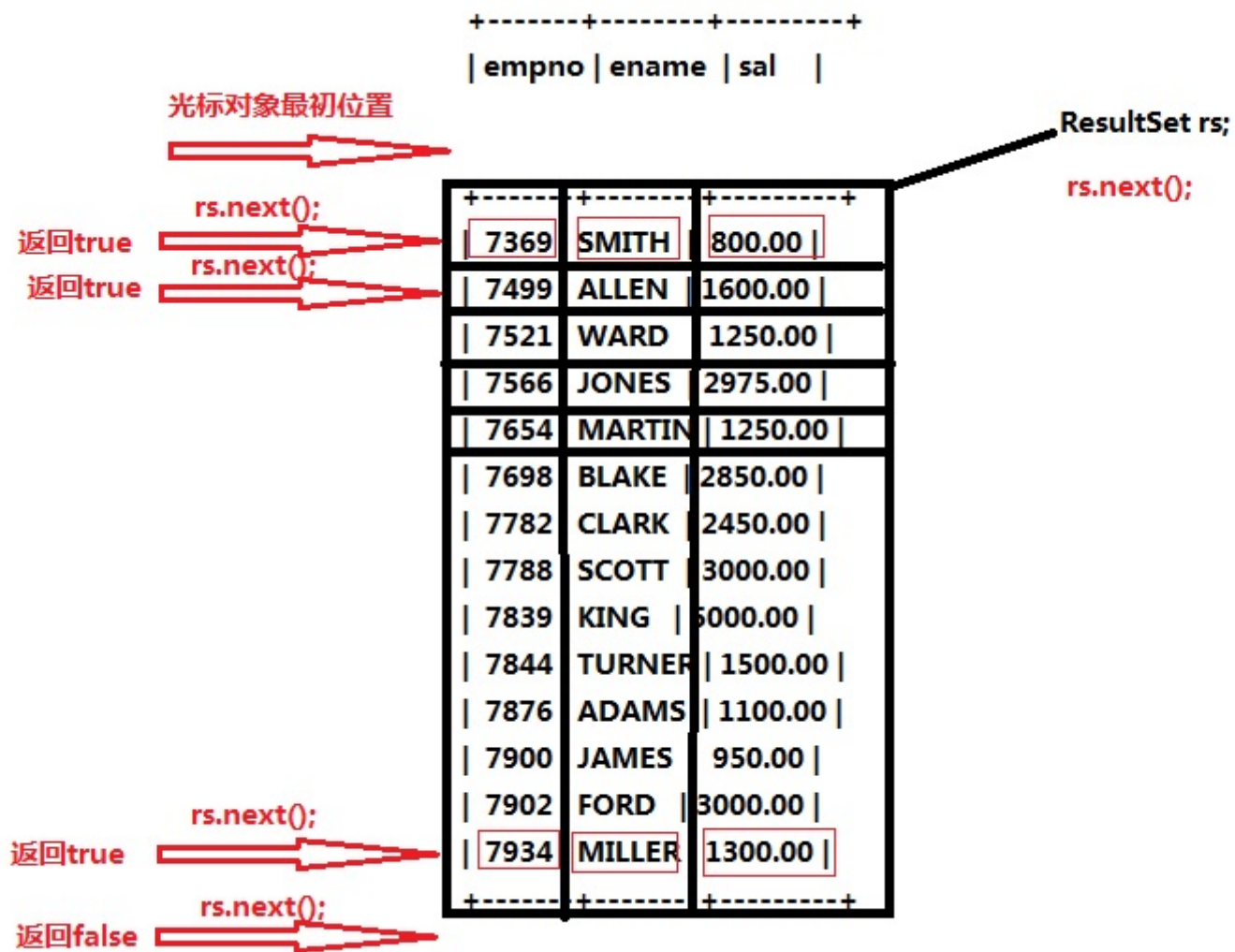
```



```

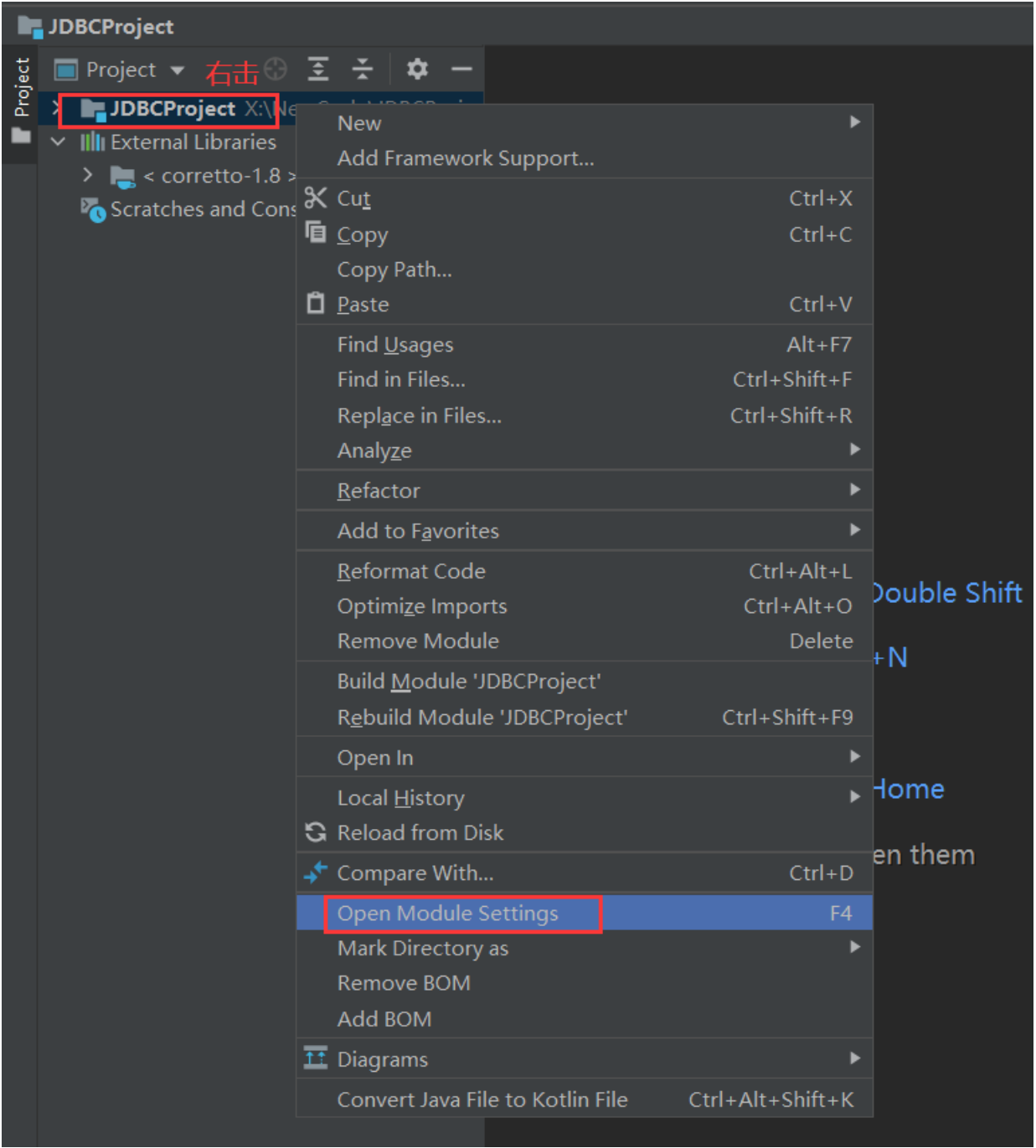
66     }
67 }
68 if(stmt != null){
69     try {
70         stmt.close();
71     } catch (SQLException e) {
72         e.printStackTrace();
73     }
74 }
75 if(conn != null){
76     try {
77         conn.close();
78     } catch (SQLException e) {
79         e.printStackTrace();
80     }
81 }
82 }
83 }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

```

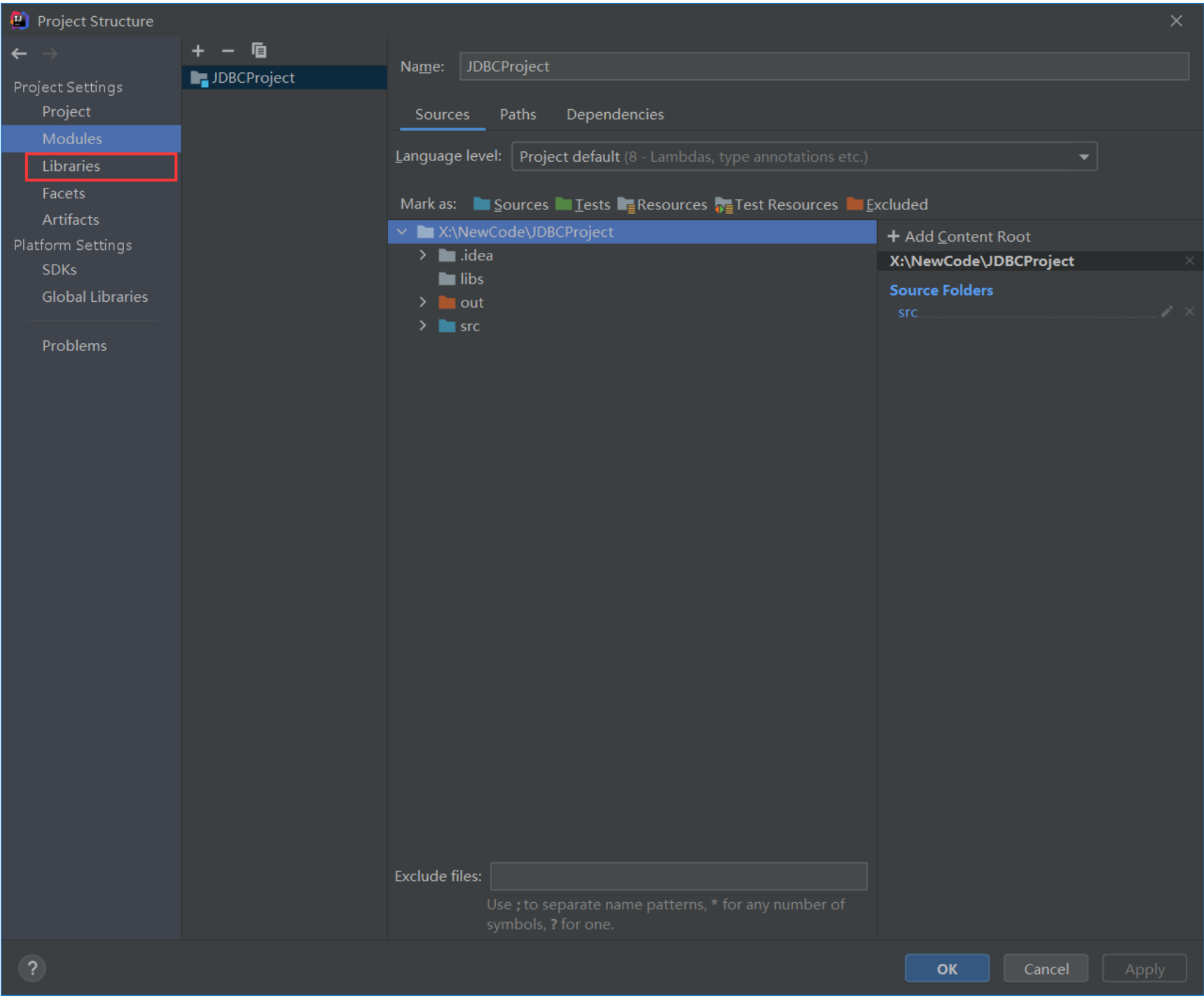


6.使用IDEA开发JDBC代码配置驱动

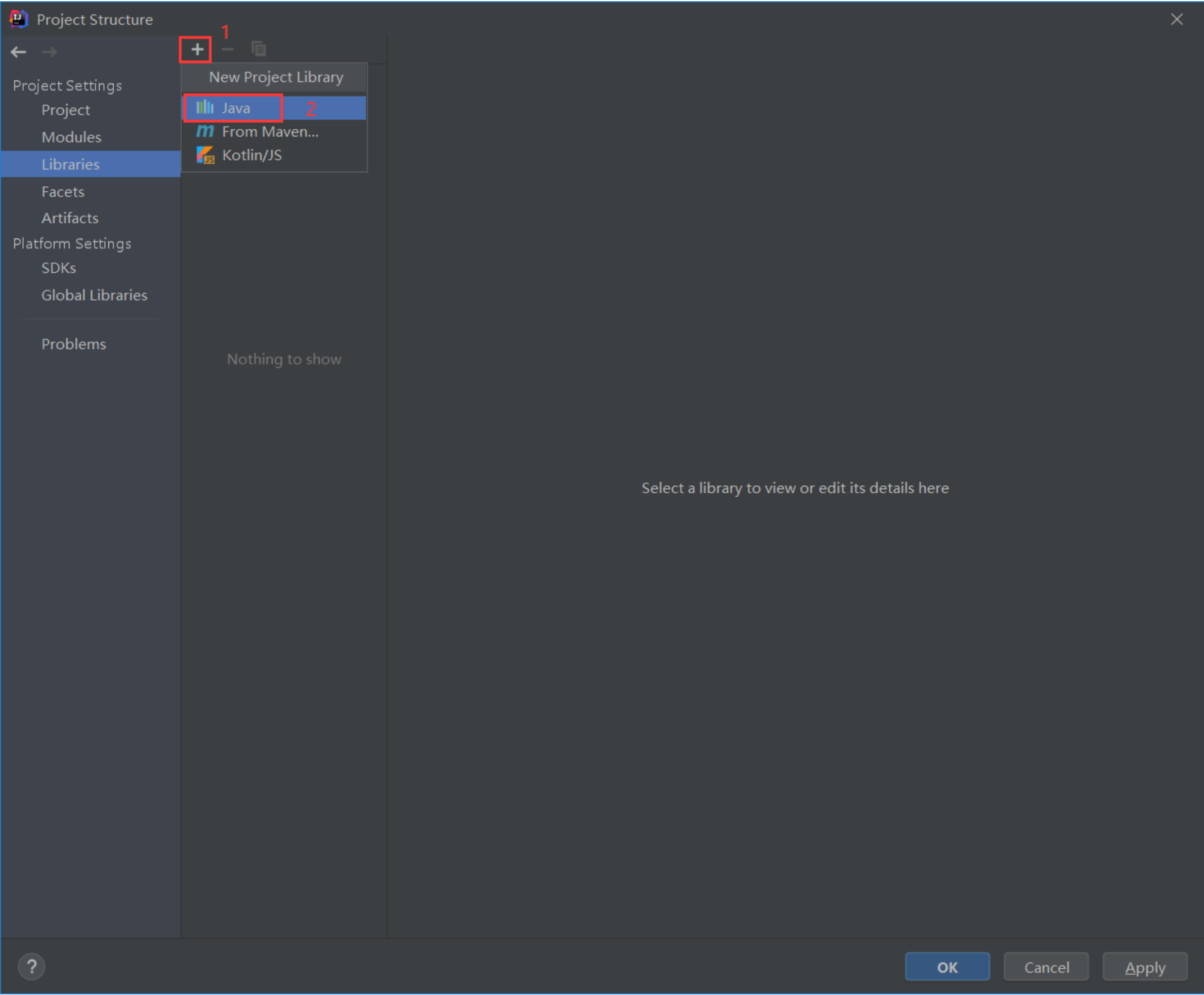
a



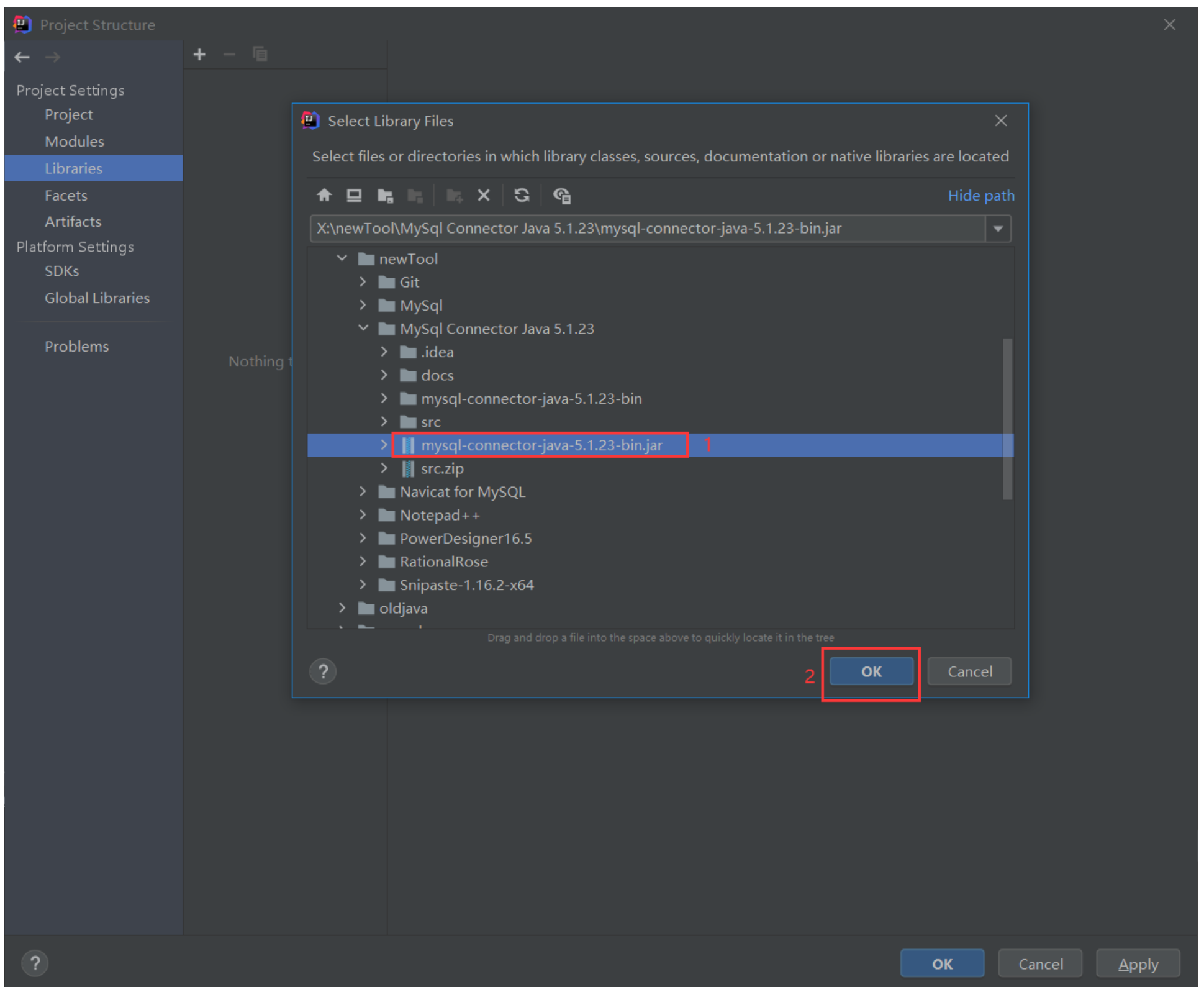
b



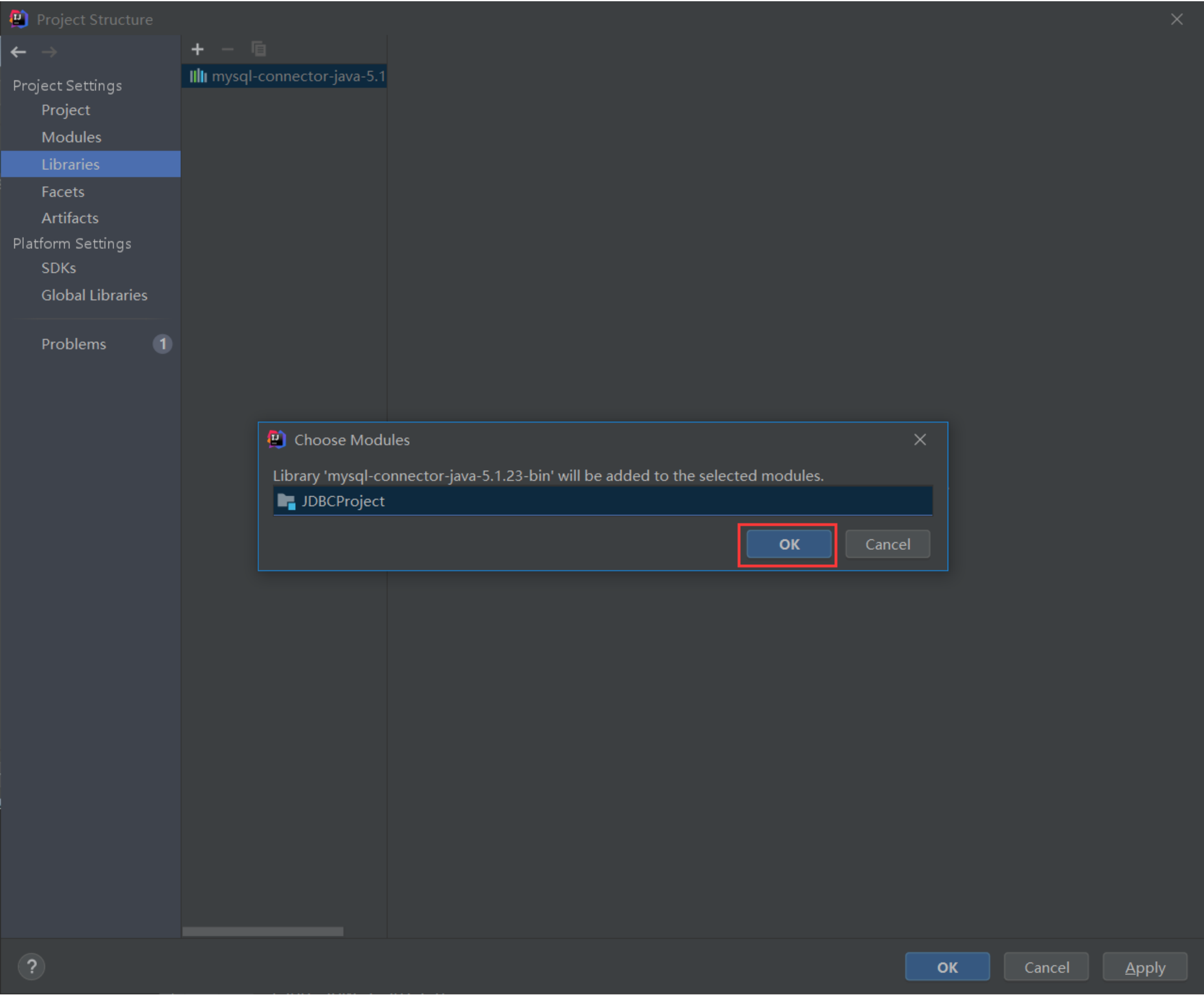
c



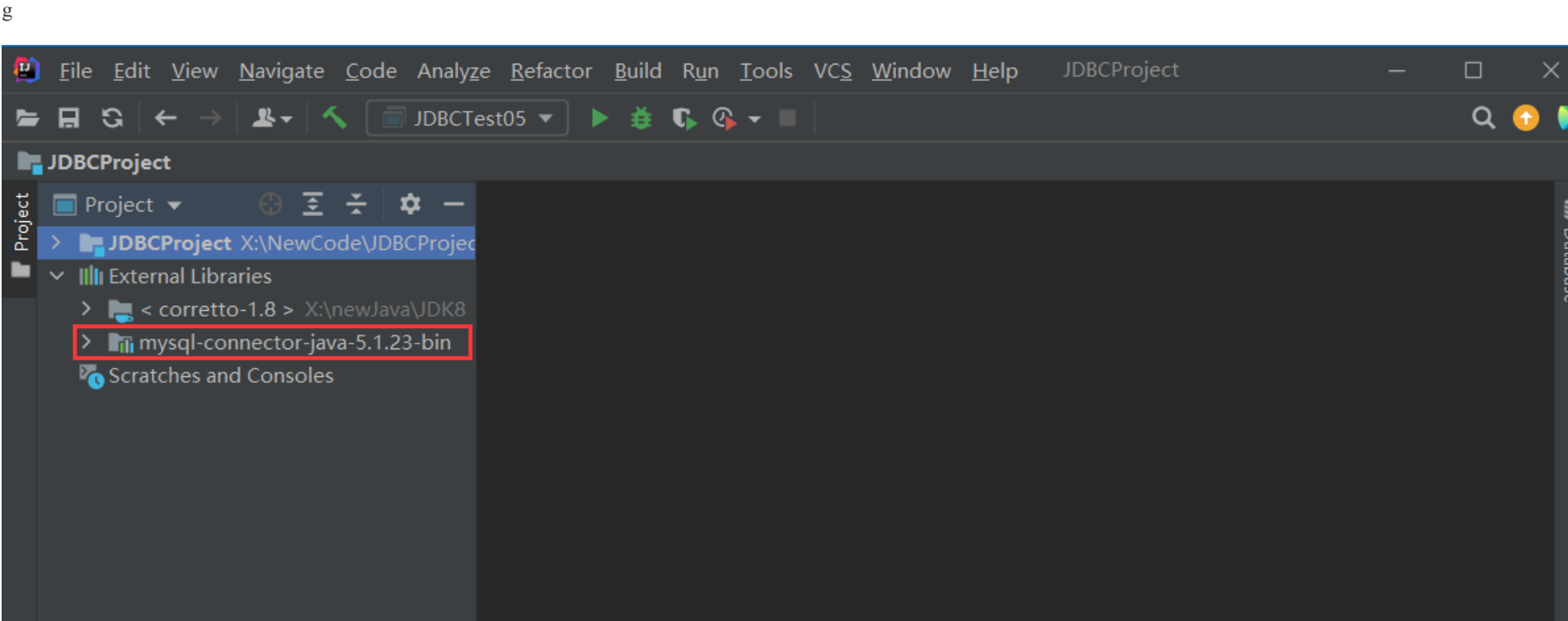
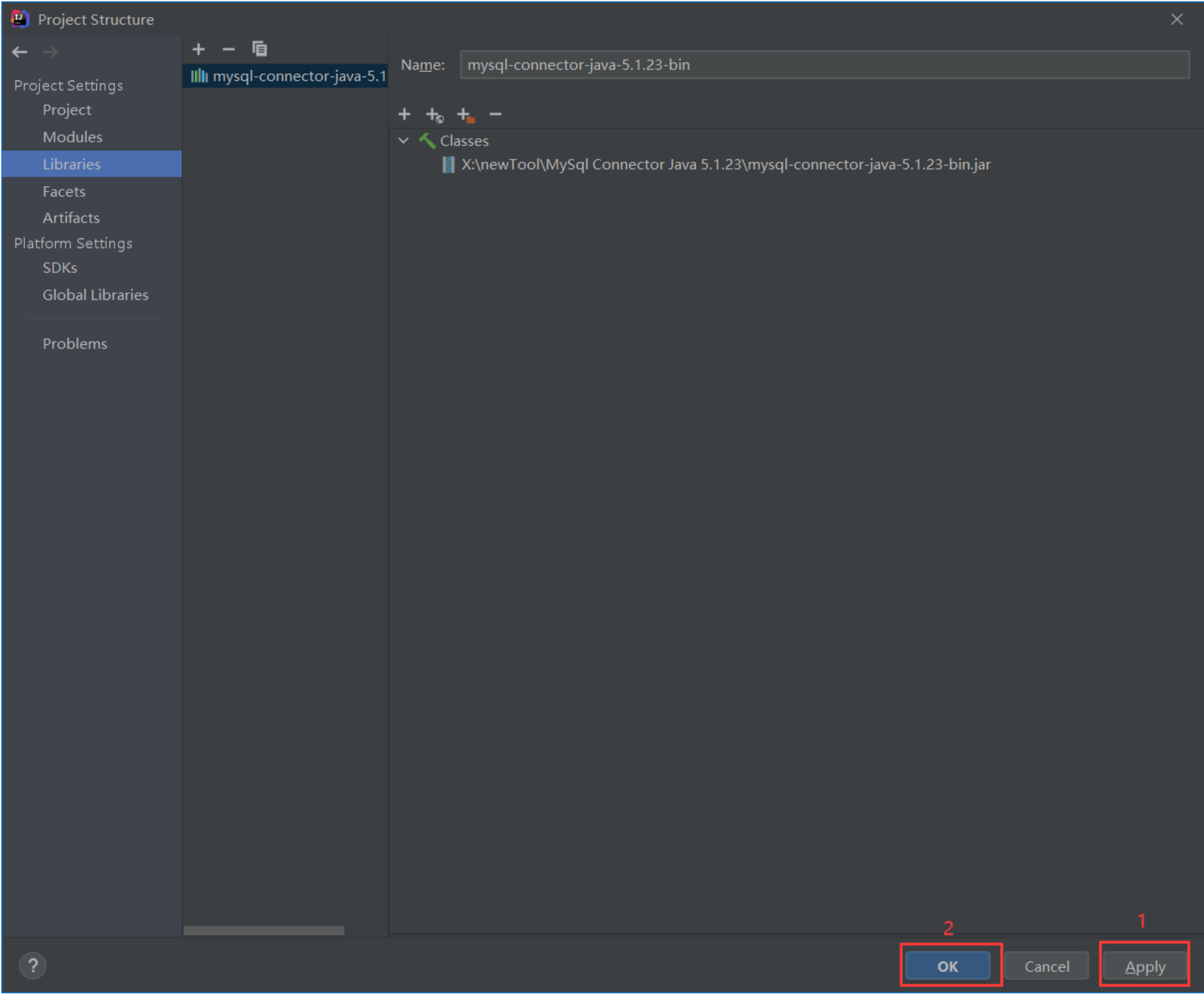
d



c



f



7.用户登录业务

```
1  #sql脚本
2  drop table if exists t_user;
3  /*=====*/
4  /* Table: t_user */
5  /*=====*/
6  create table t_user
7  (
8      id                bigint auto_increment,
9      loginName         varchar(255),
10     loginPwd           varchar(255),
11     realName           varchar(255),
12     primary key (id)
13 );
14
15 insert into t_user(loginName,loginPwd,realName) values('zhangsan', '123', '张三');
```

```
16 insert into t_user(loginName,loginPwd,realName) values('jack', '456', '杰克');
17 commit;
18 select * from t_user;
19 +-----+-----+-----+-----+
20 | id | loginName | loginPwd | realName |
21 +-----+-----+-----+-----+
22 | 1 | zhangsan | 123      | 张三     |
23 | 2 | jack     | 456      | 杰克     |
24 +-----+-----+-----+-----+
```

```
1  /**
2   * 实现功能
3   *      1、需求：模拟用户登录功能的实现
4   *      2、业务描述
5   *          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
6   *          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
7   *          Java程序连接数据库验证用户名和密码是否合法
8   *          合法：显示登录成功
9   *          不合法：显示登录失败
10  *      3、数据的准备
11  *          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
12  *          使用PD工具来进行数据库表的设计。
13  *      4、这种代码存在的问题
14  *          用户名：aaa
15  *          密码：aaa' or '1'='1
16  *          登录成功
17  *          这种现象称为SQL注入(安全隐患)。黑客经常使用
18  *      5、导致SQL注入的原因是什么？
19  *          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
20  *          导致sql语句的原意被扭曲，从而达到sqlzhuru
21  *          select * from t_user where loginName = 'aaa' and loginPwd = 'aaa' or '1'='1';
22  */
23 public class JDBCTest06 {
24     public static void main(String[] args) {
25         //初始化一个界面
26         Map<String,String> userLoginInfo = initUI();
27         //验证用户名和密码
28         boolean loginSuccess = login(userLoginInfo);
29         //输出最后结果
30         System.out.println(loginSuccess ? "登录成功" : "登录失败");
31     }
32
33     /**
34     * 用户登录
35     * @param userLoginInfo 用户登录信息
36     * @return false表示失败，true表示成功
37     */
38     private static boolean login(Map<String, String> userLoginInfo) {
39         //打标记的意识
40         boolean loginSuccess = false;
41
42         //单独定义变量
43         String loginName = userLoginInfo.get("loginName");
44         String loginPwd = userLoginInfo.get("loginPwd");
45
46         //JDBC代码
47         Connection conn = null;
48         Statement stmt = null;
49         ResultSet rs = null;
50         try {
51             //1、注册驱动
52             Class.forName("com.mysql.jdbc.Driver");
53             //2、获取连接
54             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
55             //3、获取数据库操作对象
56             stmt = conn.createStatement();
57             //4、执行SQL
58             String sql = "select * from t_user where " +
59                 "loginName = '" + loginName + "' and " +
60                 "loginPwd = '" + loginPwd + "';" ;
61             rs = stmt.executeQuery(sql);
62             //5、处理结果集
63             if(rs.next()){
64                 // 登录成功
65                 loginSuccess = true;
66             }
67         } catch (ClassNotFoundException e) {
68             e.printStackTrace();
69         } catch (SQLException e) {
70             e.printStackTrace();
71         } finally {
72             //6、释放资源
73             if(rs != null){
74                 try {
75                     rs.close();
76                 } catch (SQLException e) {
77                     e.printStackTrace();
78                 }
79             }
80         }
81     }
82 }
```

```

78         }
79     }
80     if(stmt != null){
81         try {
82             stmt.close();
83         } catch (SQLException e) {
84             e.printStackTrace();
85         }
86     }
87     if(conn != null){
88         try {
89             conn.close();
90         } catch (SQLException e) {
91             e.printStackTrace();
92         }
93     }
94 }
95 return loginSuccess;
96 }
97
98 /**
99  * 初始化用户界面
100  * @return 用户输入的用户名和密码等登录信息
101  */
102 private static Map<String, String> initUI() {
103     Scanner scanner = new Scanner(System.in);
104
105     System.out.print("用户名: ");
106     String loginName = scanner.nextLine();
107
108     System.out.print("密码: ");
109     String loginPwd = scanner.nextLine();
110
111     Map<String,String> userLoginInfo = new HashMap<>();
112     userLoginInfo.put("loginName", loginName);
113     userLoginInfo.put("loginPwd", loginPwd);
114
115     return userLoginInfo;
116 }
117 }

```

8.PreparedStatement (预编译数据库操作对象)

```

1  /**
2  * 1、解决SQL注入问题
3  *     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。
4  *     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。
5  *     要想用户信息不参与SQL语句的编译，那么必须使用 java.sql.PreparedStatement
6  *     PreparedStatement接口继承了java.sql.Statement
7  *     PreparedStatement是属于预编译的数据库操作对象
8  *     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"
9  * 2、测试结果
10 *     用户名: aaa
11 *     密码: aaa' or '1'='1
12 *     登录失败
13 * 3、解决SQL注入的关键是什么？
14 *     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。
15 * 4、对比一下 Statement 和 PreparedStatement？
16 *     - Statement存在SQL注入问题，PreparedStatement解决了SQL注入问题
17 *     - 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
18 *     - PreparedStatement会在编译阶段做类型的安全检查
19 *
20 *     综上所述：PreparedStatement使用较多。只有极少数的情况下需要使用Statement
21 * 5、什么情况下必须使用Statement呢？
22 *     业务方面要求必须支持SQL注入的时候。
23 *     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
24 */
25 public class JDBCTest07 {
26     public static void main(String[] args) {
27         //初始化一个界面
28         Map<String,String> userLoginInfo = initUI();
29         //验证用户名和密码
30         boolean loginSuccess = login(userLoginInfo);
31         //输出最后结果
32         System.out.println(loginSuccess ? "登录成功" : "登录失败");
33     }
34
35     /**
36     * 用户登录
37     * @param userLoginInfo 用户登录信息
38     * @return false表示失败，true表示成功
39     */
40     private static boolean login(Map<String, String> userLoginInfo) {
41         //打标记的意识
42         boolean loginSuccess = false;
43
44         //单独定义变量
45         String loginName = userLoginInfo.get("loginName");

```



```

46     String loginPwd = userLoginInfo.get("loginPwd");
47
48     //JDBC代码
49     Connection conn = null;
50     PreparedStatement ps = null;//这里使用PreparedStatement （预编译的数据库操作对象）
51     ResultSet rs = null;
52     try {
53         //1、注册驱动
54         Class.forName("com.mysql.jdbc.Driver");
55
56         //2、获取连接
57         conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
58
59         //3、获取预编译的数据库操作对象
60         //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
61         String sql = "select * from t_user where loginName = ? and loginPwd = ?";
62         //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
63         ps = conn.prepareStatement(sql);
64         //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。)
65         ps.setString(1, loginName);
66         ps.setString(2, loginPwd);
67
68         //4、执行SQL
69         rs = ps.executeQuery();
70
71         //5、处理结果集
72         if(rs.next()){
73             // 登录成功
74             loginSuccess = true;
75         }
76     } catch (ClassNotFoundException e) {
77         e.printStackTrace();
78     } catch (SQLException e) {
79         e.printStackTrace();
80     } finally {
81         //6、释放资源
82         if(rs != null){
83             try {
84                 rs.close();
85             } catch (SQLException e) {
86                 e.printStackTrace();
87             }
88         }
89         if(ps != null){
90             try {
91                 ps.close();
92             } catch (SQLException e) {
93                 e.printStackTrace();
94             }
95         }
96         if(conn != null){
97             try {
98                 conn.close();
99             } catch (SQLException e) {
100                 e.printStackTrace();
101             }
102         }
103     }
104     return loginSuccess;
105 }
106
107 /**
108  * 初始化用户界面
109  * @return 用户输入的用户名和密码等登录信息
110  */
111 private static Map<String, String> initUI() {
112     Scanner scanner = new Scanner(System.in);
113
114     System.out.print("用户名: ");
115     String loginName = scanner.nextLine();
116
117     System.out.print("密码: ");
118     String loginPwd = scanner.nextLine();
119
120     Map<String,String> userLoginInfo = new HashMap<>();
121     userLoginInfo.put("loginName", loginName);
122     userLoginInfo.put("loginPwd", loginPwd);
123
124     return userLoginInfo;
125 }
126 }

```

9.演示Statement的用途

```
1  /**
2   * 演示Statement的用途
3   */
4  public class JDBCTest08 {
5      public static void main(String[] args) {
6          // 用户在控制台输入desc就是降序，输入asc就是升序
7          Scanner s = new Scanner(System.in);
8          System.out.println("请输入desc或asc, desc表示降序, asc表示升序");
9          System.out.print("请输入: ");
10         String keyWords = s.nextLine();
11
12         // 执行SQL
13         Connection conn = null;
14         Statement stmt = null;
15         ResultSet rs = null;
16         try {
17             // 1、注册驱动
18             Class.forName("com.mysql.jdbc.Driver");
19             // 2、获取连接
20             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
21             // 3、获取数据库操作对象
22             stmt = conn.createStatement();
23             // 4、执行SQL语句
24             String sql = "select ename from emp order by ename " + keyWords;
25             rs = stmt.executeQuery(sql);
26             // 5、遍历结果集
27             while (rs.next()){
28                 System.out.println(rs.getString("ename"));
29             }
30         } catch (ClassNotFoundException e) {
31             e.printStackTrace();
32         } catch (SQLException e) {
33             e.printStackTrace();
34         } finally {
35             // 6、释放资源
36             if(rs != null){
37                 try {
38                     rs.close();
39                 } catch (SQLException e) {
40                     e.printStackTrace();
41                 }
42             }
43             if(stmt != null){
44                 try {
45                     stmt.close();
46                 } catch (SQLException e) {
47                     e.printStackTrace();
48                 }
49             }
50             if(conn != null){
51                 try {
52                     conn.close();
53                 } catch (SQLException e) {
54                     e.printStackTrace();
55                 }
56             }
57         }
58     }
59 }
```

10.PreparedStatement完成增删改

```
1  /**
2   * PreparedStatement完成insert delete update
3   */
4  public class JDBCTest09 {
5      public static void main(String[] args) {
6          Connection conn = null;
7          PreparedStatement ps = null;
8          try {
9              //1、注册驱动
10             Class.forName("com.mysql.jdbc.Driver");
11             //2、获取连接
12             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
13             //3、获取预编译的数据库操作对象
14             /*
15             // insert
16             String sql = "insert into t_user(loginName,loginPwd,realName) values(?,?,?)";
17             ps = conn.prepareStatement(sql);
18             ps.setString(1, "xukang");
19             ps.setString(2, "123456");
20             ps.setString(3, "徐康");
21             */
22
23             /*
```

```

24         // update
25         String sql = "update t_user set loginPwd = ?,realName = ? where id = ?";
26         ps = conn.prepareStatement(sql);
27         ps.setString(1, "789789");
28         ps.setString(2, "虚空");
29         ps.setInt(3, 3);
30         */
31
32         // delete
33         String sql = "delete from t_user where loginName = ? and loginPwd = ?";
34         ps = conn.prepareStatement(sql);
35         ps.setString(1, "xukang");
36         ps.setString(2, "789789");
37         //4、执行SQL
38         int count = ps.executeUpdate();
39         System.out.println(count);
40         //5、处理结果集
41     } catch (ClassNotFoundException e) {
42         e.printStackTrace();
43     } catch (SQLException e) {
44         e.printStackTrace();
45     } finally {
46         //6、释放资源
47         if(ps != null){
48             try {
49                 ps.close();
50             } catch (SQLException e) {
51                 e.printStackTrace();
52             }
53         }
54         if(conn != null){
55             try {
56                 conn.close();
57             } catch (SQLException e) {
58                 e.printStackTrace();
59             }
60         }
61     }
62 }
63 }

```

11.JDBC事务机制

```

1  /**
2   * JDBC事务机制
3   *      1、JDBC中的事务是自动提交的，什么是自动提交？
4   *          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。
5   *          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
6   *          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。
7   *      2、以下程序先来验证一下JDBC的事务是否是自动提交机制！
8   *          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）
9   */
10 public class JDBCTest10 {
11     public static void main(String[] args) {
12         Connection conn = null;
13         PreparedStatement ps = null;
14         try {
15             //1、注册驱动
16             Class.forName("com.mysql.jdbc.Driver");
17             //2、获取连接
18             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
19             //3、获取预编译的数据库操作对象
20             String sql1 = "update dept set dname = ? where deptno = ?";
21             ps = conn.prepareStatement(sql1);
22
23             // 第一次给占位符传值
24             ps.setString(1, "X部门");
25             ps.setInt(2, 30);
26             int count = ps.executeUpdate();// 执行第一条UPDATE语句
27             System.out.println(count);
28
29             // 重新给占位符传值
30             ps.setString(1, "Y部门");
31             ps.setInt(2, 20);
32             count = ps.executeUpdate();// 执行第二条UPDATE语句
33             System.out.println(count);
34
35         } catch (ClassNotFoundException e) {
36             e.printStackTrace();
37         } catch (SQLException e) {
38             e.printStackTrace();
39         } finally {
40             //6、释放资源
41             if(ps != null){
42                 try {
43                     ps.close();
44                 } catch (SQLException e) {
45                     e.printStackTrace();

```

```

46         }
47     }
48     if(conn != null){
49         try {
50             conn.close();
51         } catch (SQLException e) {
52             e.printStackTrace();
53         }
54     }
55 }
56 }
57 }

```

```

1  /**
2   * sql脚本:
3   *   drop table if exists t_act;
4   *   create table t_act(
5   *       actno      bigint,
6   *       balance    double(7,2) //7表示有效数字的个数, 2表示小数位的个数
7   *   );
8   *   insert into t_act(actno,balance) values(111,20000);
9   *   insert into t_act(actno,balance) values(222,0);
10  *   commit;
11  *   select * from t_act;
12  *
13  * 重点: 三行代码
14  *   conn.setAutoCommit(false);//开启事务
15  *   conn.commit();//手动提交事务
16  *   conn.rollback();//手动回滚事务
17  */
18 public class JDBCTest11 {
19     public static void main(String[] args) {
20         Connection conn = null;
21         PreparedStatement ps = null;
22         try {
23             //1、注册驱动
24             Class.forName("com.mysql.jdbc.Driver");
25             //2、获取连接
26             conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
27
28             //将自动提交机制改为手动提交
29             conn.setAutoCommit(false);//开启事务
30
31             //3、获取预编译的数据库操作对象
32             String sql = "update t_act set balance = ? where actno = ?";
33             ps = conn.prepareStatement(sql);
34             //给?传值
35             ps.setDouble(1, 10000);
36             ps.setInt(2, 111);
37             int count = ps.executeUpdate();
38
39             //String s = null;
40             //s.toString();// 空指针异常 会导致111用户平白无故损失10000元
41
42             ps.setDouble(1, 10000);
43             ps.setInt(2, 222);
44             count += ps.executeUpdate();
45
46             System.out.println(count == 2 ? "转账成功" : "转账失败");
47
48             //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
49             conn.commit();//提交事务
50         } catch (ClassNotFoundException e) {
51             e.printStackTrace();
52         } catch (SQLException e) {
53             if(conn != null){
54                 try {
55                     conn.rollback();//回滚事务
56                 } catch (SQLException throwables) {
57                     throwables.printStackTrace();
58                 }
59             }
60             e.printStackTrace();
61         } finally {
62             //6、释放资源
63             if(ps != null){
64                 try {
65                     ps.close();
66                 } catch (SQLException e) {
67                     e.printStackTrace();
68                 }
69             }
70             if(conn != null){
71                 try {
72                     conn.close();
73                 } catch (SQLException e) {
74                     e.printStackTrace();

```

```
75         }
76     }
77 }
78 }
79 }
```

12.JDBC工具类的封装

```
1  /**
2   * JDBC工具类，简化JDBC编程
3   */
4  public class JDBCUtil {
5      /**
6       * 工具类的构造方法都是私有的
7       * 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
8       */
9      private JDBCUtil(){}
10
11     // 静态代码块在类加载时执行，并且只执行一次
12     // 注册驱动
13     static {
14         try {
15             Class.forName("com.mysql.jdbc.Driver");
16         } catch (ClassNotFoundException e) {
17             e.printStackTrace();
18         }
19     }
20
21     /**
22     * 获取数据库连接对象
23     * @return 连接对象
24     * @throws SQLException
25     */
26     public static Connection getConnection() throws SQLException {
27         Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bjpowernode","root","111");
28         return conn;
29     }
30
31     /**
32     * 关闭资源
33     * @param conn 连接对象
34     * @param ps 数据库操作对象
35     * @param rs 结果集
36     */
37     public static void close(Connection conn, Statement ps, ResultSet rs){
38         if(rs != null){
39             try {
40                 rs.close();
41             } catch (SQLException e) {
42                 e.printStackTrace();
43             }
44         }
45         if(ps != null){
46             try {
47                 ps.close();
48             } catch (SQLException e) {
49                 e.printStackTrace();
50             }
51         }
52         if(conn != null){
53             try {
54                 conn.close();
55             } catch (SQLException e) {
56                 e.printStackTrace();
57             }
58         }
59     }
60 }
```

```
1  /**
2   * 两个任务：
3   *      1、测试JDBCUtil工具类是否好用
4   *      2、模糊查询怎么写？
5   */
6  public class JDBCTest12 {
7      public static void main(String[] args) {
8          Connection conn = null;
9          PreparedStatement ps = null;
10         ResultSet rs = null;
11         try {
12             //获取连接
13             conn = JDBCUtil.getConnection();
14             //获取预编译的数据库操作对象
15             //错误的写法
16             /*
17             String sql = "select ename from emp where ename like '_?%'";
18             ps = conn.prepareStatement(sql);
```

```
19         ps.setString(1, "A");
20         */
21         //正确的写法
22         String sql = "select ename from emp where ename like ?";
23         ps = conn.prepareStatement(sql);
24         ps.setString(1, "_A%");//第二个字母为A的
25         //执行SQL
26         rs = ps.executeQuery();
27         //遍历结果集
28         while (rs.next()){
29             System.out.println(rs.getString("ename"));
30         }
31     } catch (SQLException e) {
32         e.printStackTrace();
33     } finally {
34         // 释放资源
35         JDBCUtil.close(conn, ps, rs);
36     }
37 }
38 }
```

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。

```
1  /**
2   * 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
3   */
4  public class JDBCTest13 {
5      public static void main(String[] args) {
6          Connection conn = null;
7          PreparedStatement ps = null;
8          ResultSet rs = null;
9          try {
10             conn = JDBCUtil.getConnection();
11             //开启事务
12             conn.setAutoCommit(false);
13
14             String sql = "select ename, job, sal from emp where job = ? for update";
15             ps = conn.prepareStatement(sql);
16             ps.setString(1, "MANAGER");
17
18             rs = ps.executeQuery();
19             while (rs.next()){
20                 System.out.println(rs.getString("ename") + ", " + rs.getString("job") + ", " + rs.getDouble("sal"));
21             }
22             //提交事务（事务结束）
23             conn.commit();
24         } catch (SQLException e) {
25             if(conn != null){
26                 try {
27                     //回滚事务（事务结束）
28                     conn.rollback();
29                 } catch (SQLException throwables) {
30                     throwables.printStackTrace();
31                 }
32             }
33             e.printStackTrace();
34         } finally {
35             JDBCUtil.close(conn, ps, rs);
36         }
37     }
38 }
```

```
1  /**
2   * 这个程序负责修改被锁定的记录
3   */
```



```
4 public class JDBCTest14 {
5     public static void main(String[] args) {
6         Connection conn = null;
7         PreparedStatement ps = null;
8         try {
9             conn = JDBCUtil.getConnection();
10            conn.setAutoCommit(false);
11
12            String sql = "update emp set sal = sal * 1.1 where job = ?";
13            ps = conn.prepareStatement(sql);
14            ps.setString(1, "MANAGER");
15
16            int count = ps.executeUpdate();
17            System.out.println(count);
18
19            conn.commit();
20        } catch (SQLException e) {
21            if(conn != null){
22                try {
23                    conn.rollback();
24                } catch (SQLException throwables) {
25                    throwables.printStackTrace();
26                }
27            }
28            e.printStackTrace();
29        } finally {
30            JDBCUtil.close(conn, ps, null);
31        }
32    }
33 }
```