1、课程名称：Java 数据库编程（JDBC）

2、知识点

2.1、上次课程的主要知识点

1、 类集的主要结构
2、 类集中各个子类的区别

2.2、本次预计讲解的知识点

1、 JDBC 分类，并使用 JDBC 连接 Oracle 操作
2、 JDBC 主要接口的使用没，并可以利用这些接口完成数据的 CRUD
3、 事务处理
4、 批处理的使用

3、具体内容

3.1、JDBC 简介（理解）

JDBC（Java DataBase Connective）Java 的数据库连接，JDBC 本身提供的是一套与平台无关的数据库的操作标准。所以在整个 JDBC 中充斥着大量的操作接口。而且 JDBC 本身不是技术，而是一种服务。

由于 JDBC 本身属于一个标准，所以一个数据库如果希望使用 Java 进行程序开发的话，那么各个数据库的生产商必须实现这些标准 —— 提供专门的数据库的操作包。

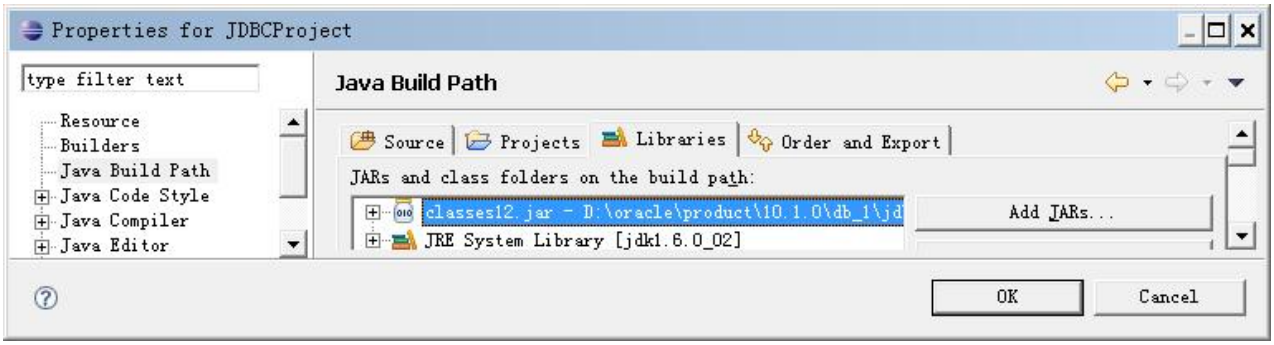
根据 JDBC 操作方式的不同，一共有以下三种常见形式：

- JDBC-ODBC 桥连接：利用微软的 ODBC 技术进行操作
  - |- 操作过程：程序 à JDBC à ODBC à 数据库
  - |- 此方式属于 Java 默认支持的方式，但是由于其中间加入了 ODBC 端，所以性能很差
- JDBC 连接：使用各个数据库生产商提供的数据库驱动程序
  - |- 操作过程：程序 à JDBC à 数据库
  - |- 由于中间缺少了 ODBC 环节，所以性能将有着明显的提升
- JDBC 网络连接：通过网络协议进行数据库的连接操作

一定要记住的是，虽然 JDBC-ODBC 方式不被使用，但是此种操作中由于是 SUN 默认支持的，所以 JDBC 的版本是最高的，但是如果使用的是纯粹的各个数据库生产商提供的驱动程序，那么肯定不会与最新的技术同步。

一般对于大型的数据库，所有的驱动程序会随着安装程序一起安装上来（DB2、Oracle），Oracle 的驱动程序路径：D:\oracle\product\10.1.0\db_1\jdbc\lib\classes12.jar，那么要想使用，肯定需要在 classpath 中配置。

如果现在使用的是 Eclipse 的话，则需要在 Build Path 中配置此驱动程序的开发包。



此时配置完成驱动程序之后，就可以通过以下的接口和类进行 JDBC 操作了：

- 类：DriverManager
- 接口：Connection、PreparedStatement、Statement、ResultSet

3.2、连接数据库（重点）

在进行数据库连接的时候需要使用以下的几个信息：

- 数据库的驱动程序：oracle.jdbc.driver.OracleDriver
- 连接地址：jdbc:oracle:thin:@localhost:1521:mldn
- 用户名：scott
- 密码：tiger

要想连接需要使用 Connection 接口进行连接对象的保存，但是此接口必须依靠 DriverManager 类才可以完成对象的实例化操作。

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
public class ConnectionDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连接数据库
        System.out.println(conn); // 如果不为null，表示已连接
        conn.close(); // 数据库的操作必须关闭
    }
}
```

要想真正的进行数据库连接必须依靠 Oracle 中的 Listener 服务。

3.3、Statement 接口（重点）

如果要进行数据库的操作，在 JDBC 中依然使用的是 SQL 语句完成，而用于执行这些 SQL 语句的接口就是 Statement 接口了，但是如果使用 Statement 接口操作的话也分为两种形式：查询、更新，为了方便起见，建立以下的一张表完成数据库的更新：

```
DROP TABLE myuser ;
DROP SEQUENCE myseq ;
PURGE RECYCLEBIN ;
CREATE SEQUENCE myseq ;
CREATE TABLE myuser(
    id          NUMBER          PRIMARY KEY ,
    name        VARCHAR2(30)NOT NULL ,
    birthday    DATE            NOT NULL
) ;
```

3.3.1、增加数据

如果要想增加数据则编写增加的 SQL 语句：

```
INSERT INTO 表名称 （列名称 1，列名称 2，…） VALUES (值 1，值 2，…)；
```

范例：编写增加操作

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
public class InsertDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        PreparedStatement pstmt = null; // 定义数据库操作
        String sql = "INSERT INTO myuser(id,name,birthday) VALUES"
            + " (myseq.nextVal,'张三',TO_DATE('1998-09-19','yyyy-mm-dd')) ";
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连据库
        pstmt = conn.prepareStatement(sql); // 创建数据库操作
        int len = pstmt.executeUpdate(sql);
        System.out.println("更新了" + len + "条记录。");
        conn.close(); // 数据库的操作必须关闭
    }
}
```

3.3.2、修改数据

修改数据的 SQL 语法：

```
UPDATE 表名称 SET 字段=值，字段=值，.. [WHERE 更新条件]
```

如果不写更新条件，表示的是全部数据都要更新。

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
public class UpdateDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
```

```
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        PreparedStatement pstmt = null; // 定义数据库操作
        String sql = "UPDATE myuser SET name='李四',birthday=sysdate WHERE id=1";
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连据库
        pstmt = conn.prepareStatement(sql); // 创建数据库操作
        int len = pstmt.executeUpdate(sql); // 更新操作，返回更新的记录数
        System.out.println("更新了" + len + "条记录。");
        conn.close(); // 数据库的操作必须关闭
    }
}
```

3.3.3、删除数据

删除数据的 SQL 语法如下：

```
DELETE FROM 表名称 [WHERE 删除条件]
```

执行删除操作：

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
public class DeleteDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        PreparedStatement pstmt = null; // 定义数据库操作
        String sql = "DELETE FROM myuser WHERE id=1";
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连据库
        pstmt = conn.prepareStatement(sql); // 创建数据库操作
        int len = pstmt.executeUpdate(sql); // 更新操作，返回更新的记录数
        System.out.println("更新了" + len + "条记录。");
        conn.close(); // 数据库的操作必须关闭
    }
}
```

以上的的三段代码几乎都是在一个固定的操作格式中完成的，唯一更改的地方只是 SQL 语句而已。

更新操作使用的就是 Statement 接口中的 executeUpdate()方法，此方法返回的是更新记录的行数。

3.3.4、查询数据

数据库的更新操作本身非常的容易，直接使用 executeUpdate()语句，但是如果要是查询的话就不一样，查询使用的方法是 executeQuery()方法完成，方法：public ResultSet executeQuery(String sql) throws SQLException

查询的时候是将所有的内容都保存在了 ResultSet 接口的对象之中。取得 ResultSet 对象之后，就可以通过 next()方法往下取每一行的数据，但是需要注意的是，如果要想取出某一列的内容，则使用 getXxx()的形式，其中 Xxx 表示的是具体的数据类型，例如：getFloat()、getInt()、getDate()。

必须记住的是，在程序中使用 SELECT 语句的时候绝对 100%不允许出现“*”，必须明确的给出要查询的具体列的名称。

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
public class QueryDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        PreparedStatement pstmt = null; // 定义数据库操作
        ResultSet rs = null; // 保存查询结果
        String sql = "SELECT id,name,birthday FROM myuser";
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连据库
        pstmt = conn.prepareStatement(sql); // 创建数据库操作
        rs = pstmt.executeQuery(sql); // 查询
        while (rs.next()) { // 判断是否有下一个记录
            int id = rs.getInt("id");
            String name = rs.getString("name");
            Date birthday = rs.getDate("birthday");
            System.out.println("编号：" + id + "，姓名：" + name + "，生日：" + birthday);
        }
        rs.close() ;
        pstmt.close() ;
        conn.close(); // 数据库的操作必须关闭
    }
}
```

此时，表中的内容已经可以查询出来了，而且在查询的时候还有另外一种更方便的做法。

```
        while (rs.next()) { // 判断是否有下一个记录
            int id = rs.getInt(1);
```

```
            String name = rs.getString(2);
            Date birthday = rs.getDate(3);
            System.out.println("编号：" + id + "，姓名：" + name + "，生日：" + birthday);
        }
```

3.3.5、作业

现在要求通过输入流输入一个用户的姓名和生日，并将此信息保存在数据库之中。

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class InsertDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        Statement stmt = null; // 定义数据库操作
        InputData input = new InputData();
        String name = input.getString("请输入姓名：");
        String birthday = input.getString("请输入生日：");
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连接数据库
        stmt = conn.createStatement(); // 创建数据库操作
        String sql = "INSERT INTO myuser(id,name,birthday) VALUES"
                + " (myseq.nextVal,'" + name + "',TO_DATE('" + birthday
                + "','yyyy-mm-dd')) ";
        System.out.println(sql) ;
        int len = stmt.executeUpdate(sql); // 更新操作，返回更新的记录数
        System.out.println("更新了" + len + "条记录。");
        conn.close(); // 数据库的操作必须关闭
    }
}
```

此时，输入正确的数据是可以执行插入操作的，但是此时也会有一种问题，由于以上的 SQL 语句是采用了拼凑代码的形式，所以一旦输入的数据有问题，则可能也无法插入：

```
请输入姓名：Mr'Smith
请输入生日：2009-09-18
INSERT INTO myuser(id,name,birthday) VALUES
(myseq.nextVal,'Mr'Smith',TO_DATE('2009-09-18','yyyy-mm-dd'))
Exception in thread "main" java.sql.SQLException: ORA-00917: missing comma
```

3.4、PreparedStatement（重点）

在开发中 100%不会使用 Statement 进行操作，而都使用其子接口 PreparedStatement 完成。

PreparedStatement 操作实际上属于预处理的操作。

如果要创建 PreparedStatement 接口的对象需要依靠 Connection 接口中的 prepareStatement()方法完成，而且必须传入一个预处理的 SQL 语句，所有的占位符使用“?”表示。

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Date;
public class PreparedInsertDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        PreparedStatement pstmt = null; // 定义数据库操作
        InputData input = new InputData();
        String name = input.getString("请输入姓名：");
        Date birthday = input.getDate("请输入生日：", "输入的日期格式有问题，");
        String sql = "INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,?,?) ";
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连接数据库
        pstmt = conn.prepareStatement(sql);
        int len = 0; // 接收更新的记录行数
        pstmt.setString(1, name);
        pstmt.setDate(2, new java.sql.Date(birthday.getTime()));
        len = pstmt.executeUpdate(); // 执行更新操作
        System.out.println("更新了" + len + "条记录。");
        conn.close(); // 数据库的操作必须关闭
    }
}
```

既然可以更新，那么现在使用此代码完成数据库的查询操作。

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Date;
public class PreparedQueryDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
```

```
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        PreparedStatement pstmt = null; // 定义数据库操作
        ResultSet rs = null; // 保存查询结果
        String sql = "SELECT id,name,birthday FROM myuser";
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连接数据库
        pstmt = conn.prepareStatement(sql) ;
        rs = pstmt.executeQuery(); // 查询
        while (rs.next()) { // 判断是否有下一个记录
            int id = rs.getInt(1);
            String name = rs.getString(2);
            Date birthday = rs.getDate(3);
            System.out.println("编号：" + id + "，姓名：" + name + "，生日：" + birthday);
        }
        rs.close() ;
        pstmt.close() ;
        conn.close(); // 数据库的操作必须关闭
    }
}
```

如果现在执行的是模糊查询，要使用 LIKE 语句。

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Date;
public class PreparedQueryDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        PreparedStatement pstmt = null; // 定义数据库操作
        ResultSet rs = null; // 保存查询结果
        String keyWord = "张" ;
        String sql = "SELECT id,name,birthday FROM myuser WHERE name LIKE ?";
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连接数据库
        pstmt = conn.prepareStatement(sql) ;
```

```
        pstmt.setString(1, "%" + keyWord + "%");
        rs = pstmt.executeQuery(); // 查询
        while (rs.next()) { // 判断是否有下一个记录
            int id = rs.getInt(1);
            String name = rs.getString(2);
            Date birthday = rs.getDate(3);
            System.out.println("编号：" + id + "，姓名：" + name + "，生日：" + birthday);
        }
        rs.close() ;
        pstmt.close() ;
        conn.close(); // 数据库的操作必须关闭
    }
}
```

3.5、批处理（重点）

之前的所有的操作实际上都属于 JDBC 1.0 的操作，而且以上代码的操作形式也几乎是固定的，而且不管 JDBC 如何发展，以上的代码是绝对会一直出现的，在 JDBC 2.0 之后增加了许多的新功能，例如：可滚动的结集、使用 ResultSet 更新数据库、批处理等等，但是唯一现在能使用的只有批处理了。

所谓的批处理就是指所有的操作可以一次性的提交到数据库之中。

如果要使用批处理则需要使用 Statement 接口中的 addBatch()方法。

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class BatchDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        Statement stmt = null; // 定义数据库操作
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连接数据库
        stmt = conn.createStatement(); // 创建数据库操作
        stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'测试
-A',TO_DATE('1998-09-01','yyyy-mm-dd'))") ;
        stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'测试
-B',TO_DATE('1998-09-02','yyyy-mm-dd'))") ;
        stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'测试
-C',TO_DATE('1998-09-03','yyyy-mm-dd'))") ;
        stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'测试
```

```
-D',TO_DATE('1998-09-04','yyyy-mm-dd'))") ;
        stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'测试
-E',TO_DATE('1998-09-05','yyyy-mm-dd'))") ;
        int len[] = stmt.executeBatch(); // 更新操作，返回更新的记录数
        System.out.println("更新了" + len.length + "条记录。");
        conn.close(); // 数据库的操作必须关闭
    }
}
```

此时，一次性的会将所有的 SQL 语句发送到数据库之中执行，一次性更新 5 条记录。那么如果现在假设这 5 条记录都是有关联的 5 条。

在使用批处理的操作中发现，如果中间有一条语句出错了，则默认情况下是将出错之前的代码进行提交，这是由于 JDBC 采用了自动的事务提交的方式才造成的结果。

如果此时要进行事务处理的话，则需要按照如下的方式进行：

1、 取消自动提交：public void setAutoCommit(boolean autoCommit) throws SQLException
2、 执行更新操作：
3、 如果没有错误，则提交事务：public void commit() throws SQLException
4、 如果有错误，则进行回滚：public void rollback() throws SQLException

```
package org.lxh.jdbcdemo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
public class BatchDemo {
    public static final String DBDRIVER = "oracle.jdbc.driver.OracleDriver";
    public static final String DBURL = "jdbc:oracle:thin:@localhost:1521:mldn";
    public static final String DBUSER = "scott";
    public static final String DBPASSWORD = "tiger";
    public static void main(String[] args) throws Exception {
        Connection conn = null; // 表示的是数据库连接
        Statement stmt = null; // 定义数据库操作
        Class.forName(DBDRIVER); // 1、加载数据库驱动程序
        conn = DriverManager.getConnection(DBURL, DBUSER, DBPASSWORD); // 连接数据库
        conn.setAutoCommit(false) ;// 取消自动提交
        stmt = conn.createStatement(); // 创建数据库操作
        try{
            stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'
测试-A',TO_DATE('1998-09-01','yyyy-mm-dd'))") ;
            stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'
测试-B',TO_DATE('1998-09-02','yyyy-mm-dd'))") ;
            stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'
测试-'C',TO_DATE('1998-09-03','yyyy-mm-dd'))") ;
            stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'
测试-D',TO_DATE('1998-09-04','yyyy-mm-dd'))") ;
            stmt.addBatch("INSERT INTO myuser(id,name,birthday) VALUES (myseq.nextVal,'
测试-E',TO_DATE('1998-09-05','yyyy-mm-dd'))") ;
```

```
            int len[] = stmt.executeBatch(); // 更新操作，返回更新的记录数
            System.out.println("更新了" + len.length + "条记录。");
            conn.commit() ;// 提交事务
        }catch(Exception e){
            conn.rollback() ;
        }
        conn.close(); // 数据库的操作必须关闭
    }
}
```

4、总结

1、 JDBC 的操作代码几乎都是固定的，麻烦就在于 SQL 语句的编写上；
2、 在开发中绝对不会使用 Statement 完成操作，而永远都只使用 PreparedStatement 接口完成；
3、 所有的结集合都使用 ResultSet 进行保存。

5、作业

现在要求使用 JDBC 完成雇员表的 CRUD 操作，使用 emp 表中的如下字段：empno、ename、job、hiredate、sal、comm。

完成的功能：增加数据、修改数据、删除数据、模糊查询数据（数据的部分显示），查询全部数据量，可以根据编号查询一个雇员的信息。

完成的时候要求充分的考虑类的设计问题，把所有关于类设计的方式都用上，一定要注意，主方法中的代码越少或者是越简单越好，而且在操作的时候，主方法所在的操作类中，不能导入 java.sql 包。