1、课程名称：实例讲解

2、题目要求

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。

3、代码开发

3.1、开发口诀

为了保证可以快速的上手开发，下面给出一些基本的思考步骤：

1、 类的开发要求：

- 要根据需求写类的名称，根据要求编写属性，所有的属性必须封装，封装之后的属性如果需要设置和取得的话，则编写 setter 和 getter 方法。
- 类中可以编写构造方法，通过构造设置内容，但是必须保证一个类（简单类）中不管有多少个构造，一定要保留有一个无参构造方法，而且如果有多个构造的话，就需要按照参数的个数排列。
- 一个类永远不能去继承一个已经实现好的类，而只能继承抽象类或实现接口
- 当抽象类和接口都可以使用的时候优先考虑接口
- 没有包的类是不存在的，每个类都必须放在一个包之中
- 异常处理的标准格式
- 以后碰见取得接口实例化操作的时候一定要通过工厂返回
- 看见集合想都不想使用 Iterator 接口输出

2、 设计要求：

- 一个程序开发的时候一定要先设置标准，标准就是接口。
- 各个类之间靠接口进行交互，接口是进行解耦合操作的。
- 主方法中代码越少越简单越好，一般主方法中是不允许导入 java.sql 包的。
- 根据要求细分类的组成，类的设计原则跟现实生活是一样的。
- 在进行 JDBC 的操作中只能使用 PreparedStatement 接口。

3.2、分析题目

现在操作的是数据库中的表，那么既然是表的话，肯定属于资源层的操作，数据库实际上操作的都是数据，对于这种操作数据的开发的程序，我们可以将其称为数据访问对象（Data Access Object，DAO）。

之后又发现，表的组成感觉和类的组成非常的相似。

- 表 à 类
- 数据 à 对象

从这种对应关系上可以发现，一个对象实际上就可以表示每一条记录。

现在将一张表通过一个类的形式映射出来：

- emp 表 à Emp 类

不管以后如何操作每一个 Emp 的对象肯定都表示一张表的一条记录，储存的是值，既然是值，就可以将其用另外一个名字替代：VO（Value Object，值对象）。

接口的命名规范和类的命名规范是完全一样的，那么如果现在给出一个 EmpDAO。那么为了区分，在接口前加一个字母“I”，所以接口：IEmpDAO。

对于数据库而言程序中关心的就是 CRUD，其中增加、修改、删除都属于数据库的更新操作，而查询属于另外一种操作，那么如果以后要想从方法中区分两种类型的不同，就给出一个命名规范：

- 所有的更新操作的方法命名格式：doXxx()
- 所有查询操作的方法命名格式：findXxx()、getXxx()

```
package org.lxh.oracle.dao;
import java.util.List;
import org.lxh.oracle.vo.Emp;
public interface IEmpDAO {
    /**
     * 完成数据库的增加操作
     * @param emp 增加的VO对象，里面保存着具体的内容
     * @return 是否成功的标记
```

```
     * @throws Exception 如果有异常交给被调用处处理
     */
    public boolean doCreate(Emp emp) throws Exception;
    /**
     * 完成数据库的修改操作
     * @param emp 增加的VO对象，里面保存着具体的内容
     * @return 是否成功的标记
     * @throws Exception 如果有异常交给被调用处处理
     */
    public boolean doUpdate(Emp emp) throws Exception;
    /**
     * 数据库的删除操作
     * @param empno 要删除的编号
     * @return 是否删除成功的标记
     * @throws Exception 有异常交给被调用处处理
     */
    public boolean doDelete(int empno) throws Exception;

    /**
     * 根据编号查询单条记录
     * @param empno 雇员编号
     * @return 一个对象，如果没有查询到返回null
     * @throws Exception 如果有异常交给被调用处处理
     */
    public Emp findById(int empno) throws Exception ;
    /**
     * 查询全部记录
     * @param keyWord 查询关键字
     * @return 查询的结果集
     * @throws Exception
     */
    public List<Emp> findAll(String keyWord) throws Exception ;
    /**
     * 查询数据库，可以进行部分的数据显示
     * @param keyWord 查询关键字
     * @param currentPage 当前所在的页
     * @param lineSize 每页显示多少条记录
     * @return 查询结果集
     * @throws Exception
     */
    public List<Emp> findAll(String keyWord,int currentPage,int lineSize) throws
Exception ;

    /**
```

```
     * 查询符合条件的记录数
     * @param keyWord 模糊查询关键字
     * @return 查询的记录数
     * @throws Exception 有异常交给被调用处处理
     */
    public int getAllCount(String keyWord) throws Exception ;
}
```

下面肯定要实现此标准，但是在实现此标准中有一个问题需要考虑了。

在进行 JDBC 的操作中，可以发现步骤如下：

1、 加载驱动程序

2、 取得数据库连接

3、 进行数据库操作 à 这个是操作的重点

4、 关闭数据库

对于数据库的打开和关闭的操作一定要注意，由于其是固定的代码，所以，可以将其单独定义成一个新的类，此类专门负责数据库的连接与关闭：DatabaseConnection。

理论上讲，当一个接口定义完成之后，下面要定义的是实现接口的具体实现类。

```
    @Override
    public boolean doCreate(Emp emp) throws Exception {
        String sql = "INSERT INTO emp(empno,ename,job,hiredate,sal,comm) VALUES
(?,?,?,?,?,?)";
        boolean flag = false;
        DatabaseConnection dbc = new DatabaseConnection();
        pstmt = dbc.getConnection().prepareStatement(sql);
        pstmt.setInt(1, emp.getEmpno());
        pstmt.setString(2, emp.getEname());
        pstmt.setString(3, emp.getJob());
        pstmt.setDate(4, new java.sql.Date(emp.getHiredate().getTime()));
        pstmt.setFloat(5, emp.getSal());
        pstmt.setFloat(6, emp.getComm());
        if (pstmt.executeUpdate() > 0) {
            flag = true ;
        }
        pstmt.close() ;
        dbc.close();
        return flag;
    }
```

以上确实实现了具体的接口的操作，但是这种代码会存在以下问题：

- 危险性：因为本代码一旦出错之后数据库无法关闭。
- 代码结构：应该划分出代理主题和真实主题。代理负责数据库打开和关闭，并且调用真实主题。

此时，可以发现程序中的主方法的操作代码非常的简单，那么这种设计模式在开发中就称为 DAO 设计模式，DAO 设计模式中有以下几部分组成：

1、 VO 类：用于映射与一张表的关系

2、 DAO 接口：定义一个项目中一张表的全部操作方法

3、 编写实现类：只完成具体的数据库的操作，而不关心如何打开和连接

4、 编写代理类：代理类负责数据库的打开和关闭，并且调用真实实现类

5、 数据库连接类：专门用于取得和关闭数据库连接的操作

6、 工厂类：取得接口实例

3.3、任务二

在 emp 表中还存在着一个 mgr 的字段，现在要求加入此字段的操作，即：一个雇员添加的时候可以输入领导编号，查询全部雇员信息的时候也要求可以查询出领导的编号和姓名。

在增加的时候就是在 Emp 类中增加了一个自身的关联，而且可以发现，表中的关系一定要在类中有所体现，但是对于实现类操作代码肯定就麻烦了，因为要考虑到关系的维护，因为之前讲解所有 Java 类的关系的时候都是采用手工的方式设置的，而现在是通过数据的保存自动进行设置的。

所有的关系先从 VO 下手修改，因为 VO 本身与表就是完全对应的，之后再考虑实现类的问题。

当然，在实际的开发中，一般接口定义完之后是不会轻易更改，但是现在的开发是采用逐步的方式分析的，所以以后的接口有可能更改。

3.4、任务三

将部门表的操作也形成这种 DAO 的操作模式，但是操作的方法可以简单：增加、修改、删除、模糊查询、根据编号查询，不需要分页显示。

部门的基本操作和 EMP 是完全一样的。

3.5、任务四

在 emp 表中还存在着一个 deptno 的字段，要求也可以将其的关系设置上去，而且可以在添加的时候添加部门的编号，查询的时候可以查询出一个部门的信息。

一个部门要包含多个雇员。每个雇员属于一个部门。

如果现在要想取得一个部门的完整信息，则一定要取出一个部门中所有的雇员信息。

4、总结

本程序作为一个较为常用的设计思路，在各个开发中都会其作用，主要是研究思想。

VO 的关系就等同于表的关系。

本程序只是在数据层上的操作：

客户端 **à** 工厂 **à** 代理 **à** 真实