**数据库课程设计报告**

**-------物业管理系统**

目 录

[一、系统开发平台](#_Toc26619)

[二、数据库规划](#_Toc14844)

[2.1 任务陈述](#_Toc23857)

[2.2 任务目标](#_Toc2711)

[三、系统定义](#_Toc21830)

[3.1 系统边界](#_Toc29896)

[3.2 用户视图](#_Toc28420)

[四、需求分析](#_Toc12215)

[4.1 用户需求说明](#_Toc29271)

[4.1.1数据需求](#_Toc9088)

[4.1.2 事务需求](#_Toc20317)

[4.2 系统需求说明](#_Toc20077)

[五、数据库逻辑设计](#_Toc18231)

[5.1 ER图](#_Toc16663)

[5.2 数据字典](#_Toc5167)

[5.2.1.Boss表格](#_Toc20585)

[5.2.2.Employee表格](#_Toc21545)

[5.2.3.Manager表格](#_Toc21718)

[5.2.4.Worker表格](#_Toc20342)

[5.2.5.House表格](#_Toc18510)

[5.2.6.Owner表格](#_Toc18317)

[5.2.7.Fee表格](#_Toc24748)

[5.3 关系模式图](#_Toc5168)

[六、数据库物理设计](#_Toc29358)

[6.1 索引](#_Toc19204)

[6.2 视图](#_Toc2501)

[6.3 安全机制](#_Toc19625)

[6.4 其他](#_Toc17205)

[七、应用程序设计](#_Toc11221)

[7.1 功能模块](#_Toc12125)

[7.2 界面设计](#_Toc17916)

[7.3 事务设计](#_Toc21028)

[7.3.1 找回密码](#_Toc25231)

[**7.3.2 注册**](#_Toc25559)

[**7.3.3 登录**](#_Toc5335)

[**7.3.4 boss**](#_Toc15873)

[**7.3.5 manager**](#_Toc15856)

[**7.3.6 worker**](#_Toc31753)

[八、测试和运行](#_Toc14206)

[8.1身份选择](#_Toc21901)

[8.2 找回密码](#_Toc13574)

[8.3 注册](#_Toc10459)

[8.4 boss登录](#_Toc6419)

[8.5 boss管理界面](#_Toc7671)

[8.6 manager登录](#_Toc9364)

[8.7 manager管理界面](#_Toc20240)

[8.8 worker登录](#_Toc23060)

[8.9 worker管理界面](#_Toc2383)

[九、总结](#_Toc31671)

[附. 参考文献](#_Toc4451)

# 一、系统开发平台

题目：物业管理系统

开发工具：Eclipse Java Oxygen

数据库：MySql

# 二、数据库规划

## 2.1 任务陈述

物业管理人员在日常工作中中需要对员工个人信息，业主信息，房屋信息进行管理，还要收取各户所产生的费用等工作，利用物业信息系统及时了解各个环节中信息的变更，有利于提高管理效率。

## 2.2 任务目标

系统开发的总体任务是实现物业管理部门各种信息的系统化、规范化和自动化。主要有以下几个方面：

\*主要信息管理：业主信息、房屋信息

\*\*费用收取： 物业费、水电煤气、有线电视、供暖、车位

\*\*\*信息查询及报表：月报表、季报表、年报表

根据功能设计如下数据库结构：

Boss：编号(bno)，姓名(bname)，照片(bimage)，密码(secret)

employee：编号(eno)，姓名(ename)，性别(esex)，年龄(eage)，手机号(ephone)，照片(eimage)，密码(secret)，老板编号(bno)

manager：编号(eno)，薪水(salary)

worker：编号(eno)，薪水(salary)，主管编号(mno)

owner: 编号(ono)，姓名(oname)，性别(osex)，年龄(oage)，手机号(ophone)，照片(oimage)

House：编号(hno)，楼号(buildingNo)，单元号(unitNo)，房间号(roomNo)，屋主编号(oNo)

Fee：房屋编号(hno)，物业费(propertyFee)，有线电视费(cableFee)，水电气费(utilityFee)，暖气费(heatingFee)，车位费(parkingFee)

# 三、系统定义

## 3.1 系统边界

## 3.2 用户视图

用户 需求

Boss 基本信息（编号，姓名，照片，密码）

（老板） 修改个人信息，雇员（录取/开除），主管（擢升/降职），设置工资（员 工/主管）

Employee 基本信息（员工编号，姓名，性别，年龄，手机号，照片，密码，老

（雇员） 板编号）

Manager 基本信息（工号，工资）

（主管） 修改信息（个人/员工），查看报表

Worker 基本信息（工号，工资，主管编号）

（职员） 修改个人密码，修改业主信息（业主/房屋），收取业主缴纳费用，添加 加房子待缴纳费用

# 四、需求分析

## 4.1 用户需求说明

### 4.1.1数据需求

1.物业管理部门上属小区所有者，归一个或多个老板（boss）所有，老板有最大权力，包括

\*个人信息管理，修改姓名/照片/密码

\*录取员工开除员工，也只有老板可以录取或开除员工，录取是模拟将投递简历的雇员（employee）设置为正式员工（worker），开除是将所有表里（employee+worker+manager）这个员工的信息都删除掉，

\*擢升主管降职主管，也只有老板可以擢升或降职主管，擢升是将正式员工（worker）的信息添加到主管（manager），降职是将主管里信息删除保留员工表里信息

\*设置工资，包括员工和主管

2.老板下管主管和员工，主管由老板从正式职员提拔而来，主要负责

\*信息管理，主管可以修改个人信息，也可以修改自己管理的员工信息

\*查看月/季/年报表

1. 员工是由老板正式录取的雇员，负责大量物业部门日常业务

\*个人信息管理，此处员工只可以修改个人密码

\*业主服务，修改所选业主信息，收取该业主名下房屋的费用，可收取部分费用，满足业主手头现金不足的情况

\*房屋管理，修改所选房屋信息，添加房屋的费用，按月是有默认配额的，超过则需要手动由职员添加

### 4.1.2 事务需求

数据录入

1. 录入雇员（employee）信息
2. 录入房屋费用
3. 录入房屋业主
4. 录入业主名下房屋

数据更新/删除

1.更新基本信息（老板/雇员/职员/主管/房屋/业主）

2.更新费用

3.删除员工所有信息（开除）

数据查询

1.查询业主/房屋信息

2.查询报表信息（月/季/年）

## 4.2 系统需求说明

初始数据库大小：

1. 三名老板
2. 大约100名求职雇员
3. 大约100个房屋
4. 大约20户业主
5. 0个正式职员
6. 0个主管
7. 五种费用

数据库变化速度：

1.每天大约有10人录取，需添加职员信息。

2.每天大约有10人开除，需删除员工信息。

记录查找的类型和平均数量：

1.查询员工情况大约一天20次

2.查询主管情况大约一天20次

3.查询房屋情况大约一天30次

4.查询业主情况大约一天30次

网络和共享需求：

1. 所有用户必须安全的和总部中央数据库网络互连

2. 必须能够支持至少3名成员同时访问.需要考虑这么大数量并发访问的许可需求。

性能：

1.单个记录查询时间少于1秒，高峰期少于5秒

2. 多个记录查询时间少于5秒，高峰期少于10秒

3. 更新/保存记录时间少于1秒，高峰期少于5秒

安全性：

1. 数据库必须有密码保护

2. 每个用户分配特定的用户视图所应有的访问权限

3. 用户只能在适合他们完成工作的需要的窗口中看到需要的数据

备份和恢复

* 每天24点备份

用户界面：

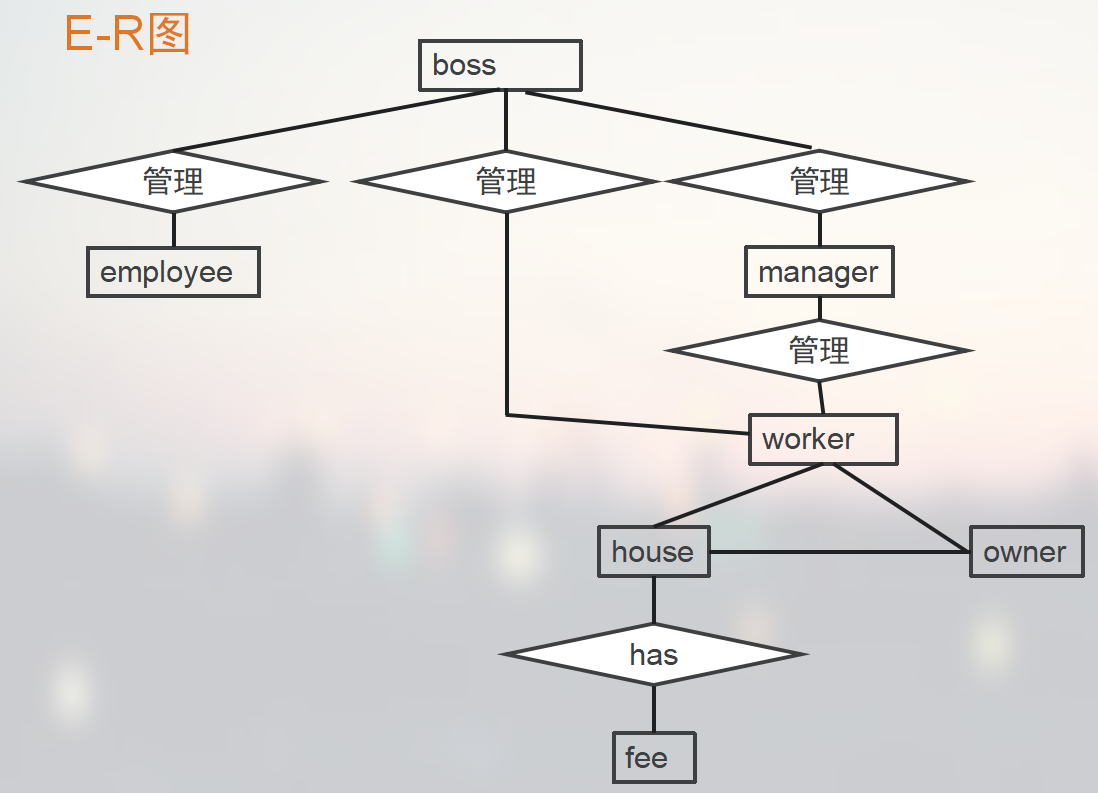
* 菜单驱动，联机帮助

法律问题：

* 对员工和业主信息管理，遵守法律

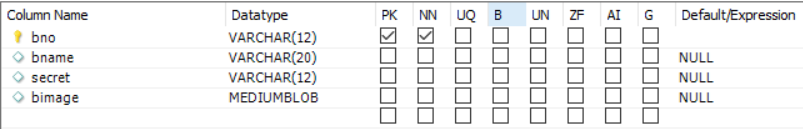
# 五、数据库逻辑设计

## 5.1 ER图



## 5.2 数据字典

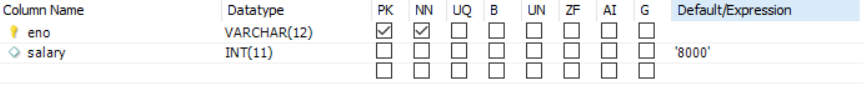
### 5.2.1.Boss表格



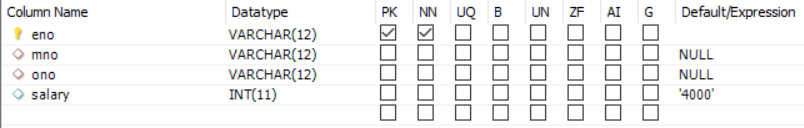
### 5.2.2.Employee表格



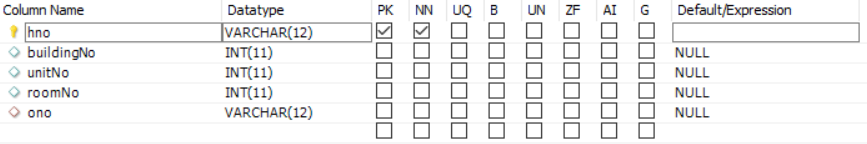
### 5.2.3.Manager表格



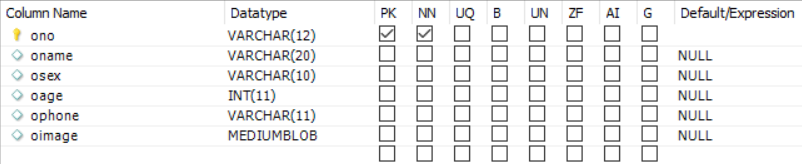
### 5.2.4.Worker表格



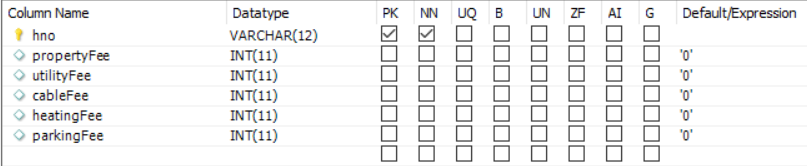
### 5.2.5.House表格



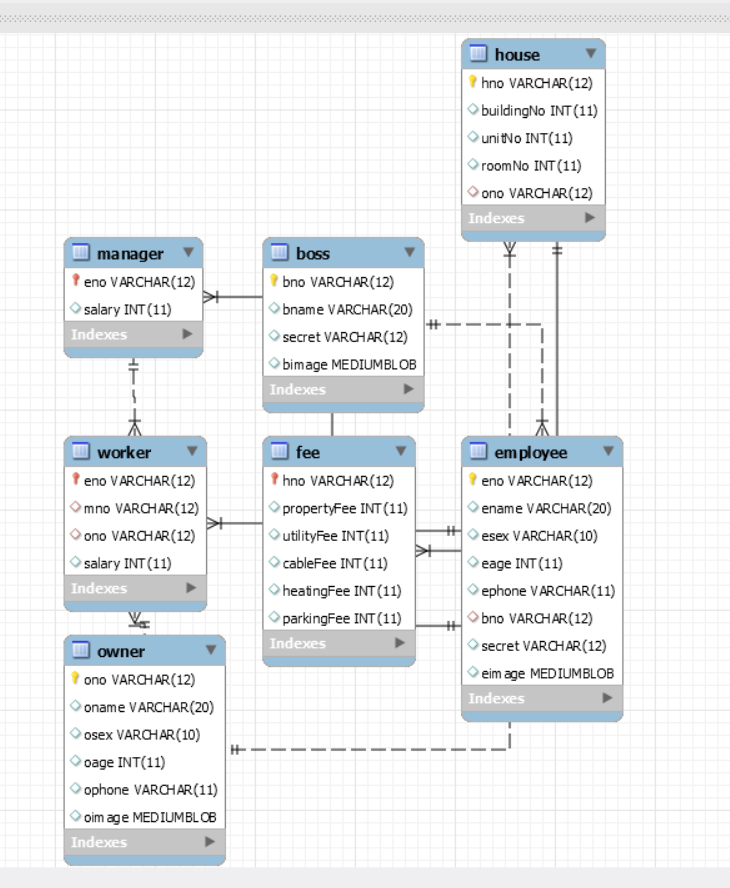
### 5.2.6.Owner表格



### 5.2.7.Fee表格



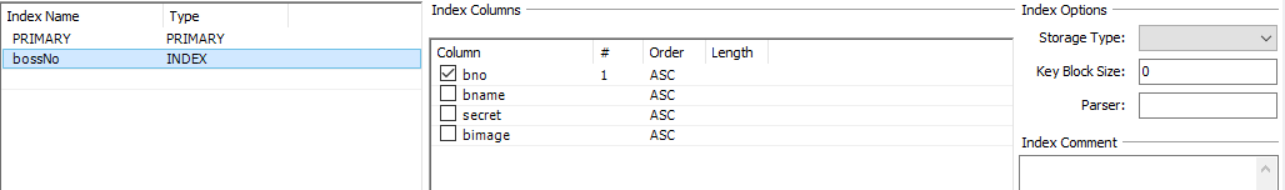
## 5.3 关系模式图



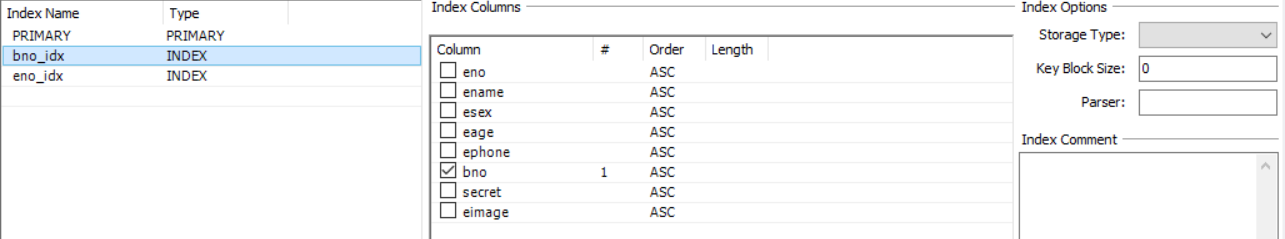
# 六、数据库物理设计

## 6.1 索引

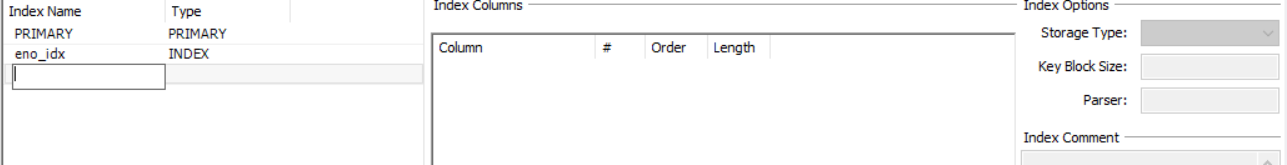
boss表，以bno为唯一索引



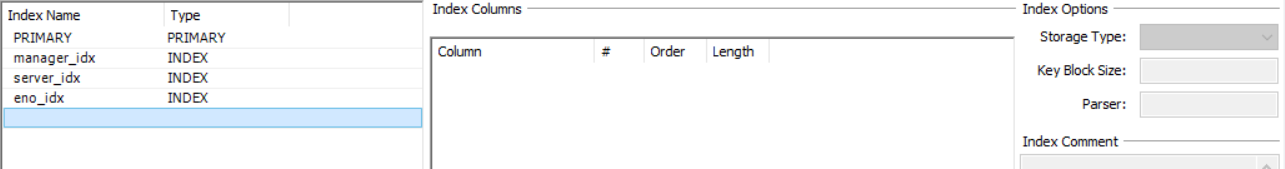
Employee表，已eno和bno 为索引



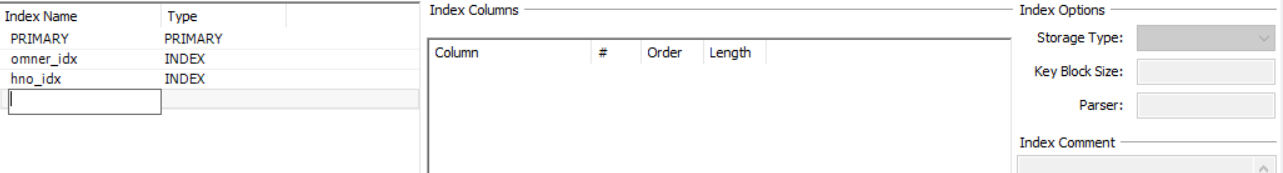
Manager表，以eno为唯一索引



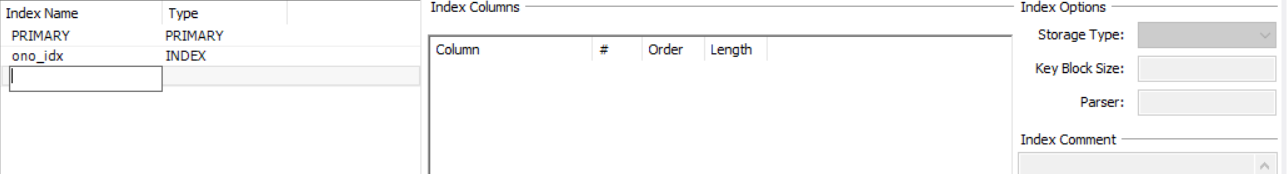
Worker表，以eno，hno，mno为索引



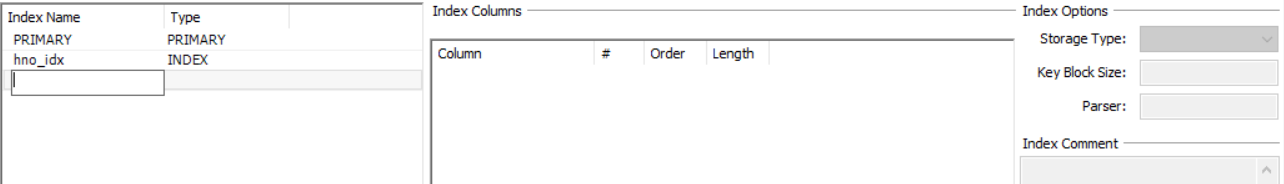
House表，以hno和ono为索引



Owner表，以ono为唯一索引



Fee表，以hno为唯一索引



## 6.2 视图

没有使用视图

## 6.3 安全机制

系统安全、数据安全

数据安全：

程序启动需要经过MySql 登陆认证，登陆ID 和密码正确才能启动程序。

程序内部的任何会引起数据库改动的操作（增删改）均经过了严密的审查判定，以确保数据库的准确性和一致性。

系统安全：

登陆需经过身份认证，即身份口令密码一致才可登录。

不同的用户有不同的权限，用户只能使用分配给他的权限，无法越权操作。

## 6.4 其他

删除雇员数据时：

为保证数据完整性，需要同时从employee worker manager删除信息

——解决方案？

在删除后加入完整性约束语句

String sql1 = "delete from employee where eno = \'"+wNo+"\'";

String sql2 = "delete from worker,manager where"

+ "( (wno not in(select eno from employee)) or "

+ " (mno not in(select eno from employee) )";//数据完整性

又报错：

Cannot delete or update a parent row: a foreign key constraint fails

(`managesystem`.`worker`, CONSTRAINT `isa` FOREIGN KEY (`eno`) REFERENCES `employee` (`eno`) ON DELETE NO ACTION ON UPDATE NO ACTION)

将外键中删除约束由NOACTION改为CASCADE：

1. CASCADE: 从父表中删除或更新对应的行，同时自动的删除或更新自表中匹配的行。ON DELETE CANSCADE和ON UPDATE CANSCADE都被InnoDB所支持。
2. SET NULL: 从父表中删除或更新对应的行，同时将子表中的外键列设为空。注意，这些在外键列没有被设为NOT NULL时才有效。ON DELETE SET NULL和ON UPDATE SET SET NULL都被InnoDB所支持。
3. NO ACTION: InnoDB拒绝删除或者更新父表。

**--**删除外键 ALTER TABLE worker DROP FOREIGN KEY isa

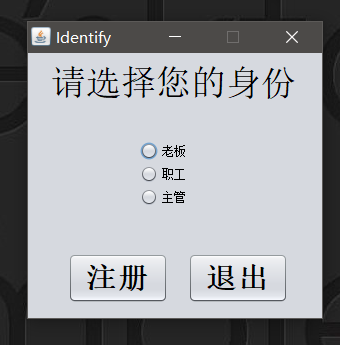
**--**添加外键 ALTER TABLE `worker` ADD CONSTRAINT `isa` FOREIGN KEY ( `eno` ) REFERENCES `category` ( `eno` ) ON DELETE CASCADE ON UPDATE CASCADE

# 七、应用程序设计

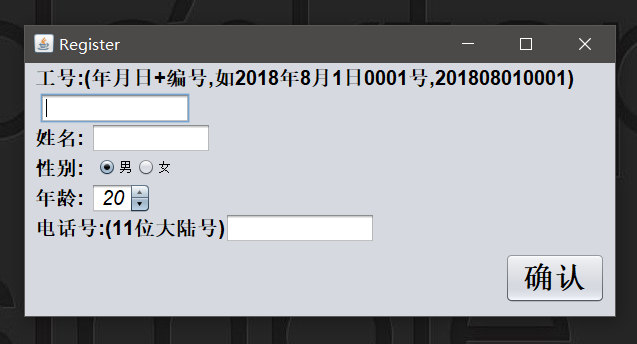
## 7.1 功能模块

## 7.2 界面设计

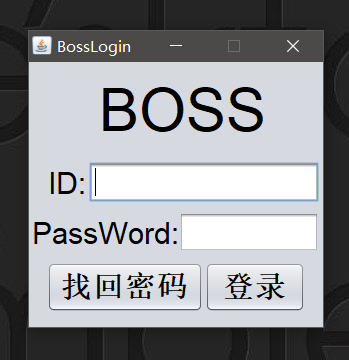
**7.2.1 身份选择界面**



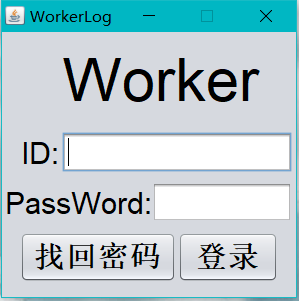
**7.2.2 注册界面**



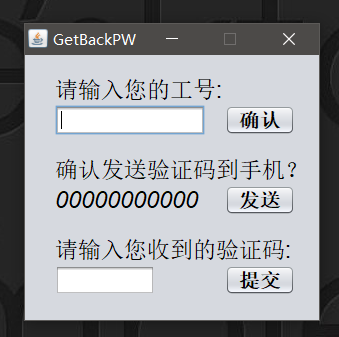
**7.2.3 登录界面**







**7.2.4 找回密码界面**



**7.2.5 boss界面**



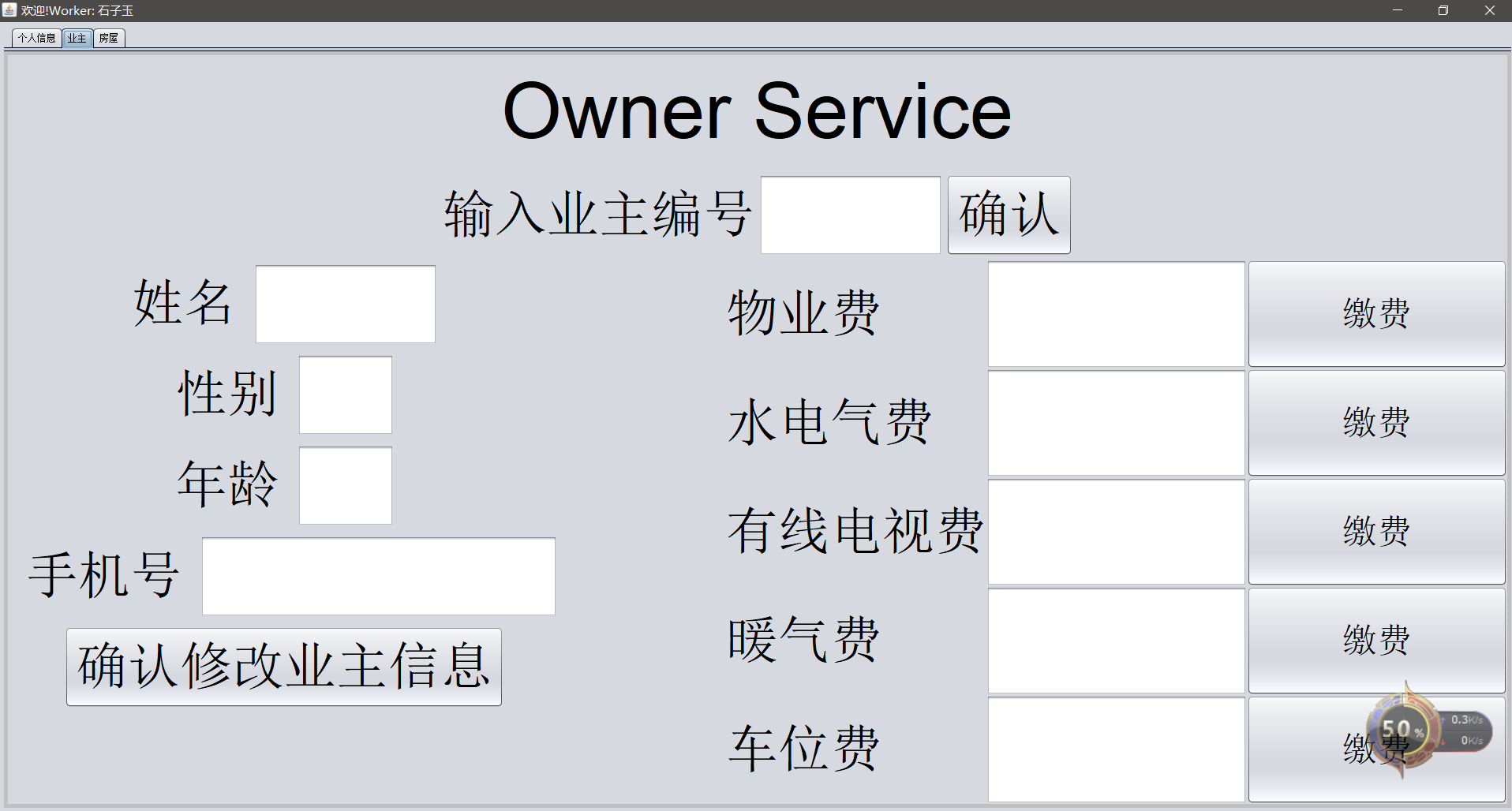


**7.2.6 manager界面**



**7.2.7 worker界面**







## 7.3 事务设计

关键或复杂的事务操作的设计，可附相应SQL语句进行分析介绍。

### 7.3.1 找回密码



主要代码：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//生成六位随机数

**public** **static** String getRandNum() {

String randNum = **new** Random().nextInt(1000000)+"";

//System.out.println("生成"+randNum);

**if** (randNum.length()!=6) {

**return** *getRandNum*();

}

**return** randNum;

}

//

**public** **static** String queryArguments(String ACCOUNT\_SID,String AUTH\_TOKEN, String smsContent,String to) {

String timestamp = *getTimestamp*(); //时间戳

String sig = *MD5*(ACCOUNT\_SID,AUTH\_TOKEN,timestamp);//签名认证

String str = "accountSid="+ACCOUNT\_SID+"&smsContent="+

smsContent+"&to="+to+"&timestamp="+timestamp+"&sig="+sig;

**return** str;

}

/\*MD5加密 \*/

**public** **static** String MD5(String... args){ //动态参数

StringBuffer result = **new** StringBuffer();

**if** (args == **null** || args.length == 0) {

**return** "";

} **else** {

StringBuffer str = **new** StringBuffer();

**for** (String string : args) {

str.append(string);

}

System.***out***.println("加密前：\t"+str.toString());

**try** {

MessageDigest digest = MessageDigest.*getInstance*("MD5");//信息摘要算法

**byte**[] bytes = digest.digest(str.toString().getBytes());

**for** (**byte** b : bytes) {

String hex = Integer.*toHexString*(b&0xff); //转化十六进制

**if** (hex.length() == 1) {

result.append("0"+hex);

}**else**{

result.append(hex);

}

}

} **catch** (NoSuchAlgorithmException e) {

e.printStackTrace();

}

}

System.***out***.println("加密后：\t"+result.toString());

**return** result.toString();

}

/\*获取时间戳\*/

**public** **static** String getTimestamp(){

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyyMMddHHmmss");

Date date = **new** Date();

**return** sdf.format(date);

}

///////////////////////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////////////////////////////

/\*用户ID\*/

**public** **static** **final** String ***ACCOUNT\_SID*** = "c42e54d811274649b45bcc0270ea510c";

/\*密钥\*/

**public** **static** **final** String ***AUTH\_TOKEN*** = "a9b0405\*\*\*\*\*\*\*ab6";

/\*请求地址前半部分\*/

**public** **static** **final** String ***BASE\_URL*** = "https://api.miaodiyun.com/20150822/industrySMS/sendSMS";//请求地址

**public** **static** String *randNum* = *getRandNum*();

**public** **static** String *smsContent* = "【浩龙科技】您的验证码为"+*randNum*+"，请于"+5+"分钟内正确输入，如非本人操作，请忽略此短信。";

/\*(获取短信验证码)\*/

**public** **static** String getResult(String to) {

*randNum* = *getRandNum*();

String smsContent = "【浩龙科技】您的验证码为"+*randNum*+"，请于"+5+"分钟内正确输入，如非本人操作，请忽略此短信。";

//这里的randNum 和 smsContent和上面的静态变量是一样的，可删除可保留

String args = *queryArguments*(***ACCOUNT\_SID***, ***AUTH\_TOKEN***, smsContent, to);

OutputStreamWriter out = **null**;

//InputStream in = null;

BufferedReader br = **null**;

StringBuffer sb = **new** StringBuffer();

**try** {

URL url = **new** URL(***BASE\_URL***);

URLConnection connection = url.openConnection(); //打开链接

connection.setDoOutput(**true**);

connection.setDoInput(**true**);

connection.setConnectTimeout(5000); //设置链接超时

connection.setReadTimeout(10000); //设置读取超时

//提交数据

out = **new** OutputStreamWriter(connection.getOutputStream(),"utf-8");

out.write(args);

out.flush();

//读取返回数据

br = **new** BufferedReader(**new** InputStreamReader(connection.getInputStream()));

String line = "";

**while**((line = br.readLine())!=**null**){

sb.append(line);

}

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

**try** {

**if** (br!=**null**) {

br.close();

}

**if** (out!=**null**) {

out.close();

}

} **catch** (Exception e2) {

e2.printStackTrace();

}

}

/\*JSONObject jsonObject = JSONObject.fromObject(sb.toString());

System.out.println(jsonObject);

Object object = jsonObject.get("respCode");

System.out.println("状态码："+object+"验证码："+randNum);

System.out.println(!object.equals("00000"));

if (!object.equals("00000")) {

return object.toString();

}else{

return randNum;

}\*/**return** *randNum*;

}

### 7.3.2 注册



主要代码;

NO = noIn.getText();

System.***out***.println(NO);

NAME = nameIn.getText();

System.***out***.println(NAME);

//String enable="false";

Enumeration<AbstractButton> radioBtns=sexBox.getElements();

**while** (radioBtns.hasMoreElements()) {

AbstractButton btn = radioBtns.nextElement();

**if**(btn.isSelected()){

SEX=btn.getText();

**break**;

}

}

//if(sexBox.isSelected(m))

System.***out***.println(SEX);

AGE = ageJsp.getValue().toString();

System.***out***.println(AGE);

PHONE = phoneIn.getText();

System.***out***.println(PHONE);

String sql = "insert into employee(eno,ename,esex,eage,ephone) "

+ "values(\'" +NO+ "\',\'" +NAME+ "\',\'" +SEX+ "\'," +AGE+ ",\'" +PHONE+"\')";

**try** {

DBLink.*st*.execute(sql);

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

### 7.3.3 登录



主要代码：

**\*登录**

String id=textField.getText();

String password=**new** String(passwordField.getPassword());

System.***out***.println("Login! id: "+id+",password: "+password);

//String PASSWORD=DBManager.getPassword(id);

String PASSWORD=*getPW*(id);

DBLink.*dbLink*();

**try** {

String sql = "select bno from boss where bno = \'"+id+"\'";

ResultSet rs;

rs = DBLink.*st*.executeQuery(sql);

**if**(password.equals(PASSWORD) && rs.next()){

//启动主窗口

**try** {

**new** BossWindow(id);

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

//关闭当前按钮

//getContntPane().exit(0);

dispose();

}

**else** **if**(rs.next() == **false**){

dispose();

**final** String s= "\*\*\*不存在该boss\*\*\*\n"

+ "\*\*\*\*\*\*\*请重新输入\*\*\*\*\*\*\*";

**try**{

**new** Thread(**new** Runnable(){

**public** **void** run() {

// **TODO** Auto-generated method stub

JOptionPane.*showConfirmDialog*(**new** BossLogin(),s, "错误",

JOptionPane.***DEFAULT\_OPTION***);

}

}).start();

Thread.*sleep*(100);

}**catch** (InterruptedException e1) {

// **TODO** Auto-generated catch block

//new Login();

}

}

**else**{

dispose();

**final** String s= "\*\*用户名和密码不匹配\*\*\n"

+ "\*\*\*\*\*\*\*请重新输入\*\*\*\*\*\*\*";

**try**{

**new** Thread(**new** Runnable(){

**public** **void** run() {

// **TODO** Auto-generated method stub

JOptionPane.*showConfirmDialog*(**new** BossLogin(),s, "错误",

JOptionPane.***DEFAULT\_OPTION***);

}

}).start();

Thread.*sleep*(100);

}**catch** (InterruptedException e1) {

// **TODO** Auto-generated catch block

//new Login();

}

}

} **catch** (SQLException e2) {

// **TODO** Auto-generated catch block

e2.printStackTrace();

}

### 7.3.4 boss



主要代码;

**\*修改照片**

String sql = "update boss set bimage = ? where bno = \'"+bno+"\'";

ImageDemo.*putImg*(sql);

String sql1 = "select bimage from boss where bno = \'"+bno+"\'";

String p;

**try** {

p = ImageDemo.*blobRead*(sql1, "test1.jpg");

//new ManagerWindow(no,p);

photo.setIcon(**new** ImageIcon(p));

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**\*录取**

wNo = chooseNo.getText();

String sql1 = "insert into worker(eno) value("

+ "(select eno from employee where eno = \'"+wNo+"\') )";

String sql = "select eno from employee where eno = \'"+wNo+"\'";

String sql2 = "update employee set bno = \'"+bno+"\' where eno = \'"+wNo+"\'";

**try** {

DBLink.*dbLink*();

ResultSet rs = DBLink.*st*.executeQuery(sql);

**if**(rs.next()){

DBLink.*st*.execute(sql2);

DBLink.*st*.execute(sql1);

System.***out***.println("Add to Worker: "+wNo);

}**else**{

**final** String s= "\*\*\*不存在该employee\*\*\*\n"

+ "\*\*\*\*\*\*\*请重新输入\*\*\*\*\*\*\*";

**try**{

**new** Thread(**new** Runnable(){

**public** **void** run() {

// **TODO** Auto-generated method stub

JOptionPane.*showConfirmDialog*(worker,s, "错误",

JOptionPane.***DEFAULT\_OPTION***);

chooseNo.setText("");

}

}).start();

Thread.*sleep*(100);

}**catch** (InterruptedException e1) {

// **TODO** Auto-generated catch block

//new Login();

}

}

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**\*开除**

wNo = chooseNo.getText();

String sql1 = "delete from employee where eno = \'"+wNo+"\'";

String sql = "select eno from employee where eno = \'"+wNo+"\'";

**try** {

ResultSet rs = DBLink.*st*.executeQuery(sql);

**if**(rs.next()){

//DBLink.st.execute(sql3);

//DBLink.st.execute(sql5);

DBLink.*st*.execute(sql1);

//DBLink.st.execute(sql4);

//DBLink.st.execute(sql6);

//DBLink.st.execute(sql2);

System.***out***.println("Delete from Employee: "+wNo);

}**else**{

**final** String s= "\*\*\*不存在该employee\*\*\*\n"

+ "\*\*\*\*\*\*\*请重新输入\*\*\*\*\*\*\*";

**try**{

**new** Thread(**new** Runnable(){

**public** **void** run() {

// **TODO** Auto-generated method stub

JOptionPane.*showConfirmDialog*(worker,s, "错误",

JOptionPane.***DEFAULT\_OPTION***);

chooseNo.setText("");

}

}).start();

Thread.*sleep*(100);

}**catch** (InterruptedException e1) {

// **TODO** Auto-generated catch block

//new Login();

}

}

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**\*设置员工工资**

wSalary = setSalary.getText();

String sql = "update worker set salary = \'"+wSalary+"\'";

**try** {

DBLink.*st*.execute(sql);

System.***out***.println("Set Worker Salary: "+wSalary);

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**\*擢升管理员**

wNoM = chooseNo1.getText();

String sql = "insert into manager(eno) values("

+ "(select eno from employee where eno = \'"+wNoM+"\') )";

String sql1 = "select eno from worker where eno = \'"+wNoM+"\'";

**try** {

ResultSet rs = DBLink.*st*.executeQuery(sql1);

**if**(rs.next()){

DBLink.*st*.execute(sql);

System.***out***.println("Add to Manager: "+wNoM);

}**else**{

**final** String s= "\*\*\*不存在该worker\*\*\*\n"

+ "\*\*\*\*\*\*\*请重新输入\*\*\*\*\*\*\*";

**try**{

**new** Thread(**new** Runnable(){

**public** **void** run() {

// **TODO** Auto-generated method stub

JOptionPane.*showConfirmDialog*(worker,s, "错误",

JOptionPane.***DEFAULT\_OPTION***);

chooseNo1.setText("");

}

}).start();

Thread.*sleep*(100);

}**catch** (InterruptedException e1) {

// **TODO** Auto-generated catch block

//new Login();

}

}

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**\*降职主管**

wNoM = chooseNo1.getText();

String sql = "select eno from manager where eno = \'"+wNoM+"\'";

String sql1 = "delete from manager where eno = \'"+wNoM+"\'";

//String sql2 = "insert into worker(en where eno = \'"+wNoM+"\'";

**try** {

ResultSet rs = DBLink.*st*.executeQuery(sql);

**if**(rs.next()){

DBLink.*st*.execute(sql1);

System.***out***.println("GoDown to Worker: "+wNoM);

}**else**{

**final** String s= "\*\*\*不存在该manager\*\*\*\n"

+ "\*\*\*\*\*\*\*请重新输入\*\*\*\*\*\*\*";

**try**{

**new** Thread(**new** Runnable(){

**public** **void** run() {

// **TODO** Auto-generated method stub

JOptionPane.*showConfirmDialog*(worker,s, "错误",

JOptionPane.***DEFAULT\_OPTION***);

chooseNo1.setText("");

}

}).start();

Thread.*sleep*(100);

}**catch** (InterruptedException e1) {

// **TODO** Auto-generated catch block

//new Login();

}

}

} **catch** (SQLException e2) {

// **TODO** Auto-generated catch block

e2.printStackTrace();

}

**\*设置主管工资**

mSalary = setSalary1.getText();

String sql = "update manager set salary = \'"+mSalary+"\'";

**try** {

DBLink.*st*.execute(sql);

System.***out***.println("Set Manager Salary: "+mSalary);

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**//提取boss 信息**

**public** **static** String getBossSecret(String bno){

String secret = "";

//DBLink.dbLink();

String sql = "select \* from boss where bno = \'"+bno+"\'";

**try** {

DBLink.*dbLink*();

ResultSet st = DBLink.*st*.executeQuery(sql);

**if**(st.next()){

secret = st.getString("secret");

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

**return** secret;

}

**public** **static** String getBossName(String bno){

String name = "";

//DBLink.dbLink();

String sql = "select \* from boss where bno = \'"+bno+"\'";

**try** {

DBLink.*dbLink*();

ResultSet st = DBLink.*st*.executeQuery(sql);

**if**(st.next()){

name = st.getString("bname");

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

**return** name;

}

**//修改boss信息**

**public** **static** **void** SetBossName(String name,String bno){

//DBLink.dbLink();

String sql = "update boss set bname = \'"+name+"\' where bno = \'"+bno+"\'";

**try** {

DBLink.*dbLink*();

DBLink.*st*.executeUpdate(sql);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

}

**public** **static** **void** SetBossNo(String no,String bno){

//DBLink.dbLink();

String sql = "update boss set bno = \'"+no+"\' where bno = \'"+bno+"\'";

**try** {

DBLink.*dbLink*();

DBLink.*st*.executeUpdate(sql);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

}

**public** **static** **void** SetBossSecret(String sec,String bno){

//DBLink.dbLink();

String sql = "update boss set secret = \'"+sec+"\' where bno = \'"+bno+"\'";

**try** {

DBLink.*dbLink*();

DBLink.*st*.executeUpdate(sql);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

}

### 7.3.5 manager



主要代码：

**\*修改照片**

**同boss**

**\***  **//提取manager 信息**

**public** **static** String getSecret(String mno){

String secret = "";

//DBLink.dbLink();

String sql = "select \* from employee where eno = \'"+mno+"\'";

**try** {

DBLink.*dbLink*();

ResultSet st = DBLink.*st*.executeQuery(sql);

**if**(st.next()){

secret = st.getString("secret");

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

**return** secret;

}

**public** **static** String getName(String mno){

String name = "";

//DBLink.dbLink();

String sql = "select \* from employee where eno = \'"+mno+"\'";

**try** {

DBLink.*dbLink*();

ResultSet st = DBLink.*st*.executeQuery(sql);

**if**(st.next()){

name = st.getString("ename");

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

**return** name;

}

**public** **static** String getSex(String mno){

String sex = "";

DBLink.*dbLink*();

String sql = "select \* from employee where eno = \'"+mno+"\'";

**try** {

ResultSet st = DBLink.*st*.executeQuery(sql);

**if**(st.next()){

sex = st.getString("esex");

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

**return** sex;

}

**public** **static** String getAge(String mno){

String age = "";

DBLink.*dbLink*();

String sql = "select \* from employee where eno = \'"+mno+"\'";

**try** {

ResultSet st = DBLink.*st*.executeQuery(sql);

**if**(st.next()){

age = st.getInt("eage")+"";

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

**return** age;

}

**public** **static** String getPhone(String mno){

String phone = "";

DBLink.*dbLink*();

String sql = "select \* from employee where eno = \'"+mno+"\'";

**try** {

ResultSet st = DBLink.*st*.executeQuery(sql);

**if**(st.next()){

phone = st.getString("ephone");

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

**return** phone;

}

**//修改manager 信息**

**public** **static** **void** Set(String mno,String name,String no,String sec,String sex,String age,String phone){

DBLink.*dbLink*();

String sql1 = "update employee set ename = \'"+name+"\' where eno = \'"+mno+"\'";

String sql2 = "update employee set eno = \'"+no+"\' where eno = \'"+mno+"\'";

String sql3 = "update employee set secret = \'"+sec+"\' where eno = \'"+mno+"\'";

String sql4 = "update employee set esex = \'"+sex+"\' where eno = \'"+mno+"\'";

String sql5 = "update employee set eage = "+age+" where eno = \'"+mno+"\'";

String sql6 = "update employee set ephone = \'"+phone+"\' where eno = \'"+mno+"\'";

**try** {

DBLink.*st*.executeUpdate(sql1);

DBLink.*st*.executeUpdate(sql2);

DBLink.*st*.executeUpdate(sql3);

DBLink.*st*.executeUpdate(sql4);

DBLink.*st*.executeUpdate(sql5);

DBLink.*st*.executeUpdate(sql6);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

}

**\*报表生成**

//利润表报表生成

**public** **static** JTable createIncomeTable(JTable table){

String[] columns={"项目","时间","收入","支出","缴税","利润"};

**int**[] columnWidth={100,80,80,60,60,60};

**int** rowHeight = 30;

table.setRowHeight(rowHeight);

DefaultTableModel model=**new** DefaultTableModel(columns,20);

table.setFont(**new** Font("",Font.***BOLD***,18));

table.setModel(model);

String[] rowData = {"","","收入总额","支出总额","缴税总额","合计利润"};

model.addRow(rowData);

String[] rowData1 = {"","","","","","",""};

model.addRow(rowData1);

TableColumnModel columnModel=table.getColumnModel();

**int** count=columnModel.getColumnCount();

**for**(**int** i=0;i<count;i++){

javax.swing.table.TableColumn column=columnModel.getColumn(i);

column.setPreferredWidth(columnWidth[i]);

}

**return** table;

}

//资产负债表生成

**public** **static** JTable createCashflowTable(JTable table){

String[] columns = {"行次","负债业主","房屋编号","欠款数","应还日期"};

**int**[] columnWidth={100,80,80,60,80};

**int** rowHeight = 30;

table.setRowHeight(rowHeight);

DefaultTableModel model=**new** DefaultTableModel(columns,20);

table.setFont(**new** Font("",Font.***BOLD***,18));

table.setModel(model);

String[] rowData = {"","","收入总额","支出总额","缴税总额","合计利润"};

model.addRow(rowData);

String[] rowData1 = {"","","","","","",""};

model.addRow(rowData1);

TableColumnModel columnModel=table.getColumnModel();

**int** count=columnModel.getColumnCount();

**for**(**int** i=0;i<count;i++){

javax.swing.table.TableColumn column=columnModel.getColumn(i);

column.setPreferredWidth(columnWidth[i]);

}

**return** table;

}

//资金流动表

**public** **static** JTable createBalanceTable(JTable table){

String[] columns = {"行次","时间","流入资金","流出资金","资金合计"};

**int**[] columnWidth={100,80,80,60,60,60};

**int** rowHeight = 30;

table.setRowHeight(rowHeight);

DefaultTableModel model=**new** DefaultTableModel(columns,20);

table.setFont(**new** Font("",Font.***BOLD***,18));

table.setModel(model);

String[] rowData = {"","","收入总额","支出总额","缴税总额","合计利润"};

model.addRow(rowData);

String[] rowData1 = {"","","","","","",""};

model.addRow(rowData1);

TableColumnModel columnModel=table.getColumnModel();

**int** count=columnModel.getColumnCount();

**for**(**int** i=0;i<count;i++){

javax.swing.table.TableColumn column=columnModel.getColumn(i);

column.setPreferredWidth(columnWidth[i]);

}

**return** table;

}

### 7.3.6 worker

****

**主要代码：**

**\*缴费**

String no = ownerNo.getText();

String sql = "select propertyFee from fee where hno in ("

+ "select hno from house where ono = \'"+no+"\')";

**try** {

DBLink.*dbLink*();

ResultSet rs = DBLink.*st*.executeQuery(sql);

**if**(rs.next()){

String propertyFee = rs.getInt("propertyFee")+"";

String propertyAccept = property.getText();

String sql1 = "update fee set propertyFee = "+propertyFee+"-"+propertyAccept

+ " where hno in (select hno from house where ono = \'"+no+"\')";

DBLink.*st*.executeUpdate(sql1);

}

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

**\*添加费用**

String hNo = houseNo.getText();

String utility = utilityIn.getText();

**try** {

DBLink.*dbLink*();

String sql = "select utilityFee from fee where hno = \'"+hNo+"\'";

ResultSet rs = DBLink.*st*.executeQuery(sql);

**if**(rs.next()){

utilityAdd = rs.getInt("utilityFee");

}

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

String sql = "update fee set utilityFee = "+utility+"+"+utilityAdd+" where hno = \'"+hNo+"\'";

**try** {

DBLink.*dbLink*();

DBLink.*st*.executeUpdate(sql);

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}**finally**{

DBLink.*release*(DBLink.*con*, DBLink.*st*, DBLink.*rs*, DBLink.*pst*);

}

# 八、测试和运行

介绍系统的测试情况，可提供几个测试用例。

可选：生成大数据量的仿真数据，进行性能测试的情况。

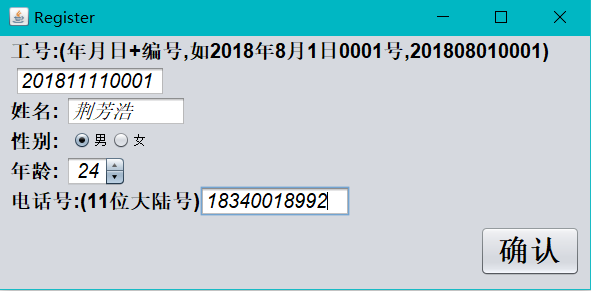
### 8.1身份选择

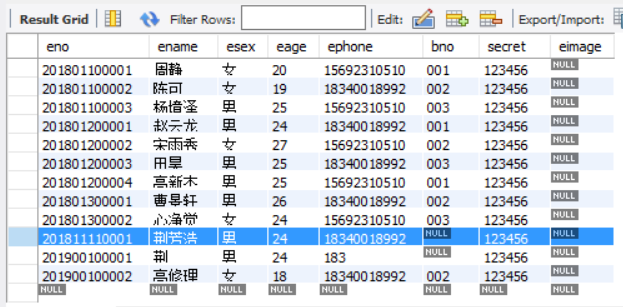
**正常**

### 8.2 找回密码

****

### 8.3 注册

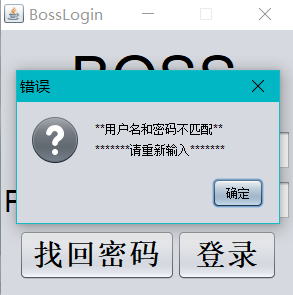
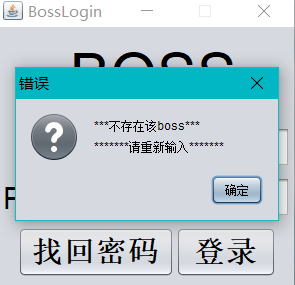
****

****

### 8.4 boss登录

当工号不存在会报错，正常

当密码和工号不匹配会报错，正常

****

测试完异常状况，正常登录

id:001

passsword:666001001(默认为’666+id+id’)

### 8.5 boss管理界面

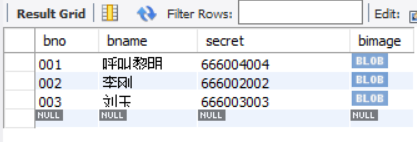
**登入初始界面**



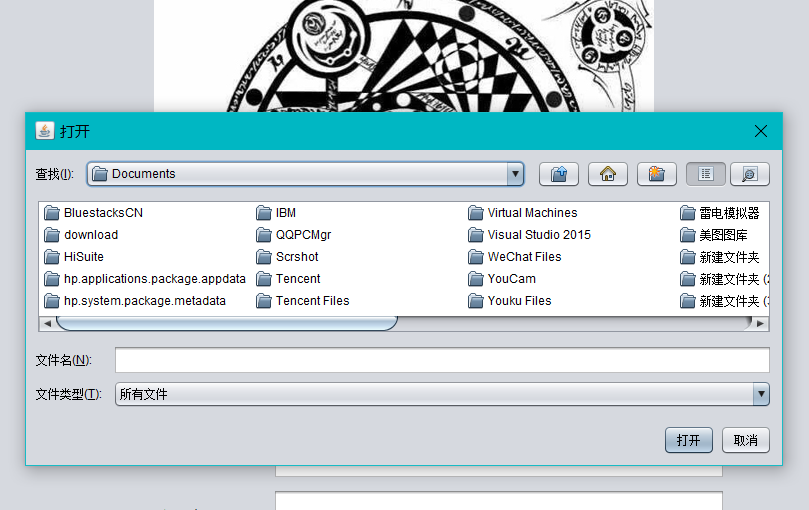
**修改基本信息**



**修改成功**



**上传照片。。。上传成功。。。**



**录取新注册的雇员 id:201811110001**



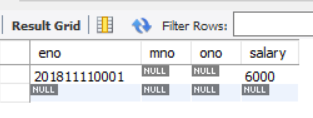
**录取成功。**



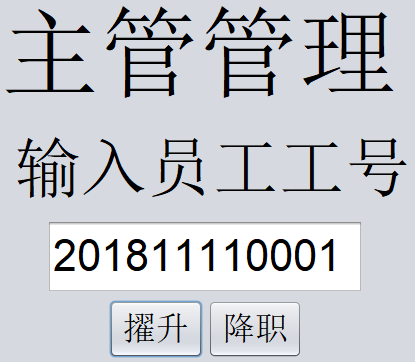
**设置员工工资：由默认的4000-改为6000**



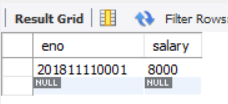
**修改成功**



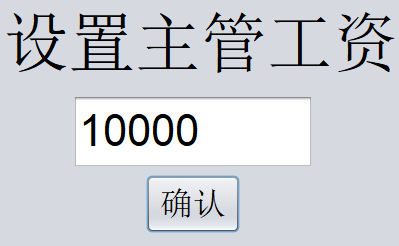
**进一步将该员工擢升为主管**



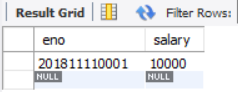
**擢升成功**



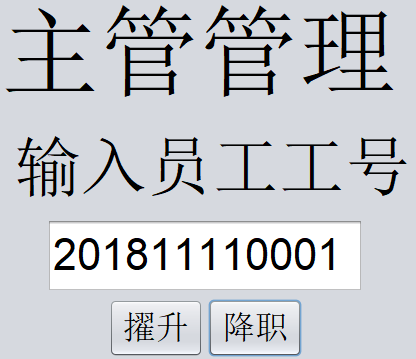
**设置主管工资：由默认的8000-改为10000**



**设置成功**



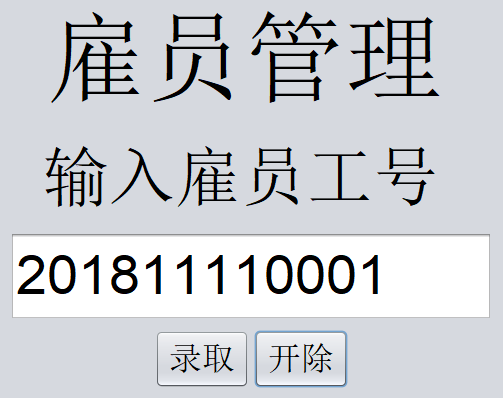
**再将这个主管降职为普通职员**



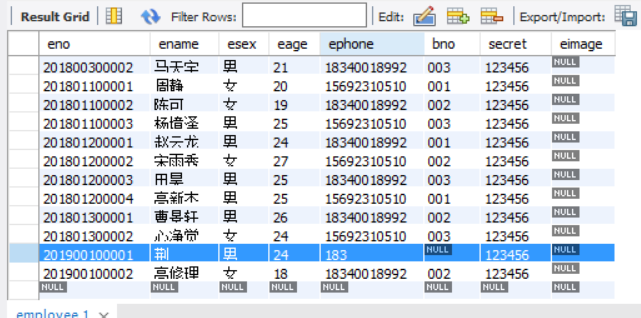
**降职成功**

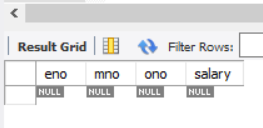


**将这个雇员直接开除**

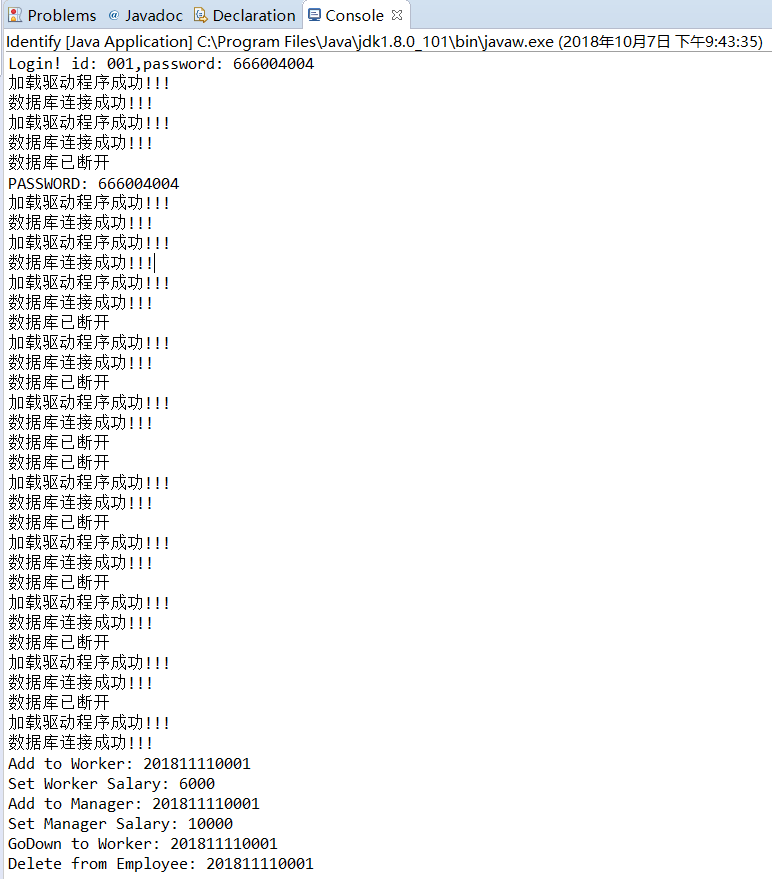


**开除成功，所有员工表(employee,manager,worker)里都无这个人的信息**





**Console运行轨迹**



### 8.6 manager登录

当工号不存在会报错，正常

当密码与工号不匹配会报错，正常



测试错误信息成功，登入manager界面

Id：201400100001（已由上述boss操作将该职员录取并擢升为主管）

Password：123456（默认为‘123456’）

### 8.7 manager管理界面

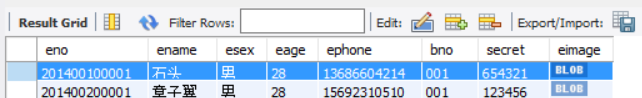
**登入初始界面**

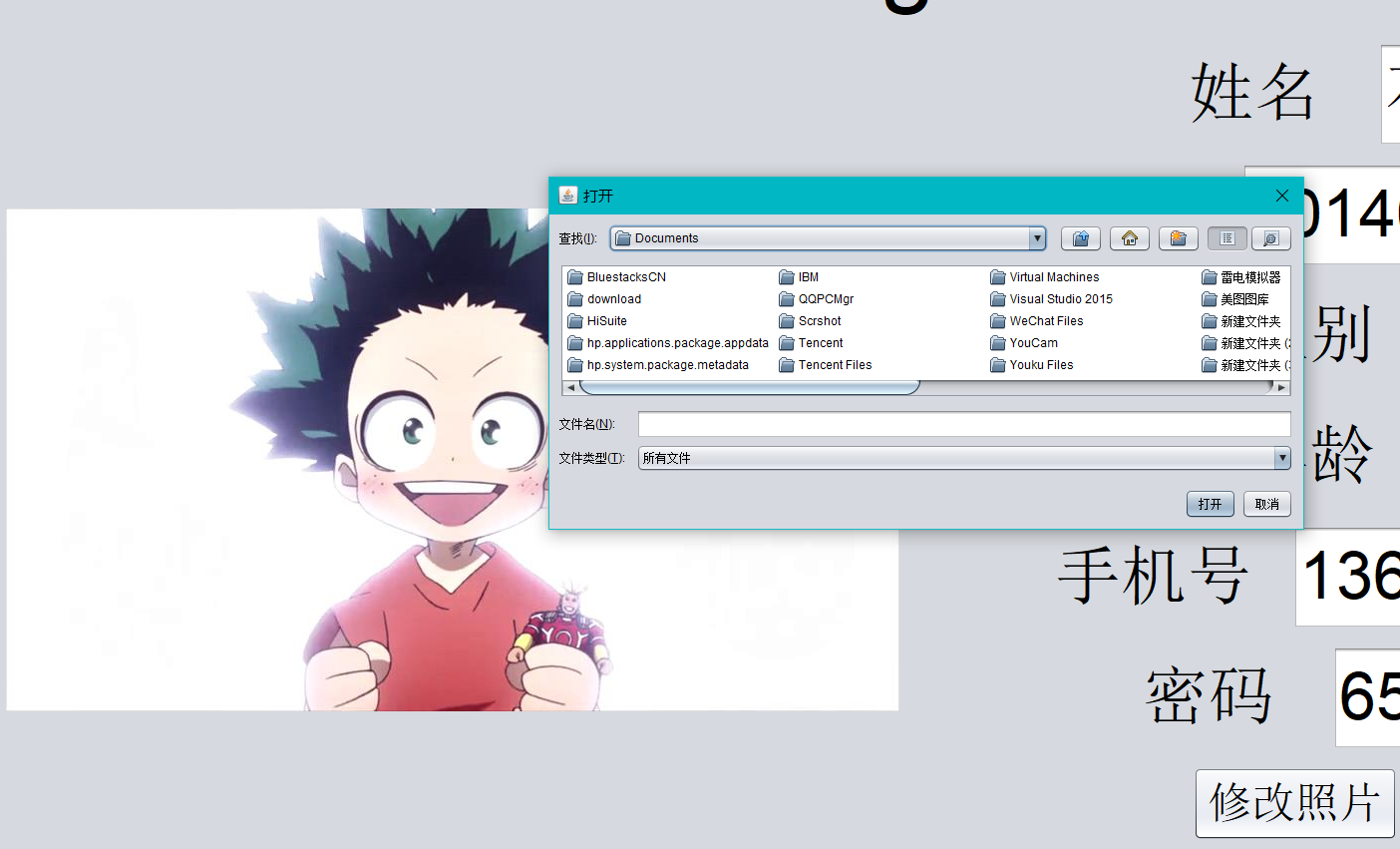
****

**修改基本信息**

****

**修改成功**

****

**上传照片。。上传成功。。**

**个人信息测试完毕。**

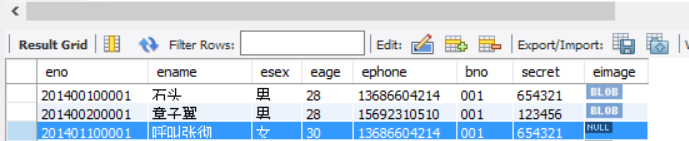
**下面进行员工管理测试。。。。。**

**选择员工，显示本员工信息，正常**

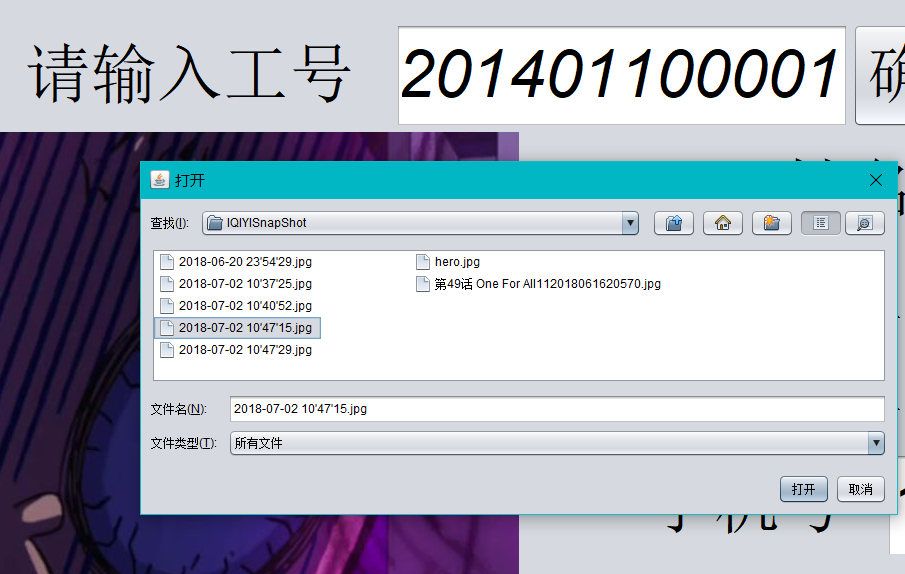
****

**修改信息。**

**修改成功**

****

**修改头像，成功。**

****

**职员功能测试完毕。正常。**

**查看三个报表，因未录入数据，无数据显示，报表信息显示正常**

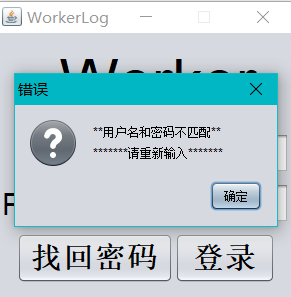
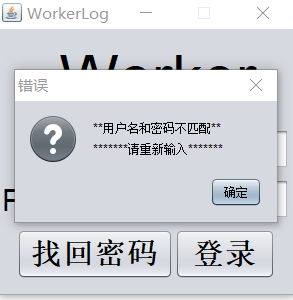
****

### 8.8 worker登录

**测试错误信息**

**当无该职员，则报错，z正常**

**当密码与工号不匹配，报错，正常**

****

**测试完毕，登入职员界面**

**Id：201500100001（由boss：001录取）**

**Password：123456（default password)**

### 8.9 worker管理界面

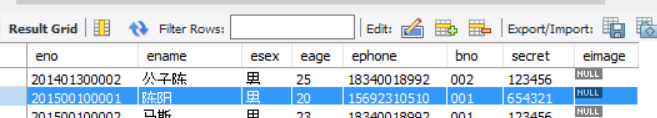
**\*登入初始界面**

****

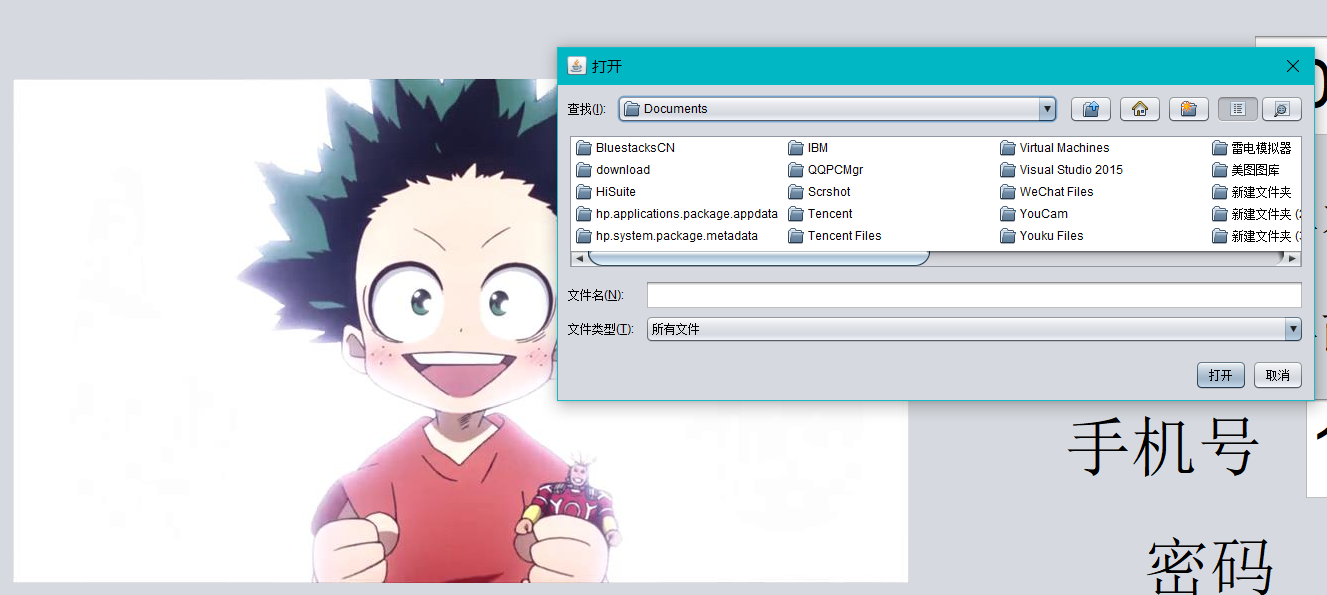
**\*修改个人密码**

****

**修改成功**

****

**修改照片。。修改成功。**

****

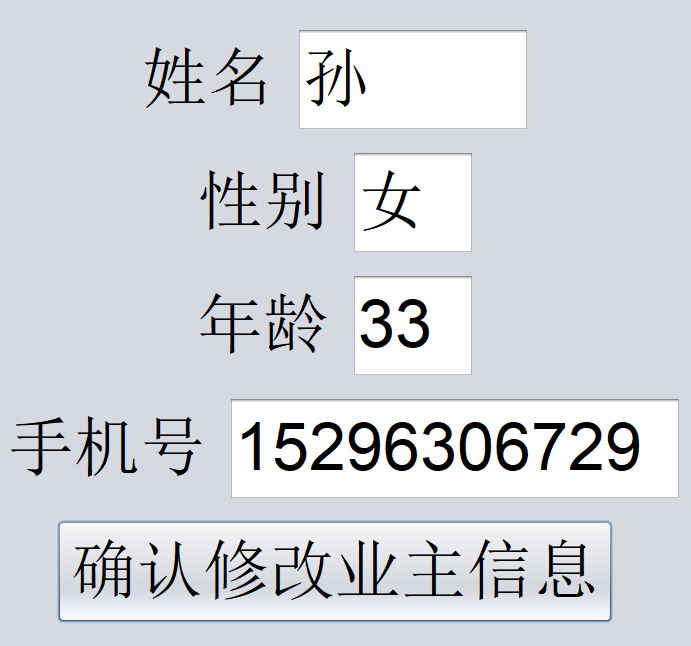
**个人信息测试完毕。正常。**

**\*业主信息管理功能测试。**

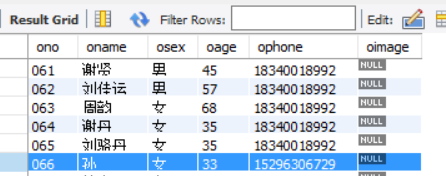
**显示所服务业主信息，正常**

****

**修改信息**

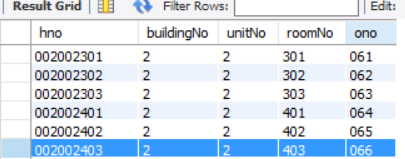
****

**修改成功**

****

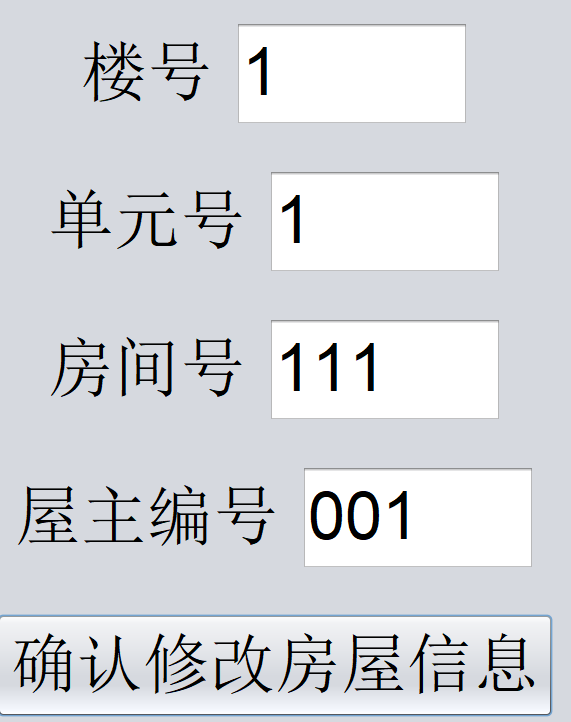
**\*房屋信息管理测试。**

**测试#66业主名下房子;002002403**

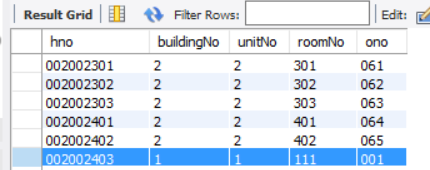
****

****

**修改信息**

****

**修改成功**

****

**\*下面模拟加费，缴费流程。**

**选择，房屋001001102**

****

**其屋主为002，进入业主服务窗口**

****

**开始缴费，假设房主都带的不够，各款项只能缴纳部分费用**

****

**各款项缴费30后，重新刷新该业主信息，各欠款均减少30，成功**

****

# 九、总结

通过本次的数据库课程设计，我在巩固了上学期学到的基础知识后，进一步学到了更多数据库的知识，特别是在编写软件时将database模块集成到一个整体内，实现数据的规范化，规模化，操作安全化，可以避免在用户数据过多时数据混乱，使增删改操作变得复杂，而且程序运行随着数据规模的不断增长运行效率越来越慢，而数据库的应运而生很好的解决了这个问题，在自己的亲手实践中，对数据库的重要性有了更多的认识。

也遇到了很多问题，都是在学习时忽略或是没有学到的知识，自己通过网络和教材里的指导一一解决，这个过程虽然痛苦，经常耽误很长的时间，可在亲自解决一个问题后，成就感足以弥补这段时间的付出，很珍惜这段时间的经历，当然所学到的知识是次要的，最重要的是经过这次系统的开发，无论是从最初的分析需求到一步步设计数据库结构，设计界面，按照流程开发，感受到了流程规范的重要性，因为很多的无用功都是浪费在最初的设计不够完备上，不完整的设计直接进行开发导致的就是后期花费大量时间来修改各种细节，而这些都是在一次次的测试中进行的。

尽管最后勉强完成了基本功能的实现，也算是差强人意，可是还是有太多的遗憾，首先就是界面设计太过粗糙，因为是自己从0到如今的“丑”界面的，当初设计时只考虑到功能的完备性和操作的简洁化，没考虑过做的花哨点优美点，因为看到公司的办公系统都是灰褐色调为主的简洁界面，可是在别的同学最后展示时看到人家的界面那么好看还是有点羞惭的，也承认自己设计不出那样好看的界面，后来了解到人家用了现成的模板，向其中组件添加了监听功能模块，也算是一份经验，也改变了我以自己的喜好设计的想法，为自己以后工作打好了基础。

自己这两个周的开发过程，也熬夜到3，4点过，也被一个功能的实现折腾过一整天，演示PPT那天临阵用了两小时做了PPT，然后临时进行了PPT演示，最后那天交给实验老师查验时很紧张，对自己没太有信心，老师一直很耐心的听还鼓励我，当最后演示完所有实现的功能后真的是松了一口气，回到家睡了整整12个小时，很感谢老师的指导，很珍惜这次经历。

# 附. 参考文献

主要的参考资料、网址。

数据库系统概念（Database System Concepts(Sixth Edition) ）

作者：（美国）西尔伯沙茨（Abraham Silberschatz） （美国）科思（Henry F.Koah） （美国）苏达尔善（S.Sudarshan）机械工业出版社