



记性—事件管理系统

《面向对象程序设计》期末考评自主创作大作业



班 级： 19130413

学 号： 1913041319

姓 名： 韩轩

2020 年 12 月

目录

1 概述	1
1.1 普遍现状	1
1.2 实际意义	1
1.3 业务范围	1
1.4 开发工具和语言	1
2 需求描述	1
2.1 需求概述	1
2.2 功能需求	2
2.3 数据需求	3
2.4 完整性约束	3
2.5 核心业务流程	3
2.5.1 事件自动删除	3
2.5.2 事件自动提醒	3
2.5.2 天气预报	3
3 概要设计	4
3.1 应用程序设计	4
3.2 数据库设计	4
3.2.1 数据库概念设计	4
4 详细设计	5
4.1 数据库对象实现	5
4.1.1 数据库连接	5
4.2 应用程序界面设计及编码	8
5 心得体会	50
6 参考文献	51

1 概述

1.1 普遍现状

人们在日常生活中常常会遇到各种各样的繁琐事件，记忆在脑海中。而一个人的记忆力是有限的，在极短时间内的大量事件往往会发生记忆偏差，导致自身错过一些比较重要的事情。随着信息时代的到来，事件的管理使得问题得以解决。

1.2 实际意义

我们在日常生活中经常面对电脑学习，娱乐。把事件按截至时间存储在电脑上就效果显著。而电脑上还未发现一个简单好用的事件管理系统，基于此我决定开发一个事件管理系统来实现此功能。以此来解决人们在日常生活中的事件管理问题。传统的记事本或实体记录方法，当时间进行变更时，无法实时更新信息。而事件管理系统的出现，可以简单轻松的完成这一点。

1.3 业务范围

本系统是面向个人的事件管理系统，全部信息存储在本地，不上传网络，从最大程度上避免了个人隐私的泄露。

1.4 开发工具和语言

数据库：MySQL 5.5

数据库管理工具：Navicat for MySQL

开发工具：Eclipse 4.16.0

开发语言：Java

2 需求描述

2.1 需求概述

人们已经进入一个全民快节奏的生活，在饱受压力生活的情况下，每个事件的记忆、执

行就成为生活中很重要的一件事。

如今的计算机技术非常发达，为了给个人提高效率和效益，因此利用相关计算机技术开发出一个事件管理系统是很必要的。本系统致力于把人们从往常的便签、记事本的固定记录模式转化为可以自动提醒，自动删除的全自动模式。

2.2 功能需求

通过与记性--事件管理系统数据库用户的深入交流与沟通，该系统主要提供如下功能。

(1) 用户信息的管理。简单的注册，登录功能和便捷的直接使用功能，包括：

用户注册、删除个人信息；

用户使用账号密码登录；

游客直接使用。

(2) 事件基本信息的管理。提供事件基本信息的增加、维护与查询功能，包括：

用户修改、增加及删除事件基本信息；

用户可以根据事件名称、事件类别和事件编号查询事件基本信息；

游客修改、增加及删除事件基本信息；

游客可以根据事件名称、事件类别和事件编号查询事件基本信息。

(3) 事件基本信息自动维护。提供事件的临近提醒和过期删除功能，包括：

语音提醒最近一小时结束的事件；

语音提醒已经过期的事件。

(4) 天气播报。提供系统城市的天气情况来判断事件的执行顺序，包括：

湿度、PM2.5、PM10、平均气温；

空气质量、感冒指数；

最低温度、最高温度、空气质量指数、风力、风向、天气情况。

事件管理系统的功能模块如图 1.1 所示。

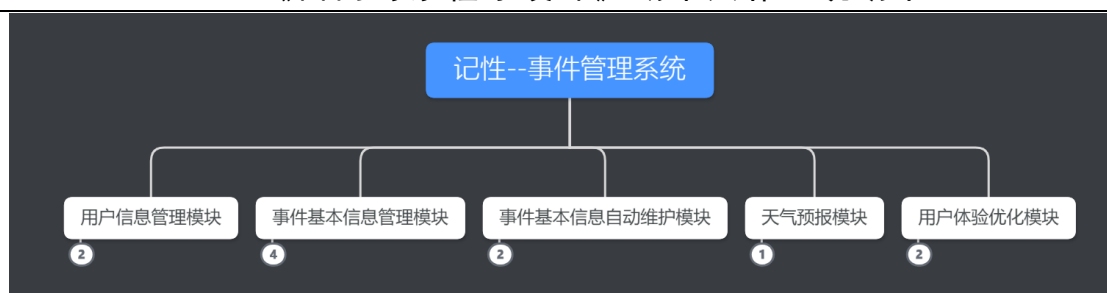


图 1.1 记性一事件管理系统的功能模块(续)

2.3 数据需求

记性一事件管理系统的用户需求分析如下。

- (1) 用户信息： 登录账号、登录密码、姓名、性别、年龄、城市。
- (2) 事件信息： 事件编号、事件使用者、事件内容、事件类别、事件截止日期。

2.4 完整性约束

记性一事件管理系统的完整性约束分析如下。

- (1) 一个用户可以有多个事件，一个事件只能属于某一个用户。

2.5 核心业务流程

2.5.1 事件自动删除

当开启事件管理系统时，自动刷新一次，检测事件是否过期，过期删除并语音提醒。

开启事件管理系统后，可选择手动刷新和自动刷新。

2.5.2 事件自动提醒

当开启事件管理系统时，自动刷新一次，检测事件是否即将过期，并语音提醒，判断时限为一小时。

开启事件管理系统后，可选择手动刷新和自动刷新。

2.5.2 天气预报

当开启事件管理系统时，使用直接登录会打开天气预报界面，内容包括：

湿度、PM2.5、PM10、平均气温；

空气质量、感冒指数；

最低温度、最高温度、空气质量指数、风力、风向、天气情况。

3 概要设计

3.1 应用程序设计

(1) 系统方案：java Swing

根据自己所掌握的知识还有时间的考虑，采用 java 图形界面实现该系统。

(2) 工具：

Eclipse

(3) 模式

MVC 模式：MVC 是一种强制性地把应用程序的数据表示、数据处理和流程控制分开的设计模式。MVC 模式的出现不仅实现了功能模块和显示模块的分离，同时它还提高了应用系统的可维护性、可扩展性、可移植性和可复用性。

Dao 模式：Dao 模式是指位于业务逻辑和持久化数据之间实现对持久化数据的访问。就是将数据库操作都封装起来。隔离了数据访问代码和业务逻辑代码。业务逻辑代码直接调用 DAO 方法即可，完全感觉不到数据库表的存在。分工明确，数据访问层代码变化不影响业务逻辑代码，这符合单一职能原则，降低了藕合性，提高了可复用性。

3.2 数据库设计

3.2.1 数据库概念设计

1. 确定实体集及属性

(1) 用户实体集： 登录账号、登录密码、姓名、性别、年龄、城市属性。

(2) 事件实体集： 事件编号、事件使用者、事件内容、事件类别、事件截止日期属性。

2. 确定联系集及属性

3. 完整系统模块如图 3.1 所示。

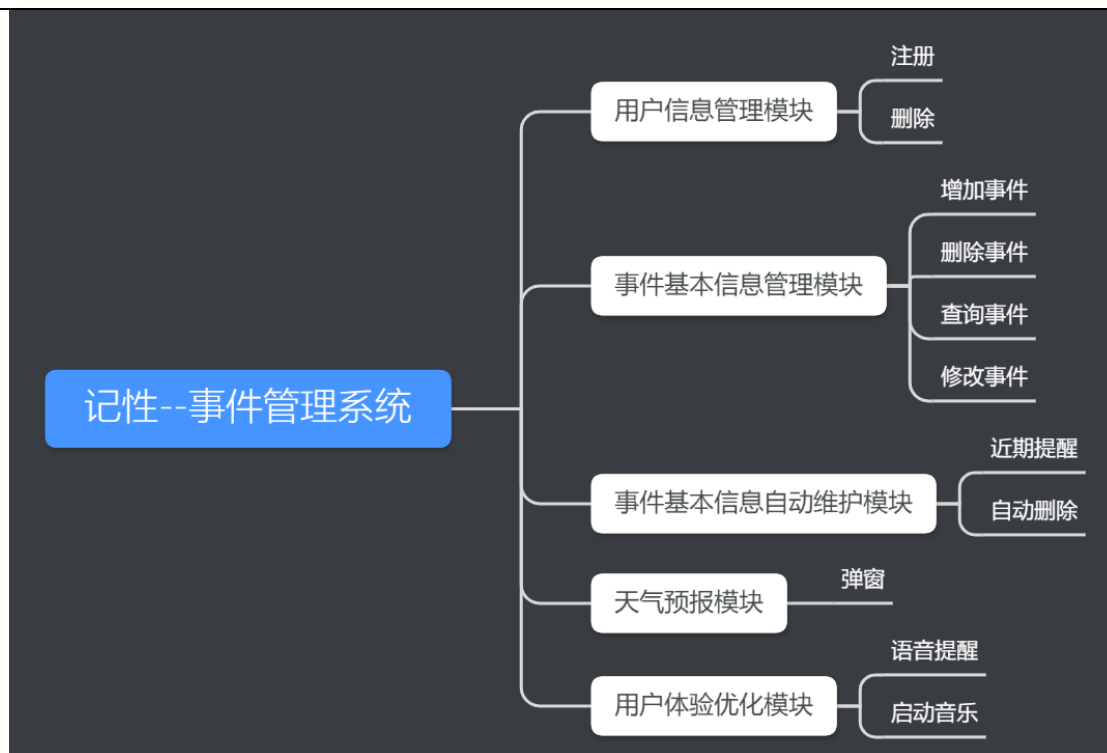


图 3.1 记性-管理系统(续)

4 详细设计

4.1 数据库对象实现

4.1.1 数据库连接

(1) 将数据库参数写入 db.properties 文件中

```
private String dbUrl="jdbc:mysql://localhost:3308/jixing"; // 数据库连接地址
private String dbUserName="root"; // 用户名
private String dbPassword="root"; // 密码
private String jdbcName="com.mysql.jdbc.Driver"; // 数据库驱动名称
```

(2) 建立数据库

/*

Navicat MySQL Data Transfer

Source Server : 192.168.43.12

Source Server Version : 50537

《面向对象程序设计》期末大作业说明书

Source Host : localhost:3308

Source Database : jixing

Target Server Type : MYSQL

Target Server Version : 50537

File Encoding : 65001

Date: 2020-12-17 22:14:01

*/

SET FOREIGN_KEY_CHECKS=0;

--- -----
-- Table structure for thing
--- -----

```
DROP TABLE IF EXISTS `thing`;  
CREATE TABLE `thing` (  
  `id` int(11) NOT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  `category` varchar(255) DEFAULT NULL,  
  `endingdate` datetime DEFAULT NULL,  
  `userid` varchar(20) NOT NULL,  
  PRIMARY KEY (`id`,`userid`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

--- -----
-- Records of thing
--- -----


```
-- Table structure for user
```

```
DROP TABLE IF EXISTS `user`;  
CREATE TABLE `user` (  
  `username` varchar(255) NOT NULL,  
  `password` varchar(255) DEFAULT NULL,  
  `name` varchar(255) DEFAULT NULL,  
  `sex` varchar(255) DEFAULT NULL,  
  `age` int(11) DEFAULT NULL,  
  `city` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`username`) USING BTREE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=DYNAMIC;
```

```
-- Records of user
```

```
INSERT INTO `user` VALUES ('admin', 'admin', '游客', '0', '0', '0');
```

数据库设计模型，如图 4.1 所出示。

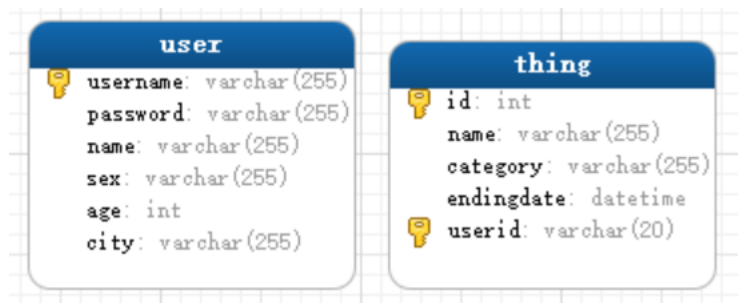


图 4.1 数据库模型图

数据库设计效果，如图 4.2 所出示。

id	name	category	endingdate	userid
2	毛中考试	考试	2021-01-09 07:0	游客
3	球赛	活动	2020-12-26 16:0	游客

图 4.2 数据库效果图

username	password	name	sex	age	city
admin	admin	游客	0	0 0	
石一歌	123456	韩轩	男	20	太原
诗亦歌	123456	诗诗	女	0	北京

图 4.3 数据库效果图（续）

4.2 应用程序界面设计及编码

(1) 用户登录界面，如图 4.4 所示。

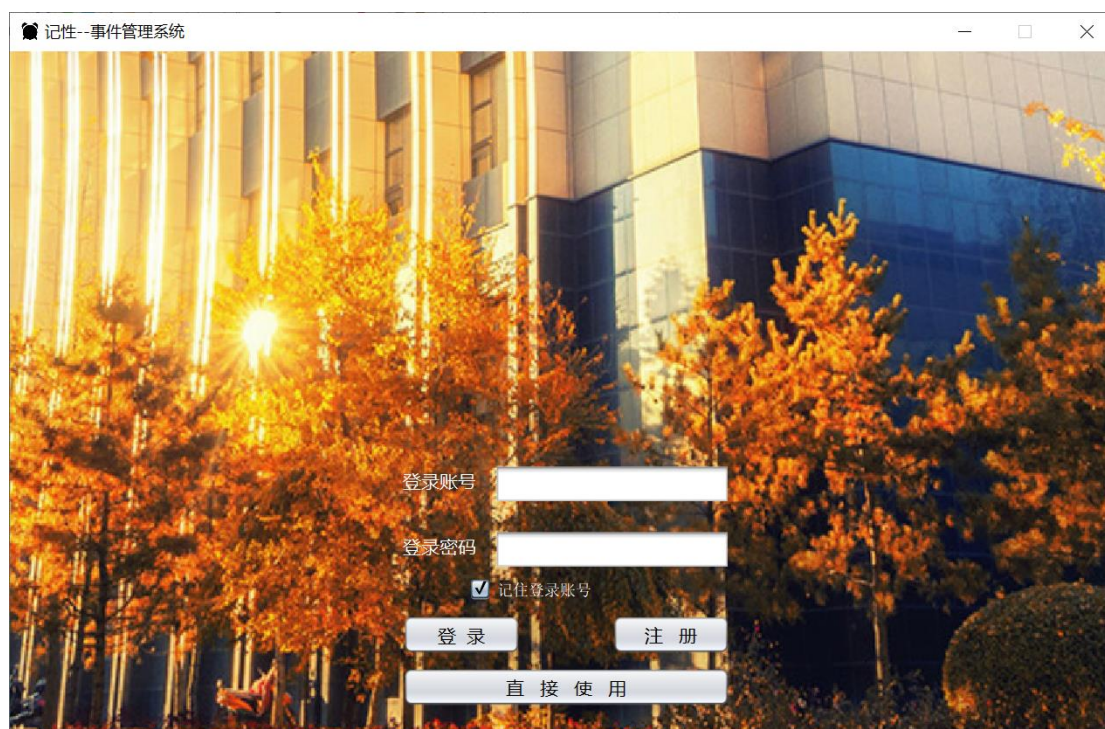


图 4.4 用户登录界面图

核心代码片段：

```
public void btLoginActionPerformed() {
```

```
    // 获取用户名和密码
```

```
    String username = tfUsername.getText().trim();
```

```
    String password = new String(pfPassword.getPassword()).trim();
```

```
    if (username.equals("admin") && password.equals("admina")) {
```

```
        new BrowseFrame(null);
```

```
        this.dispose();
```

```
    } else {
```

```
        User user = userAction.login(username, password); // 调用具体的login
```

方法 判断用户名和密码

```
        if (user == null) {
```

```
            new ProgressBarFrame(user);
```

```
            JOptionPane.showMessageDialog(this, "账号或密码不正确!"); // 提示
```

```
            tfUsername.requestFocus(); // 获取焦点
```

```
            // 增加用户体验, 选中文本框中的文字
```

```
            tfUsername.selectAll();
```

```
            return;
```

```
        } else {
```

```
            new ProgressBarFrame(user);
```

```
            this.setLogin(true); // login状态为true
```

```
            // 用户信息完整, 则进入事件查询界面
```

```
        }
```

```
        this.dispose(); // 关闭当前界面
```

```
    }
```

```
}
```

```
private JPanel pnbackground = new JPanel(); // 定义背景面板
```

```
private ImageIcon image; // 定义图片
```

```
private JLabel background; // 定义背景
```

```
private JLayeredPane pane = new JLayeredPane(); // 分层网格用来设置透
```

明

```
image = new ImageIcon("image\\background.png");
```

```
background = new JLabel(image); // 把背景图片添加到标签里
```

```
pnbackground.setBounds(0, 0, image.getIconWidth(),
```

image.getIconHeight()); // 把背景面板设置为和图片等高等宽

```
pnbackground = (JPanel) this.getContentPane(); // 把此面板设置为内容面
```

板

```
pnbackground.add(background);

jlUser.setOpaque(false); // 设置组件透明
tfUsername.setOpaque(false);
jlPassword.setOpaque(false);
pfPassword.setOpaque(false);
ckbRemember.setOpaque(false);
btLogin.setOpaque(false);
btRegister.setOpaque(false);
btDirectuse.setOpaque(false);

pane.add(pnbackground, JLayeredPane.DEFAULT_LAYER); // 最底层
pane.add(jlUser, JLayeredPane.MODAL_LAYER); // 模式对话层
pane.add(tfUsername, JLayeredPane.MODAL_LAYER); // 模式对话层
pane.add(jlPassword, JLayeredPane.MODAL_LAYER); // 模式对话层
pane.add(pfPassword, JLayeredPane.MODAL_LAYER); // 模式对话层
pane.add(ckbRemember, JLayeredPane.MODAL_LAYER); // 模式对话层
pane.add(btLogin, JLayeredPane.MODAL_LAYER); // 模式对话层
pane.add(btRegister, JLayeredPane.MODAL_LAYER); // 模式对话层
pane.add(btDirectuse, JLayeredPane.MODAL_LAYER); // 模式对话层

this.setBounds(100, 100, image.getIconWidth(),
image.getIconHeight());
this.setLayeredPane(pane); // 第二层设置为pane
this.setVisible(true);
```

```
package jixing.controller;
```

```
import java.io.File;
```

```
import java.io.IOException;
```

```
import javax.sound.sampled.AudioFormat;
```

```
import javax.sound.sampled.AudioInputStream;
```

```
import javax.sound.sampled.AudioSystem;
```

```
import javax.sound.sampled.LineUnavailableException;
```

```
import javax.sound.sampled.SourceDataLine;
```

```
import javax.sound.sampled.UnsupportedAudioFileException;
```

```
public class MusicAction {
```

```
    public void sing() {
```

```
        try {
```

```
            File file = new File("music/1.wav");
```

```
            // 2、定义一个AudioInputStream用于接收输入的音频数据
```

```
AudioInputStream am;
// 3、使用AudioSystem来获取音频的音频输入流(处理(抛出)异常)
am = AudioSystem.getAudioInputStream(file);
// 4、使用AudioFormat来获取AudioInputStream的格式
AudioFormat af = am.getFormat();
// 5、一个源数据行
SourceDataLine sd;
// 6、获取受数据行支持的音频格式DataLine.Info
// DataLine.Info dl = new DataLine.Info(SourceDataLine.class, af);
// 7、获取与上面类型相匹配的行 写到源数据行里 二选一
sd = AudioSystem.getSourceDataLine(af); // 便捷写法
// sd = (SourceDataLine) AudioSystem.getLine(dl);
// 8、打开具有指定格式的行，这样可以使行获得资源并进行操作
sd.open();
// 9、允许某个数据行执行数据i/o
sd.start();
// 10、写数据
int sumByteRead = 0; // 读取的总字节数
byte[] b = new byte[320]; // 设置字节数组大小
// 11、从音频流读取指定的最大数量的数据字节，并将其放入给定的字节
```

数组中。

```
while (sumByteRead != -1) { // -1代表没有 不等于-1时就无限读取
    sumByteRead = am.read(b, 0, b.length); // 12、读取哪个数组
    if (sumByteRead >= 0) { // 13、读取了之后将数据写入混频器,开始
```

播放

```
        sd.write(b, 0, b.length);
```

```
    }
}
} catch (UnsupportedAudioFileException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
} catch (LineUnavailableException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
// sd.drain();
// sd.close();
```

```
}
```

(2) 用户注册界面，如图 4.5 所示。



The image shows a user registration window titled "记性注册界面" (Memory Registration Interface). It contains the following fields and controls:

- 用户名** (Username): A text input field.
- 输入密码** (Enter Password): A password input field.
- 再次输入...** (Enter again...): A second password input field.
- 姓名** (Name): A text input field.
- 性别:** (Gender): Radio buttons for **男** (Male) and **女** (Female). The **男** option is selected.
- 所在地:** (Location): A dropdown menu currently showing **北京** (Beijing).
- 注册** (Register): A button at the bottom left.
- 返回** (Return): A button at the bottom right.

图 4.5 用户注册界面图



图 4.6 用户注册成功效果图（续）

核心代码部分：

```
public void btRegisterActionPerformed() { // 注册方法实现
```

```
    String username = tfUsername.getText().trim(); // 获取 username 的值
```

```
    String name = tfName.getText().trim(); // 获取用户名
```

```
    String pass = new String(pfPassword.getPassword()).trim(); // 获取第一次输入的密码
```

```
    String rePass = new String(pfRePassword.getPassword()).trim(); // 获取第二次输入的密码
```

```
    // 获取所有的用户列表
```

```
    List<User> userList = userService.selectAll();
```

```
for (User user : userList) { // 遍历
    if (!user.getUsername().equals(username)) { // 校验 username 唯一
        if (pass.equals(rePass)) {
            User users = new User();
            users.setUsername(username);
            users.setPassword(pass);
            users.setName(name);
            users.setSex(String.valueOf(rdoMale.isSelected()) ? '男' : '女'); // 获取性别 1 是男 0 是女
            users.setCity(cboCity.getSelectedItem().toString()); // 获取城市

            userService.add(users); // 调用添加用户方法

            JOptionPane.showMessageDialog(this, "注册成功!"); // 提示
            this.dispose(); // 关闭当前页面
            new UserInfoFrame(); // 打开登录页面
            break;
        } else {
            JOptionPane.showMessageDialog(this, "密码不一致请重新输入!"); // 提示
        }
    } else {
        JOptionPane.showMessageDialog(null, "用户名已存在, 请重新输入!"); // 提示
        break;
    }
}
```



```
}  
  
}
```

```
public User login(String username, String password) {  
    // 获取用户列表  
    List<User> users = userService.selectAll();  
    // 循环判断用户输入的账号和密码在数据集合中是否存在。  
    for (int i = 0; i < users.size(); i++) {  
        User user = users.get(i);  
        if (user.getUsername().equals(username) &&  
user.getPassword().equals(password))  
            return user;  
    }  
    // 无对应用户数据返回 null  
    return null;  
}
```

(3) 用户信息界面，如图 4.7 所示。



图 4.7 用户信息界面图

《面向对象程序设计》期末大作业说明书



图 4.8 用户信息界面删除效果图

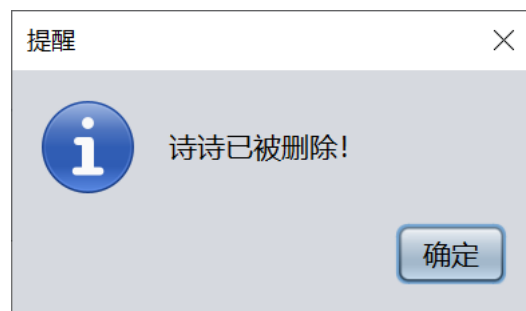


图 4.9 用户信息界面删除效果图（续）

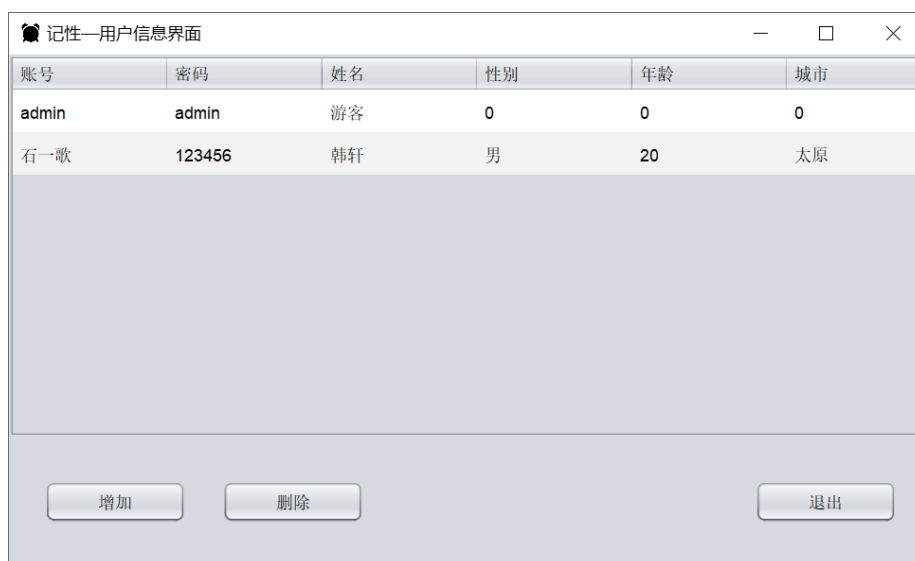


图 4.9 用户信息界面删除效果图(续)

核心代码部分:

```
void init() {  
    Object[] title = {"账号", "密码", "姓名", "性别", "年龄", "城市",};  
    Object[] userarray = userService.selectAll().toArray();  
    Object[][] userinfo = new Object[userarray.length][6];  
    User u = null;  
    for (int i = 0; i < userarray.length; i++) {  
        u = (User)userarray[i];  
        userinfo[i][0] = u.getUsername();  
        userinfo[i][1] = u.getPassword();  
        userinfo[i][2] = u.getName();  
        userinfo[i][3] = u.getSex();  
        userinfo[i][4] = u.getAge();  
        userinfo[i][5] = u.getCity();  
    }  
    t = new JTable(userinfo, title);  
    t.setRowHeight(30); // 行高  
    t.setColumnSelectionAllowed(false); // 列选禁用  
    t.setCellSelectionEnabled(false); // 行列同选禁用  
    t.setRowSelectionAllowed(true); // 行选开启  
    JScrollPane scrollPane = new JScrollPane(t);  
  
    scrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLB  
AR_AS_NEEDED);  
  
    scrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_A  
S_NEEDED);  
  
    scrollPane.setBounds(0, 0, 660, 270);
```

```
this.add(scrollPane);
```

```
bt_insert = new JButton("增加");  
bt_insert.setBounds(25, 300, 100, 30);  
this.add(bt_insert); // 添加到本界面
```

```
bt_del = new JButton("删除");  
bt_del.setBounds(150, 300, 100, 30);  
this.add(bt_del); // 添加到本界面
```

```
bt_out = new JButton("退出");  
bt_out.setBounds(525, 300, 100, 30);  
this.add(bt_out); // 添加到本界面
```

```
}
```

```
void mouseClicked() {
```

```
    bt_del.addMouseListener(new MouseListener() {
```

```
        @Override
```

```
        public void mouseClicked(MouseEvent e) {
```

```
            // TODO Auto-generated method stub
```

```
            // 点击调用
```

```
            index = t.getSelectedRow(); // 选择的行
```

```
            sel_num = t.getSelectedRowCount(); // 选择的行数
```

```
            if (sel_num == 0) {
```

```
                JOptionPane.showMessageDialog(null, "请先选择使用者！", "提醒", 1); // 提示
```

```
        } else if (sel_num > 1) {
            JOptionPane.showMessageDialog(null, "单次只能操作一行", "提醒", 1); // 提示
        } else {
            userService.delete((String)t.getValueAt(index, 2));
            dispose();
            JOptionPane.showMessageDialog(null, t.getValueAt(index, 2) +
"已被删除！", "提醒", 1); // 提示
            new UserInfoFrame();
        }
    }

@Override
public void mousePressed(MouseEvent e) {} // 按下调用

@Override
public void mouseReleased(MouseEvent e) {} // 释放调用

@Override
public void mouseEntered(MouseEvent e) {
    bt_del.setBackground(Color.orange);
} // 进入调用

@Override
public void mouseExited(MouseEvent e) {
    bt_del.setBackground(null);
} // 离开调用

});
```

```
bt_insert.addMouseListener(new MouseListener() {

    @Override
    public void mouseClicked(MouseEvent e) {
        // TODO Auto-generated method stub
        // 点击调用
        dispose();
        new RegisterFrame();
    }

    @Override
    public void mousePressed(MouseEvent e) {} // 按下调用

    @Override
    public void mouseReleased(MouseEvent e) {} // 释放调用

    @Override
    public void mouseEntered(MouseEvent e) {
        bt_insert.setBackground(Color.orange);
    } // 进入调用

    @Override
    public void mouseExited(MouseEvent e) {
        bt_insert.setBackground(null);
    } // 离开调用

});

bt_out.addMouseListener(new MouseListener() {
```

```
@Override
public void mouseClicked(MouseEvent e) {
    // TODO Auto-generated method stub
    // 点击调用
    dispose();
    new LoginFrame();
}

@Override
public void mousePressed(MouseEvent e) {} // 按下调用

@Override
public void mouseReleased(MouseEvent e) {} // 释放调用

@Override
public void mouseEntered(MouseEvent e) {
    bt_out.setBackground(Color.orange);
} // 进入调用

@Override
public void mouseExited(MouseEvent e) {
    bt_out.setBackground(null);
} // 离开调用
});
}
```

(4) 登录进度条，如图 4.10 所示。



图 4.10 进度条效果图

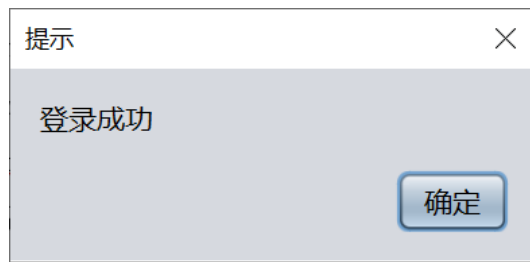


图 4.11 登录成功效果图

核心代码部分：

```
public User login(String username, String password) {  
    // 获取用户列表  
    List<User> users = userService.selectAll();  
  
    // 循环判断用户输入的账号和密码在数据集合中是否存在。  
    for (int i = 0; i < users.size(); i++) {  
        User user = users.get(i);  
        if (user.getUsername().equals(username) &&  
user.getPassword().equals(password))  
            return user;  
    }  
  
    // 无对应用户数据返回 null  
    return null;  
}
```

```
public ProgressBarFrame(User user) {  
    this.setTitle("登陆中");  
    this.setSize(250, 75);  
    this.setResizable(false);  
    this.setLocationRelativeTo(null);  
    this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); // 真正关
```


闭界面

```
JPanel panel = new JPanel();

// 创建一个进度条
JProgressBar progressBar = new JProgressBar();
// 设置进度的 最小值 和 最大值
progressBar.setMinimum(MIN_PROGRESS);
progressBar.setMaximum(MAX_PROGRESS);
// 设置当前进度值
progressBar.setValue(currentProgress);
// 绘制百分比文本（进度条中间显示的百分数）
progressBar.setStringPainted(true);
// 添加进度改变通知
progressBar.addChangeListener(new ChangeListener() {
    @Override
    public void stateChanged(ChangeEvent e) {
        if (progressBar.getValue() == 100) {
            JOptionPane.showInternalMessageDialog(null, "登录成功", "提示",
JOptionPane.PLAIN_MESSAGE); // 提示
            dispose();
            new BrowseFrame(user); // 打开注册界面
        }
    }
});
progressBar.setUI(new GradientProgressBarUI());
// 添加到内容面板
panel.add(progressBar);
```

```
this.setContentPane(panel);

this.setVisible(true);


// 模拟延时操作进度, 每隔 0.04 秒更新进度
new Timer(40, new ActionListener() {

    @Override

    public void actionPerformed(ActionEvent e) {

        currentProgress++;

        progressBar.setValue(currentProgress);

    }

}).start();

}


/**
 *
 * @Title: ProgressBarFrame.java
 * @Package: jixing.view
 * @ClassName: GradientProgressBarUI
 * @Description: TODO
 * @author: 石一歌
 * @date: 2020 年 12 月 21 日 下午 5:18:48
 * @version: V6.0
 */

public class GradientProgressBarUI extends BasicProgressBarUI {

    @Override

    protected void paintDeterminate(Graphics g, JComponent c) {

        Graphics2D graphics2d = (Graphics2D)g;

        Insets b = progressBar.getInsets();

        // JProgressBar 的边界区域
```

```
int width = progressBar.getWidth();
int height = progressBar.getHeight();
int barRectWidth = width - (b.right + b.left);
int barRectHeight = height - (b.top + b.bottom);
int arcSize = height / 2 - 1;
int amountFull = getAmountFull(b, barRectWidth, barRectHeight);
// 已完成的进度
graphics2d.setColor(Color.white);
graphics2d.fillRoundRect(0, 0, width - 1, height, arcSize, arcSize);
// 绘制 JProgressBar 的背景
// 用 GradientPaint 类来实现渐变色显示进度
// 设置了开始点和终止点，并设置好这两个点的颜色，系统会自动在这两个
点中用渐变色来填充

Point2D start = new Point2D.Float(0, 0);
Point2D end = new Point2D.Float(amountFull - 1, barRectHeight - 1);
// 这里设置的终止点是当前已经完成的进度的那个点
GradientPaint gradientPaint = new GradientPaint(start, Color.gray,
end, Color.gray);

graphics2d.setPaint(gradientPaint);

graphics2d.fillRoundRect(b.left, b.top, amountFull - 1, barRectHeight,
arcSize, arcSize); // 这里实现的是圆角的效果将 arcSize 调成 0 即可实现矩形效果
}
}
```

(5) 事件管理界面，如图 4.12 所示。

《面向对象程序设计》期末大作业说明书

记性

游客,您好,欢迎登录记性 来自于: 0 退出

事件名: 分类: - 请选择 - 查询

编号	使用者名称	类别	具体内容	截止时间
2	游客	考试	毛中考试	2021-01-09 07:00:22
3	游客	活动	球赛	2020-12-26 16:00:46

代办事件数: 2件 查看详情 持续刷新 刷新 修改

图 4.12 事件管理界面图(续)

记性

游客,您好,欢迎登录记性 来自于: 0 退出

事件名: 分类: 活动 查询

编号	使用者名称	类别	具体内容	截止时间
3	游客	活动	球赛	2020-12-26 16:00:46

代办事件数: 1件 查看详情 持续刷新 刷新 修改

图 4.13 事件管理界面查询效果图

《面向对象程序设计》期末大作业说明书



图 4.14 事件管理界面刷新效果图(续)



图 4.15 事件管理界面最小化效果图

核心代码部分:

```
if (SystemTray.isSupported()) // 如果操作系统支持托盘
{
    this.tray();
}
this.setVisible(true);
// 窗口关闭时触发事件
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
    public void windowIconified(WindowEvent e) {
```

```
        try {
            tray.add(trayIcon); // 将托盘图标添加到系统的托盘实例中
            setVisible(false); // 使窗口不可视
            dispose();
        } catch (AWTException ex) {
            ex.printStackTrace();
        }
    }
});

btnQuery.addActionListener(new ActionListener() { // 查询监听事件
    public void actionPerformed(ActionEvent e) {

        String name = txtThingName.getText().toString(); // 获取查询的
name
        Object category = cboCategory.getSelectedItem(); // 获取查询选择
的分类

        if (category.toString().equals("- 请选择 -")) { // 如果不选择 或
者说选择的是“请选择”
            initThingTableModel(); // 初始化表格
            showThing(name, null, user); // 根据 name 去查询
        } else {
            initThingTableModel(); // 初始化表格
            showThing(name, category.toString(), user); // 展示商品信息
        }
    }
});

btnRefreshContinued = new JButton("持续刷新"); // 定义购买按钮
btnRefreshContinued.setBounds(330, 394, 140, 32); // 设置按钮位置
```

```
this.add(btnRefreshContinued); // 将按钮添加到界面
```

btnRefreshContinued.addActionListener(new ActionListener() { // 按钮的
监听事件

```
    @Override
    public void actionPerformed(ActionEvent e) {
        reminder(5);
    }
});
```

```
btnRefresh = new JButton("刷新"); // 定义购买按钮
```

```
btnRefresh.setBounds(500, 394, 70, 32); // 设置按钮位置
```

```
this.add(btnRefresh); // 将按钮添加到界面
```

```
btnRefresh.addActionListener(new ActionListener() { // 按钮的监听事件
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        refresh();
    }
});
```

```
/**
```

```
*
```

```
* @Title: tray
```

```
* @Description: TODO    void
```

```
*/
```

```
private void tray() {
```

```
    // TODO Auto-generated method stub
```

```
    tray = SystemTray.getSystemTray(); // 获得本操作系统托盘的实例
```

```
    ImageIcon icon = new ImageIcon("image/tixing.png"); // 将要显示到托盘中的图标
```

```
    // 24px 的 png 格式图片
```

```
PopupMenu pop = new PopupMenu(); // 构造一个右键弹出式菜单
```

```
String open = new String("打开记事");
```

```
String close = new String("退出");
```

```
MenuItem show = new MenuItem(open);
```

```
MenuItem exit = new MenuItem(close);
```

```
pop.add(show);
```

```
pop.add(exit);
```

```
show.addActionListener(new ActionListener() // 点击“显示窗口”菜单后将
```

窗口显示出来

```
{  
    public void actionPerformed(ActionEvent e) {  
        tray.remove(trayIcon); // 从系统的托盘实例中移除托盘图标  
        setExtendedState(JFrame.NORMAL);  
        setVisible(true); // 显示窗口  
        toFront();  
    }  
});
```

```
exit.addActionListener(new ActionListener() // 点击“退出演示”菜单后退
```

出程序

```
{  
    public void actionPerformed(ActionEvent e) {  
        System.exit(0); // 退出程序  
    }  
});
```

```
trayIcon = new TrayIcon(icon.getImage(), "记事 v7.0", pop);
```

```
// 添加鼠标监听器，当鼠标在托盘图标上双击时，默认显示窗口
```

```
trayIcon.addMouseListener(new MouseAdapter() {  
    public void mouseClicked(MouseEvent e) {
```



```
        if (e.getClickCount() == 2) // 鼠标双击
        {
            tray.remove(trayIcon); // 从系统的托盘实例中移除托盘图标
            setExtendedState(JFrame.NORMAL);
            setVisible(true); // 显示窗口
            toFront();
        }
    }
});

}

/**
 *
 * @Title: initThingTableModel
 * @Description: TODO    void
 */
private void initThingTableModel() {
    // 创建一个 JTable 的默认显示模式
    DefaultTableModel dt = new DefaultTableModel();
    // 设置 JTable 的列的个数和列的名字
    dt.setColumnIdentifiers(new Object[] {"编号", "使用者名称", "类别", "具体内容", "截止时间",});

    tblThing.setBackground(new Color(255, 255, 255));
    // 设置 JTable 表格对象被选中行的背景色
    tblThing.setSelectionBackground(new Color(255, 255, 255));
    // 设置 JTable 表格对象被选中行的字体色
    tblThing.setSelectionForeground(new Color(255, 103, 0));
}
```

```
// 为表格设置商品信息表格模型
```

```
tblThing.setModel(dt);
```

```
// 设置表格的列
```

```
tblThing.getColumnModel().getColumn(0).setPreferredWidth(70);
```

```
tblThing.getColumnModel().getColumn(1).setPreferredWidth(70);
```

```
tblThing.getColumnModel().getColumn(2).setPreferredWidth(70);
```

```
tblThing.getColumnModel().getColumn(3).setPreferredWidth(70);
```

```
tblThing.getColumnModel().getColumn(4).setPreferredWidth(150);
```

```
}
```

```
/**
```

```
 *
```

```
 * @Title: showThing
```

```
 * @Description: TODO
```

```
 * @param name
```

```
 * @param category
```

```
 * @param user
```

```
 * @return void
```

```
 */
```

```
private void showThing(String name, String category, User user) { // 展示所有商品信息
```

```
    List<Thing> thing = browseAction.queryThing(name, category, user); // 调用查询方法
```

```
    Iterator<Thing> it = thing.iterator();
```

```
    while (it.hasNext()) {
```

```
        Thing value = it.next();
        if (deleteaction.isOut(value)) {
            it.remove();
        }
        if (remindAction.isOut(value)) {
            speak(myFmt.format(value.getEndingdate()) + "的" +
value.getName() + value.getCategory() + "马上要截止了!");
        }
    }

    setList(thing);

    lblQuantity.setText(list.size() + "件");

    DefaultTableModel dt = (DefaultTableModel)tblThing.getModel(); // 在表
格中增加内容
    for (int i = 0; i < thing.size(); i++) { // 遍历 thing 列表 将数据添加到
表格中
        dt.insertRow(i, new Object[] {thing.get(i).getId(),
thing.get(i).getUserId(), thing.get(i).getCategory(),
            thing.get(i).getName(),
myFmt.format(thing.get(i).getEndingdate())});
    }
}

/**
 *
 * @Title: refresh
 * @Description: TODO
```

```
* @return void
*/
public void refresh() { // 刷新界面数据
    DefaultTableModel model = (DefaultTableModel)tblThing.getModel();
    cboCategory.setModel(new DefaultComboBoxModel<String>(new String[] {"-
请选择 -", "作业", "考试", "活动", "讲座", "约会"})); // 下拉列表属性
    model.setRowCount(0); // 清空表格数据
    List<Thing> list = browseAction.showAll(null); // 获取所有的商品
    Iterator<Thing> it = list.iterator();
    while (it.hasNext()) {
        Thing value = it.next();
        if (deleteaction.isOut(value)) {
            speak(myFmt.format(value.getEndingdate()) + "的" +
value.getName() + value.getCategory() + "已过期!");
            it.remove();
        }
        if (remindAction.isOut(value)) {
            speak(myFmt.format(value.getEndingdate()) + "的" +
value.getName() + value.getCategory() + "马上要截止了!");
        }
    }
    lblQuantity.setText(list.size() + "件");

    for (int i = 0; i < list.size(); i++) { // 遍历 list
        // 重新为表格填数据
        Thing thing = (Thing)list.get(i); // 获取每个 thing
        model.addRow(// 添加到表格中
            new Object[] {thing.getId(), thing.getUserId(),
thing.getCategory(), thing.getName(),
```

```
        myFmt.format(thing.getEndingdate()))});
    }
}

public List<Thing> queryThing(String name, String category, User user) {

    List<Thing> thing = new ArrayList<>(); // 定义查询列表
    String thingName = name;
    if (thingName == null)
        thingName = "";
    List<Thing> thingList;
    if (user == null)
        thingList = this.showAll(null);
    else
        thingList = this.showAll(user.getName()); // 获取所有的事件
    for (Thing things : thingList) { // 遍历所有事件
        if ((things.getName().contains(thingName)) // 找到符合条件的
            && (category == null ||
things.getCategory().equals(category))) {
            thing.add(things); // 添加到集合中
        }
    }
    // 返回结果
    return thing;

}

public class DeleteAction {
    static BrowseAction browseAction = new BrowseAction();

    /**
```

```
*
* @Title: isOut
* @Description: TODO
* @param thing
* @return boolean
*/
public boolean isOut(Thing thing) {
    Date date = new Date();
    // System.out.println(date);
    if (date.after(thing.getEndingdate())) {
        browseAction.deleteThing(thing.getId());
        return true;
    }
    return false;
}
}

public class RemindAction {
    Calendar cal_start = Calendar.getInstance();
    Calendar cal_end = Calendar.getInstance();

    /**
    *
    * @Title: isOut
    * @Description: TODO
    * @param thing
    * @return boolean
    */
    public boolean isOut(Thing thing) {
        Date date = new Date();
```

```
Date date_start, date_end;

cal_start.setTime(thing.getEndingdate()); // 赋值
cal_end.setTime(thing.getEndingdate()); // 赋值
cal_start.add(Calendar.SECOND, -1);
cal_end.add(Calendar.SECOND, +1);
date_start = cal_start.getTime();
date_end = cal_end.getTime();

if (date.getTime() < date_end.getTime() && date.getTime() >
date_start.getTime())

    return true;

return false;

}

}
```

(6) 事件修改界面，如图 4.16 所示。

The image shows a Java Swing window titled "记事" (Journal). It contains the following elements:

- 使用者名称** (User Name): A text input field.
- 事件编号** (Event Number): A text input field.
- 事件内容** (Event Content): A text input field.
- 分类** (Category): A dropdown menu with the text "- 请选择 -" (Please select).
- 日期** (Date): A date and time field showing "2020-12-25 03:12:26".
- Buttons**: Four buttons arranged in a 2x2 grid: "增加" (Add), "删除" (Delete), "查询" (Query), and "修改" (Modify).

图 4.16 事件修改界面图

《面向对象程序设计》期末大作业说明书



记性

使用者名称: 游客 事件编号: 1

事件内容: 课设 分类: 作业

日期: 2021-01-25 12:01:26

增加 删除

查询 修改

图 4.17 事件增加效果图(续)



记性

您好,记性系统欢迎您! 来自于: 登录

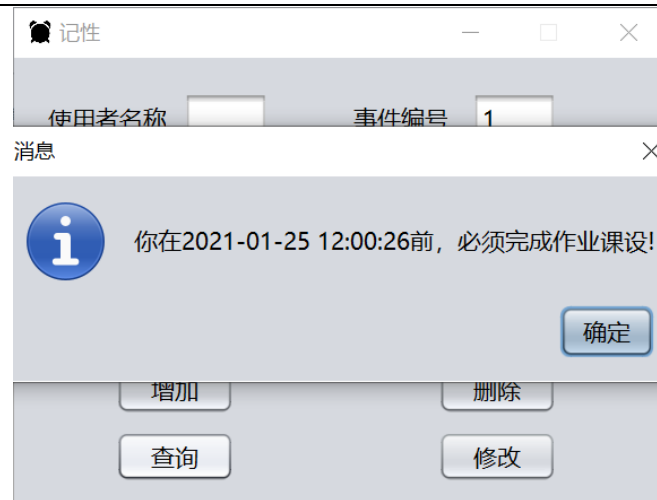
事件名: 分类: - 请选择 - 查询

编号	使用者名称	类别	具体内容	截止时间
1	游客	作业	课设	2021-01-25 12:00:26
2	游客	考试	毛中考试	2021-01-09 07:00:22
3	游客	活动	球赛	2020-12-26 16:00:46

代办事件数: 3件 [查看详情](#) 持续刷新 刷新 修改

图 4.18 事件增加效果图(续)

《面向对象程序设计》期末大作业说明书



记性

使用者名称 事件编号

消息

你在2021-01-25 12:00:26前, 必须完成作业课设!

确定

增加 删除

查询 修改

图 4.18 事件查询效果图



记性

使用者名称 事件编号


事件内容 分类:

日期:

增加 删除

查询 修改

图 4.19 事件修改效果图



记性

您好, 记性系统欢迎您! 来自于:

事件名: 分类:

编号	使用者名称	类别	具体内容	截止时间
1	游客	作业	课设	2021-01-16 12:00:20
2	游客	考试	毛中考试	2021-01-09 07:00:22
3	游客	活动	球赛	2020-12-26 16:00:46

代办事件数: 3件 [查看详情](#)

图 4.20 事件修改效果图(续)

《面向对象程序设计》期末大作业说明书

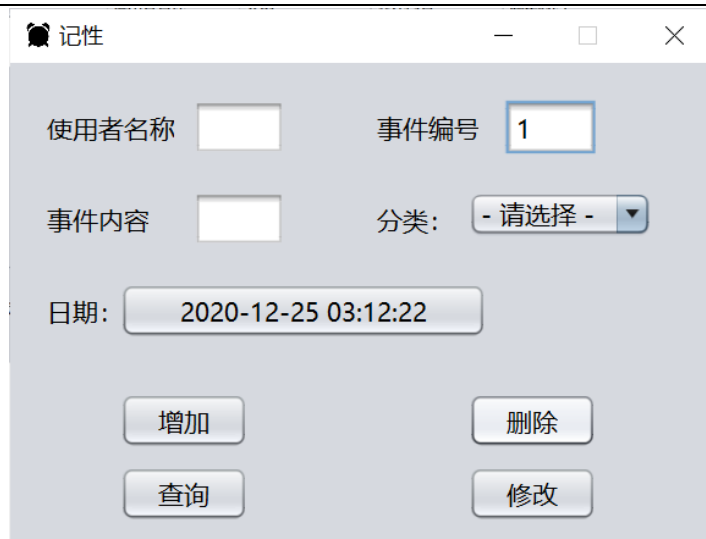


图 4.21 事件删除效果图



编号	使用者名称	类别	具体内容	截止时间
2	游客	考试	毛中考试	2021-01-09 07:00:22
3	游客	活动	球赛	2020-12-26 16:00:46

图 4.22 事件删除效果图(续)

核心代码部分:

```
/**  
*  
* @Title: btFixActionPerformed  
* @Description: TODO    void
```

```
*/  
  
protected void btFixActionPerformed() {  
    // TODO Auto-generated method stub  
  
    Thing thing = new Thing(tfUserId.getText().toString(),  
Integer.valueOf(tfId.getText()),  
    tfName.getText().toString(),  
cboCategory.getSelectedItemAt(0).toString(), btChoosDate.getDate());  
    browseAction.updateThing(thing);  
}  
  
/**  
 *  
 * @Title: btCheckByIdActionPerformed  
 * @Description: TODO    void  
 */  
  
protected void btCheckByIdActionPerformed() {  
    // TODO Auto-generated method stub  
  
    Thing thing =  
browseAction.findThingById(Integer.valueOf(tfId.getText()));  
    // System.out.println(myFmt.format(thing.getEndingdate()));  
    speak("你在" + myFmt.format(thing.getEndingdate()) + "前，必须完成" +  
thing.getCategory() + thing.getName() + "!!");  
    JOptionPane.showMessageDialog(this,  
        "你在" + myFmt.format(thing.getEndingdate()) + "前，必须完成" +  
thing.getCategory() + thing.getName() + "!!");// 提示  
}  
  
/**  
 *
```

```
* @Title: btDeleteActionPerformed
* @Description: TODO    void
*/
protected void btDeleteActionPerformed() {
    // TODO Auto-generated method stub
    browseAction.deleteThing(Integer.valueOf(tfId.getText()));
}

/**
 *
 * @Title: btAddActionPerformed
 * @Description: TODO    void
 */
public void btAddActionPerformed() {
    Thing thing = new Thing(tfUserId.getText().toString(),
Integer.valueOf(tfId.getText()),
        tfName.getText().toString(),
cboCategory.getSelectedItem().toString(), btChoosDate.getDate());
    browseAction.addThing(thing);
}

/**
 *
 * @Title: speak
 * @Description: TODO
 * @param s
 *
    void
 */
public void speak(String s) {
```

```
ActiveXComponent sap = new ActiveXComponent("Sapi.SpVoice");
Dispatch sapo = sap.getObject();
try {
    // 音量 0-100
    sap.setProperty("Volume", new Variant(100));
    // 语音朗读速度 -10 到 +10
    sap.setProperty("Rate", new Variant(0));
    // 执行朗读
    Dispatch.call(sapo, "Speak", new Variant(s));
} catch (Exception e) {
    e.printStackTrace();
} finally {
    sapo.safeRelease();
    sap.safeRelease();
}

}

private JPanel createWeekAndDayPanal() {
    String colname[] = {"日", "一", "二", "三", "四", "五", "六"};
    JPanel result = new JPanel();
    // 设置固定字体, 以免调用环境改变影响界面美观
    result.setFont(new Font("宋体", Font.PLAIN, 14));
    result.setLayout(new GridLayout(7, 7));
    result.setBackground(Color.white);
    JLabel cell;

    for (int i = 0; i < 7; i++) {
        cell = new JLabel(colname[i]);
        cell.setHorizontalAlignment(JLabel.CENTER);
    }
}
```

```
        if (i == 0 || i == 6)
            cell.setForeground(weekendFontColor);
        else
            cell.setForeground(weekFontColor);
        result.add(cell);
    }

    int actionCommandId = 0;
    for (int i = 0; i < 6; i++)
        for (int j = 0; j < 7; j++) {
            JButton numberButton = new JButton();
            numberButton.setBorder(null);

            numberButton.setHorizontalAlignment(SwingConstants.CENTER);

            numberButton.setActionCommand(String.valueOf(actionCommandId));

            numberButton.addActionListener(this);
            numberButton.setBackground(palletTableColor);
            numberButton.setForeground(dateFontColor);
            if (j == 0 || j == 6)
                numberButton.setForeground(weekendFontColor);
            else
                numberButton.setForeground(dateFontColor);
            daysButton[i][j] = numberButton;
            result.add(numberButton);
            actionCommandId++;
        }

    return result;
```

```
}

private JDialog createDialog(Frame owner) {
    JDialog result = new JDialog(owner, "日期时间选择", true);
    result.setDefaultCloseOperation(JDialog.HIDE_ON_CLOSE);
    result.getContentPane().add(this, BorderLayout.CENTER);
    result.pack();
    result.setSize(width, height);
    return result;
} void showDateChooser(Point position)
{
    Frame owner =
(Frame)SwingUtilities.getWindowAncestor(DateChooserJButton.this);
    if (dialog == null || dialog.getOwner() != owner)
        dialog = createDialog(owner);
    dialog.setLocation(getAppropriateLocation(owner, position));
    flushWeekAndDay();
    dialog.show();
}

Point getAppropriateLocation(Frame owner, Point position) {
    Point result = new Point(position);
    Point p = owner.getLocation();
    int offsetX = (position.x + width) - (p.x + owner.getWidth());
    int offsetY = (position.y + height) - (p.y + owner.getHeight());

    if (offsetX > 0)
        result.x -= offsetX;
    if (offsetY > 0)
        result.y -= offsetY;
    return result;
}
```

```
}

private void dayColorUpdate(boolean isOldDay) {
    Calendar c = getCalendar();
    int day = c.get(Calendar.DAY_OF_MONTH);
    c.set(Calendar.DAY_OF_MONTH, 1);
    int actionCommandId = day - 2 + c.get(Calendar.DAY_OF_WEEK);
    int i = actionCommandId / 7;
    int j = actionCommandId % 7;
    if (isOldDay)
        daysButton[i][j].setForeground(dateFontColor);
    else
        daysButton[i][j].setForeground(todayBackColor);
}

private void flushWeekAndDay() {
    Calendar c = getCalendar();
    c.set(Calendar.DAY_OF_MONTH, 1);
    int maxDayNo = c.getActualMaximum(Calendar.DAY_OF_MONTH);
    int dayNo = 2 - c.get(Calendar.DAY_OF_WEEK);
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 7; j++) {
            String s = "";
            if (dayNo >= 1 && dayNo <= maxDayNo)
                s = String.valueOf(dayNo);
            daysButton[i][j].setText(s);
            dayNo++;
        }
    }
    dayColorUpdate(false);
}
```



```
public void stateChanged(ChangeEvent e) {
    JSpinner source = (JSpinner)e.getSource();
    Calendar c = getCalendar();
    if (source.getName().equals("Hour")) {
        c.set(Calendar.HOUR_OF_DAY, getSelectedHour());
        setDate(c.getTime());
        return;
    }

    dayColorUpdate(true);

    if (source.getName().equals("Year"))
        c.set(Calendar.YEAR, getSelectedYear());
    else
        // (source.getName().equals("Month"))
        c.set(Calendar.MONTH, getSelectedMonth() - 1);
    setDate(c.getTime());
    flushWeekAndDay();
}

public void actionPerformed(ActionEvent e) {
    JButton source = (JButton)e.getSource();
    if (source.getText().length() == 0)
        return;
    dayColorUpdate(true);
    source.setForeground(todayBackColor);
    int newDay = Integer.parseInt(source.getText());
    Calendar c = getCalendar();
    c.set(Calendar.DAY_OF_MONTH, newDay);
    setDate(c.getTime());
}
```

```
}
```

(7) 天气预报界面，如图 4.23 所示。

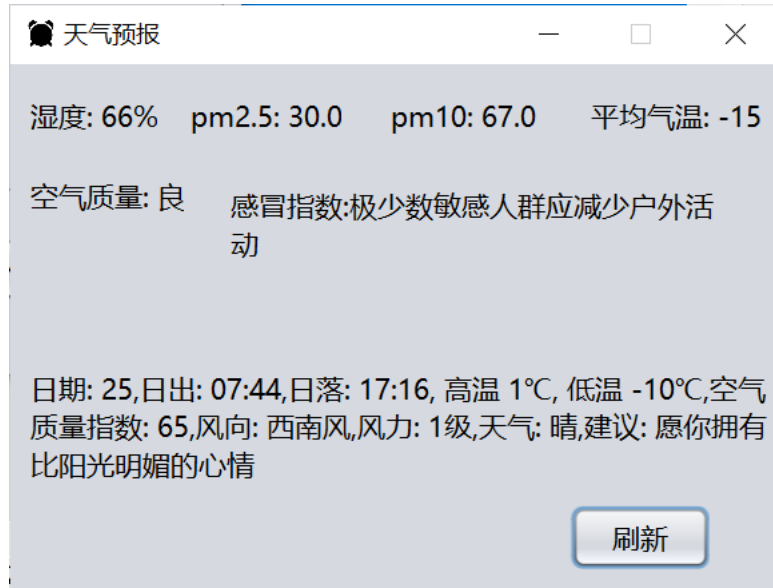


图 4.23 天气预报效果图

核心代码部分：

```
public class FindForecastAction {

    private String dealResponseResult(InputStream inputStream) {
        StringBuilder html = new StringBuilder(); // 存储处理结果
        try {
            InputStreamReader inputStreamReader = new
InputStreamReader(inputStream);
            BufferedReader reader = new BufferedReader(inputStreamReader);
            html.append(reader.readLine());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return html.toString();
    }
}
```

```
public JSONObject get_weather(String city_id) {
    try {
        URL url = new URL("http://t.weather.itboy.net/api/weather/city/" +
city_id);
        HttpURLConnection connection =
(HttpURLConnection)url.openConnection();
        connection.setRequestMethod("GET");
        connection.setConnectTimeout(5 * 1000);
        connection.setReadTimeout(5 * 1000);
        connection.connect();
        // 获得服务器的响应码
        int response = connection.getResponseCode();
        if (response == HttpURLConnection.HTTP_OK) {
            InputStream inputStream = connection.getInputStream();
            String html = dealResponseResult(inputStream);
            JSONObject json = JSONObject.parseObject(html);
            JSONObject data = json.getJSONObject("data");
            return data;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
}
```

```
FindForecastAction jsondata = new FindForecastAction();
JSONObject data = jsondata.get_weather("101100101");
```

```
String humidity = data.getString("shidu");
float pm25 = Float.parseFloat(data.getString("pm25"));
float pm10 = Float.parseFloat(data.getString("pm10"));
String quality = data.getString("quality");
int temp = Integer.parseInt(data.getString("wendu"));
String Cold_index = data.getString("ganmao");
for (Object weather : data.getJSONArray("forecast")) {
    JSONObject object = (JSONObject)weather;
    System.out.println();
    lb_content = new JLabel("<html>" + "日期: " + object.get("date")
+ ", 日出: " + object.get("sunrise")
+ ", 日落: " + object.get("sunset") + "</br>" + ", " +
object.get("high") + ", " + object.get("low")
+ ", 空气质量指数: " + object.get("aqi") + "</br>" + ", 风向:
" + object.get("fx") + ", 风力: "
+ object.get("fl") + ", 天气: " + object.get("type") + "</br>"
+ ", 建议: " + object.get("notice")
+ "</html>");
    break;
}
```

5 心得体会

本次面向对象程序设计大作业遇到了许多问题。在系统的实现过程中发现了对上课内容学习的不扎实，对一些结构的使用不熟练，尤其是此次使用的 DAO 模式和 MVC 模式，在连接数据库的过程中浪费了大量的时间；由于不熟悉 Eclipse 调试，面对一些可以快速排查的 Bug，只能边猜边改，造成工作效率下降。

在本次的课程实验中，我通过查阅资料，完成了一次事件管理系统的设计开发实现流程，在开发过程中，又学习了之前不曾掌握的一些技术，如 Swing 中一些高级组件 MenuItem、

JProgressBar 以及 Calendar 类的使用,我还学会了使用第三方 jar 包—jacob 实现语音播报,以及最小化托盘的实现方式。又复习使用了之前实验中学习到的 JDBC 调用 MySQL 数据库的知识,时间的 SimpleDateFormat 表达方式收获颇丰。

纸上得来终觉浅,绝知此事要躬行。通过亲自动手制作,使我掌握的知识不再是纸上谈兵。在今后社会的发展和学习实践过程中,一定要不懈努力,不能遇到问题就想到要退缩,一定要不厌其烦的发现问题所在,然后一进行解决。同时,也明白了在发现问题之后,不要直接百度,而是自己独立思考,只有自己想出来的,才是真正掌握的知识。代码的搬运工只会明白怎么做,而永远不懂为什么。





在这学期的实验中,不仅培养了独立思考、动手操作的能力,而且构建了我们面向对象的程序设计思想,加深了我们对类与对象概念的理解。

6 参考文献

- [1] Java 中改变应用程序界面外观 (javax.swing.UIManager 类和 LookAndFeel 类)
<https://blog.csdn.net/u010995220/article/details/49847307/>, 2015. 11. 15
- [2] 实现渐变色的 JProgressBar
https://blog.csdn.net/Mr_Pang/article/details/47976443?locationNum=2, 2015, 8, 25
- [3] java Timer (定时调用、实现固定时间执行)
<https://www.cnblogs.com/0201zcr/p/4703061.html>, 2015. 8. 04
- [4] Java 中 Date 时区的转换
<https://www.cnblogs.com/ganbo/p/11244306.html>, 2019. 7. 25
- [5] JAVA 实现日期选择控件
<https://www.cnblogs.com/swek/articles/4338077.html>, 2015. 3. 14
- [6] 耿祥义 张跃平. JAVA 面向对象程序设计. 第三版. 清华大学出版社. 2010. 508
- [7] 刘爽英 王丽芳 李欣然 张元. 数据库原理及应用. 清华大学出版社. 2013. 269

附录

1. 使用时，应将修改db.properties文件，数据库端口默认3308，账号默认root，密码默认root. 使用jixing, mysql文件建立数据库.

2. 本项目使用的第三方包为：
 mysql-connector-java-8.0.19.jar 、  jacob.jar 、
 fastjson-1.1.34.jar 、  gson-2.8.6.jar 。

3. 语音包使用需要将(64位系统添加) jacob-1.19-M2-x64.dll添加到JDK的bin目录和Windows的system32目录（32位系统添加jacob-1.19-M2-x86.dll）。
4. 由于Eclipse自身的Bug, 最小化托盘的右击菜单会出现中文乱码, 在运行配置--自变量中添加-Dfile.encoding=GBK, 可消除乱码。
5. 天气预报接口稳定性有待增强。