華中科技大學

课 程 实 验 报 告

课程名称：Java 语言程序设计

实验名称：医院简易挂号管理系统

院　　系 ：计算机科学与技术

专业班级 ： CS1703

学　　号 ： U201714608

姓　　名 ： 彭凯杰

指导教师 ： 纪俊文

2020 年 05 月 23 日

一、需求分析

1. 题目要求

采用桌面应用程序模式，开发一个医院挂号系统，管理包括人员、号种及其挂号费用，挂号退号等信息，完成登录、挂号、查询和统计打印功能。数据库表如下所示，建立索引的目的是加速访问，请自行确定每个索引要涉及哪些字段。

T_KSXX (科室信息表)

字段名称	字段类型	主键	索引	可空	备注
KSBH	CHAR(6)	是	是	否	科室编号，数字
KSMC	CHAR(10)	否	否	否	科室名称
PYZS	CHAR(8)	否	否	否	科室名称的拼音字首

T_BRXX (病人信息表)

字段名称	字段类型	主键	索引	可空	备注
BRBH	CHAR(6)	是	是	否	病人编号，数字
BRMC	CHAR(10)	否	否	否	病人名称
DLKL	CHAR(8)	否	否	否	登录口令
YCJE	DECIMAL(10,2)	否	否	否	病人预存金额
DLRQ	DateTime	否	否	是	最后一次登录日期及时间

T_KSYS (科室医生表)

字段名称	字段类型	主键	索引	可空	备注
YSBH	CHAR(6)	是	是	否	医生编号，数字，第 1 索引
KSBH	CHAR(6)	否	是	否	所属科室编号，第 2 索引
YSMC	CHAR(10)	否	否	否	医生名称
PYZS	CHAR(4)	否	否	否	医生名称的拼音字首
DLKL	CHAR(8)	否	否	否	登录口令
SFZJ	BOOL	否	否	否	是否专家
DLRQ	DATETIME	否	否	是	最后一次登录日期及时间

T_HZXX (号种信息表)

字段名称	字段类型	主键	索引	可空	备注
HZBH	CHAR(6)	是	是	否	号种编号，数字，第 1 索引
HZMC	CHAR(12)	否	否	否	号种名称

PYZS	CHAR(4)	否	否	否	号种名称的拼音字首
KSBH	CHAR(6)	否	是	否	号种所属科室，第 2 索引
SFZJ	BOOL	否	否	否	是否专家号，即号种类别
GHRS	INT	否	否	否	每日限定的挂号人数
GHFY	DECIMAL(8,2)	否	否	否	挂号费

T_GHXX (挂号信息表)

字段名称	字段类型	主键	索引	可空	备注
GHBH	CHAR(6)	是	是	否	挂号的顺序编号，数字
HZBH	CHAR(6)	否	是	否	号种编号：可找号种 SFZJ
YSBH	CHAR(6)	否	是	否	医生编号
BRBH	CHAR(6)	否	是	否	病人编号
GHRC	INT	否	是	否	该号种的挂号人次
THBZ	BOOL	否	否	否	退号标志=true 为已退号码
GHFY	DECIMAL(8,2)	否	否	否	病人的实际挂号费用
RQSJ	DATETIME	否	否	否	挂号日期时间

为了减少编程工作量，T_KSXX、T_BRXX、T_KSYS、T_HZXX 的信息手工录入数据库，每个表至少录入 6 条记录，所有类型为 CHAR(6)的字段数据从“000001”开始，连续编码且中间不得空缺。为病人开发的桌面应用程序要实现的主要功能具体如下：

（1）病人登录：输入自己的病人编号和密码，经验证无误后登录。

（2）病人挂号：病人处于登录状态，选择科室、号种和医生（非专家医生不得挂专家号，专家医生可以挂普通号）；输入缴费金额，计算并显示找零金额后完成挂号。所得挂号的编号从系统竞争获得生成，挂号的顺序编号连续编码不得空缺。

功能（2）的界面如下所示，在光标停在“科室名称”输入栏时，可在输入栏下方弹出下拉列表框，显示所有科室的“科室编号”、“科室名称”和“拼音字首”，此时可通过鼠标点击或输入科室名称的拼音字首两种输入方式获得“科室编号”，用于插入 T_GHXX 表。注意，采用拼音字首输入时可同时完成下拉列表框的科室过滤，使得下拉列表框中符合条件的科室越来越少，例如，初始为“内一科”和“内二科”。其它输入栏，如“医生姓名”、“号种类别”、“号种名称”也可同时支持两种方式混合输入。

每种号种挂号限定当日人次，挂号人数超过规定数量不得挂号。一个数据一致的程序要保证：挂号总人数等于当日各号种的挂号人次之和，病人的账务应保证开支平衡。已退号码不得用于重新挂号，每个号种的 GHRC 数据应连续不间断，GHRC 从 1 开始。若病人有预存金额则直接扣除挂号费，此时“交款金额”和“找零金额”处于灰色不可操作状态。

为医生开发的桌面应用程序要实现的主要功能具体如下：

（1）医生登录：输入自己的医生编号和密码，经验证无误后登录。

（2）病人列表：医生处于登录状态，显示自己的挂号病人列表，按照挂号编号升序排列。显示结果如下表所示。

挂号编号	病人名称	挂号日期时间	号种类别
000001	章紫衣	2018-12-30　11:52:26	专家号
000003	范冰冰	2018-12-30　11:53:26	普通号
000004	刘德华	2018-12-30　11:54:28	普通号

（3）收入列表：医生处于登录状态，显示所有科室不同医生不同号种起止日期内的收入合计，起始日期不输入时默认为当天零时开始，截止日期至当前时间为止。时间输入和显示结果如下表所示。

起始时间：2018-12-30　00:00:00　　截止时间：2018-12-30　12:20:00

科室名称	医生编号	医生名称	号种类别	挂号人次	收入合计
感染科	000001	李时珍	专家号	24	48
感染科	000001	李时珍	普通号	10	10
内一科	000002	扁鹊	普通号	23	23
保健科	000003	华佗	专家号	10	20

病人应用程序和医生应用程序可采用主窗口加菜单的方式实现。例如，医生应用程序有三个菜单项，分别为“病人列表”、“收入列表”和“退出系统”等。

考虑到客户端应用程序要在多台计算机上运行，而这些机器的时间各不相同，客户端程序每次在启动时需要同数据库服务器校准时间，可以建立一个时间服务程序或者直接取数据库时间校准。建议大家使用 MS　SQL 数据库开发。

挂号时锁定票号可能导致死锁，为了防止死锁或系统响应变慢，建议大家不要锁死数据库表或者字段。程序编写完成后，同时启动两个挂号程序进行单步调试，以便测试两个病人是否会抢到同一个号、或者有号码不连续或丢号的现象。

系统考核目标：（1）挂号后数据库数据包括挂号时间不会出现不一致或时序颠倒现象，以及挂号人次超过该号种当日限定数量的问题；（2）挂号号码和挂号人次不会出现不连续或丢号问题；（3）病人的开支应平衡，并应和医院的收入平衡；（4）系统界面友好、操作简洁，能支持全键盘操作、全鼠标操作或者混合操作；（5）能支持下拉列表框过滤输入；（6）系统响应迅速，不会出现死锁；（7）统计报表应尽可能不采用多重或者多个循环实现；（8）若采用时间服务器程序校准时间，最好能采用心跳检测机制，显示客户端的上线和下线情况。

思考题：当病人晚上 11:59:59 秒取得某号种的挂号价格 10 元，当他确定保存时价格在第 2 天 00:00:00 已被调整为 20 元，在编程时如何保证挂号费用与当天价格相符？

2. 需求分析

整体需求在要求中已有完整体现，在此仅做总结，需要支持病人挂号和对医生登录两种功能，在统一的数据库上进行管理；需要对登录提供的编号和口令提供正确性检查；在病人挂号界面，需要提供根据拼音字首的补全功能，例如当用户选择医生时可以仅提供医生姓名的首字母；挂号界面同样要求正确性判断；医生界面需要能提供挂号列表和收入列表两个界面，且序号、时间、总金额等信息不能错误。所有功能的实现界面必须具有用户友好性。

二、系统设计

1. 概要设计

本实验项目主要涉及四个模块，登录模块、挂号模块、收入报表模块和数据库操作模块。登录模块为医生或病人提供登录的功能，需要提供编号和密码；患者登录成功后转至挂号模块，提供相应信息和余额充足的情况下可进行挂号；医生登录成功后可查看收入报表及自己的患者名单；数据库管理模块中维护科室信息表、病人信息表、号种信息表、挂号信息表和科室医生 5 张表，其余模块均需要该模块为其提供数据的增删改查功能。整个挂号系统的流程图见图 2.1.数据库管理模块中的 5 张表见图 2.2.

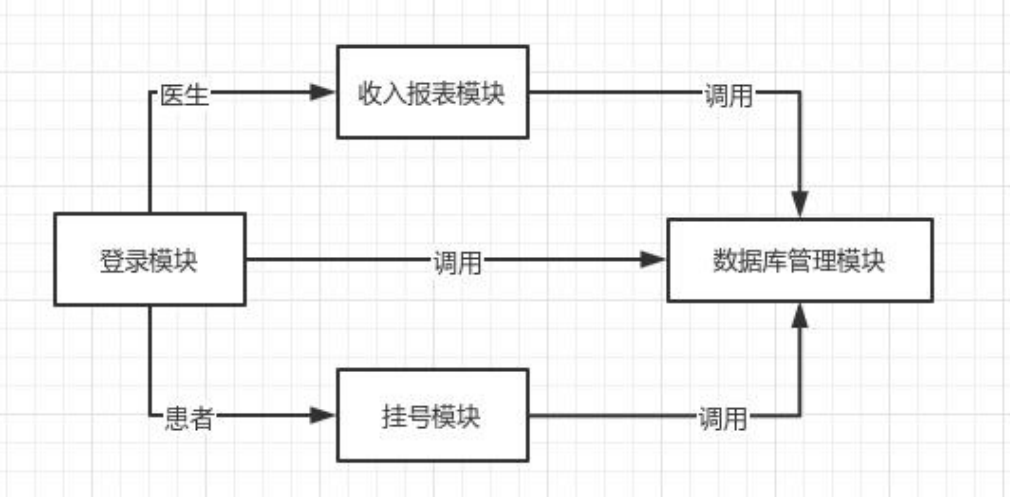


图 2.1 系统流程图

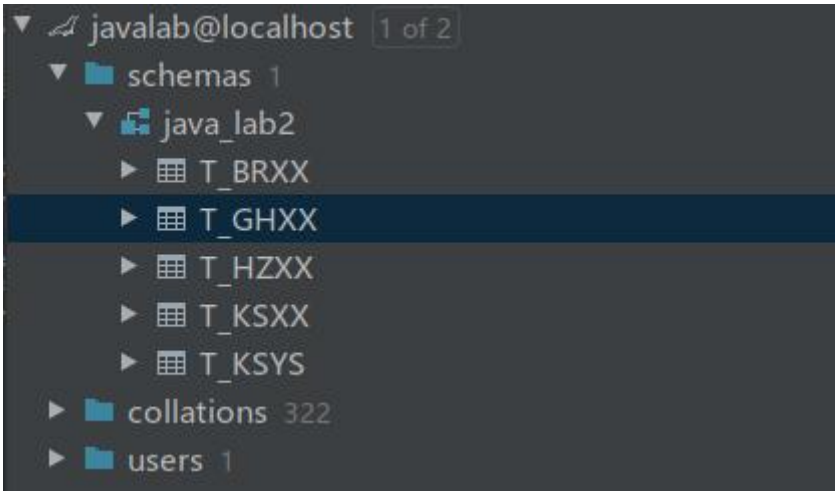


图 2.2 数据库管理模块

2. 详细设计

本实验采取 JavaFx 的 MVC 编程，界面的实现借助工具生成.fxml 文件，界面的控制采取对应的 Controller 文件，二者通过在 fxml 文件中指定 fx:controller 实现绑定，具体的各模块的实现如下。

1) 登录模块

函数从 start 函数开始执行，首先加载 fxml 文件，设置 root 节点，接着新建一个场景 scene，调用 stage 的 setScene 方法后，使用 show 方法即可看到如图 2.3 的登录界面。

图 2.3 登录界面

身份一栏设置的是 ChoiceBox，在 fxml 文件中指定两个值为“患者”和“医生”，默认

为患者，接着是接收编号和密码的文本框，在密码对应的文本框中设置一个对键盘回车按键的响应函数，当接受到回车按键时，默认调用登录 button 对应的函数，即实现了回车登录的功能需求。

当按下登录按键后，首先检查编号和密码是否为空，若不为空则调用数据库管理模块中的获取对应编号的医生或患者的信息，若找不到信息则编号输入错误，若密码不符则密码输入错误，均通过 alert 提醒重新输入。密码正确后根据登录类型选择下一场景，并将当前窗口关闭。

2) 挂号模块

挂号模块的设计如图 2.4 所示，包括 4 个 TextField、4 个 Combo Box 和 2 个 Button，需要首先选择科室名称、医生姓名、号种类别和应缴金额后才会根据余额和挂号费的关系进行判断，应缴金额显示的是号种对应的费用。若余额充足，则 4 个文本狂均不可编辑，防止死锁，挂号号码也只能在挂号成功后才会返回；若余额不足，则能对交款金额进行编辑，点击挂号即充值，若充值后余额充足才能成功，否则继续充值，所有提示信息均在底部通过红色的文字提示。

图 2.4 挂号界面

在科室名称、医生姓名和号种类别三个选项中均要求提供对应的拼音字首来查找对应的信息，以科室名称为例，如内科，提供 n 或 N 应该能过滤复选框中的内容，这里需要调用数据库管理模块提供的功能，根据输入的拼音在数据库中根据相应的`PYZS`字段查找，比较的功能通过 String 类的 contains 方法即可，查找后将相应信息返回，所有插入到对应的复选框中即可，代码片段见图 2.5,详情请见源代码部分。

```
if (pingyin != null) {
    inputDepartment.getItems().clear();
    inputDepartment.getItems().addAll(DBConnector.getInstance().getInfo( table: "department", pingyin));
} else {
    inputDepartment.getItems().clear();
    inputDepartment.getItems().addAll(DBConnector.getInstance().getDepInfor());
}
```

图 2.5 根据拼音过滤

3) 医生报表模块

该模块的界面见图 2.6.主要包括两个 Tab 存放病人列表和收入列表的内容，以及两个 DatePicker 选择查询的开始日期和截止日期，以及若干个 Label 标签和两个 Button。在对应的 Controller 文件中完成两个内部类 Register 和 Income 类分别存放 Tab 对应的病人挂号信息和收入信息。

图 2.6 报表模块界面

在报表界面初始化时使用 TableColumn 中的 setCellValueFactory 方法将内部类中的成员与 TableColumn 进行关联，为了能将信息加载到 Tab 中，还在 Controller 文件中定义了数组成员 patientLIst 和 incomeList，通过数据库管理模块提供的 getPatientInfo 和 getIncomeInfo 方法获取满足条件的记录后，使用 ObservableList 提供的 addAll 方法将所有信息加入到 Tab 中，部分代码的实现见图 2.7,详情请见源代码部分。

```
private void setPatientInfo(){
    registerNum.setCellValueFactory((TableColumn.CellDataFeatures<Register, String> param) ->
    patientName.setCellValueFactory((TableColumn.CellDataFeatures<Register, String> param) ->
    registerDate.setCellValueFactory((TableColumn.CellDataFeatures<Register, String> param) ->
    isSp.setCellValueFactory((TableColumn.CellDataFeatures<Register, String> param) -> param.g
    patientList.clear();
    patientList.addAll(DBConnector.getInstance().
            getPatientInfo(doctorId, startTime: pickDateStart.getValue().format(DateTimeFormatter
            LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"))));
    patientTable.setItems(patientList);
}
```

图 2.7 加入患者列表

4) 数据库连接模块

该模块负责根据传入的参数在数据库中获取对应的信息。在该模块中首先通过反射机制加载数据库驱动，然后根据用户名、密码、数据库名创建连接，本次实验采用线程安全的单例模式，创建连接的构造函数为 private 类型，通过 getInstance 方法获取一个数据库连接的实例，在获取时首先将 DBConnector 类上锁，防止其他进程的获取导致数据库操作的紊乱，实现线程安全。

需要提供获取患者信息、获取医生信息、更新登录日期、获取科室信息、获取挂号信息、获取医生姓名、根据医生编号获取患者信息、根据科室 ID 选择医生、挂号记录以及获取收入信息的功能，因此本模块主要操作为通过 SQL 语句进行查询，故代码见源代码部分，在此不做展示。通过该模块实现了本实验对数据库访问和维护操作的分离，模块之间的功能更加清晰。

三、软件开发

开发平台：Manjaro Linux x86_64 Intellij-IDEA-Ultimate-Edition-2020.1.1-1

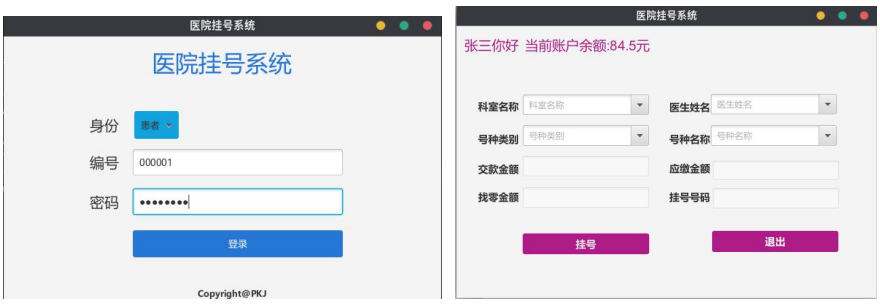
编译和连接方法：使用 IDEA 自带的编译和连接

调试方法：使用 IDEA 自带的 debug 功能

外部库:openjfx-11.0.2_linux-x64_bin-sdk

四、软件测试

首先测试患者部分，选择登录类型为“患者”，输入编号“000001”和密码“12345678”后点击登录按钮即可证明登录成功，登录成功后会显示账户余额。如图 2.8(a)(b)所示。



(a) 输入帐号和密码　　(b) 登录成功后界面

图 2.8 患者登录成功

点击退出后回到登录界面，尝试输入错误密码，点击登录按钮后会得到报错信息，如图 2.9 所示。

图 2.9 密码错误

重新登录后首先选择科室名称，在此验证拼音字首的实现功能，输入 n 得到过滤后的内科提示信息，测试结果如图 2.10 所示。

图 2.10 补全功能

选择专家医生张内科后，可以看到号种类别可选为普通号或专家号，而选择普通医生里内科后，可以看到号种类别可选仅为普通号，测试如图 2.11(a)(b),可知该功能实现正确。

(a) 普通医生

(b) 专家医生

图 2.11 号种选择

选择专家号后，可以看到应缴金额处出现了号种对应的金额，如图 2.12 所示。

图 2.12 应缴金额

点击挂号按钮，若余额充足会弹出提示框提示挂号成功，如图 2.13 所示。

图 2.13 挂号成功

挂号成功后，在找零金额，挂号号码处会显示结果，其中找零金额对应账户余额，如图 2.14 所示。

图 2.14 找零金额及挂号号码的显示

连续挂 5 次专家号，到第六次挂号时，首先会显示余额不足，要求充值，交款金额部分可编辑，选择交款金额为 10，再次点击挂号按钮，会显示“该号种已经挂满”的提示信息，如图 2.15 所示。

图 2.15 号种挂满

选择其他科室的专家号，在交款金额部分输入 1,测试充值效果，点击挂号后会在底部显示“余额不足或充值不足”的提示信息，如图 2.16 所示。

图 2.16 缴费不足

输入缴费金额为 10 后重新挂号可以挂号成功，如图 2.17 所示。

图 2.17 缴费足够

至此患者部分测试成功，开始测试医生部分，在登录界面选择登录类型为“医生”，输

入编号和姓名，按回车按键进入登录后的界面，如图 2.18 所示。可以看到挂号编号按升序排列，无丢号和不连续现象，且挂号编号与挂号日期时间无紊乱现象发生，符合考核标准。

图 2.18 医生界面

默认列表为病人列表，在选择日期内选择自己的患者名单，选择日期为当前日期。可以查询过去的挂号名单，如选择开始日期为 2020/5/25 日后，点击查询按钮，可以看到在 2020/5/25 日挂号的患者新增入名单，同时会新增标签提示开始日期和截止日期，均精确到秒。测试如图 2.19 所示。

图 2.19 查询病人列表

点击收入列表项，可以查看指定日期内的各科室收入及挂号人次，如图 2.20 所示，根据计算得知与挂号缴费金额总额一致。

图 2.20 收入列表

同样选择开始日期为 2020/5/25 日后，点击查询按钮，可以看到在 2020/5/25 日挂号的费用增加至收入列表中，如图 2.21 所示。

图 2.21 收入列表查询

五、特点与不足

1. 技术特点

本实验采用 JavaFx 的 MVC 编程模型，将针对数据库的操作和针对页面请求处理的操作分离开来，各模块间接口设置的非常清晰，调用关系十分清楚；在数据库管理模块中采用单例的设计模式，这是第一次在项目中真正实践这种设计模式，感觉在针对数据库的处理上使

用十分方便，也能够提供线程安全的访问；整体设计的页面具有用户友好性。

2. 不足和改进的建议

不足在于挂号页面中，当通过拼音字首去选择科室的时候，可能程序控制台会出现一些报错，因为还未选择完成之后输入框的内容同时会被其他处理程序调用，不过不影响整体功能的使用，只是在控制台会有报错，有时间会改进；页面设计不能随意更改大小，以最原始的大小效果最好；完成时间较久，导致延期了两天提交，在此感到抱歉。

六、过程和体会

1. 遇到的主要问题和解决方法

首先遇到的问题是不知道如何向复选框中添加默认的值，在网络上搜索后得到的解决方案是在 fxml 文件中添加即可，如图 6.1 所示。

```
<FXCollections fx:factory="observableArrayList">
    <String fx:value="患者" />
    <String fx:value="医生" />
```

图 6.1 复选框中添加默认内容

在设计页面的过程中还遇到过很多问题，例如如何将 TableColumn 中的内容与内部类的内容相关联、如何能让回车键有效果，这些问题都是在网络上搜索的答案，还有一些是参考了 github 上解决方案。

2. 课程设计的体会

本次实验设计的医院挂号系统难度比较高，这是第一次完成 JavaFx 的 MVC 编程实验，与其他类型的 MVC 编程模型有类似之处，但是整个实验的页面设计和控制器的设计还是很花时间的，其中关于数据库版块的设计更加繁琐，需要用使用嵌入式的 SQL 语言，且要解决很多格式上的问题，这一部分也是借鉴了很多网络上搜索到的解决方案才实现的；通过这次实验，我也懂得了实现一个客户端的数据库管理软件的难点，在此感谢给予我帮助的老师和同学，感谢老师没有因为我的延期责怪我；Java 课程的质量真的非常之高，我一定会向学弟学妹们推荐这门课程，再次感谢老师的辛勤付出！

七、源码和说明

1. 文件清单及其功能说明

[CS1703]_[U201714608]_[彭凯杰].mp4 文件为演示视频文件，测试同样可以看文档中的测试部分。

CS1703_201714608_彭凯杰.pdf 为实验报告。

UIDemo 为实验项目工程文件，其中 src 目录下为源代码文件。

2. 用户使用说明书

打开 Intellij IDEA，导入项目 UIDemo，在 Main 类中选择右键，点击“Run Main.main()”

即可运行，同时可以观看演示视频进行检查。

3. 源代码

Main.java

```
package hospital;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
public class Main extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        DBConnector.getInstance().connectDB("localhost",3306,"java_lab2",
                "pkj", "123456");
        FXMLLoader loader = new FXMLLoader(getClass().getResource("Login.fxml"));
        Parent root = loader.load();
        stage.setTitle("医院挂号系统");
        Scene scene = new Scene(root, 600, 360);
        stage.setScene(scene);
        stage.show();
    }
}
```

DBConnector.java

```
package hospital;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.sql.*;
import java.util.LinkedList;

public class DBConnector {
    private static volatile DBConnector instance = null;
    private Connection connection;
    private Statement statement;
```

```
/**
 * 加载 mariadb 驱动
 */
private DBConnector(){
    try {
        Class.forName("org.mariadb.jdbc.Driver");
    } catch (ClassNotFoundException e) {
        System.out.println("not found mariadb driver");
        e.printStackTrace();
    }
}

/**
 * 线程安全的单例模式
 * @return 数据库连接实例
 */
public static DBConnector getInstance(){
    if(instance == null){
        synchronized (DBConnector.class){
            if (instance == null) {
                instance = new DBConnector();
            }
        }
    }
    return instance;
}

/**
 * 建立数据库连接
 * @param hostName 主机名
 * @param port 端口号
 * @param dbName 数据库名
 * @param user 用户名
 * @param passwd 密码
 */
```

```
    public void connectDB(String hostName,int port,String dbName,String user,String
passwd){
        String url = "jdbc:mariadb://"+hostName+":"+port+"/" +

dbName+"?autoReconnect=true&characterEncoding=UTF-8&characterSetResults=UTF-8&serverT
imezone=UTC";
        try {
            connection = DriverManager.getConnection(url,user,passwd);
            statement = connection.createStatement();
        } catch (SQLException throwables) {
            System.out.println("cannot connect to mariadb");
            throwables.printStackTrace();
        }
    }

    /**
     * 关闭数据库连接
     */
    public void closeConnect(){
        try{
            connection.close();
            statement.close();
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    /**
     * 获取病人信息
     * @param pid 病人编号
     * @return 病人信息
     */
    public ResultSet getPatientInfo(String pid) {
        try {
            return statement.executeQuery("SELECT * FROM T_BRXX WHERE BRBH
=" + pid);
```

```
            } catch (SQLException e) {
                return null;
            }
        }

        /**
         * 获取医生信息
         * @param docid 医生编号
         * @return 医生信息
         */
        public ResultSet getDoctorInfo(String docid) {
            try {
                return statement.executeQuery("SELECT * FROM    T_KSYS WHERE YSBH
=" + docid);
            } catch (SQLException e) {
                return null;
            }
        }

        /**
         * 更新登录日期
         * @param loginType 登录身份
         * @param user 用户
         */
        public void updateDLRQ(String loginType, String user){
            try{
                String table = loginType.equals("患者")?"T_BRXX" : "T_KSYS";
                String bhType = loginType.equals("患者")?"BRBH" : "YSBH";
                statement.executeUpdate("update " + table + " set DLRQ =
DATE_FORMAT(NOW(),'%Y-%m-%d %H:%m:%s') " +
                        "where " + bhType + "=" + user);
                //System.out.println("updated");
            }catch (Exception e){
                e.printStackTrace();
            }
        }
```

```
    /**
     * 获取科室信息表
     * @return 科室编号+科室名称
     */
    public LinkedList<String> getDepInfor(){
        try{
            ResultSet resultSet = statement.executeQuery("SELECT * FROM T_KSXX");
            LinkedList<String> depInfo = new LinkedList<>();
            while (resultSet.next()){
                String depid = resultSet.getString("KSBH").trim();
                String depName = resultSet.getString("KSMC").trim();
                depInfo.add(depid + " " + depName);
            }
            return depInfo;
        }catch (Exception e){
            e.printStackTrace();
        }
        return null;
    }

    /**
     * 获取挂号信息
     * @param department 科室编号
     * @param isSP 是否为专家号
     * @param regTypePY 号种名称的拼音
     * @return 号种编号+号种+挂号费用
     */
    public LinkedList<String> getRegisterInfo(String department, String isSP, String
regTypePY){
        try{
            int ifSp = isSP.equals("专家号") ? 1 : 0;
            ResultSet result = statement.executeQuery("SELECT
HZBH,HZMC,PYZS,GHRS,GHFY FROM T_HZXX " +
                    "WHERE KSBH = " + department + " AND SFZJ = " + ifSp +
                        " AND (GHRS > (SELECT IFNULL(MAX(GHRC),0) FROM
```

```
T_GHXX " +
                                "WHERE HZBH=T_HZXX.HZBH AND
TO_DAYS(RQSJ)=TO_DAYS(NOW())))");

                    LinkedList<String> registerInfo = new LinkedList<>();
                    String registerPY;
                    while (result.next()){
                        String registerId = result.getString("HZBH").trim();
                        String registerName = result.getString("HZMC").trim();
                        registerPY = result.getString("PYZS").trim();
                        Double registerPay = result.getDouble("GHFY");
                        if(regTypePY == null) {
                            registerInfo.add(registerId+" "+registerName+" "+isSP+"
"+registerPay+"  元");
                        }
                        else{
                            if(registerPY.contains(regTypePY)) {
                                registerInfo.add(registerId+" "+registerName+" "+isSP+"
"+registerPay+"  元");
                            }
                        }
                    }
                    return registerInfo;
                }catch (Exception e){
                    e.printStackTrace();
                }
                return null;
            }

            /**
             * 获取医生名称
             * @param docid 医生编号
             * @return 医生名称
             */
            public String getDoctorName(String docid){
                try {
```

```
                ResultSet resultSet = statement.executeQuery("SELECT YSMC FROM
T_KSYS WHERE YSBH = "+ docid);
                while (resultSet.next())
                    return resultSet.getString("YSMC");
            }catch (Exception e){
                e.printStackTrace();
            }
            return null;
        }

        /**
         * 获取病人信息
         * @param docid 医生编号
         * @return 病人信息
         */
        public ObservableList<DoctorController.Register> getPatients(String docid){
            try {
                ObservableList<DoctorController.Register> patients =
FXCollections.observableArrayList();
                ResultSet resultSet = statement.executeQuery("SELECT
GHBH,BRMC,RQSJ,SFZJ " +
                        "FROM T_GHXX,T_BRXX,T_HZXX WHERE T_GHXX.YSBH=
"+ docid +
                        " AND T_GHXX.BRBH=T_BRXX.BRBH AND
T_GHXX.HZBH=T_HZXX.HZBH ORDER BY GHBH");
                while (resultSet.next()){
                    String registerId = resultSet.getString("GHBH").trim();
                    String patientName = resultSet.getString("BRMC").trim();
                    String regTime = resultSet.getString("RQSJ").trim();
                    int isSP = resultSet.getInt("SFZJ");
                    DoctorController.Register patient = new
DoctorController.Register(registerId,patientName,regTime,isSP);
                    patients.add(patient);
                }
                return patients;
            }catch (Exception e){
```

```
            e.printStackTrace();
        }
        return null;
    }

    /**
     * 根据输入表名或者拼音得到具体科室信息、号种信息或挂号信息
     * @param table 要访问的表
     * @param tablePY 表拼音
     * @return 具体信息
     */
    public LinkedList<String> getInfo(String table, String tablePY){
        LinkedList<String> info = new LinkedList<>();
        String queryTable = "";
        String number = "";                //编号
        String name = "";
        String fee = "";
        if(!table.equals("")){
            switch (table){
                case "department":
                    queryTable = "T_KSXX";
                    number = "KSBH";
                    name = "KSMC";
                    break;
                case "haozhong":
                    queryTable = "T_HZXX";
                    number = "HZBH";
                    name = "HZMC";
                    fee = "GHFY";
                    break;
                case "doctor":
                    queryTable = "T_KSYS";
                    number = "YSBH";
                    name = "YSMC";
                    break;
                default:
```

```
                        queryTable = null;
                        break;
                    }
                }
                try{
                    ResultSet resultSet = statement.executeQuery("SELECT * FROM " +
queryTable);
                    while (resultSet.next()){
                        if(tablePY.isEmpty()){
                            info.add(resultSet.getString(number) + " " +
resultSet.getString(name));
                            if(!fee.equals("")) {
                                info.add(info.getLast() + " " + resultSet.getDouble(fee) + "元");
                            }
                        }else{
                            //设置为大写
                            tablePY = tablePY.toUpperCase();
                            if(resultSet.getString("PYZS").contains(tablePY)){
                                info.add(resultSet.getString(number)+"
"+resultSet.getString(name));
                                if(!fee.equals("")){
                                    info.add(info.getLast()+" "+resultSet.getDouble(fee)+"元");
                                }
                            }
                        }
                    }
                }catch (Exception e){
                    e.printStackTrace();
                }
                return info;
            }

            /**
             * 根据输入的科室号选择医生
             * @param depId 科室编号
             * @param docPY 医生拼音
```

```
     * @return  医生信息
     */
    public LinkedList<String> getDepartDoctor(String depId, String docPY){
        LinkedList<String> info = new LinkedList<>();
        try{
            ResultSet resultSet = statement.executeQuery("SELECT YSBH,YSMC,PYZS
FROM T_KSYS " +
                    "WHERE KSBH = " + depId);
            while (resultSet.next()){
                if(docPY == null){
                    info.add(resultSet.getString("YSBH")+" " +
resultSet.getString("YSMC"));
                }else
                {
                    //设置为大写
                    docPY = docPY.toUpperCase();
                    if(resultSet.getString("PYZS").contains(docPY)){
                        info.add(resultSet.getString("YSBH")+" " +
resultSet.getString("YSMC"));
                    }
                }

            }
        }catch (Exception e){
            e.printStackTrace();
        }
        return info;
    }

    /**
     *  尝试挂号
     * @param HZBH  号种编号
     * @param YSBH  医生编号
     * @param BRBH  病人编号
     * @param GHFY  挂号费用
     * @param patientPay  病人付款
```

```
     * @return 状态信息
     */
    public String tryRegister(String HZBH,String YSBH,String BRBH,double GHFY,double patientPay){
        try{
            ResultSet resultSet = statement.executeQuery("SELECT * FROM T_GHXX ORDER BY GHBH DESC limit 1");
            int registerId,count,maxRegister = 0;   //挂号 id 和当前挂号人数，最大挂号人数
            if(!resultSet.next()){
                registerId = 1;
            } else{
                registerId = Integer.parseInt(resultSet.getString("GHBH"))+1;
            }
            resultSet = statement.executeQuery("SELECT * FROM T_GHXX WHERE HZBH = " +
                    HZBH + " and RQSJ >= DATE_FORMAT(NOW(),'%Y-%m-%d 00:00:00') ORDER BY GHRC DESC limit 1");
            if(!resultSet.next()){
                count = 0;
            } else {
                count = resultSet.getInt("GHRC");
            }
            resultSet = statement.executeQuery("SELECT * FROM T_HZXX WHERE HZBH = " + HZBH);
            while (resultSet.next()){
                maxRegister = resultSet.getInt("GHRS");
            }
            if(count >= maxRegister){
                return "当前号种挂号人数已满,请选择其他号种";
            }
            statement.executeUpdate(
                    String.format("INSERT INTO T_GHXX VALUE (\"%06d\",\"%s\",\"%s\",\"%s\",%d,0,%.2f, current_timestamp)",
                            registerId, HZBH, YSBH, BRBH, count+1, GHFY));
```

```
                statement.executeUpdate(String.format("UPDATE T_BRXX SET YCJE = %.2f
WHERE BRBH = %s",patientPay-GHFY,BRBH));
                double refund = Double.parseDouble(String.format("%.2f",patientPay-GHFY));
                RegisterController.setPatientYCYE(refund);
                return "success "+registerId;
            }catch (Exception e){
                e.printStackTrace();
            }
            return null;
        }

        /**
         * 医生查询时间病人信息
         * @param doctorID 医生编号
         * @param startTime 开始时间
         * @param endTime 结束时间
         * @return 病人列表
         */
        public LinkedList<DoctorController.Register> getPatientInfo(String doctorID, String
startTime, String endTime){
            LinkedList<DoctorController.Register> patients = new LinkedList<>();
            try{
                String query = "SELECT
T_GHXX.GHBH,T_BRXX.BRMC,T_GHXX.RQSJ,T_HZXX.SFZJ " +
                        "FROM T_GHXX,T_BRXX,T_HZXX WHERE T_GHXX.YSBH =
"+ doctorID +
                        " AND T_HZXX.HZBH = T_GHXX.HZBH AND T_GHXX.BRBH
= T_BRXX.BRBH AND RQSJ >= '" + startTime + "' and RQSJ <= '"+ endTime + "'";
                ResultSet resultSet = statement.executeQuery(query);
                while (resultSet.next()){
                    patients.add(new
DoctorController.Register(resultSet.getString("GHBH"),resultSet.getString("BRMC"),resultSet.get
String("RQSJ"),resultSet.getInt("SFZJ")));
                }
            }catch (Exception e){
                e.printStackTrace();
```

```
            }
            return patients;
        }

        /**
         * 获取收入信息
         * @param startTime 开始时间
         * @param endTime 结束时间
         * @return 收入信息列表
         */
        public ObservableList<DoctorController.Income> getIncomeInfo(String startTime, String
endTime){
            ObservableList<DoctorController.Income> incomes =
FXCollections.observableArrayList();
            try {
                String query = "SELECT
KSMC,T_GHXX.YSBH,YSMC,T_HZXX.SFZJ,COUNT(*) AS CNT ,SUM(T_GHXX.GHFY) AS
INCOME FROM T_GHXX,T_KSXX,T_KSYS,T_HZXX WHERE " +
                        "T_GHXX.YSBH=T_KSYS.YSBH AND
T_KSYS.KSBH=T_KSXX.KSBH AND T_GHXX.HZBH=T_HZXX.HZBH AND
T_GHXX.RQSJ >= '" +
                        startTime + "' AND T_GHXX.RQSJ <= '" + endTime + "' AND
T_GHXX.THBZ = 0 GROUP BY T_GHXX.YSBH";
                ResultSet resultSet = statement.executeQuery(query);
                while (resultSet.next()){
                    incomes.add(new
DoctorController.Income(resultSet.getString("KSMC"),resultSet.getString("YSBH"),resultSet.getSt
ring("YSMC"),

resultSet.getInt("SFZJ"),resultSet.getInt("cnt"),resultSet.getDouble("income")));
                }
            }catch (Exception e){
                e.printStackTrace();
            }
            return incomes;
        }
```

```
}
```

LoginController.java

```
package hospital;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.input.KeyCode;
import javafx.stage.Stage;

import java.sql.ResultSet;


public class LoginController {
    @FXML
    private Button loginButton;
    @FXML
    private TextField inputUsername;
    @FXML
    private PasswordField inputPassword;
    @FXML
    public ChoiceBox<String> loginType;

    String user = "患者"; //默认用户为患者

    /* initialize 方法，这个方法则会自动在构造函数之后调用
    * 使用了匿名函数、lambda 表达式
    * 此处将按下回车键等同于点击 loginButton
    */
    @FXML
    public void initialize(){
        inputPassword.setOnKeyPressed(e -> {
            if (e.getCode() == KeyCode.ENTER){
```

```
                    onLoginClick();
                }
            });
        }

        public void onLoginClick(){
            if(loginType.getValue().equals("医生")) //若在下拉栏选择医生，则用户切换为医
生
                user = "医生";
            try{
                if(tryLogin()){
                    String nextScene = "Register.fxml";              //下一切换场景
                    String title = "医院挂号系统";
                    if(user.equals("医生")){ //医生成功登录
                        DoctorController.setDoctorId(inputUsername.getText());

DoctorController.setDoctorName(DBConnector.getInstance().getDoctorName(inputUsername.getTe
xt()));
                        nextScene = "Doctor.fxml";
                        title = "医生系统";
                    }
                    else{ //患者成功登录
                        RegisterController.setPatientId(inputUsername.getText().trim());
                        ResultSet resultSet =
DBConnector.getInstance().getPatientInfo(inputUsername.getText().trim());
                        while (resultSet.next()){

RegisterController.setPatientName(resultSet.getString("BRMC"));

RegisterController.setPatientYCYE(resultSet.getDouble("YCJE"));
                        }
                    }
                    Stage nowStage = (Stage) loginButton.getScene().getWindow();
                    nowStage.close();
                    Parent root = FXMLLoader.load(getClass().getResource(nextScene));
                    Scene scene = new Scene(root, 620, 400);
```

```
                scene.setRoot(root);
                Stage stage = new Stage();
                stage.setTitle(title);
                stage.setScene(scene);
                stage.show();
            }
        }catch (Exception e){
            e.printStackTrace();
        }
    }

    /**
     * 登录
     * @return   登录是否成功
     */
    public boolean tryLogin(){

        //检查是否输入姓名和密码
        if(inputUsername.getText().isEmpty()){
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText("姓名为空！");
            alert.showAndWait();
            return false;
        }

        if(inputPassword.getText().isEmpty()){
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setHeaderText("密码为空！");
            alert.showAndWait();
            return false;
        }

        //去除多余的空格
        inputUsername.setText(inputUsername.getText().trim());
        inputPassword.setText(inputPassword.getText().trim());
```

```
ResultSet result = null;
if(user.equals("患者")){
    result = DBConnector.getInstance().getPatientInfo(inputUsername.getText());
}
else{
    result = DBConnector.getInstance().getDoctorInfo(inputUsername.getText());
}

if(result == null){
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setHeaderText("数据库读取错误！");
    alert.showAndWait();
}
try {
    assert result != null;
    if(!result.next()){
        Alert alert = new Alert(Alert.AlertType.INFORMATION);          //提示
信息

        alert.setHeaderText("用户不存在！");
        alert.showAndWait();
        return false;
    }
    else if(!result.getString("DLKL").equals(inputPassword.getText())){ //登录口令
不对

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setHeaderText("密码错误！");
        alert.showAndWait();
        return false;
    }
}catch (Exception e){
    e.printStackTrace();
    return false;
}

//登录成功
DBConnector.getInstance().updateDLRQ(user, inputUsername.getText());//  更新登
```

录事件

```
            return true;
        }
    }
```

RegisterController.java

```
package hospital;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;


import javax.swing.*;
import java.io.IOException;
import java.sql.ResultSet;

public class RegisterController {

    @FXML
    private ComboBox<String> inputDepartment;
    @FXML
    private ComboBox<String> inputDoctor;
    @FXML
    private ComboBox<String> inputIfSp;
    @FXML
    private ComboBox<String> inputNameRegister;
    @FXML
    private TextField feeLabel;   //费用
    @FXML
    private Label statusLabel;   //状态信息
```

```
    @FXML
    private TextField refundLabel;   //找零
    @FXML
    private Label welcomeLabel;   //欢迎 label
    @FXML
    private TextField inputMoney;   //输入的交款金额
    @FXML
    private TextField registerNum;   //挂号号码
    @FXML
    private Button registerButton;   //挂号
    @FXML
    private Button exitButton;   //退出按钮
    @FXML
    private AnchorPane pane;

    private static String patientName;   //病人姓名
    private static String patientId;   //病人编号
    private static double patientYCYE;   //病人预存金额

    private static double registerFee;   //挂号费用
    private static double patientPay;   //病人付费费用

    private int lastInputDepartment = -1;
    private int lastInputDoctor = -1;
    private int lastInputIfSP = -1;
    private int lastInputRegister = -1;

    @FXML
    void initialize() {
        welcomeLabel.setText( patientName + "你好\t 当前账户余额:" + patientYCYE + "
元");
        inputDepartment.addEventHandler(ComboBox.ON_SHOWING, event -> {
            if (!inputDepartment.getEditor().getText().isEmpty()) {
                return;
            }
            setInputDepartment();
```

```
                event.consume();
            });
            inputDepartment.getEditor().setOnKeyReleased(keyEvent -> {
                if (keyEvent.getCode() == KeyCode.ENTER || keyEvent.getCode() ==
KeyCode.UP
                        || keyEvent.getCode() == KeyCode.DOWN){
                    return;
                }
                setInputDepartment();
                if (!inputDepartment.isShowing()){
                    inputDepartment.show();
                }
                else {
                    inputDepartment.hide();
                    inputDepartment.show();
                }
            });

            inputMoney.setOnKeyReleased(keyEvent -> {
                if (keyEvent.getCode() == KeyCode.ENTER || keyEvent.getCode() ==
KeyCode.BACK_SPACE)
                    return;
                String s = inputMoney.getText().trim();
                if (!s.matches("([1-9]+[0-9]*|0)(\\.[\\d]+)?")) {
                    setStatusLabel("输入非法，请重新输入");
                    return;
                }
                setStatusLabel("");
                patientPay = Double.parseDouble(s);
            });

            exitButton.setOnKeyReleased(keyEvent -> {
                if (keyEvent.getCode() == KeyCode.ENTER) {
                    try {
                        onExitButtonClick();
                    } catch (IOException e) {
```

```
                    e.printStackTrace();
                }
            }
        });
        registerButton.setOnKeyReleased(keyEvent -> {});
    }

    /**
     * 退出按钮点击事件
     * @throws IOException 异常
     */
    @FXML
    void onExitButtonClick() throws IOException {
        Stage nowStage = (Stage) exitButton.getScene().getWindow();
        nowStage.close();
        Parent root = FXMLLoader.load(getClass().getResource("Login.fxml"));
        Scene scene = new Scene(root, 600, 360);
        scene.setRoot(root);
        Stage stage = new Stage();
        stage.setTitle("医院挂号系统");
        stage.setScene(scene);
        stage.show();
    }

    /**
     * 设置复选框科室信息,可以根据拼音选择科室
     */
    private void setInputDepartment() {
        String pingyin = inputDepartment.getEditor().getText();//获取输入的拼音
        if (pingyin != null) {
            inputDepartment.getItems().clear();

inputDepartment.getItems().addAll(DBConnector.getInstance().getInfo("department", pingyin));
        } else {
            inputDepartment.getItems().clear();
            inputDepartment.getItems().addAll(DBConnector.getInstance().getDepInfor());
```

```
            }
        }

        /**
         * 设置复选框医生信息，可以根据拼音选择医生
         */
        private void setInputDoctor() {
            String depart = inputDepartment.getEditor().getText();
            if (depart.isEmpty()) {
                setStatusLabel("请先选择科室");
                return;
            }
            String[] strings = depart.split(" ");
            depart = strings[0];
            String pinyin = inputDoctor.getEditor().getText();
            if (pinyin != null) {
                inputDoctor.getItems().clear();

inputDoctor.getItems().addAll(DBConnector.getInstance().getDepartDoctor(depart, pinyin));
            } else {
                inputDoctor.getItems().clear();

inputDoctor.getItems().addAll(DBConnector.getInstance().getDepartDoctor(depart, null));
            }
        }

        /**
         * 设置复选框专家号普通号
         */
        private void setInputIfSp() {
            String depart = inputDepartment.getEditor().getText();
            if (depart.isEmpty()) {
                setStatusLabel("请先选择科室");
                return;
            }
            String doctor = inputDoctor.getEditor().getText();
```

```
        if (doctor.isEmpty()) {
            setStatusLabel("请先选择医生");
            return;
        }
        String[] strings = doctor.split(" ");
        doctor = strings[0];
        boolean flag = false;
        try {
            ResultSet resultSet = DBConnector.getInstance().getDoctorInfo(doctor);
            while (resultSet.next()) {
                if (resultSet.getInt("SFZJ") == 1) {
                    flag = true;
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        inputIfSp.getItems().clear();

        String pinyin = inputIfSp.getEditor().getText().trim();
        if (pinyin.isEmpty()) {
            if (flag) {
                inputIfSp.getItems().add("专家号");
            }
            inputIfSp.getItems().add("普通号");
        } else {
            pinyin = pinyin.toLowerCase();
            if (flag && "zjh".contains(pinyin)) {
                inputIfSp.getItems().add("专家号");
            }
            if ("pth".contains(pinyin)) {
                inputIfSp.getItems().add("普通号");
            }
        }
    }
```

```
        /**
         * 设置复选框挂号信息：号种编号+号种名称+医生+号种类别+挂号费用
         */
        private void setInputNameRegister() {
            String pinyin = inputNameRegister.getEditor().getText();
            String depart = inputDepartment.getEditor().getText();
            if (depart.isEmpty()) {
                setStatusLabel("请先选择科室");
                return;
            }
            String doctor = inputDoctor.getEditor().getText();
            if (doctor.isEmpty()) {
                setStatusLabel("请先选择医生");
                return;
            }
            String specialist = inputIfSp.getEditor().getText();
            if (specialist.isEmpty()) {
                setStatusLabel("请先选择挂号类别");
                return;
            }
            String[] departs = depart.split(" ");
            if (pinyin.isEmpty()) {
                inputNameRegister.getItems().clear();

inputNameRegister.getItems().addAll(DBConnector.getInstance().getRegisterInfo(departs[0],
inputIfSp.getEditor().getText(), null));
            } else {
                pinyin = pinyin.toLowerCase();
                inputNameRegister.getItems().clear();

inputNameRegister.getItems().addAll(DBConnector.getInstance().getRegisterInfo(departs[0],
inputIfSp.getEditor().getText(), pinyin));
            }
        }

        /**
```

```
     * 检查预存金额是否够付款
     */
    private void checkYCJE() {
        if (patientYCYE < registerFee) {
            setStatusLabel("余额不足,请充值");
            inputMoney.setText("");
            inputMoney.setEditable(true);
            inputMoney.setDisable(false);
        } else {
            inputMoney.setText("余额充足，使用余额付款");
            inputMoney.setText("");
            inputMoney.setEditable(false);
            inputMoney.setDisable(true);
        }
    }

    /**
     * 挂号按钮点击事件
     */
    @FXML
    void onRegisterClick() {
        String depart = inputDepartment.getEditor().getText();
        if (depart.isEmpty()) {
            setStatusLabel("请先选择科室");
            return;
        }
        String doctor = inputDoctor.getEditor().getText();
        if (doctor.isEmpty()) {
            setStatusLabel("请先选择医生");
            return;
        }
        String specialist = inputIfSp.getEditor().getText();
        if (specialist.isEmpty()) {
            setStatusLabel("请先选择挂号类别");
            return;
        }
```

```
            String registerName = inputNameRegister.getEditor().getText();
            if (registerName.isEmpty()) {
                setStatusLabel("请选择挂号信息");
                return;
            }
            if (!inputMoney.isDisable() && !inputMoney.getText().isEmpty()) {
                if (!inputMoney.getText().matches("([1-9]+[0-9]*|0)(\\.[\\d]+)?")) {
                    setStatusLabel("输入非法，请重新输入");
                    return;
                }
                patientPay = Double.parseDouble(inputMoney.getText());
            } else {
                patientPay = 0;
            }
            if (patientYCYE < registerFee && patientYCYE + patientPay < registerFee) {
                setStatusLabel("缴费金额不足或余额不足");
                inputMoney.setText("");
                inputMoney.setEditable(true);
                inputMoney.setDisable(false);
                return;
            }
            String[] doctors = doctor.split(" ");
            String[] registers = registerName.split(" ");
            pane.setDisable(true);
            double tmp = patientPay;
            patientPay += patientYCYE;

            String status = DBConnector.getInstance().tryRegister(registers[0], doctors[0],
patientId, registerFee, patientPay);
            if (status.contains("success")) {
                welcomeLabel.setText(patientName + "你好\t 当前账户余额:" + patientYCYE
+ "元");
                if (tmp == 0){
                    refundLabel.setText(patientYCYE + "  元");
                }
                else {
```

```
                    refundLabel.setText(String.format("%.2f", patientPay - registerFee) + "  元
");
                    statusLabel.setText("找零已存入余额");
                    statusLabel.setStyle("-fx-text-fill: #000000;");
                }
                String[] temps = status.split(" ");
                registerNum.setText(temps[1]);
                JOptionPane.showMessageDialog(null, "挂号成功!", "医院门诊挂号系统",
JOptionPane.PLAIN_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, status, "医院门诊挂号系统",
JOptionPane.ERROR_MESSAGE);
            }
            pane.setDisable(false);
        }

        public static String getPatientName() {
            return patientName;
        }

        public static void setPatientName(String patientName) {
            RegisterController.patientName = patientName;
        }

        public static String getPatientId() {
            return patientId;
        }

        public static void setPatientId(String patientId) {
            RegisterController.patientId = patientId;
        }

        public static void setPatientYCYE(double patientYCYE) {
            RegisterController.patientYCYE = patientYCYE;
        }
```

```
    public static double getPatientYCYE() {
        return patientYCYE;
    }
}
```