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 ：　校交 1601 班

学　　号 ：　U201610504

姓　　名 ：　刘逸帆

指导教师 ：　吕新桥

2019 年 5 月 12 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	TINYINT	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(8)	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	TINYINT	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	该号种当天已挂号人次
THBZ	TINYINT	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
KBSJ	DATETIME	否	否	是	看病时间：“空”未看 已看病则不能退号

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“内二课”。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，每个号重的 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	章子怡	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

对于病人的操作界面，需要实现挂号、退号功能，同时需要包涵人性化的过滤功能和提示信息；对于医生的操作界面，需要实现两种统计功能和额外的更便捷的过滤功能。

对于程序的功能而言，不仅需要程序具有健壮性，在发生错误的时候不能崩溃，而且要求 UI 对用户友好，需要支持鼠标和键盘混合输入的操作方式。

二、系统设计

1. 概要设计

医院挂号信息管理系统的结构可分为如下 2 个部分：管理系统程序部分以及数据库部分。程序部分负责相应用户的输入，与数据库沟通、处理数据并返回处理的结果，其可分为 4 个模块：登录界面、病人操作界面、医生操作界面以及数据库连接模块；登录界面用于通过数据库连接模块检查用户的登录信息是否正确，确认登陆后根据用户的选择唤醒医生操作界面或病人操作界面，医生操作界面或病人操作界面则继续调用数据库模块与数据库进行沟通，处理数据并返回结果。数据库部分则用于存储数据并可同时服务多个客户端。总体的模块图如图 2.1 所示。

此外，程序入口被封装为一个较小的 main 模块，用于执行包括加载数据库驱动、连接数据库、启动化图形界面引擎在内的初始化操作，最终唤醒登录界面。如果不能成功连接数据库则此模块进行相应的处理并退出程序。

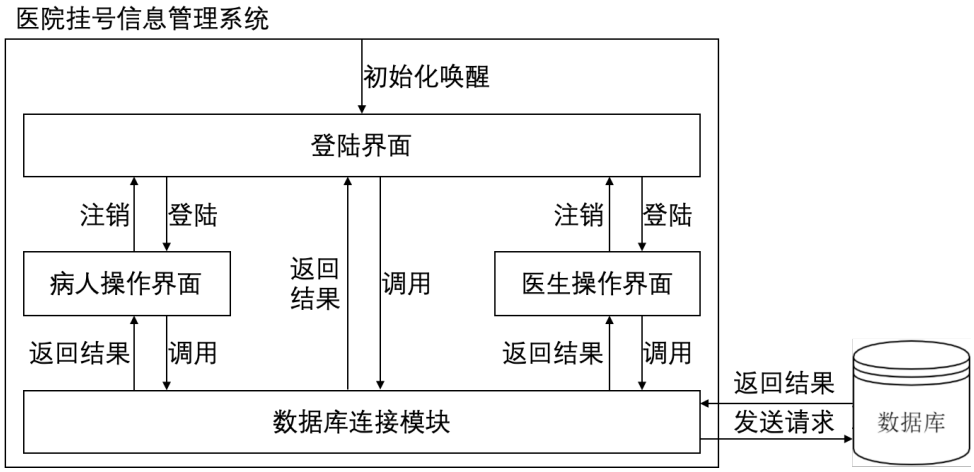

图 2.1 总体模块图

系统总体流程的状态转移图如图 2.2 所示，系统初始化成功后首先进入登录界面并等待用户输入登录信息，然后通过查询数据库判断登录信息是否匹配，如果匹配则登录，否则提示错误并等待用户重新输入登录信息。登录成功后通过用户选择的登陆方式来加载医生操作界面或病人操作界面。此后进入图形界面引擎控制的事件循环。当有事件到来时（如用户输入、点击等）处理事件、显示返回结果并继续等待下一个事件。所有的状态都是可逆的，也就是说用户可以通过退出按钮来进行注销并回到上一个界面，以此来增加界面的可操作性。

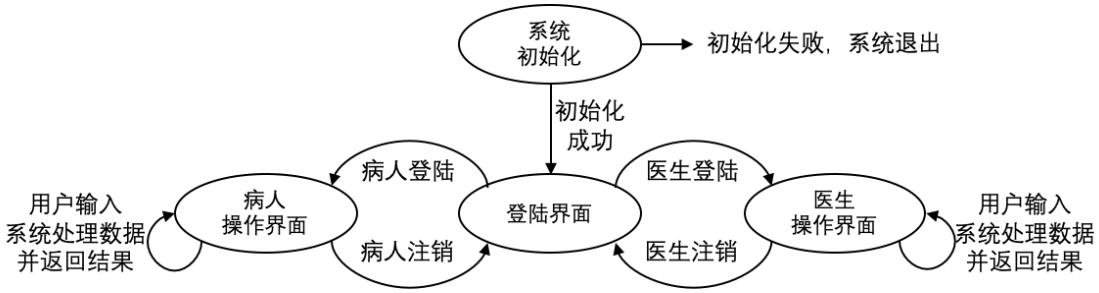

图 2.2 总体状态转移图

2. 详细设计

根据 1.2 小节中的需求分析以及 2.1 小节中的总体设计，分别设计具体的数据库连接模块、登陆界面、病人操作界面、医生操作界面如下：

1) 数据库连接模块详细设计

模块功能：数据库连接模块是沟通系统程序与数据库的重要环节，同时其位于医院挂号信息管理系统的最底层，是整个图形界面交互系统的基础。此模块中具体需要包含：包含注册 JDBC 驱动行为的数据库连接模块构造函数、通过驱动与数据库建立连接的 connectDB 方法、断开与数据库连接的 disconnectDB 方法、执行 SQL 查询语句的 runQueryCommand 方法、执行 SQL 更新语句的 runUpdateCommand 方法。数据库连接模块应该在操作界面初始化前被初始化。

模块实现：在数据库连接模块类中设置两个私有成员 conn 和 stmt，分别用于保持程序的数据库连接和执行数据库的查询更新等操作；在建立程序与数据库的连接时，参照官方文档构造字符串形式的数据库链接并调用 DriverManager 类的静态方法 getConnection 即可，通过

构造成功的连接 conn 的 createStatement 方法即可创建用于具体执行查询和更新等 SQL 语句的连接状态 stmt；在方法 runQueryCommand 和 runUpdateCommand 中，将 String 格式的 SQL 语句作为输入参数传入，随后分别通过 stmt 的 executeQuery 和 executeUpdate 方法即可得到对应的操作结果，从而实现对数据库的查询或更新。

2) 登陆界面详细设计

模块功能：登陆界面在系统初始化与数据库的连接后会被首先唤醒，需要在该模块中判断用户选择的登陆方式，并通过查询数据库来判断登陆是否成功，从而在系统中给出对应的提示信息或登陆成功后的不同操作界面。此模块中具体需要包含：病人登陆或医生登陆的选择框、账号输入栏、密码输入栏、登录按钮。

模块实现：对于界面中病人身份与医生身份的选框，需要做到同一时刻必须选择且仅可选择一个选框；对于密码输入框，应该将输入的密码以密文样式回显在界面中；在用户单击登陆按钮时，首先对用户输入是否为空进行判断，若账号或密码输入为空则在系统中进行对应提示，若输入合法则通过数据库连接查询是否有对应的账号信息，此时若查找到与输入一致的账号信息则根据用户选择的身份唤醒病人操作界面或医生操作界面，否则提示账号或密码输入有误。

流程图：登陆逻辑的流程图如图 2.3 所示。

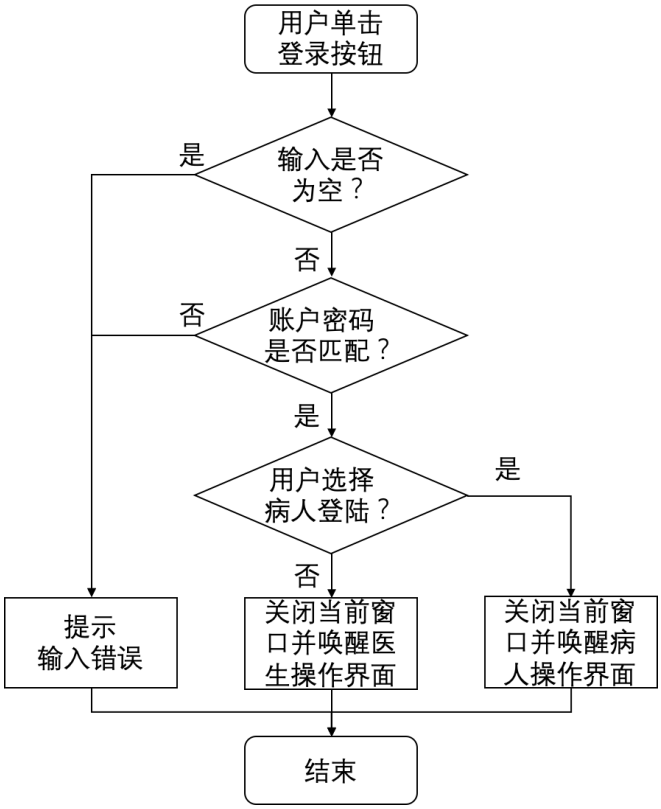

图 2.3 登陆流程

3) 病人操作界面详细设计

模块功能：病人操作界面需要为病人提供基本的挂号功能。在实现挂号功能的过程中需要提供对输入信息的核对以及不同缴费方式的支持。

模块实现：对于挂号功能，需要在已有输入信息的基础上对当前输入信息以下拉选项的形式进行预测；若交款金额输入框的输入为空，则系统认为使用账户余额付费；应交金额、找零金额和挂号号码输入框应禁止用户对其进行修改，其中挂号号码输入框应在挂号成功后回显系统自动分配的号码。在用户单击确定按钮时，系统首先将输入的号种信息与数据库中信息进行匹配，若未匹配到号种则输出相应错误提示；在病人使用余额支付并余额不足时，系统输出相应的错误提示；在病人提交挂号申请时再次对数据库中对应号种的应缴金额进行确认，若数据不一致，提示用户重新确认挂号申请。

流程图：病人操作界面逻辑的流程图如图 2.4 所示。

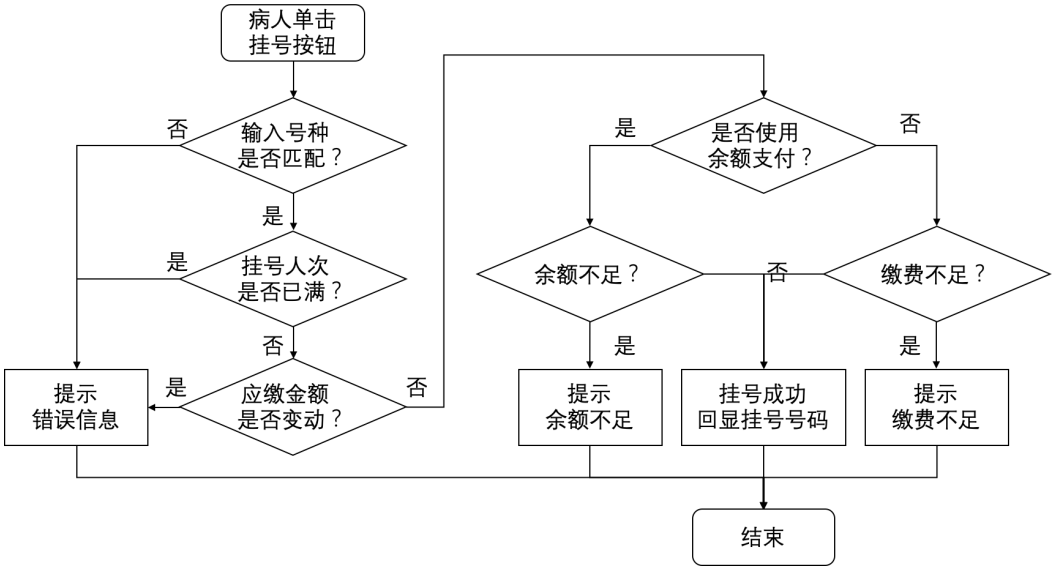

图 2.4 挂号流程

4) 医生操作界面详细设计

模块功能：医生操作界面需要为医生提供挂号信息的检索以及收入信息的检索。在实现挂号信息以及收入信息检索的过程中，检索到的数据均以表格的形式在系统中进行输出展示，展示信息涵盖的时间范围可以通过操作界面中的控件进行设置，从而筛选出指定数据。

模块实现：对于挂号信息，需要依据当前登陆医生的信息对数据库中信息进行匹配，从而在表单中显示指定范围的信息；对于收入信息，需要在数据库中对各条符合要求的记录进行分类与计算，最后将指定格式的数据传入表单进行输出。在设计过程中为简化对表格的更新操作，设计 Register 挂号信息类与 Income 医生收入类，分别为两个类构造实例后，构造的实例能够被绑定为界面中用于展示的表格的数据来源，从而对两个实例进行更新队列内容即可实现对输出控件中数据的刷新，而不用重新渲染控件。

三、软件开发

本程序以及测试程序的编写于测试均在 Mac OS X 系统下完成，具体开发以及测试环境如下：

1) 开发环境：

开发操作系统：Mac OS X

JDK：OpenJDK 1.8.0

集成开发环境：Eclipse 2018.12

2) 测试环境：

测试操作系统：Mac OS X

JRE：JAVA SE 8

数据库(OS X)：Mysql 8.0.11

四、软件测试

对软件的三个主要功能界面依次进行测试，具体测试内容与测试结果如下：

1) 登录测试

在登陆界面中对各种可能出现的边界情况进行测试，首先测试若输入为空时尝试进行登陆操作的结果，测试结果如图 4.1 所示。

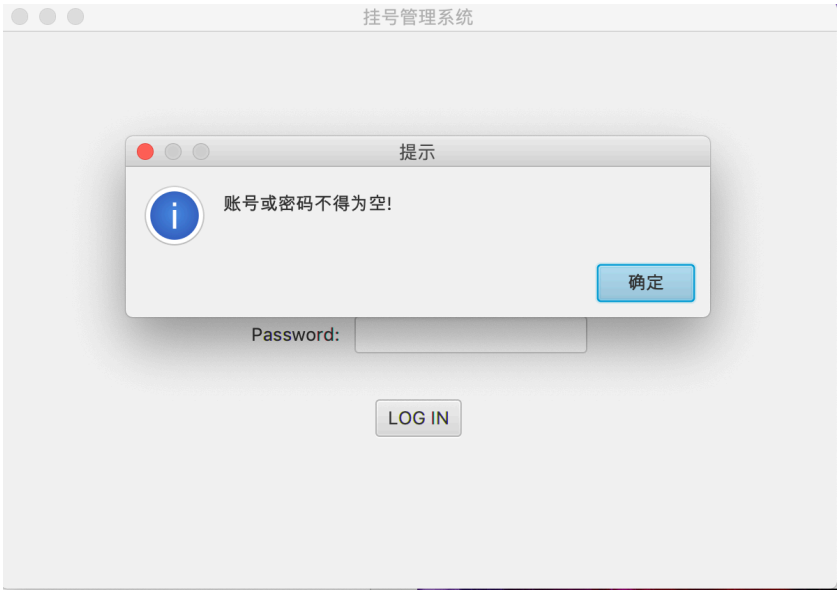

图 4.1 登陆账号或密码为空

尝试使用错误用户名或错误密码登陆系统，测试结果如图 4.2 所示。

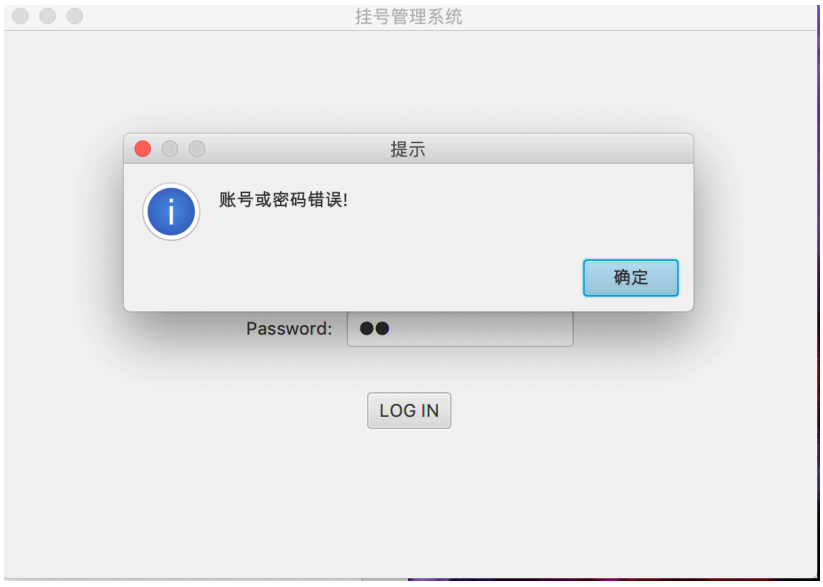

图 4.2 登陆账号或密码错误

尝试以正确的病人账号进行登陆，登陆成功后的病人操作界面如图 4.3 所示。

图 4.3 病人操作界面

尝试以正确的医生账号进行登陆，登陆成功后的医生操作界面如图 4.4 所示。

图 4.4 医生操作界面

经过对各种边界条件和正确操作的测试，说明登陆界面能够正确对不规范输入进行检测并提示错误，同时能够以正确的逻辑进行页面跳转，验证了登陆逻辑的正确性。

2) 病人操作界面测试

对病人操作界面的挂号功能进行测试，首先测试在文本框中进行输入时系统的自动补全功能，测试结果如图 4.5 所示。

图 4.5 挂号信息自动补全

在对后续文本框进行输入的过程中，文本框自动补全的候选项应关联于已有的输入，对这种关联关系进行测试，测试结果如图 4.6 所示。

图 4.6 自动补全依赖于已有输入

在挂号时输入错误组合的号种信息，测试系统能否正常报错，测试结果如图 4.7 所示。

图 4.7 系统识别错误号种

在挂号时使用余额进行支付，测试余额不足时系统能否正常报错，测试结果如图 4.8 所示。

图 4.8 系统识别余额不足

在挂号时使用现金缴费方式进行支付，测试支付金额不足时系统能否正常报错，测试结果如图 4.9 所示。

图 4.9 系统识别缴费金额不足

在输入号种信息过程中该号种的应缴金额可能会发生变化，系统对此情形应给予报错提示，在数据库中修改号种的应缴金额信息并测试挂号报错信息，测试结果如图 4.10 所示。

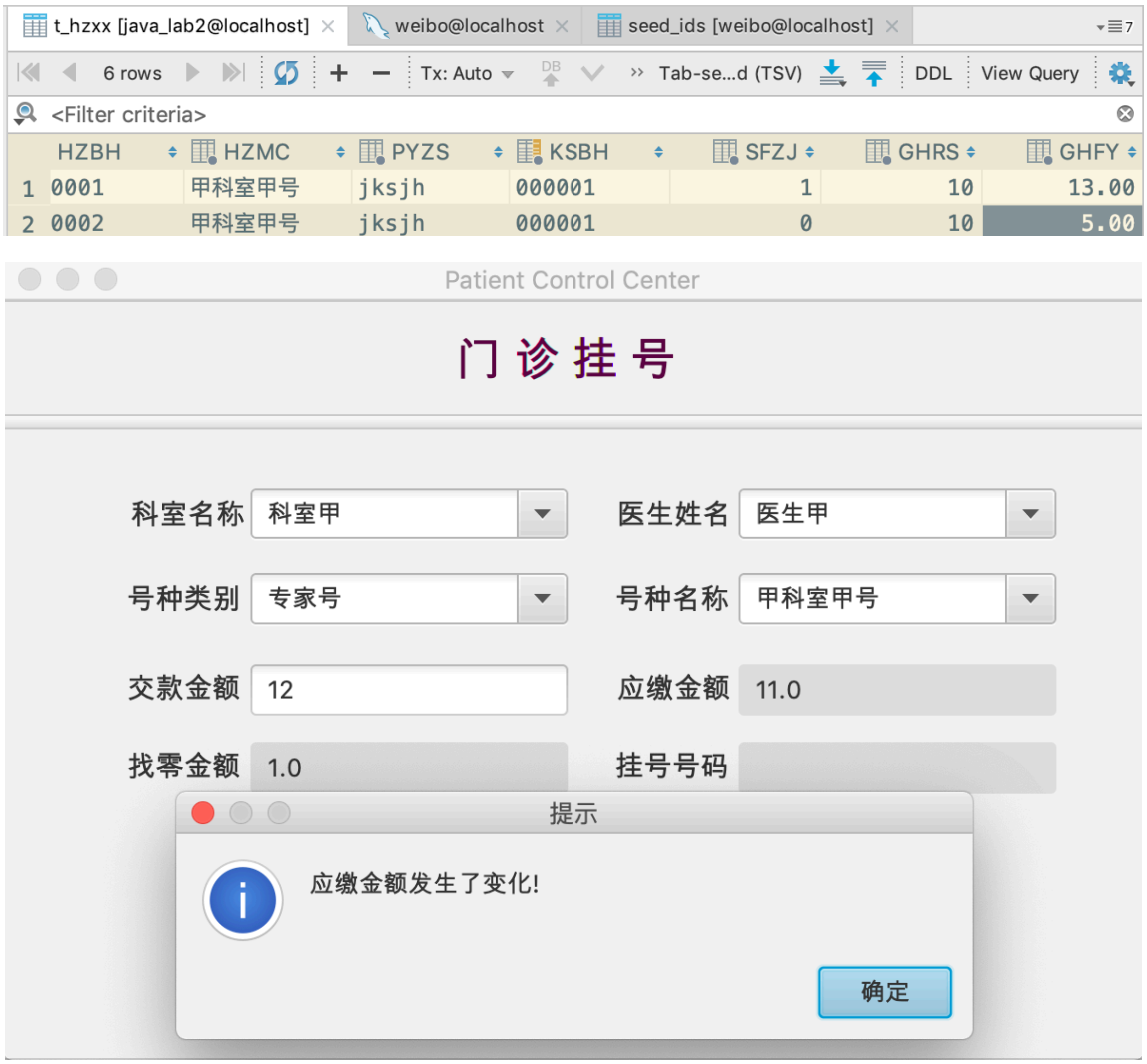

图 4.10 系统识别应缴金额发生变化

在系统中输入正确的号种信息，测试成功挂号结果如图 4.11 所示。

图 4.11 系统成功挂号

挂号数量达到上限时系统会给出相应报错，测试结果如图 4.12 所示。

图 4.12 系统识别挂号达到上限

经过对挂号操作中可能出现的各种边界情况以及正常使用流程进行测试，说明系统对于不规范的挂号请求能够正确拒绝并提示对应的错误信息。同时对于正常的输入流程，系统能

够通过自动补全等方式提供人性化的服务，验证了管理系统中挂号逻辑的正确性。

3) 医生操作界面测试

在医生操作界面中，首先对挂号信息的展示进行测试，依次测试按三种可选时间筛选挂号信息，分别查询当日、所有时间、指定时间的测试结果如图 4.13 所示。

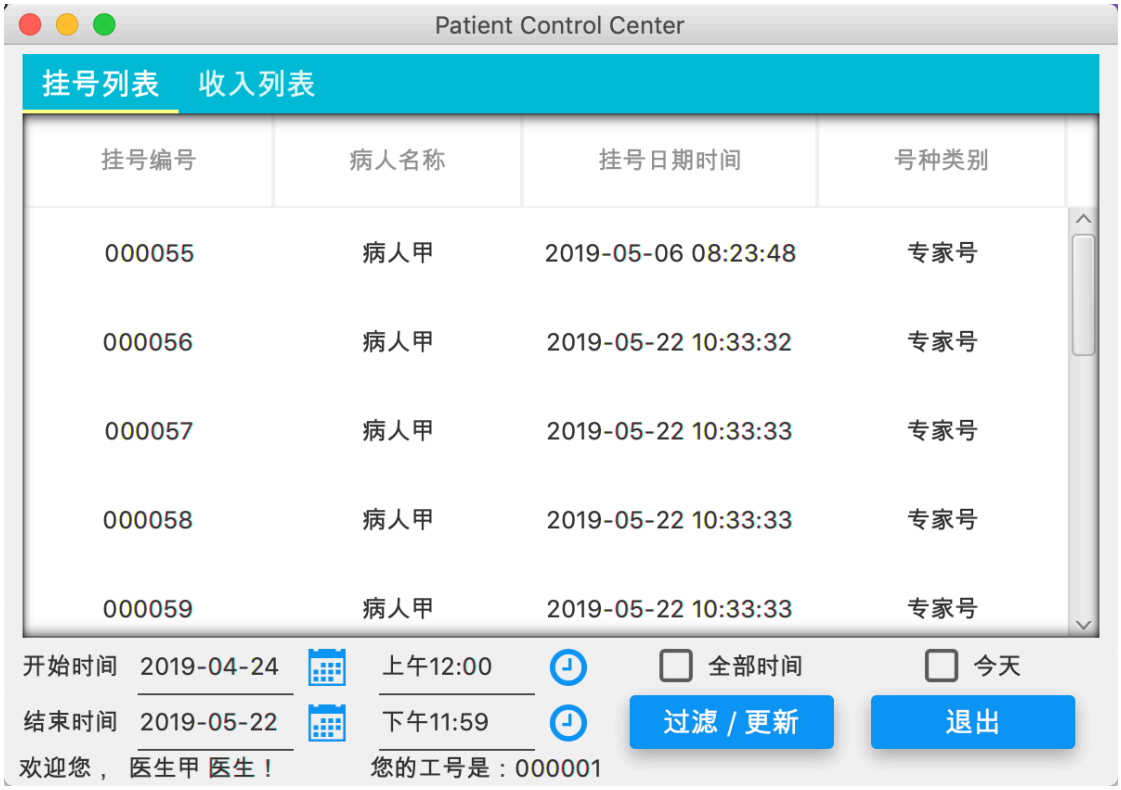

图 4.13 医生挂号信息展示

依次测试按三种时间筛选收入信息，分别查询当日、所有时间、指定时间的测试结果如图 4.14 所示。

图 4.14 医生收入信息展示

经过对医生操作界面中各种筛选条件下的挂号信息和收入信息进行查询，说明了系统能够通过不同空间对需要展示的数据进行正确的筛选并更新表格控件，验证了系统医生操作界面工作逻辑的正确性。

五、特点与不足

1. 技术特点

本系统的设计有如下的特点：

1. 病人操作界面中在输入挂号信息时支持自动补全，同时自动补全的内容依赖于先前的输入，这提高了系统的可用性、便捷性，使本系统更加的人性化；

2. 系统支持键盘和鼠标的混合输入，操作便捷十分人性化人性化；

3. 系统在各种输入环境下对边界情况的考虑比较完整，具有不错的健壮性；

4. 在挂号过程中，考虑了数据库变动的可能并作出了预防措施，提升了系统的健壮性；

5. 在医生界面中，使用多个控件来支持筛选日期的输入，这使程序的操作能够更加多样性、人性化；

6. 在系统的设计过程中，将所编写的程序分解为多个模块，使得整个系统的耦合度较低、易于修改且可扩展性强；

7. 使用 java 语言进行开发，全平台通用。

2. 不足和改进的建议

本系统仍有有以下的不足和改进空间：

1. 数据库中所有的密码为明文存储且长度受字段限制，安全性较低。在改进时可以考虑将加密后存储；

2. 在系统的设计过程中没有使用 C/S 的开发方式，所有客户端直接与数据库建立连接。同时在客户端需要查询或更新数据库时均使用了直接传送查询语句的方式，这都降低了系统的工作效率并可能产生包括 SQL 注入在内的安全性隐患。在改进过程中可为系统中维护的数据库连接 connector 添加各种定制方法以增强对系统的管理；

3. 本系统在挂号操作时使用的是本机时间，存在多个客户端的时间未校准的问题。在改进过程中可以令客户端统一从数据库服务器拉取时间信息从而保持数据一致；

4. 以系统中下拉菜单的更新为例，数据库查询操作十分频繁，影响系统效率。在改进的过程中，下拉预测框的更新算法可以改为增量更新，从而降低对数据库直接操作的频率并提升性能；

5. 在系统中还可以添加更多的统计功能供不同角色进行不同角度的查询操作。

6. 可在系统中新增注册用户功能。

六、过程和体会

1. 遇到的主要问题和解决方法

问题一：在编写病人操作界面时需要支持自动补全，用于触发自动补全操作的信号需要来自于键盘输入，但通过 setOnPressed 为按下按键信号添加事件监听器并不能实现对应功能。

解决方法：经过查询资料发现 ComboBox 控件接管了 setOnPressed 事件，接管后该代理

仅在敲击特殊按键（如 Ctrl、Enter 等）时生效，将绑定的信号更改为 setOnReleased 即可。

问题二：在注销后通过键盘再次登陆并重启病人操作界面时，系统对控件的聚焦位置与编写程序时在 runLater 中设置的聚焦位置不同。

解决方法：经过调试发现程序在登陆界面中若通过回车按钮激活登录按钮，在进入病人操作界面后该回车键按下的信号会被再次读入，从而更改被聚焦的控件。在两个界面中分别对不同的控件切换聚焦逻辑为“回车按下”和“回车释放”后解决了问题。

问题三：起初在登陆界面中只能通过鼠标进行控件的聚焦，操作不便。

解决方法：在查询文档后通过 requestFocus 方法能够自动切换系统聚焦的控件。

问题四：起初系统中的提示框并不能锁定用户的操作范围，从而能够不断产生新的错误提示框。

解决方法：通过 alarm 器件替换原先直接生成的新的 scence 的方式，从而在提示消息框出现后限制用户只能在关闭提示框后继续操作系统。

2. 课程设计的体会

通过此次程序设计，我最大的收获是学习了如何使用 JavaFX 搭建一个现代化的、友好的界面，以及如何让 Java 程序与数据库协作来完成复杂的功能。

此外，虽然本次程序设计中主要的困难在于对图形化界面 GUI 的编写，但在需要具体组织数据结构并完成业务逻辑的过程中还是需要用到不少 JAVA 程序设计中的基础知识与高级知识。总的来说在本次课程设计中我的 JAVA 基础知识得到了夯实，同时对于 JAVA 语言的一些高级特性进行了了解，如匿名函数和泛型等，还顺带复习了有关数据库的知识以及许多软件工程的经验与知识，这对于我今后的学习生活都能够奠定基础。

七、源码和说明

1. 文件清单及其功能说明

项目文件夹的结构及其说明如下：

1. create_tables.py：用于构造系统运行所需的数据库结构，数据库类型为 mysql；
2. src 文件夹：程序源码；
3. jfoenix-8.0.8.jar：jetbrain 公司的开源库，包含更多的 javafx 控件；
4. bin 文件夹：程序编译生成的字节码等。

其中 src 源码文件夹的内容及说明如下：

1. Main.java：挂号系统入口模块，包含数据库连接的初始化；
2. DBConnctor.java：系统与数据库的连接类，负责维护连接并提供查询接口；
3. Login.fxml：登陆界面；
4. LoginController.java：登陆界面的控制逻辑代码；
5. Patient.fxml：病人操作界面；
6. PatientController.java：病人操作界面的控制逻辑代码；

7. Doctor.fxml：医生操作界面；
8. DoctorController.java：医生操作界面的控制逻辑代码。

2. 用户使用说明书

使用 Eclipse 对项目进行导入并编译执行即可。

3. 源代码

整个系统的代码过于冗长，每个文件中的代码仅截取部分。

1. Main.java

```
package java_lab2;


import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.Parent;
import javafx.stage.Stage;


public class Main extends Application {
 public static DBConnector db_con = null;


 @Override
 public void start(Stage primaryStage) throws Exception{
        // Initialize connector
        try{
                if(db_con == null)
                        db_con = new DBConnector();
                db_con.connectDB("localhost", 3306, "java_lab2", "java", "java");
        }
        catch(ClassNotFoundException ce){
                System.err.print("Cannot load sql driver. Details:\n");
                ce.printStackTrace();
                   System.exit(1);
        }


            Parent root = FXMLLoader.load(getClass().getResource("Login.fxml"));
            primaryStage.setTitle("挂号管理系统");
        // Scene scene = new Scene(root, 300, 275);
```

```
            primaryStage.setMinWidth(400);
            primaryStage.setMinHeight(190);
        primaryStage.setScene(new Scene(root));
            primaryStage.show();
        // listen close button click event
        primaryStage.setOnCloseRequest(event->{
                System.out.println("closing from left-top button...\n");
                db_con.disconnectDB();
        });
 }

 public static void main(String[] args) {
        launch(args);
 }
}
```

2. LoginController.java

```
package java_lab2;

import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.control.TextField;
import javafx.scene.control.PasswordField;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.Label;

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

import java.sql.*;
```

```
public class LoginController {
    @FXML TextField account;
    @FXML PasswordField password;
    @FXML CheckBox doctor_set;
    @FXML CheckBox patient_set;
    @FXML Button login;

    /**
     * The constructor.
     * The constructor is called before the initialize() method.
     */
    public LoginController() {
    }

    /**
     * Initializes the controller class. This method is automatically called
     * after the fxml file has been loaded.
     */
    @FXML
    private void initialize() {
        patient_set.setSelected(true);
        doctor_set.setSelected(false);
        // set "enter" key for textfield
        account.setOnKeyPressed(keyEvent -> {
            if (keyEvent.getCode() == KeyCode.ENTER)
                password.requestFocus();
        });
        password.setOnKeyPressed(keyEvent -> {
            if (keyEvent.getCode() == KeyCode.ENTER)
                login.fire();
        });

        // initial the focus
        Platform.runLater(new Runnable(){
            @Override public void run(){
                account.requestFocus();
```

```
            }
        });
    }

    @FXML
    void click_doctor(){
        patient_set.setSelected(!patient_set.isSelected());
    }

    @FXML
    void click_patient(){
        doctor_set.setSelected(!doctor_set.isSelected());
    }

    @FXML
    void click_login(){
        // check account and password
        String account_input = account.getText();
        String password_input = password.getText();
        if(account_input.length() == 0 || password_input.length() == 0){// empty input
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("提示");
            alert.setHeaderText(null);
            alert.setContentText("账号或密码不得为空!");
            alert.showAndWait();
            return;
        }
        else {// check account and password
            System.out.print("account: "+account_input+"\n");
            System.out.print("password: "+password_input+"\n");
            try{
                String sql = "SELECT * FROM " + (patient_set.isSelected()?"T_BRXX ":"T_KSYS ") +
                                "WHERE " + (patient_set.isSelected()?"BRBH ":"YSBH ") + "= \'" + account_input + "\' "
+
                                "AND DLKL = \'" + password_input +"\'";
                System.out.println(sql+"\n");
```

```
            ResultSet rs = Main.db_con.runQueryCommand(sql);
            if(!rs.next()){// login failed
                Alert alert = new Alert(AlertType.INFORMATION);
                alert.setTitle("提示");
                alert.setHeaderText(null);
                alert.setContentText("账号或密码错误!");
                alert.showAndWait();
                return;
            }
            else{
                if(patient_set.isSelected()){
                    String name = rs.getString("BRMC");
                    // 输出数据
                    System.out.println("Login Success, BRMC: " + name +"\n");
                }
                else{
                    String name = rs.getString("YSMC");
                    // 输出数据
                    System.out.println("Login Success, YSMC: " + name +"\n");
                }
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    // login success, close login window
    // // my DEBUG
    // Main.db_con.disconnectDB();
    // get a handle to the stage
    Stage stage = (Stage) login.getScene().getWindow();
    // do what you have to do
    // stage.close();

    // check login method
    if(patient_set.isSelected()){// login as patient
```

```
                try{
                    Parent root = FXMLLoader.load(getClass().getResource("Patient.fxml"));
                    root.requestFocus();
                    stage.setTitle("Patient Control Center");
                    stage.setScene(new Scene(root));
                    stage.setUserData(account.getText());
                    stage.show();
                }catch(Exception e){
                    e.printStackTrace();
                }
            }
            else{// login as doctor
                try{
                    Parent root = FXMLLoader.load(getClass().getResource("Doctor.fxml"));
                    root.requestFocus();
                    stage.setTitle("Patient Control Center");
                    stage.setScene(new Scene(root));
                    stage.setUserData(account.getText());
                    stage.show();
                }catch(Exception e){
                    e.printStackTrace();
                }
            }
        }

}
```

3. PatientController.java

```
package java_lab2;

import javafx.application.Platform;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.scene.control.TextField;
import javafx.scene.control.PasswordField;
```

```
import javafx.scene.control.CheckBox;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ComboBox;

import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.Parent;
import javafx.stage.Stage;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.InputEvent;

import java.sql.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class PatientController {
 @FXML ComboBox<String> ksmc;
      @FXML ComboBox<String> ysxm;
      @FXML ComboBox<String> hzlb;
      @FXML ComboBox<String> hzmc;
      @FXML TextField jkje;
      @FXML TextField yjje;
      @FXML TextField zlje;
      @FXML TextField ghhm;
      @FXML Button submit;
      @FXML Button clear;
      @FXML Button quit;

      private String ksmc_temp = "";
      private String ysxm_temp = "";
      private String hzlb_temp = "";
```

```
private String hzmc_temp = "";

/**
 * The constructor.
 * The constructor is called before the initialize() method.
 */
public PatientController() {
}

@FXML
private void initialize(){
    yjje.setEditable(false);
    zlje.setEditable(false);
    ghhm.setEditable(false);

    ksmc.setOnMouseClicked(mouseEvent -> {
        set_ksmc();
    });
    ysxm.setOnMouseClicked(mouseEvent -> {
        set_ysxm();
    });
    hzlb.setOnMouseClicked(mouseEvent -> {
        set_hzlb();
    });
    hzmc.setOnMouseClicked(mouseEvent -> {
        set_hzmc();
    });

    ksmc.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> arg0, String oldvalue,String newvalue) {
            if(newvalue!=null){
                ksmc_temp = newvalue;
```

```
                    // System.out.println(newvalue);
            }
            ksmc.getEditor().setText(ksmc_temp);
            ksmc.getEditor().positionCaret(ksmc.getEditor().getText().length());
        }
    });
    ysxm.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> arg0, String oldvalue,String newvalue) {
            if(newvalue!=null){
                ysxm_temp = newvalue;
                    // System.out.println(newvalue);
            }
            ysxm.getEditor().setText(ysxm_temp);
            ysxm.getEditor().positionCaret(ysxm.getEditor().getText().length());
        }
    });
    hzlb.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> arg0, String oldvalue,String newvalue) {
            if(newvalue!=null){
                hzlb_temp = newvalue;
                    // System.out.println(newvalue);
            }
            hzlb.getEditor().setText(hzlb_temp);
            hzlb.getEditor().positionCaret(hzlb.getEditor().getText().length());
            set_yjje();
           set_zlje();
        }
    });
    hzmc.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> arg0, String oldvalue,String newvalue) {
            if(newvalue!=null){
                hzmc_temp = newvalue;
                    // System.out.println(newvalue);
```

```
                    }
                    hzmc.getEditor().setText(hzmc_temp);
                    hzmc.getEditor().positionCaret(hzmc.getEditor().getText().length());
                    set_yjje();
                   set_zlje();
                }
            });



            ksmc.setOnKeyReleased(keyEvent -> {
                if (keyEvent.getCode().isDigitKey() || keyEvent.getCode().isLetterKey() ||
keyEvent.getCode()==KeyCode.BACK_SPACE) {
                    ksmc_temp = ksmc.getEditor().getText();
                    ksmc.show();
                    set_ksmc();
                }
            });
        ksmc.setOnKeyPressed(keyEvent -> {
            if (keyEvent.getCode() == KeyCode.ENTER)
                    ysxm.requestFocus();
        });
            ysxm.setOnKeyReleased(keyEvent -> {
                if (keyEvent.getCode().isDigitKey() || keyEvent.getCode().isLetterKey() ||
keyEvent.getCode()==KeyCode.BACK_SPACE) {
                    ysxm_temp = ysxm.getEditor().getText();
                    ysxm.show();
                    set_ysxm();
                }
            });
        ysxm.setOnKeyPressed(keyEvent -> {
            if (keyEvent.getCode() == KeyCode.ENTER)
                    hzlb.requestFocus();
        });
            hzlb.setOnKeyReleased(keyEvent -> {
                if (keyEvent.getCode().isDigitKey() || keyEvent.getCode().isLetterKey() ||
```

```
keyEvent.getCode()==KeyCode.BACK_SPACE) {
                        hzlb_temp = hzlb.getEditor().getText();
                        hzlb.show();
                        set_hzlb();
                       // set_yjje();
                    }
                });
            hzlb.setOnKeyPressed(keyEvent -> {
                if (keyEvent.getCode() == KeyCode.ENTER)
                        hzmc.requestFocus();
            });
                hzmc.setOnKeyReleased(keyEvent -> {
                    if (keyEvent.getCode().isDigitKey() || keyEvent.getCode().isLetterKey() ||
keyEvent.getCode()==KeyCode.BACK_SPACE) {
                        hzmc_temp = hzmc.getEditor().getText();
                        hzmc.show();
                        set_hzmc();
                       // set_yjje();
                    }
                });
            hzmc.setOnKeyPressed(keyEvent -> {
                if (keyEvent.getCode() == KeyCode.ENTER)
                        jkje.requestFocus();
            });
                jkje.setOnKeyReleased(keyEvent -> {
                    if (keyEvent.getCode().isDigitKey() || keyEvent.getCode().isLetterKey() ||
keyEvent.getCode()==KeyCode.BACK_SPACE) {
                        set_zlje();
                    }
                });
            jkje.setOnKeyPressed(keyEvent -> {
                if (keyEvent.getCode() == KeyCode.ENTER)
                        submit.fire();
            });


                Platform.runLater(new Runnable(){
```

```
            @Override public void run(){
                set_ksmc();
                System.out.println("run later");
                ksmc.requestFocus();
            }
        });
}

@FXML
private void set_ksmc() {
  try{
            String sql = "SELECT * FROM T_KSXX " +
                        "WHERE KSMC LIKE \'%" + ksmc.getEditor().getText() + "%\' " +
                        "OR PYZS LIKE \'%" + ksmc.getEditor().getText() + "%\'";
            System.out.println(sql);
            ResultSet rs = Main.db_con.runQueryCommand(sql);
            if(!rs.next()){// search failed
                // Do nothing but clear the items
                ksmc.getItems().clear();
            }
            else{
                // clear first
                ksmc.getItems().clear();
                // read the first record
                String item_ks_name = rs.getString("KSMC");
                System.out.print(item_ks_name + "\n");
                ksmc.getItems().addAll(item_ks_name);
                // read the rest records
                while(rs.next()){
                    item_ks_name = rs.getString("KSMC");
                    System.out.print(item_ks_name + "\n");
                    ksmc.getItems().addAll(item_ks_name);
                }
            }
        }catch(Exception e){
            e.printStackTrace();
```

```
        }
    }
```

4. DoctorController.java

```
package java_lab2;

import com.jfoenix.controls.*;
import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
import com.sun.javafx.robot.impl.FXRobotHelper;
import javafx.application.Platform;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.util.StringConverter;
import javafx.event.*;

import javafx.scene.control.*;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;
import java.sql.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;



public class DoctorController {
    // Register Item Define
```

```
        private static final class Register extends RecursiveTreeObject<Register> {
            public StringProperty number;
            public StringProperty namePatient;
            public StringProperty dateTimeDisplay;
            public StringProperty isSpecialistDisplay;
            public Register(String number, String namePatient, Timestamp dateTime, boolean isSpec){
                this.number = new SimpleStringProperty(number);
                this.namePatient = new SimpleStringProperty(namePatient);
                this.dateTimeDisplay = new
SimpleStringProperty(dateTime.toLocalDateTime().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));
                this.isSpecialistDisplay = new SimpleStringProperty(isSpec ? "专家号" : "普通号");
            }
        }

        // Income Item Define
        private static final class Income extends RecursiveTreeObject<Income> {
            public StringProperty departmentName;
            public StringProperty doctorNumber;
            public StringProperty doctorName;
            public StringProperty registerType;
            public StringProperty registerPopulation;
            public StringProperty incomeSum;
            public Income(String depName, String docNum, String docName, boolean isSpec, int regNumPeople, Double
incomSum){
                this.departmentName = new SimpleStringProperty(depName);
                this.doctorNumber = new SimpleStringProperty(docNum);
                this.doctorName = new SimpleStringProperty(docName);
                this.registerType = new SimpleStringProperty(isSpec ? "专家" : "普通");
                this.registerPopulation = new SimpleStringProperty(Integer.toString(regNumPeople));
                this.incomeSum = new SimpleStringProperty(String.format("%.2f", incomSum));
            }
        }

        String doctorName = "";
        String doctorNum = "";
```

```
@FXML private Label labelWelcome;
@FXML private Label labelName;
@FXML private Label labelAccount;
@FXML private JFXDatePicker pickerDateStart;
@FXML private JFXDatePicker pickerDateEnd;
@FXML private JFXTimePicker pickerTimeStart;
@FXML private JFXTimePicker pickerTimeEnd;

@FXML private JFXTabPane mainPane;
@FXML private Tab tabRegister;
@FXML private Tab tabIncome;

@FXML private JFXTreeTableView<Register> tableRegister;
@FXML private TreeTableColumn<Register, String> columnRegisterNumber;
@FXML private TreeTableColumn<Register, String> columnRegisterPatientName;
@FXML private TreeTableColumn<Register, String> columnRegisterDateTime;
@FXML private TreeTableColumn<Register, String> columnRegisterType;
private TreeItem<Register> rootRegister;

@FXML private JFXTreeTableView<Income> tableIncome;
@FXML private TreeTableColumn<Income, String> columnIncomeDepartmentName;
@FXML private TreeTableColumn<Income, String> columnIncomeDoctorNumber;
@FXML private TreeTableColumn<Income, String> columnIncomeDoctorName;
@FXML private TreeTableColumn<Income, String> columnIncomeRegisterType;
@FXML private TreeTableColumn<Income, String> columnIncomeRegisterPopulation;
@FXML private TreeTableColumn<Income, String> columnIncomeSum;
private TreeItem<Income> rootIncome;

private ObservableList<Register> listRegister = FXCollections.observableArrayList();
private ObservableList<Income> listIncome = FXCollections.observableArrayList();

@FXML JFXCheckBox checkBoxAllTime;
@FXML JFXCheckBox checkBoxToday;
@FXML JFXButton buttonFilter;
@FXML JFXButton buttonExit;
```

```
    @FXML
    void initialize(){
        // set two date converter (formatter)
        pickerDateStart.setConverter(new DateConverter());
        pickerDateEnd.setConverter(new DateConverter());
        // default to current date
        pickerDateStart.setValue(LocalDate.now());
        pickerDateEnd.setValue(LocalDate.now());

        // set time selector to 24h
        pickerTimeStart.set24HourView(true);
        pickerTimeEnd.set24HourView(true);
        // default to 00:00 to 23:59
        pickerTimeStart.setValue(LocalTime.MIN);
        pickerTimeEnd.setValue(LocalTime.MAX);

        // initiailze register list
        columnRegisterNumber.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Register, String> param) -> param.getValue().getValue().number);
        columnRegisterPatientName.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Register, String> param) ->
param.getValue().getValue().namePatient );
        columnRegisterDateTime.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Register, String> param) ->
param.getValue().getValue().dateTimeDisplay );
        columnRegisterType.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Register, String> param) ->
param.getValue().getValue().isSpecialistDisplay );

        rootRegister = new RecursiveTreeItem<>(listRegister, RecursiveTreeObject::getChildren);
        tableRegister.setRoot(rootRegister);
        tableRegister.setCenterShape(true);


        // template
        // listRegister.add(new Register("a", "b", new Timestamp(1000),true));
```

```
        columnIncomeDepartmentName.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Income, String> param) ->
param.getValue().getValue().departmentName);
        columnIncomeDoctorNumber.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Income, String> param) ->
param.getValue().getValue().doctorNumber);
        columnIncomeDoctorName.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Income, String> param) -> param.getValue().getValue().doctorName);
        columnIncomeRegisterType.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Income, String> param) -> param.getValue().getValue().registerType);
        columnIncomeRegisterPopulation.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Income, String> param) ->
param.getValue().getValue().registerPopulation);
        columnIncomeSum.setCellValueFactory(
                (TreeTableColumn.CellDataFeatures<Income, String> param) -> param.getValue().getValue().incomeSum);

        rootIncome = new RecursiveTreeItem<>(listIncome, RecursiveTreeObject::getChildren);
        tableIncome.setRoot(rootIncome);

        Platform.runLater(new Runnable(){
            @Override public void run(){
                String sql = "";
                ResultSet rs = null;

                // initialize the welcome inform
                Stage stage = (Stage) labelWelcome.getScene().getWindow();
                doctorNum = (String) stage.getUserData();
                try{
                    sql = "SELECT * FROM T_KSYS " +
                                "WHERE T_KSYS.YSBH = \'" + doctorNum + "\'";
                    rs = Main.db_con.runQueryCommand(sql);
                    if(!rs.next()){}
                    else{
                        doctorName = rs.getString("YSMC");
                        System.out.println("Login success, YSMC: " + doctorName);
                    }
```

```
                }catch(Exception e){
                    e.printStackTrace();
                }
                labelWelcome.setText(new String("欢迎您，"));
                labelName.setText(String.format("%s 医生！", doctorName));
                labelAccount.setText(String.format("您的工号是：%s", doctorNum));

                // initialize the page
                // buttonFilter.fire();
            }
        });
    }
```