

**课 程 实 验 报 告**

**课程名称：Java语言程序设计**

**实验名称：泛型栈模拟泛型队列和**

**医院简易挂号管理系统**

**院 系 ：计算机科学与技术**

**专业班级 ： IOT1501**

**学 号 ： U201514881**

**姓 名 ： 赵浩东**

**指导教师 ： 吕新桥**

**2018 年 5 月 19 日**

# 泛型栈模拟泛型队列

1. **需求分析**
2. **题目要求**

**参见<https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html>，Java提供的java.util.Queue是一个接口没有构造函数。试用java.util.Stack<E>泛型栈作为父类，用另一个泛型栈对象作为成员变量，模拟实现一个泛型子类Queue<E>，当存储元素的第1个栈的元素超过dump时，再有元素入队列就倒入第2栈。除提供无参构造函数Queue( )外，其它所有队列函数严格参照java.util.Queue的接口定义实现。**

**import java.util.Stack;**

**import java.util.NoSuchElementException;**

**public class Queue<E> extends Stack<E>{**

**public final int dump=10;**

**private Stack<E> stk;**

**public Queue( ){ /\* 在此插入代码\*/ }**

**public boolean add(E e) throws IllegalStateException, ClassCastException,**

**NullPointerException, IllegalArgumentException{ /\* 在此插入代码\*/ }**

**public boolean offer(E e) throws ClassCastException, NullPointerException,**

**IllegalArgumentException{ /\* 在此插入代码\*/ }**

**public E remove( ) throws NoSuchElementException { /\* 在此插入代码\*/ }**

**public E poll( ) { /\* 在此插入代码\*/ }**

**public E peek ( ) { /\* 在此插入代码\*/ }**

**public E element( ) throws NoSuchElementException { /\* 在此插入代码\*/ }**

**}**

**考虑到Stack<E>只能存储类型为E的元素，以及Stack是一个存储能力(capacity, 参见有关说明)理论上无限的类型，这可能会影响到相关方法的异常处理，请适当处理上述异常（也许某些异常从来都不会发生）。**

**思考：Queue<E>是否应该提供clone和equals函数，以及其它一些函数如addAll等？**

1. **需求分析**

**remove方法和poll方法内部实现方式相同，但在异常处理时remove方法会抛出异常而poll方法则是直接返回，element和peek方法同理。**

**由于Stack<E>只能存储类型为E的元素，以及Stack是一个存储能力理论上无限的类型，故ClassCastException和IllegalArgumentException从来不会发生，当我们模拟的队列满（由于模拟的方法，满时队内元素的个数不确定）时抛出IllegalStateException，当队为空抛出NoSuchElementException，当入队元素为空抛出NullPointerException。**

1. **系统设计**
2. **概要设计**

利用栈模拟队列，由于栈为先进后出结构，队列为先进先出结构，故使用两个栈一个执行入队操作，记为栈1，一个执行出队操作，记为栈2。要实现的方法已经给出，对这些方法进行填充即可。

1. **详细设计**

**add/offer方法设计：**

add/offer为入队操作，如果栈1满了则将入队栈元素依次pop再push到栈2，之后再把新插入的元素push到栈1中。add和offer的区别在于当队列为满时add会抛出异常而offer返回false。

**remove/poll方法设计：**

remove/poll是出队，即取队首元素并删除，若栈2不为空则直接调用栈2的pop方法，若栈2为空则先将栈1元素依次pop再push到栈2中，再调用栈2的pop方法。remove和poll的区别和add和offer的区别在于当队列为空时remove会抛出异常而poll返回null。

**element/peek方法设计：**

element/peek是取队首元素但不删除，和remove/poll方法一样，不过是把栈2的pop方法换成peek方法。element和peek的区别也和remove和poll的区别相同，当队列为空时element会抛出异常而peek返回null。

**printQueue方法设计：**

为了便于测试加入了这个方法来打印队列，若队列为空抛出NoSuchElementException异常，分两种情况，如果栈2为空则用一个for循环倒序打印入队栈中的元素，否则用两个for循环按相反的顺序打印两个栈中的元素。

**main方法设计：**

使用Scanner类从控制台获取输入当做菜单选项来选择调用的方法，以此测试实现的Queue类中方法的正确性。

1. **软件开发**

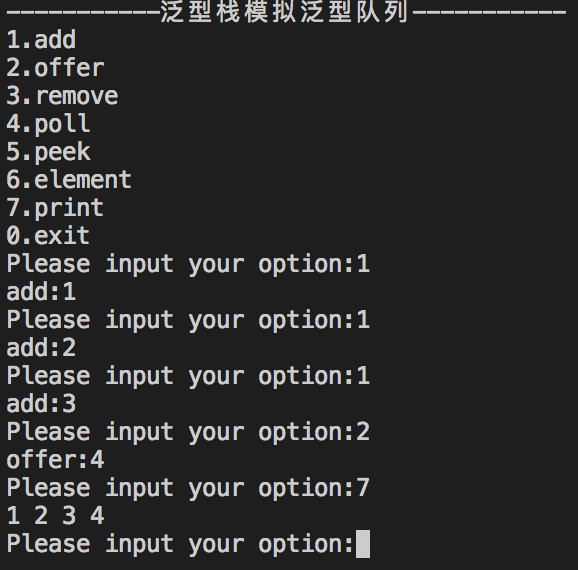
平台为macOS，使用命令行编译及运行java源代码，具体方式见文件夹中的README.txt。

1. **软件测试**

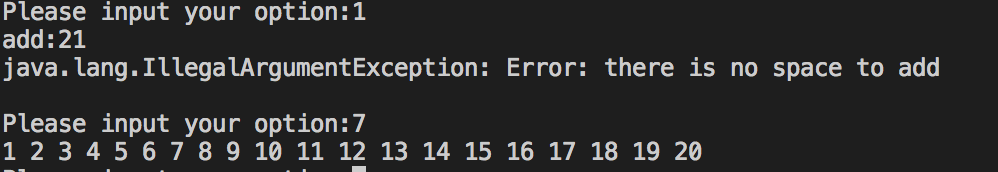
在命令行中输入：

java Queue

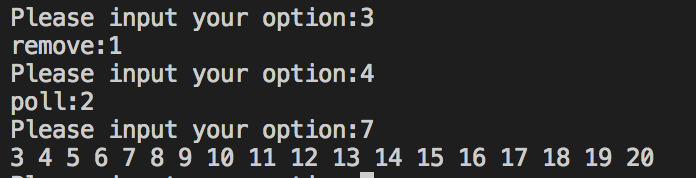
以运行程序，初始界面如下图所示，会显示一个菜单并提示输入选项来测试，测试入队操作，如下图所示：



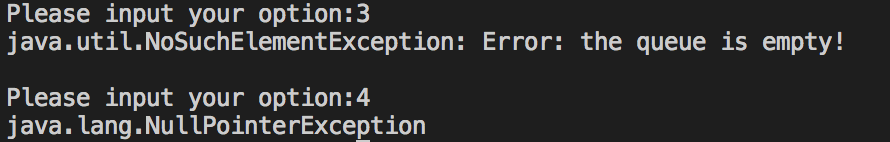
若队列已满，再向队内插入元素会抛出IllegalArgumentException，如下图所示：



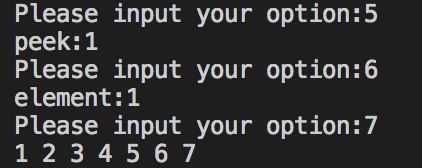
此时队列是直接插入20个元素中间没有出队的，所以队列上限就是20个元素，接下来测试出队方法，出队两个元素，如下图所示：



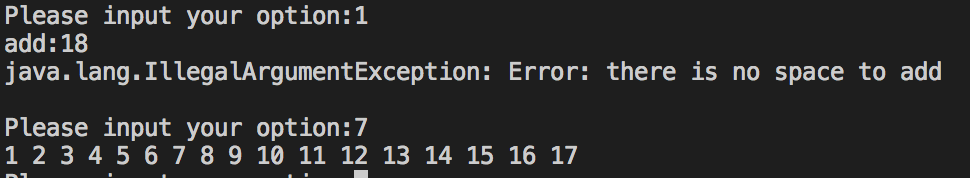
全部出队直至队列为空后再次出队会发现此时remove方法抛出NoSuchElementException异常，而poll方法返回null值使得主方法中catch到一个NullPointerException异常。



测试peek和element方法，队列元素为1-7，如下图：



若插入过程中有出队或查询队头操作，则队列上限会发生变化，如下图所示，17个元素时队列就已满。



1. **特点与不足**
2. **技术特点**

在实验要求实现的Queue类的几个方法的基础上增加了主方法和printQueue方法，便于测试。

1. **不足和改进的建议**

无

1. **过程和体会**
2. **遇到的主要问题和解决方法**

一开始在实现栈1出栈的操作时调用父类Stack的pop方法，但是却在运行时出错，经过对Stack类pop源码的查询发现pop中有this.peek()这个方法的调用，而子类Queue也有peek这个方法，相当于对父类的peek进行重写，这时调用父类的pop方法会执行子类的peek，导致错误，解决方法就是使用peek和removeElementAt来间接实现pop。

1. **课程设计的体会**

本次实验使我对Java的面向对象的思想有了一定的了解，以及对栈和队列的数据结构有了更深的认识，掌握了通过两个栈来实现队列的方法，本次试验难度较小，能够比较轻松地完成。Java给我的感受是比C++更加简单和方便易用，编写程序的难度也相较于C++低一些。

1. **源码和说明**
2. **文件清单及其功能说明**

Queue.java 实验源代码

Queue.class 字节码文件

1. **用户使用说明书**

见实验提交文件夹中的README.txt。

1. **源代码**

import java.util.Stack;

import java.util.NoSuchElementException;

import java.util.Scanner;

public class Queue<E> extends Stack<E>{

public final int dump=10;

private Stack<E> stk;

public Queue( ){

this.stk = new Stack<E>();

}

public boolean add(E e) throws IllegalStateException, ClassCastException,

NullPointerException, IllegalArgumentException{

if (e == null) {

throw new NullPointerException("Error: the element is null!\n");

}

if (super.elementCount >= 10 && !(stk.empty())) {

throw new IllegalArgumentException("Error: there is no space to add\n");

} else {

if (super.elementCount < 10) {

super.push(e);

} else {

while(!(super.empty()))

            {

                E t = super.peek();

                super.removeElementAt(super.size()-1);

                stk.push(t);

            }

            super.push(e);

}

}

return true;

}

public boolean offer(E e) throws ClassCastException, NullPointerException,

IllegalArgumentException{

if (e == null) {

throw new NullPointerException("Error: the element is null!\n");

}

if (super.elementCount >= 10 && !(stk.empty())) {

return false;

} else {

if (super.elementCount < 10) {

super.push(e);

} else {

while(!(super.empty()))

            {

                E t = super.peek();

                super.removeElementAt(super.size()-1);

                stk.push(t);

            }

            super.push(e);

}

}

return true;

}

public E remove( ) throws NoSuchElementException {

if (super.empty() && stk.empty()) {

throw new NoSuchElementException("Error: the queue is empty!\n");

}

if (stk.empty()) {

while(!(super.empty()))

            {

                E t = super.peek();

                super.removeElementAt(super.size()-1);

                stk.push(t);

            }

}

return stk.pop();

}

public E poll( ) {

if (super.empty() && stk.empty()) {

return null;

}

if (stk.empty()) {

while(!(super.empty()))

            {

                E t = super.peek();

                super.removeElementAt(super.size()-1);

                stk.push(t);

            }

}

return stk.pop();

}

public E peek ( ) {

if (super.empty() && stk.empty()) {

return null;

}

if (stk.empty()) {

while(!(super.empty()))

            {

                E t = super.peek();

                super.removeElementAt(super.size()-1);

                stk.push(t);

            }

}

return stk.peek();

}

public E element( ) throws NoSuchElementException {

if (super.empty() && stk.empty()) {

throw new NoSuchElementException("Error: the queue is empty!\n");

}

if (stk.empty()) {

while(!(super.empty()))

            {

                E t = super.peek();

                super.removeElementAt(super.size()-1);

                stk.push(t);

            }

}

return stk.peek();

}

    /\* 测试用函数，打印队列 \*/

    public void printQueue() throws NoSuchElementException

    {

        if(super.empty() && stk.empty())

        {

            throw new NoSuchElementException("The Queue is empty!");

        }

        int i;

        if(stk.empty())

        {

            for(i=0;i<super.size();i++)

                System.out.print(super.elementAt(i)+" ");

        }

        else

        {

            for(i=stk.size();i>0;i--)

                System.out.print(stk.elementAt(i-1)+" ");

            for(i=0;i<super.size();i++)

                System.out.print(super.elementAt(i)+" ");

        }

        System.out.print("\n");

    }

    /\* 主方法 \*/

    public static void main(String[] args)

    {

        Queue<Integer> q = new Queue<Integer>();

        int op;

        int e;

        Scanner in = new Scanner(System.in);

        System.out.println("-----------泛型栈模拟泛型队列-----------");

        System.out.println("1.add");

        System.out.println("2.offer");

        System.out.println("3.remove");

        System.out.println("4.poll");

        System.out.println("5.peek");

        System.out.println("6.element");

        System.out.println("7.print");

        System.out.println("0.exit");

        System.out.print("Please input your option:");

        op = in.nextInt();

        while (op != 0)

        {

            try

            {

                switch(op)

                {

                    case 1:

                        System.out.print("add:");

                        e = in.nextInt();

                        q.add(e);

                        break;

                    case 2:

                        System.out.print("offer:");

                        e = in.nextInt();

                        q.offer(e);

                        break;

                    case 3:

                        e = q.remove();

                        System.out.println("remove:" + e);

                        break;

                    case 4:

                        e = q.poll();

                        System.out.println("poll:" + e);

                        break;

                    case 5:

                        e = q.peek();

                        System.out.println("peek:" + e);

                        break;

                    case 6:

                        e = q.element();

                        System.out.println("element:" + e);

                        break;

                    case 7:

                        q.printQueue();

                        break;

                    default:

                        System.out.println("Invalid Input!");

                }

            }

            catch(IllegalStateException ise)

            {

                System.out.println(ise.toString());

            }

            catch(ClassCastException cce)

            {

                System.out.println(cce.toString());

            }

            catch(NullPointerException npe)

            {

                System.out.println(npe.toString());

            }

            catch(IllegalArgumentException iae)

            {

                System.out.println(iae.toString());

            }

            catch(NoSuchElementException nsee)

            {

                System.out.println(nsee.toString());

            }

            finally

            {

                System.out.print("Please input your option:");

                op = in.nextInt();

            }

        }

    }

}

# 医院简易挂号管理系统

1. **需求分析**
2. **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** |
| **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** |
| **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** | **该病人该号种的挂号人次** |
| **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？**

1. **需求分析**

医生与病人分别通过各自的编号和口令登录进入医生或病人界面，医生能够进行上述的操作，病人同理。通过对数据库的访问来获取需要的信息。

1. **系统设计**
2. **概要设计**

分别为医生和病人开发两套操作界面，通过登录界面的选项来进行选择，登录后病人和医生进行不同的操作。一共三个模块：登录界面，医生界面，病人界面。

1. **详细设计**
2. **登录界面设计：**

判断用户编号和登录口令的输入框是否为空，若为空弹出提示信息；根据用户选择的医生登录或是病人登录来到数据库中查找对应的表中的编号和登录口令若一致则准许登录，并加载对应的医生界面或病人界面，否则提示用户名或密码错误。

1. **医生界面设计：**

医生界面登录后有一个菜单栏，可选择三项“病人列表”“收入列表”“退出系统”，每个菜单项的点击有一个对应的方法，医生界面使用DoctorView类控制，点击病人列表，会有一个TableView显示挂号的病人列表，还可以选择时间，若不选择默认显示所有病人。收入列表同理，病人和收入的信息使用javafx的Property机制来显示到TableView中，可以动态更新，分别对应PatientInfo/IncomeInfo类。退出系统则是使用当前窗口的hide方法，将当前窗口关闭，这样可以继续进行下一次登录，也防止了使用系统exit(0)方法的意外退出导致一些静态变量的清空和数据库连接未关闭等情况。

1. **病人界面设计：**

病人界面登录后会显示出交款金额（即病人的余额），之后通过选择框选择病人要挂的编号，通过选择框的addListener方法增加监听事件，当选择完科室后才能选择医生，然后依次选择号种名称和号种类别，选择完成后会给出应缴金额并自动计算找零金额，并给出挂号的号码，此时若不按确认键默认相当于退号，按了确认之后会将数据库中对应的号种信息表中退号标志置为false，并同时更新病人信息表中的余额为找零金额。

1. **软件开发**

操作系统：macOS 10.13.4。

开发工具：VS Code编辑器+命令行编译方式开发。

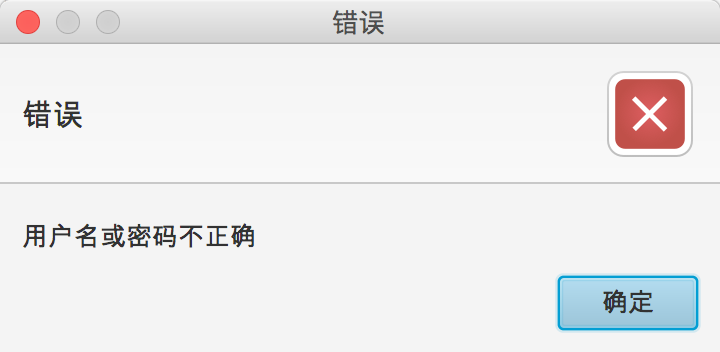
数据库：MySQL 8.0.11，用户名为root。

1. **软件测试**

运行程序，首先进入登陆界面，用户可以输入自己的编号和口令并选择身份（医生/病人）来登录系统，如果输入为空或者编号或密码错误则会弹出提示。

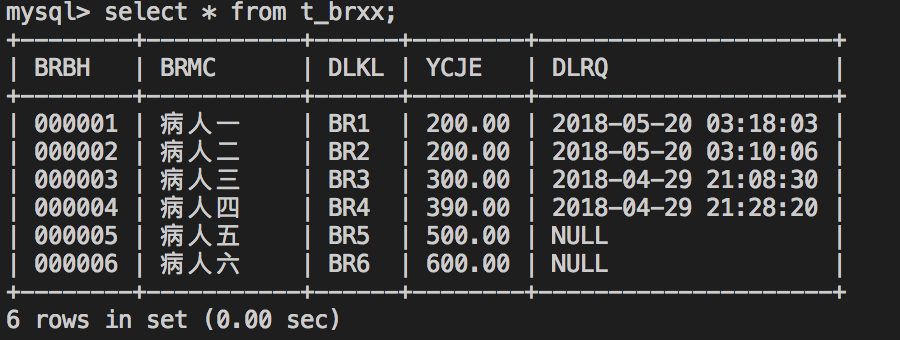






登录成功后，比如以病人一登录，控制台会输出登录信息，进入数据库后也可以发现登录时间已更新。





登录后的界面如下，可以看到交款金额显示为200.00，此时可以通过选择框来选择挂号的科室医生号种等等，以挂感染科号种二为例。

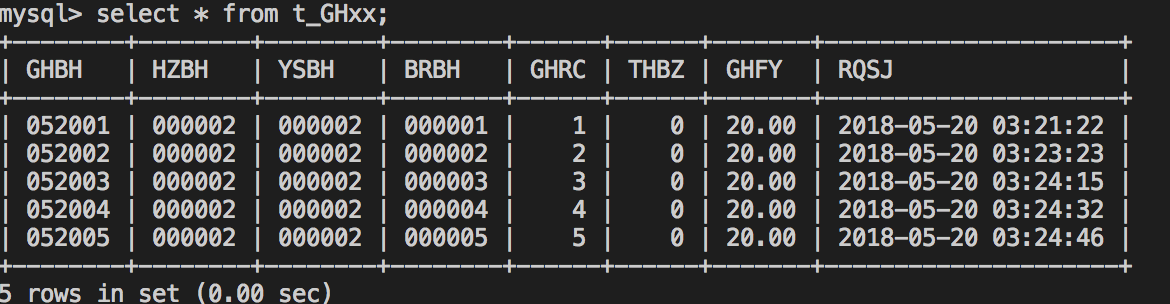


选择完成后应缴金额、找零金额和挂号号码会自动生成，挂号号码根据日期和当天的挂号人次自动生成，点击确定后弹出提示信息：挂号成功。





此时数据库t\_ghxx表中产生了一条新的记录。登录多个病人再重复上述过程，感染科上限人数为5人，若当日挂号人次已满，则会提示：挂号人次已达上限。



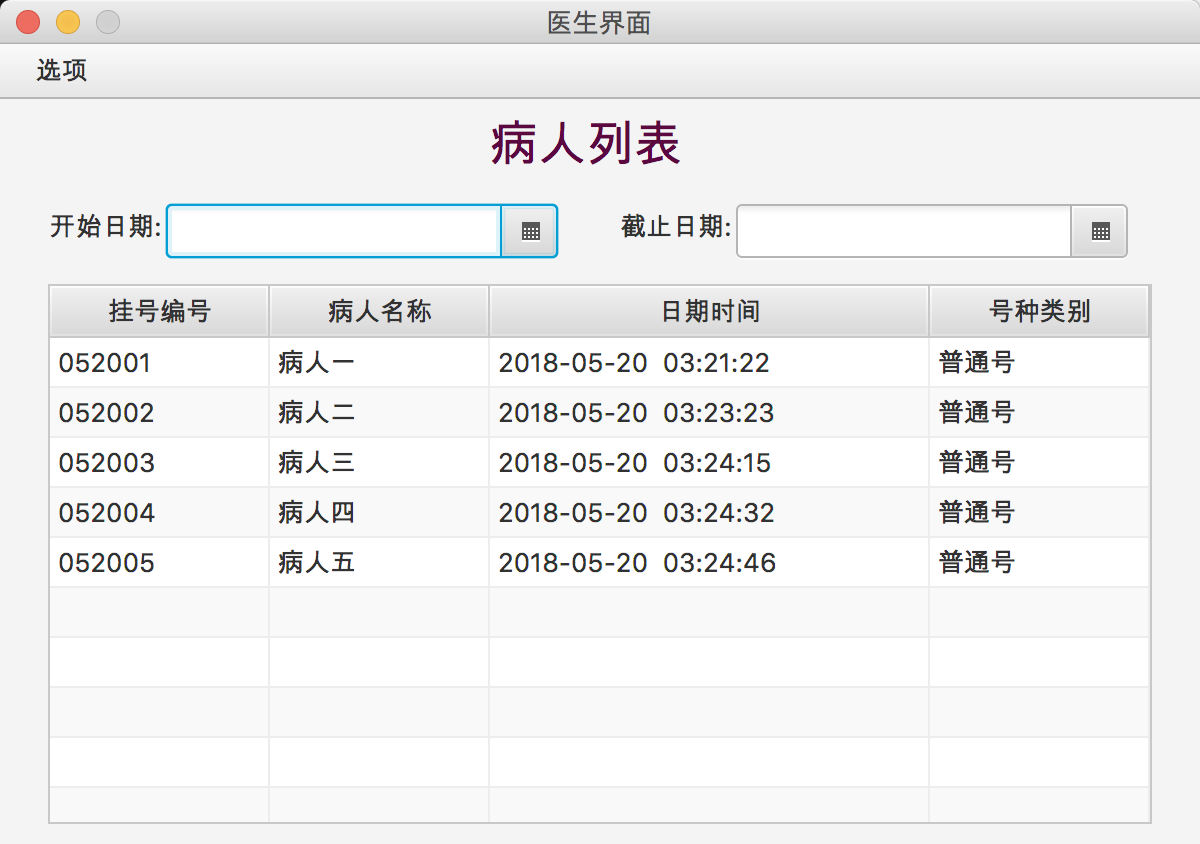


接下来测试医生界面，医生界面登录后有一个菜单栏供选择，以医生二为例。登录后选择菜单栏中的病人列表和收入列表可以显示对应的信息，退出系统和病人界面的退出一样。





如果没有选择开始日期和截止日期，则默认显示全部的病人信息或收入信息。





如果选择日期，还可以筛选日期范围内的挂号信息，比如只看5月20日的信息。

1. **特点与不足**
2. **技术特点**

未使用IDE开发，绘制界面的fxml文件也是纯手打，通过命令行编译以及运行。

1. **不足和改进的建议**

程序尚存一些逻辑上的bug，不过不影响功能，并且使用正确的操作流程在一般情况下不易发现。另外在登录时不能防止sql注入攻击，比如输入登录口令 “ ’ or ‘1=1’ #”，这会导致编号为000001的医生和病人可能在错误的登录口令输入情况下正确登录并操作。

还有就是因为挂号人次使用一个静态变量来计数，如果整个程序退出就会导致挂号编号重新从1开始，所以在使用时必须保证程序一直运行，实际应用中医院挂号系统应该也是一直运行的，不过要考虑到停电或自然灾害等突发情况，改进方式可以在初始化时读数据库挂号信息表中的最后一条记录，将挂号人次置为对应的编号，但是本程序中没有实现这个功能。

病人信息界面当挂号的信息改变是会抛出一个IndexOutofBundleException，是因为其他本来有选择的Index的选择框的内容需要清空，而ChoiceBox类经过查询后我没有找到清空Index的方法，就没有管这个，直接抛给主方法处理异常，所以如果一个病人不小心选错了号重新选择的时候控制台会有异常信息打印出来，不过不影响程序功能。

1. **过程和体会**
2. **遇到的主要问题和解决方法**

本次实验虽然是GUI应用开发，但是由于系统的问题，没有使用IDE来做开发，遇到了许许多多的困难，比如javafx加载的fxml文件因为在Eclipse等IDE中可以直接导入，而在命令行运行时它们的路径应该是在java命令下加-cp参数来寻找，或者用相对路径，把fxml文件和class文件放在一起而不是和源码放在一起。

1. **课程设计的体会**

本次实验相比实验一要难很多，需要用Java实现一个基于数据库的GUI应用程序，虽然使用Java编程相对于以前学过的C++来说更为简单，但是开发过程中还是遇到了诸多问题。比如Java中哪些方法需要import却不知道。需要不断地试错才能将bug清除，还有Java没有类似Qt这种可视化的图形界面设计工具，所以图形界面的设计相对来说比较繁琐。不过不得不承认Java面相对象的编程方式以及高度封装的API给开发简单的软件带来了极大的便利。而且java比C++更容易学习和掌握。面向对象的思想对我有极大的启发。

通过本次实验我对Java的熟练度有了一个很大的提升，对软件工程的认知又更深了一步，最重要的是感受到了Java的魅力，可以说Java成为了我目前最喜欢的编程语言。感谢老师一个学期的教导，希望Java课程越办越好。

1. **源码和说明**

代码：见实验2文件夹中src文件夹

说明：见实验2文件夹的README.txt。