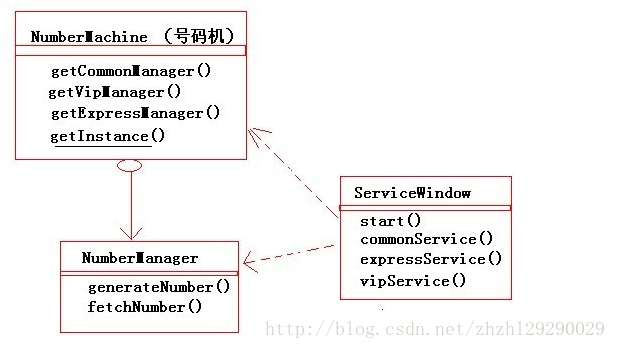
一、业务需求

模拟实现银行业务调度系统逻辑，具体需求如下：  
1、银行内有6个业务窗口，1 - 4号窗口为普通窗口，5号窗口为快速窗口，6号窗口为VIP窗口。  
2、有三种对应类型的客户：VIP客户，普通客户，快速客户（办理如交水电费、电话费之类业务的客户）。  
3、异步随机生成各种类型的客户，生成各类型用户的概率比例为：  
        VIP客户 ：普通客户 ：快速客户  =  1 ：6 ：3。  
4、客户办理业务所需时间有最大值和最小值，在该范围内随机设定每个VIP客户以及普通客户办理业务所需的时间，快速客户办理业务所需时间为最小值（提示：办理业务的过程可通过线程Sleep的方式模拟）。  
5、各类型客户在其对应窗口按顺序依次办理业务。  
6、 当VIP（6号）窗口和快速业务（5号）窗口没有客户等待办理业务的时候，这两个窗口可以处理普通客户的业务，而一旦有对应的客户等待办理业务的时候，则优先处理对应客户的业务。  
7、随机生成客户时间间隔以及业务办理时间最大值和最小值自定，可以设置。  
8、不要求实现GUI，只考虑系统逻辑实现，可通过Log方式展现程序运行结果。

二、需求分析

1，图解分析



2，面向对象分析和设计

1）有三种对应类型的客户：VIP客户，普通客户，快速客户 ，异步随机生成各种类型的客户，各类型客户在其对应窗口按顺序依次办理业务 。  
a、首先，经常在银行办理业务的人更有利于理解本系统，例如，我经常陪老婆跑银行，对银行的这个业务算是比较熟悉了，我知道每一个客户其实就是由银行的一个取号机器产生号码的方式来表示的。所以，我想到要有一个号码管理器对象，让这个对象不断地产生号码，就等于随机生成了客户。  
b、由于有三类客户，每类客户的号码编排都是完全独立的，所以，我想到本系统一共要产生三个号码管理器对象，各自管理一类用户的排队号码。这三个号码管理器对象统一由一个号码机器进行管理，这个号码机器在整个系统中始终只能有一个，所以，它要被设计成单例。  
2）各类型客户在其对应窗口按顺序依次办理业务 ，准确地说，应该是窗口依次叫号。  
a、各个窗口怎么知道该叫哪一个号了呢？它一定是问的相应的号码管理器，即服务窗口每次找号码管理器获取当前要被服务的号码。  
b、如果我不是多次亲身经历银行的这种业务，再加之积累了大量面向对象的应用开发经验，我也不知道能否轻松进行这种设计，能否发掘出其中隐含的对象信息，我真说不出具体的经验是什么，就是日积月累出来的一种感觉。难道这就是传说中的：“只可意会，不可言传？”  
(面向对象设计把握一个重要的经验：谁拥有数据，谁就对外提供操作这些数据的方法。再牢牢掌握几个典型的案例就可以了：人在黑板上画圆，列车司机紧急刹车，售货员统计收获小票的金额，你把门关上了等。  
学员的两个面向对象的面试题，用面向对象的方式设计如下情景。  
“两块石头磨成一把石刀，石刀可以砍树，砍成木材，木材做成椅子”，  
“球从一根绳子的一段移动到了另一端”，)

三、代码实现

一）：NmuberManager类的编写 ---> 产生号码的机器类

分析：  
1、定义一个动态数组即集合，用于存储未服务的号码  
2、定义一个产生新号码的方法generateNewManager和获取马上要为之服务的号码的方法fetchServiceNumber，这两个方法被不同的线程操作了相同的数据，所以，要进行同步。  
3、同步问题：  
要考虑generateNewManager和fetchServiceNumber两个方法是在两个不同的线程中，两个不同的线程要访问相同的数据lastNumber，就需要使用多线程技术  
代码实现：

package com.baidu.interview.bank;

import java.util.ArrayList;

import java.util.List;

public class NumberManager {

    //定义变量，即下一次产生的号码

    private int lastNumber = 1;

    //定义一个动态数组即集合，用于存储未服务的号码

    private List<Integer> queueNumber = new ArrayList<Integer>();

    //产生号码，返回下一次产生的号码，需要同步

    public synchronized Integer generateNewManager(){

        //将产生的号码存入集合

        queueNumber.add(++lastNumber);

        return lastNumber;

    }

    //从号码机中获取取号，需要同步

    public synchronized Integer fetchServiceNumber(){

        //取走谁就要将这个取出的值返回，取出什么类型就返回什么类型

        if(queueNumber.size()>0)

            return (Integer)queueNumber.remove(0);

        return null;

    }

}

package com.baidu.interview.bank;

import java.util.ArrayList;

import java.util.List;

public class NumberManager {

//定义变量，即下一次产生的号码

private int lastNumber = 1;

//定义一个动态数组即集合，用于存储未服务的号码

private List<Integer> queueNumber = new ArrayList<Integer>();

//产生号码，返回下一次产生的号码，需要同步

public synchronized Integer generateNewManager(){

//将产生的号码存入集合

queueNumber.add(++lastNumber);

return lastNumber;

}

//从号码机中获取取号，需要同步

public synchronized Integer fetchServiceNumber(){

//取走谁就要将这个取出的值返回，取出什么类型就返回什么类型

if(queueNumber.size()>0)

return (Integer)queueNumber.remove(0);

return null;

}

}

二）NumberMachine类的编写：

分析：  
1、定义三个成员变量分别指向三个NumberManager对象，分别表示普通、快速和VIP客户的号码管理器，定义三个对应的方法来返回这三个NumberManager对象。  
2、将NumberMachine类设计成单例。这三个号码管理器对象统一由一个号码机器进行管理，这个号码机器在整个系统中始终只能有一个，所以，它要被设计成单例。  
代码实现：

[java] [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12761375) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

[print?](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

package com.baidu.interview.bank;

//创建号码管理器类

public class NumberMachine {

    //设计单列

    //私有构造函数

    private NumberMachine(){}

    private static NumberMachine instance = new NumberMachine();

    //公有静态方法，返回这个单例对象的引用

    public static NumberMachine getInstance(){

        return instance;

    }

    //创建三种客户产生号码的机器

    private NumberManager commnManager = new NumberManager();

    private NumberManager expressManager = new NumberManager();

    private NumberManager vipManager = new NumberManager();

    //产生get方法，获取对象

    public NumberManager getCommnManager() {

        return commnManager;

    }

    public NumberManager getExpressManager() {

        return expressManager;

    }

    public NumberManager getVipManager() {

        return vipManager;

    }

}

package com.baidu.interview.bank;

//创建号码管理器类

public class NumberMachine {

//设计单列

//私有构造函数

private NumberMachine(){}

private static NumberMachine instance = new NumberMachine();

//公有静态方法，返回这个单例对象的引用

public static NumberMachine getInstance(){

return instance;

}

//创建三种客户产生号码的机器

private NumberManager commnManager = new NumberManager();

private NumberManager expressManager = new NumberManager();

private NumberManager vipManager = new NumberManager();

//产生get方法，获取对象

public NumberManager getCommnManager() {

return commnManager;

}

public NumberManager getExpressManager() {

return expressManager;

}

public NumberManager getVipManager() {

return vipManager;

}

}

三）CustomerType枚举类和Constants类的编写：

第一、CustomerType类  
分析：  
1、系统中有三种类型的客户，所以用定义一个枚举类，其中定义三个成员分别表示三种类型的客户。  
2、重写toString方法，返回类型的中文名称。这是在后面编码时重构出来的，刚开始不用考虑。  
代码实现：

[java] [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12761375) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

[print?](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

package com.baidu.interview.bank;

/\*有三种窗口：vip：普通：快速=1：6：3

     \* 可以使用枚举，表示三种窗口\*/

public enum CustomerType {

    COMMON,EXPRESS,VIP;

    public String toString(){

        String name = null;

        switch(this){

        case COMMON:

            name = "普通";

            break;

        case EXPRESS:

            name = "快速";

            break;

        case VIP:

            name = name();

            break;

        }

        return name;

    }

}

package com.baidu.interview.bank;

/\*有三种窗口：vip：普通：快速=1：6：3

\* 可以使用枚举，表示三种窗口\*/

public enum CustomerType {

COMMON,EXPRESS,VIP;

public String toString(){

String name = null;

switch(this){

case COMMON:

name = "普通";

break;

case EXPRESS:

name = "快速";

break;

case VIP:

name = name();

break;

}

return name;

}

}

第二、Constants类  
作为一个常量类，用于提供各种常量值  
代码实现：

[java] [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12761375) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

[print?](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

package com.baidu.interview.bank;

//定义一个类，作为常量值的获取

public class Constants {

    public static int MAX\_SERVICE\_TIME = 10000;

    public static int MIN\_SERVICE\_TIME = 1000;

    public static int COMMON\_CUSTOMER\_INTERVAL\_TIME = 1;

}

package com.baidu.interview.bank;

//定义一个类，作为常量值的获取

public class Constants {

public static int MAX\_SERVICE\_TIME = 10000;

public static int MIN\_SERVICE\_TIME = 1000;

public static int COMMON\_CUSTOMER\_INTERVAL\_TIME = 1;

}

说明：  
每个普通窗口服务一个客户的平均时间为5秒，一共有4个这样的窗口，也就是说银行的所有普通窗口合起来，平均1.25秒内可以服务完一个普通客户，再加上快速窗口和VIP窗口也可以服务普通客户，所以，1秒钟产生一个普通客户比较合理

四）ServiceWindow类的编写

分析：  
1、定义一个start方法，内部启动一个线程，根据服务窗口的类别分别循环调用三个不同的方法。  
2、定义三个方法分别对三种客户进行服务，为了观察运行效果，应详细打印出其中的细节信息。  
 代码实现：

[java] [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12761375) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

[print?](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

package com.baidu.interview.bank;

import java.util.Random;

import java.util.concurrent.Executors;

//创建服务窗口类

public class ServiceWindow {

    //定义一个变量，表示窗口类型，默认为普通窗口

    private CustomerType type = CustomerType.COMMON;

    private int windowId = 1;

    //定义两个私有变量的set方法，不一定非要像上面一样指定

    public CustomerType getType(){

        return type;

    }

    public void setType(CustomerType type) {

        this.type = type;

    }

    public void setWindowId(int windowId) {

        this.windowId = windowId;

    }

    //叫号的方法，

    public void start(){

        //创建单独的线程

        Executors.newSingleThreadExecutor().execute(new Runnable() {

            public void run() {

                //不停的取号

                while(true){

                    /\*if(type == CustomerType.COMMON)

                    //通过号码机要号码管理器中产生的号码

                    NumberMachine.getInstance().getCommnManager();

                    else if()...;

                    else....;\*/

                    //不同的类型客户有不同的服务

                    switch(type){

                        case COMMON:

                            commonService();

                            break;

                        case EXPRESS:

                            expressService();

                            break;

                        case VIP:

                            vipService();

                            break;

                    }

                }

            }

        });

    }

    //创建普通客户服务

    private void commonService(){

        //开始叫号

        String windowName = "第\*" + windowId + "\*号" + type + "窗口";

        System.out.println(windowName + "开始获取普通任务!");

        Integer serviceNumber = NumberMachine.getInstance().getCommnManager().fetchServiceNumber();

        if(serviceNumber != null){

            //如果有服务号码，就开始服务

            System.out.println(windowName + "开始为第" + serviceNumber + "号普通客户服务");

            //普通客户服务耗时的区间

            int maxRandom = Constants.MAX\_SERVICE\_TIME - Constants.MIN\_SERVICE\_TIME;

            //普通客户消耗的时间，随机产生

            int serviceTime = new Random().nextInt(maxRandom) + 1 + Constants.MIN\_SERVICE\_TIME;

            try {

                //运行sleep，即表示正在服务

                Thread.sleep(serviceTime);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

            //完成服务

            System.out.println(windowName + "完成为第" + serviceNumber + "号普通客户服务，总共耗时" + serviceTime/1000 + "秒");

        }else{

            //如果没有服务号码产生，则空闲

            System.out.println(windowName + "没有取到普通任务，正在空闲一秒");

            try {

                Thread.sleep(1000);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

    //创建VIP服务

    private void vipService() {

        //开始叫号

        Integer serviceNumber = NumberMachine.getInstance().getCommnManager().fetchServiceNumber();

        String windowName = "第\*" + windowId + "\*号" + type + "窗口";

        System.out.println(windowName + "开始获取VIP任务!");

        if(serviceNumber != null){

            System.out.println(windowName + "开始为第" + serviceNumber + "号VIP客户服务");

            //VIP客户服务耗时的区间

            int maxRandom = Constants.MAX\_SERVICE\_TIME - Constants.MIN\_SERVICE\_TIME;

            int serviceTime = new Random().nextInt(maxRandom) + 1 + Constants.MIN\_SERVICE\_TIME;

            try {

                //运行sleep，即表示正在服务

                Thread.sleep(serviceTime);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

            System.out.println(windowName + "完成为第" + serviceNumber + "号VIP客户服务，总共耗时" + serviceTime/1000 + "秒");

        }else{

            //如果没有服务号码产生，即快速窗口空闲时，则可为普通客户服务

            System.out.println(windowName + "没有取到VIP任务！");

            commonService();

        }

    }

    //创建快速服务

    private void expressService() {

        //开始叫号

        Integer serviceNumber = NumberMachine.getInstance().getCommnManager().fetchServiceNumber();

        String windowName = "第\*" + windowId + "\*号" + type + "窗口";

        System.out.println(windowName + "开始获取快速任务!");

        if(serviceNumber != null){

            System.out.println(windowName + "开始为第" + serviceNumber + "号快速客户服务");

            //快速客户服务耗时为最小值

            int serviceTime = Constants.MIN\_SERVICE\_TIME;

            try {

                //运行sleep，即表示正在服务

                Thread.sleep(serviceTime);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

            System.out.println(windowName + "完成为第" + serviceNumber + "号快速客户服务，总共耗时" + serviceTime/1000 + "秒");

        }else{

            //如果没有服务号码产生，即VIP窗口空闲时，则可为普通客户服务

            System.out.println(windowName + "没有取到快速任务！");

            commonService();

        }

    }

}

package com.baidu.interview.bank;

import java.util.Random;

import java.util.concurrent.Executors;

//创建服务窗口类

public class ServiceWindow {

//定义一个变量，表示窗口类型，默认为普通窗口

private CustomerType type = CustomerType.COMMON;

private int windowId = 1;

//定义两个私有变量的set方法，不一定非要像上面一样指定

public CustomerType getType(){

return type;

}

public void setType(CustomerType type) {

this.type = type;

}

public void setWindowId(int windowId) {

this.windowId = windowId;

}

//叫号的方法，

public void start(){

//创建单独的线程

Executors.newSingleThreadExecutor().execute(new Runnable() {

public void run() {

//不停的取号

while(true){

/\*if(type == CustomerType.COMMON)

//通过号码机要号码管理器中产生的号码

NumberMachine.getInstance().getCommnManager();

else if()...;

else....;\*/

//不同的类型客户有不同的服务

switch(type){

case COMMON:

commonService();

break;

case EXPRESS:

expressService();

break;

case VIP:

vipService();

break;

}

}

}

});

}

//创建普通客户服务

private void commonService(){

//开始叫号

String windowName = "第\*" + windowId + "\*号" + type + "窗口";

System.out.println(windowName + "开始获取普通任务!");

Integer serviceNumber = NumberMachine.getInstance().getCommnManager().fetchServiceNumber();

if(serviceNumber != null){

//如果有服务号码，就开始服务

System.out.println(windowName + "开始为第" + serviceNumber + "号普通客户服务");

//普通客户服务耗时的区间

int maxRandom = Constants.MAX\_SERVICE\_TIME - Constants.MIN\_SERVICE\_TIME;

//普通客户消耗的时间，随机产生

int serviceTime = new Random().nextInt(maxRandom) + 1 + Constants.MIN\_SERVICE\_TIME;

try {

//运行sleep，即表示正在服务

Thread.sleep(serviceTime);

} catch (InterruptedException e) {

e.printStackTrace();

}

//完成服务

System.out.println(windowName + "完成为第" + serviceNumber + "号普通客户服务，总共耗时" + serviceTime/1000 + "秒");

}else{

//如果没有服务号码产生，则空闲

System.out.println(windowName + "没有取到普通任务，正在空闲一秒");

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

//创建VIP服务

private void vipService() {

//开始叫号

Integer serviceNumber = NumberMachine.getInstance().getCommnManager().fetchServiceNumber();

String windowName = "第\*" + windowId + "\*号" + type + "窗口";

System.out.println(windowName + "开始获取VIP任务!");

if(serviceNumber != null){

System.out.println(windowName + "开始为第" + serviceNumber + "号VIP客户服务");

//VIP客户服务耗时的区间

int maxRandom = Constants.MAX\_SERVICE\_TIME - Constants.MIN\_SERVICE\_TIME;

int serviceTime = new Random().nextInt(maxRandom) + 1 + Constants.MIN\_SERVICE\_TIME;

try {

//运行sleep，即表示正在服务

Thread.sleep(serviceTime);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(windowName + "完成为第" + serviceNumber + "号VIP客户服务，总共耗时" + serviceTime/1000 + "秒");

}else{

//如果没有服务号码产生，即快速窗口空闲时，则可为普通客户服务

System.out.println(windowName + "没有取到VIP任务！");

commonService();

}

}

//创建快速服务

private void expressService() {

//开始叫号

Integer serviceNumber = NumberMachine.getInstance().getCommnManager().fetchServiceNumber();

String windowName = "第\*" + windowId + "\*号" + type + "窗口";

System.out.println(windowName + "开始获取快速任务!");

if(serviceNumber != null){

System.out.println(windowName + "开始为第" + serviceNumber + "号快速客户服务");

//快速客户服务耗时为最小值

int serviceTime = Constants.MIN\_SERVICE\_TIME;

try {

//运行sleep，即表示正在服务

Thread.sleep(serviceTime);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(windowName + "完成为第" + serviceNumber + "号快速客户服务，总共耗时" + serviceTime/1000 + "秒");

}else{

//如果没有服务号码产生，即VIP窗口空闲时，则可为普通客户服务

System.out.println(windowName + "没有取到快速任务！");

commonService();

}

}

}

五）主类的编写

测试：  
1、用for循环创建出4个普通窗口，再创建出1个快速窗口和一个VIP窗口。  
2、再创建三个定时器，分别定时去创建新的普通客户号码、新的快速客户号码、新的VIP客户号码。  
代码实现：

[java] [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12761375) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

[print?](http://blog.csdn.net/zhzhl29290029/article/details/12761375)

package com.baidu.interview.bank;

import java.util.concurrent.Executors;

import java.util.concurrent.TimeUnit;

public class BankTest {

    public static void main(String[] args) {

        //产生4个普通窗口

        for(int i=1;i<5;i++){

            ServiceWindow window = new ServiceWindow();

            window.setWindowId(i);

            window.start();

        }

        //产生1个快速窗口

        ServiceWindow expressWindow = new ServiceWindow();

        expressWindow.setType(CustomerType.EXPRESS);

        expressWindow.start();

        //产生一个VIP窗口

        ServiceWindow vipWindow = new ServiceWindow();

        vipWindow.setType(CustomerType.VIP);

        vipWindow.start();

        //普通客户拿号

        Executors.newScheduledThreadPool(1).scheduleAtFixedRate(

                new Runnable() {

                    public void run() {

                        // 创建线程，运行普通客户服务

                        Integer serviceNumber = NumberMachine.getInstance().getCommnManager().generateNewManager();

                        System.out.println("第" + serviceNumber + "号普通客户正在等待服务！");

                    }

                },

                0,

                Constants.COMMON\_CUSTOMER\_INTERVAL\_TIME,

                TimeUnit.SECONDS

            );

        //快速客户拿号

        Executors.newScheduledThreadPool(1).scheduleAtFixedRate(

                new Runnable() {

                    public void run() {

                        //创建线程，运行快速客户服务

                        Integer serviceNumber = NumberMachine.getInstance().getExpressManager().generateNewManager();

                        System.out.println("第" + serviceNumber + "号快速客户正在等待服务！");

                    }

                },

                0,

                Constants.COMMON\_CUSTOMER\_INTERVAL\_TIME \* 2,

                TimeUnit.SECONDS

            );

        //VIP客户拿号

        Executors.newScheduledThreadPool(1).scheduleAtFixedRate(

                new Runnable() {

                    public void run() {

                        //创建线程，运行VIP客户服务

                        Integer serviceNumber = NumberMachine.getInstance().getVipManager().generateNewManager();

                        System.out.println("第" + serviceNumber + "号VIP客户正在等待服务！");

                    }

                },

                0,

                Constants.COMMON\_CUSTOMER\_INTERVAL\_TIME \* 6,

                TimeUnit.SECONDS

            );

    }

}