## 设备管理

### 需求分析

1. 模拟操作系统中有A、B、C三种独占型设备，A设备1个，B设备2个，C设备2个。（同一种设备的实体被认为是相同的）。
2. 采用死锁预防方法来处理申请独占设备可能造成的死锁。
3. 屏幕显示：每个设备是否被使用。哪个设备在使用该设备，哪些进程在等待使用该设备。
4. 设置设备分配表和等待队列

### 概要设计

1. 抽象出了 Device.java类
2. 抽象出了DeviceTable.java类
3. 抽象出了 DeviceWaitQueue.java类





### 详细设计

设备类

getDevice( )

@申请设备

@param Uid

@param time

@param size

@return int 设备号

伪代码：

在设备表中查找要申请的设备。

改变设备的状态，分配设备。

removeDevice( )

@释放设备

@param Uid

@return int[]

伪代码：

在设备表中查找要释放的设备。

释放设备后，从等待队列中取出一个节点。

getFirstNode( )

@获取等待队列中的节点

@return int

伪代码：

在对应设备中的等待队列中取出对应的节点。

**public class** Device {  
 */\* A、B、C三种独占型设备，A设备1个，B设备2个，C设备2个。 \*/* **private int** returnIndexA1 = 1;  
 **private int** returnIndexB1 = 2;  
 **private int** returnIndexB2 = 3;  
 **private int** returnIndexC1 = 4;  
 **private int** returnIndexC2 = 5;  
 **private int** c;  
 */\* 同一设备的实体被认为是相同的 \*/* **private** LinkedList<DeviceWaitQueue> blockA ;  
 **private** LinkedList<DeviceWaitQueue> blockB ;  
 **private** LinkedList<DeviceWaitQueue> blockC ;  
 **private** DeviceTable deviceTable;  
 DeviceWaitQueue deviceWaitQueueA;  
 DeviceWaitQueue deviceWaitQueueB;  
 DeviceWaitQueue deviceWaitQueueC;  
 **public** Device() {  
 **this**.blockA = **new** LinkedList<DeviceWaitQueue>();  
 **this**.blockB = **new** LinkedList<DeviceWaitQueue>();  
 **this**.blockC = **new** LinkedList<DeviceWaitQueue>();  
 **this**.deviceTable = **new** DeviceTable();  
 }  
 */\*\*  
 \** ***@Description*** *申请A设备  
 \** ***@param*** *Uid  
 \** ***@param*** *time  
 \** ***@param*** *size  
 \*/* **public int** getDeviceA(String Uid, **int** time, **int** size ){  
 deviceWaitQueueA =**new** DeviceWaitQueue(Uid,size,time);  
 **if** (deviceTable.getA1().equals(**"设备空闲"**) ){  
 deviceTable.setA1(Uid);  
 c = returnIndexA1;  
 returnIndexA1 = 9;  
 **return** c;  
 }  
 **else** {  
 blockA.add(deviceWaitQueueA);  
 **return** 9;  
 }  
  
 }  
 */\*\*  
 \** ***@Description*** *释放A设备  
 \** ***@param*** *Uid  
 \*/* **public int**[] removeDeviceA(String Uid){  
 **int**[] re =**new int**[2];  
 **if** (Uid.equals(deviceTable.getA1())){  
 deviceTable.setA1(**"设备空闲"**);  
 **if**(!blockA.isEmpty()) {  
 re[0] = 1;  
 re[1] = gerFirstNodeA();  
 **return** re;  
 }  
 returnIndexA1 = 1;  
 **return null**;  
 }**else  
 return null**;  
 }  
  
 */\*\*  
 \** ***@Description*** *获取等待队列第一个节点  
 \*/* **public int** gerFirstNodeA(){  
 deviceWaitQueueA = blockA.removeFirst();  
 getDeviceA(deviceWaitQueueA.getUid(),deviceWaitQueueA.getTime(),deviceWaitQueueA.getSize());  
 **return** deviceWaitQueueA.getTime();  
 }  
 */\*\*  
 \** ***@Description*** *申请B设备  
 \** ***@param*** *Uid  
 \** ***@param*** *time  
 \** ***@param*** *size  
 \*/* **public int** getDeviceB(String Uid, **int** time, **int** size ){  
 deviceWaitQueueB =**new** DeviceWaitQueue(Uid,size,time);  
 **if** (deviceTable.getB1().equals(**"设备空闲"**) ){  
 deviceTable.setB1(Uid);  
 c = returnIndexB1;  
 returnIndexB1 = 9;  
 **return** c;  
 }  
 **else if**(deviceTable.getB2().equals(**"设备空闲"**) ) {  
 deviceTable.setB2(Uid);  
 c = returnIndexB2;  
 returnIndexB2 = 9;  
 **return** c;  
 }  
 **else**{  
 blockB.add(deviceWaitQueueB);  
 **return** 9;  
 }  
 }  
 */\*\*  
 \** ***@Description*** *释放B设备  
 \** ***@param*** *Uid  
 \*/* **public int**[] removeDeviceB(String Uid){  
 **int**[] re =**new int**[2];  
 **if** (Uid.equals(deviceTable.getB1())){  
 deviceTable.setB1(**"设备空闲"**);  
 **if**(!blockB.isEmpty()) {  
 re[0] = 3;  
 re[1] = gerFirstNodeB();;  
 **return** re;  
 }  
 returnIndexB1 = 3;  
 **return null**;  
 }**else if** (Uid.equals(deviceTable.getB2())){  
 deviceTable.setB2(**"设备空闲"**);  
 **if**(!blockB.isEmpty()) {  
 re[0] = 4;  
 re[1] = gerFirstNodeB();;  
 **return** re;  
 }  
 returnIndexB2 = 4;  
 **return null**;  
 }  
 **else  
 return null**;  
 }  
 */\*\*  
 \** ***@Description*** *获取B设备等待队列第一个节点  
 \*/* **public int** gerFirstNodeB(){  
 deviceWaitQueueB = blockB.removeFirst();  
 getDeviceB(deviceWaitQueueB.getUid(),deviceWaitQueueB.getTime(),deviceWaitQueueB.getSize());  
 **return** deviceWaitQueueB.getTime();  
 }  
 */\*\*  
 \** ***@Description*** *申请C设备 两个c设备  
 \** ***@param*** *Uid  
 \** ***@param*** *time  
 \** ***@param*** *size  
 \*/* **public int** getDeviceC(String Uid, **int** time, **int** size ){  
 deviceWaitQueueC = **new** DeviceWaitQueue(Uid,size,time);  
 **if** ( deviceTable.getC1().equals(**"设备空闲"**) ){  
 deviceTable.setC1(Uid);  
 c = returnIndexC1;  
 returnIndexC1 = 9;  
 **return** c;  
 }  
 **else if**( deviceTable.getC2().equals(**"设备空闲"**) ){  
 deviceTable.setC2(Uid);  
 c = returnIndexC2;  
 returnIndexC2 = 9;  
 **return** c;  
 }  
 **else**{  
 blockC.add(deviceWaitQueueC);  
 **return** 9;  
 }  
 }  
 */\*\*  
 \** ***@Description*** *释放C设备  
 \** ***@param*** *Uid  
 \*/* **public int**[] removeDeviceC(String Uid){  
 **int**[] re = **new int**[2];  
 **if** (Uid.equals(deviceTable.getC1())){  
 deviceTable.setC1(**"设备空闲"**);  
 **if**(!blockC.isEmpty()) {  
 re[0] = 6;  
 re[1] = gerFirstNodeC();;  
 **return** re;  
 }  
 returnIndexC1 = 6;  
 **return null**;  
 }**else if** (Uid.equals(deviceTable.getC2())){  
 deviceTable.setC2(**"设备空闲"**);  
 **if**(!blockC.isEmpty()) {  
 re[0] = 7;  
 re[1] = gerFirstNodeC();;  
 **return** re;  
 }  
 returnIndexC2 = 7;  
 **return null**;  
 }  
 **else  
 return null**;  
 }  
 */\*\*  
 \** ***@Description*** *获取B设备等待队列第一个节点  
 \*/* **public int** gerFirstNodeC(){  
 deviceWaitQueueC = blockB.removeFirst();  
 getDeviceC(deviceWaitQueueC.getUid(),deviceWaitQueueC.getTime(),deviceWaitQueueC.getSize());  
 **return** deviceWaitQueueC.getTime();  
 }  
  
 **public** DeviceTable getDeviceTable() {  
 **return** deviceTable;  
 }  
  
 **public void** setDeviceTable(DeviceTable deviceTable) {  
 **this**.deviceTable = deviceTable;  
 }  
  
 **public** LinkedList<DeviceWaitQueue> getBlockA() {  
 **return** blockA;  
 }  
  
 **public void** setBlockA(LinkedList<DeviceWaitQueue> blockA) {  
 **this**.blockA = blockA;  
 }  
  
 **public** LinkedList<DeviceWaitQueue> getBlockB() {  
 **return** blockB;  
 }  
  
 **public void** setBlockB(LinkedList<DeviceWaitQueue> blockB) {  
 **this**.blockB = blockB;  
 }  
  
 **public** LinkedList<DeviceWaitQueue> getBlockC() {  
 **return** blockC;  
 }  
  
 **public void** setBlockC(LinkedList<DeviceWaitQueue> blockC) {  
 **this**.blockC = blockC;  
 }  
  
 **public** DeviceWaitQueue getDeviceWaitQueueA() {  
 **return** deviceWaitQueueA;  
 }  
  
 **public void** setDeviceWaitQueueA(DeviceWaitQueue deviceWaitQueueA) {  
 **this**.deviceWaitQueueA = deviceWaitQueueA;  
 }  
  
 **public** DeviceWaitQueue getDeviceWaitQueueB() {  
 **return** deviceWaitQueueB;  
 }  
  
 **public void** setDeviceWaitQueueB(DeviceWaitQueue deviceWaitQueueB) {  
 **this**.deviceWaitQueueB = deviceWaitQueueB;  
 }  
  
 **public** DeviceWaitQueue getDeviceWaitQueueC() {  
 **return** deviceWaitQueueC;  
 }  
  
 **public void** setDeviceWaitQueueC(DeviceWaitQueue deviceWaitQueueC) {  
 **this**.deviceWaitQueueC = deviceWaitQueueC;  
 }  
}

设备表

get( )

@获取设备的状态

@return string

伪代码：

返回设备的状态

set( )

@设置设备的状态

伪代码：

设置设备的状态

**public class** DeviceTable {  
 **private** String A1;  
 **private** String B1;  
 **private** String B2;  
 **private** String C1;  
 **private** String C2;  
  
 **public** DeviceTable() {  
 A1 = **"设备空闲"**;  
 B1 = **"设备空闲"**;  
 B2 = **"设备空闲"**;  
 C1 = **"设备空闲"**;  
 C2 = **"设备空闲"**;  
 }  
  
 **public** String getA1() {  
 **return** A1;  
 }  
  
 **public void** setA1(String a1) {  
 A1 = a1;  
 }  
  
 **public** String getB1() {  
 **return** B1;  
 }  
  
 **public void** setB1(String b1) {  
 B1 = b1;  
 }  
  
 **public** String getB2() {  
 **return** B2;  
 }  
  
 **public void** setB2(String b2) {  
 B2 = b2;  
 }  
  
 **public** String getC1() {  
 **return** C1;  
 }  
  
 **public void** setC1(String c1) {  
 C1 = c1;  
 }  
  
 **public** String getC2() {  
 **return** C2;  
 }  
  
 **public void** setC2(String c2) {  
 C2 = c2;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "DeviceTable:\n"** +  
 **"A='"** + A1 + **'\''** +  
 **"\nB='"** + B1 + **'\''** +  
 **"\nB='"** + B2 + **'\''** +  
 **"\nC='"** + C1 + **'\''** +  
 **"\nC='"** + C2 + **'\''** +  
 }  
}

等待队列

getUid( )

@获取等待队列的状态

@return string

伪代码：

返回等待队列的状态

setUid( )

@设置等待队列的状态

伪代码：

设置等待队列的状态

getSize( )

@获取等待队列的大小

@return int

伪代码：

返回等待队列的大小

setSize( )

@设置等待队列的大小

伪代码：

设置等待队列的大小

getTime( )

@获取等待队列的时间

@return int

伪代码：

返回等待队列的时间

setUid( )

@设置等待队列的状态

伪代码：

设置等待队列的状态

**public class** DeviceWaitQueue {  
 **private** String uid;  
 **private int** size;  
 **private int** time;  
  
 **public** DeviceWaitQueue(String Uid, **int** size, **int** time ){  
 **this**.uid = Uid;  
 **this**.size = size;  
 **this**.time = time;  
 }  
  
 **public** String getUid() {  
 **return** uid;  
 }  
  
 **public void** setUid(String uid) {  
 **this**.uid = uid;  
 }  
  
 **public int** getSize() {  
 **return** size;  
 }  
  
 **public void** setSize(**int** size) {  
 **this**.size = size;  
 }  
  
 **public int** getTime() {  
 **return** time;  
 }  
  
 **public void** setTime(**int** time) {  
 **this**.time = time;  
 }  
}

### 运行效果



### 个人感悟

在这一个学期的操作系统课程设计中我主要做设备管理的实现，收获了很多也学会了很多：

1. 代码能力，学习了很多关于java.swing里的知识，对处理图形界面有了初步的认识。
2. 对于这次负责设备管理，让我复习了os课程里传授的知识，在实战中运用所学，巩固知识。
3. 这次团队的分工，很好的解耦，每个人各自分工，合理使得团队效益最大化。