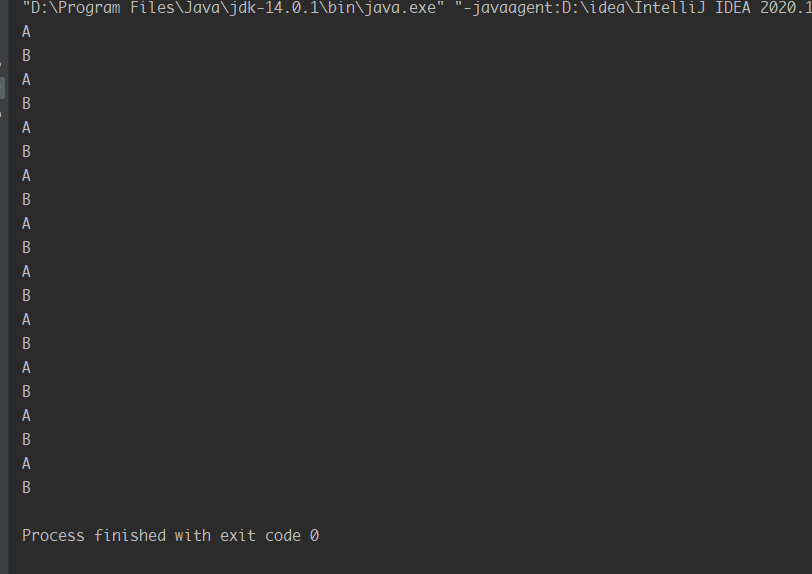
# 通过多线程实现交替打印A B A B 各10次 后程序退出

代码

package Thread\_HomeWork.HomeWork01;  
  
  
public class Printer {  
  
 // 判断是否可以打印  
 private boolean hasBufferToPrint = false;  
  
 public synchronized void PrintA() {  
 while (hasBufferToPrint) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 System.*out*.println("A");  
 hasBufferToPrint = true;  
 notify(); // 唤醒B  
 }  
  
 public synchronized void PrintB() {  
 while (!hasBufferToPrint) {  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 System.*out*.println("B");  
 hasBufferToPrint = false;  
 notify(); // 唤醒A  
 }  
  
 static class ThreadA extends Thread{  
 private Printer printer;  
  
 public ThreadA(Printer printer) {  
 this.printer = printer;  
 }  
 public void run(){  
 for (int i = 0; i < 10; i++) {  
 printer.PrintA();  
 }  
 }  
 }  
  
 static class ThreadB extends Thread{  
 private Printer printer;  
  
 public ThreadB(Printer printer) {  
 this.printer = printer;  
 }  
 public void run(){  
 for (int i = 0; i < 10; i++) {  
 printer.PrintB();  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 Printer printer = new Printer();  
 ThreadA a = new ThreadA(printer);  
 ThreadB b = new ThreadB(printer);  
 a.start();  
 b.start();  
 }  
}

运行结果

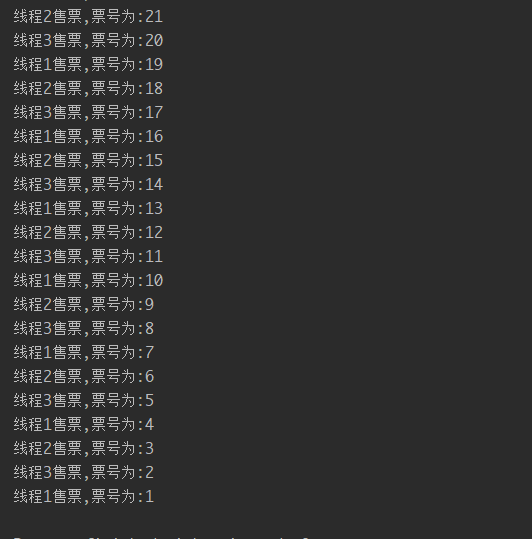


# 设计一个购票系统：可以多个线程同时进行购票，确保购票的业务正确。

方法1：

package Thread\_HomeWork.HomeWork02;  
import java.util.concurrent.locks.ReentrantLock;  
  
 */\*\*  
 \* 设计一个购票系统：可以多个线程同时进行购票，确保购票的业务正确  
 \*/* class Widow implements Runnable{  
 private int ticket = 100;  
  
 // 1. 实例化一个ReentrantLock  
 private ReentrantLock lock = new ReentrantLock(true);  
 @Override  
 public void run() {  
 while (true) {  
 try {  
 //2. 调用lock()  
 lock.lock();  
  
 if (ticket > 0) {  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println(Thread.*currentThread*().getName() + "售票,票号为:"+ ticket);  
 ticket--;  
 }else{  
 break;  
 }  
 }finally {  
 // 3. 调用解锁方法();  
 lock.unlock();  
 }  
  
 }  
 }  
 }  
  
  
public class Ticket {  
 public static void main(String[] args) {  
 Widow w = new Widow();  
 Thread t1 = new Thread(w);  
 Thread t2 = new Thread(w);  
 Thread t3 = new Thread(w);  
 t1.setName("线程1");  
 t2.setName("线程2");  
 t3.setName("线程3");  
 t1.start();  
 t2.start();  
 t3.start();  
 }  
}

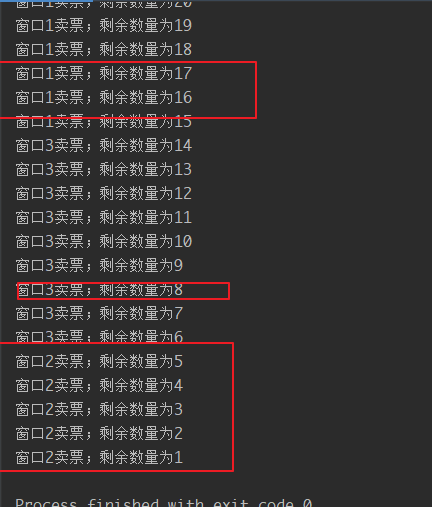
运行结果



方法2

package Thread\_HomeWork.HomeWork02;  
  
  
public class Ticket\_02 {  
 public static void main(String[] args) {  
 Good good = new Good();  
 GoodRunnable gr = new GoodRunnable(good);  
 Thread t1 = new Thread(gr, "窗口1");  
 Thread t2 = new Thread(gr, "窗口2");  
 Thread t3 = new Thread(gr, "窗口3");  
 t1.start();  
 t2.start();  
 t3.start();  
  
 }  
}  
  
class Good {  
 private Integer count = 100;  
  
 public Integer getCount() {  
 return count;  
 }  
  
 public void setCount(Integer count) {  
 this.count = count;  
 }  
  
  
 public void change() {  
 while (true) {  
 synchronized ("lockA") {  
 if (count > 0) {  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println(Thread.*currentThread*().getName() + "卖票；剩余数量为" + getCount());  
 count--;  
 } else {  
 break;  
 }  
 }  
 }  
  
 }  
}  
  
class GoodRunnable implements Runnable {  
 private Good good;  
  
 public GoodRunnable(Good good) {  
 this.good = good;  
 }  
  
 @Override  
 public void run() {  
 good.change();  
 }  
}

运行结果



# IO的使用

第三题与第四题均写完了读，没有写，第三题写完了没办法进行sort排序，还在寻找另一种思路。第四题，也是使用split进行添加。

Product类

package Thread\_HomeWork.HomeWork03;  
  
public class Product {  
 private int id;  
 private String name;  
 private float price;  
 private int number;  
  
 public Product() {  
 }  
  
 public Product(int id, String name, float price, int number) {  
 this.id = id;  
 this.name = name;  
 this.price = price;  
 this.number = number;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public float getPrice() {  
 return price;  
 }  
  
 public void setPrice(float price) {  
 this.price = price;  
 }  
  
 public int getNumber() {  
 return number;  
 }  
  
 public void setNumber(int number) {  
 this.number = number;  
 }  
  
 @Override  
 public String toString() {  
 return "Product [id=" + id + ", name=" + name + ", price=" + price + ", number=" + number + "]";  
 }  
  
  
}

读取类

package Thread\_HomeWork.HomeWork03;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.util.ArrayList;  
import java.util.List;  
  
  
public class BufferedReader\_ {  
  
 public static void main(String[] args) throws Exception {  
 String filePath = "d:\\a.txt";  
 // 创建bufferedReader  
 BufferedReader bufferedReader = new BufferedReader(new FileReader(filePath));  
 // 读取  
 String line; // 按行读取,效率高  
 int count = 1;  
 // 说明  
 // 1. readLine() 按行读取文件  
 // 2. 当返回null时,表示文件读取完毕  
 while((line = bufferedReader.readLine())!= null){  
 List<Product> list = *readLine*(line);  
 System.*out*.println(list.toString());  
 //System.out.println(line);  
 }  
 // 关闭流,这里注意:只需要关闭BufferedReader,因为底层会自动的去关闭节点流  
 // FileReader  
 bufferedReader.close();  
 }  
 public static List<Product> readLine(String line){  
 String[] split = line.split(",");  
 Product product = new Product(Integer.*parseInt*(split[0]), split[1], Float.*parseFloat*(split[2]), Integer.*parseInt*(split[3]));  
 List<Product> list = new ArrayList<>();  
 list.add(product);  
 //System.out.println(list.toString());  
 return list;  
 }  
}

运行结果：

