重庆科技学院

上机实验报告(程序设计类)

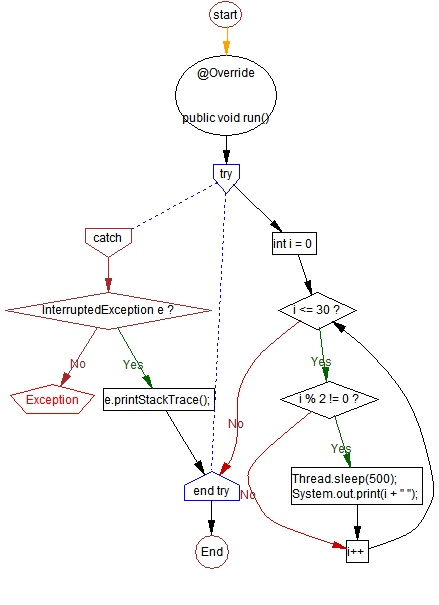
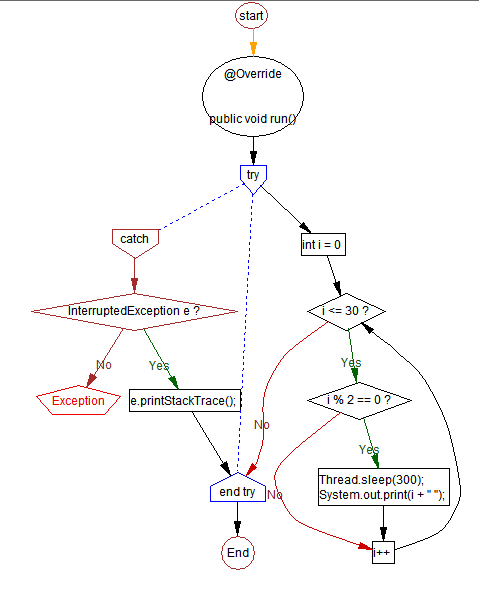
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 面向对象的序设计方法 | | 实验题目 | 实验六 Java多线程 | | |
| 机房编号 | I301 | | 上机时间 | 2018-12-6 | | |
| 指导老师 | 王双明 | | 报告序号 | 6 | 上机成绩 | . |
| 学生姓名 | 袁阊越 | 学号 | 2018520612 | 教学班级 | 计科升2016-2 | |
| **一、上机目的和要求:**  1. 掌握Java多线程的创建及其启动，多线程的两种常用创建方式及其区别；  2. 加深理解线程的生命周期及五种基本状态，分别是新建状态（New）、就绪状态（Runnable）、运行状态（Running）、阻塞状态（Blocked）、死亡状态（Dead）；  3. 掌握引起Java线程阻塞的主要方法，如：jion()方法、sleep()方法、yeild()方法；  4. 掌握线程安全及其解决机制，如：同步方法、同步代码块等。 | | | | | | |
| **二、程序开发环境:**  处理器：Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz (8 CPUs), ~3.4GHz  操作系统：Windows 7 专业版 32-bit  内存：4GB  JDK：jdk1.6.0\_04  软件：eclipse | | | | | | |
| **三、上机内容(老师布置的具体任务):**  1. 通过继承Thread类的方法创建两个线程，在Thread构造方法中指定线程的名称分别为“Thread\_one”和”Thread\_two”；其中一个线程向控制台输出奇数数列（1、3、5、7、……），时间间隔0.5秒；另一个线程向控制台输出偶数数列（2、4、6、8、……），时间间隔0.3秒。  2. 通过实现Runnable接口的方法创建一个新线程，要求main线程打印10次“main”，时间间隔0.2秒；新线程打印5次“new”，时间间隔0.4秒。  3. 下列程序代码是模拟火车售票情况，开启两个窗口销售10张火车票，程序一次运行结果如图1所示。请思考：  程序运行是否有错，若有错，错在哪里？应该如何改正？  **public** **class** SaleTicketThread **extends** Thread{  **private** String name;  **private** **int** tickets=10;  **public** SaleTicketThread() {  }  **public** SaleTicketThread(String name) {  **this**.name=name;  }  **public** **int** getTickets() {  **return** tickets;  }  **public** **void** setTickets(**int** tickets) {  **this**.tickets = tickets;  }  @Override  **public** **void** run() {  **while**(tickets>0){  System.***out***.println(name+" 正在销售第 "+tickets-- +" 票");  try{ Thread.sleep(200); }catch(InterruptedException e){}  }  }    **public** **static** **void** main(String[] args) {  SaleTicketThread sale1=**new** SaleTicketThread("窗口1");  SaleTicketThread sale2=**new** SaleTicketThread("窗口2");  sale1.start();  sale2.start();  }  }    图1 第3题某次运行情况  **错的！同一个进程的多的线程，访问共同的数据冲突，sale1和sale2线程的同时访问tickets数据时冲突。为程序添加synchronized,使其sale1线程的访问tickets数据的时候sale2线程不访问tickets数据，以及为tickets加上static变量，使其实时更改。**  4.下列程序代码是使用实现Runnable接口模拟火车售票情况，售票员休息一秒再卖出一张火车票。现开启两个窗口销售10张火车票，程序一次运行结果如图2所示。请思考：  程序运行是否有错，若有错，错在哪里？应该如何改正？  **public** **class** SaleTikcetRunnable **implements** Runnable {  **private** **int** tickets=10;  @Override  **public** **void** run() {  **while**(tickets>0){  **try** {  Thread.*sleep*(1000);  System.***out***.println(Thread.*currentThread*().getName()+" 正在  销售第 "+tickets-- +" 票");  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }    **public** **static** **void** main(String[] args) {  SaleTikcetRunnable ticket=**new** SaleTikcetRunnable();  Thread sale1=**new** Thread(ticket, "窗口1");  Thread sale2=**new** Thread(ticket, "窗口2");  sale1.start();  sale2.start();  }  }    图2 第4题某次运行情况  **错的！ 程序implements Runnable 依赖Thread实现，只有一个对象实现多个线程。程序在抢票过程中没有实现线程同步，出现票数重复抢，数据冲突。为线程中需要同步的代码加上synchronized。**  5. 有一个抽奖池，该抽奖池中存放了奖励的金额，该抽奖池用一个数组int[] arr = {10,5,20,50,100,200,500,800,2,80,300};  创建两个抽奖箱(线程)设置线程名称分别为“抽奖箱1”，“抽奖箱2”，随机从arr数组中获取奖项元素并打印在控制台上，运行结果如图3所示。    图3 模拟抽奖的运行结果  下列程序代码不完整，请补充完成。  **public** **class** ChouJiang **implements** Runnable {  //所有的奖券  **int**[] arr = { 10, 5, 20, 50, 100, 200, 500, 800, 2, 80, 300 };  **int** num = arr.length;  //记录对应的奖券是否抽过，为true表示已抽过、为false表示未抽过  **boolean**[] flag = **new** **boolean**[arr.length];  **public** **void** {  **while** (**true**) {  **synchronized** (**this**) {  **if** (num > 0) {  //产生指定范围（与奖券下标相一致）内的随机数  **int** index = (**int**) ( );  **int** get = arr[index];  // 代表这张抽奖券抽过了  **if** (flag[index] != **true**) {  flag[index] = **true**;  System.***out***.println( + " 又产生了一个" + get + "元大奖");  num--;  }  }**else**{  **return** ; //自动结束线程  }  }  }  }  **public** **static** **void** main(String[] args) {  ChouJiang c = **new** ChouJiang();  Thread t1 = **new** Thread(c, );  Thread t2 = **new** Thread(c, "抽奖箱2");  t1.start();  ;  }  }  以下为选做题：  6. 编写如图4所示的界面，当程序运行时：  （1）每隔两秒钟在显示字母区域随机显示一个字母（如图所示，显示的字母是“g”）；  （2）用户在文本框中使用输入相应的字母，如果输入正确则得1分，否则得0分；  （3）并将用户的得分累计显示在的得分栏。    图4 字母游戏运行效果图  7. 模拟实现兔子散步，如图5所示，启动程序：  （1）单击开始按钮，兔子从左端往右方散步，当走到窗体的边界时，自动结束散步；  （2）单击停止按钮，兔子停止散步。  （3）思考：兔子在水平方向来来回回散步，如何实现？  （4）思考：兔子漫无目的地随机散步，如何实现？    图5 兔子散步  8.请编写程序实现龟兔赛跑。  9.请编程实现打地鼠游戏。 | | | | | | |
| **四、上机调试中出现的错误信息、错误原因及解决办法:**  错误：SaleTikcetRunnable抢票线程重复抢票  错误原因:程序在抢票过程中没有实现线程同步，出现票数重复抢，数据冲突  解决方法：为线程中需要同步的代码加上synchronized  错误：SaleTicketThread抢票线程重复抢票  错误原因：同一个进程的多的线程，访问共同的数据冲突，sale1和sale2线程的同时访问tickets数据时冲突  解决方法：为程序添加synchronized,使其sale1线程的访问tickets数据的时候sale2线程不访问tickets数据，以及为tickets加上static变量，使其实时更改。 | | | | | | |
| **五、上机实验中的收获及心得:**  在做Java类和对象的设计的实验前，我以为不会难做，就像以前做物理实验一样，做完实验，然后两下子就将实验报告做完。直到做完测试实验时，我才明白其实并不容易做，但学到的知识与难度成正比，使我受益匪浅。  　　在做实验的过程中我还学会了 String 类和 StringBuffer 类的使用，掌握 Math 和 Random 类的使用，包装类的使用，日期和时间的处理方法，集合 List、Set 和 Map 的使用，Iterator 迭代器的使用，掌握 for…each 循环的使用；String 型与基本数据类型的转换、String 型与日期型的转换。 Scanner 类的使用。  　　透过这次Java类和对象的设计的实验，使我学到了不少实用的知识，更重要的是，做实验的过程，思考问题的方法，这与做其他的实验是通用的，真正使我们受益匪浅。 | | | | | | |

建议：正文采用五宋体，代码用小五号Times new Roman字体

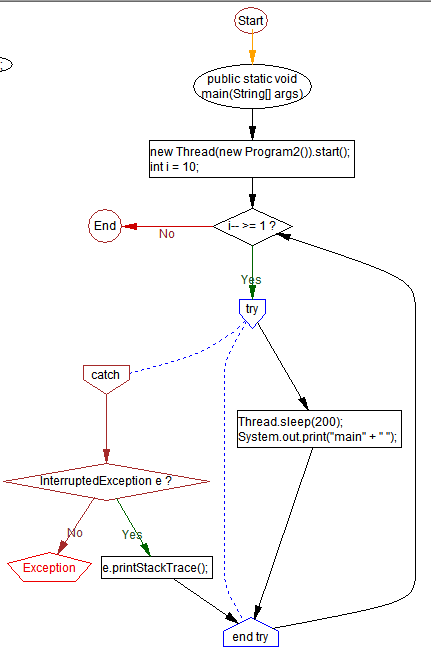
附录：

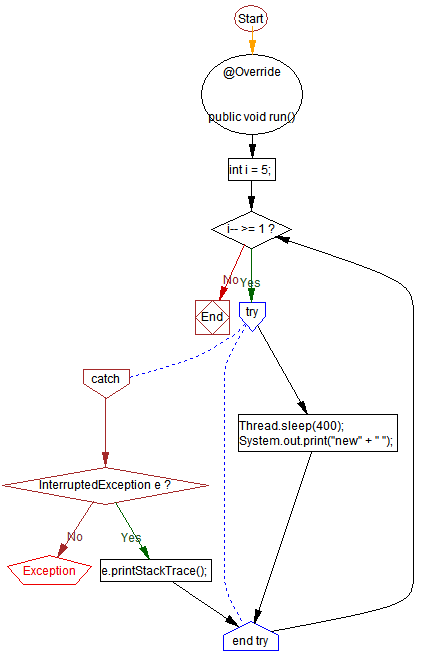
**一、程序流程图**

第一题

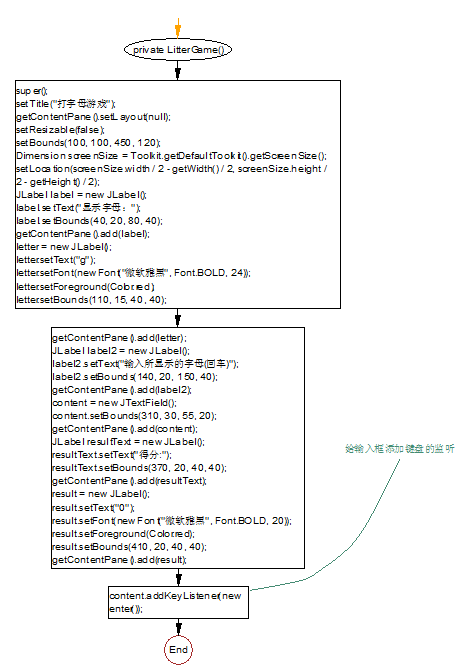


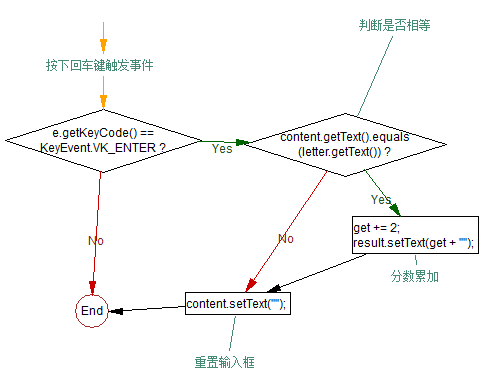
第二题



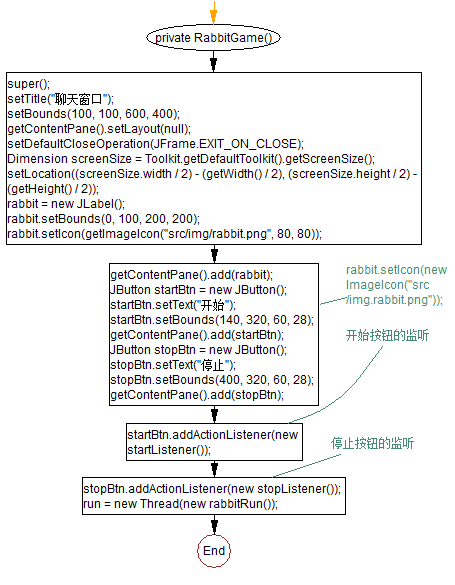


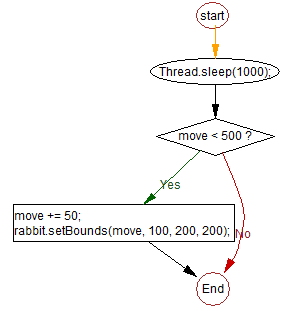
第六题





第七题





1. **主要数据结构和变量说明**

**字母游戏**

//用数组存放大小字母  
private static String[] a = {"a", "b", "c", "d", "q", "z", "w", "x", "s", "e", "r", "t", "f", "y", "h",  
 "n", "u", "j", "m", "k", "o", "p", "A", "B", "C", "D", "Q", "Z", "W", "X", "S", "E","R""T" "F", "Y", "H", "N", "U", "J", "M", "K", "O", "P"};

//初始化分数

private static int get = 0;

//创建线程每两秒钟更新字母

new Thread(new LitterChange()).start();

//给输入框添加键盘的监听

content.addKeyListener(new enter());

//重置输入框

content.setText("");

**兔子移动游戏**

//兔子

private final JLabel rabbit;

//移动量

private static int move = 0;

//控制兔子移动的线程

private final Thread run;

//开始按钮的监听

startBtn.addActionListener(new startListener());

//停止按钮的监听

stopBtn.addActionListener(new stopListener());

1. **上机调试后的源程序及注释**

第一题

package pers.ycy.test6;

public class Program1 {

public static void main(String[] args) {

new Thread\_one().start();

new Thread\_two().start();

}

private static class Thread\_one extends Thread {

@Override

public void run() {

try {

for (int i = 0; i <= 30; i++) {

if (i % 2 != 0) {

Thread.sleep(500);

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

}

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

private static class Thread\_two extends Thread{

@Override

public void run() {

try {

for (int i = 0; i <= 30; i++) {

if (i % 2 == 0) {

Thread.sleep(300);

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

}

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

第二题

package pers.ycy.test6;

public class Program2 implements Runnable {

public static void main(String[] args) {

new Thread(new Program2()).start();

int i = 10;

while (i-- >= 1) {

try {

Thread.sleep(200);

System.out.print("main" + " ");

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

@Override

public void run() {

int i = 5;

while (i-- >= 1) {

try {

Thread.sleep(400);

System.out.print("new" + " ");

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

第三题

package pers.ycy.test6;

public class SaleTicketThread extends Thread {

private String name;

private static int tickets = 10;

public SaleTicketThread() {

}

private SaleTicketThread(String name) {

this.name = name;

}

public int getTickets() {

return tickets;

}

public void setTickets(int tickets) {

this.tickets = tickets;

}

private static final Object d = new Object();

public void run() {

while (true) {

synchronized (d) {

if (tickets > 0) {

System.out.println(name + " 正在销售第 " + tickets-- + " 票");

try {

sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

} else {

break;

}

}

}

}

public static void main(String[] args) {

SaleTicketThread sale1 = new SaleTicketThread("窗口1");

SaleTicketThread sale2 = new SaleTicketThread("窗口2");

sale1.start();

sale2.start();

}

}

第四题

package pers.ycy.test6;

public class SaleTikcetRunnable implements Runnable {

private int tickets = 10;

@Override

public void run() {

while (tickets > 1) {

synchronized (this) {

System.out.println(Thread.currentThread().getName() + " 正在销售第 " + tickets-- + " 票");

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public static void main(String[] args) {

SaleTikcetRunnable ticket = new SaleTikcetRunnable();

Thread sale1 = new Thread(ticket, "窗口1");

Thread sale2 = new Thread(ticket, "窗口2");

sale1.start();

sale2.start();

}

}

第五题

package pers.ycy.test6;

public class ChouJiang implements Runnable {

//所有的奖券

private int[] arr = {10, 5, 20, 50, 100, 200, 500, 800, 2, 80, 300};

private int num = arr.length;

//记录对应的奖券是否抽过，为true表示已抽过、为false表示未抽过

private boolean[] flag = new boolean[arr.length];

@Override

public void run() {

while (true) {

synchronized (this) {

if (num > 0) {

//产生指定范围（与奖券下标相一致）内的随机数

int index = (int) (Math.random() \* 11);

int get = arr[index];

// 代表这张抽奖券抽过了

if (!flag[index]) {

flag[index] = true;

System.out.println(Thread.currentThread().getName() + " 又产生了一个" + get + "元大奖");

num--;

}

} else {

return; //自动结束线程

}

}

}

}

public static void main(String[] args) {

ChouJiang c = new ChouJiang();

Thread t1 = new Thread(c, "抽奖箱1");

Thread t2 = new Thread(c, "抽奖箱2");

t1.start();

t2.start();

}

}

第六题

package pers.ycy.test6;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

public class Game extends JFrame {

//用数组存放大小字母

private static String[] a = {"a", "b", "c", "d", "q", "z", "w", "x", "s", "e", "r", "t", "f", "y", "h",

"n", "u", "j", "m", "k", "o", "p", "A", "B", "C", "D", "Q", "Z", "W", "X", "S", "E", "R", "T",

"F", "Y", "H", "N", "U", "J", "M", "K", "O", "P"};

private static JLabel letter;

private static JLabel result;

//初始化分数

private static int get = 0;

private static JTextField content;

public static void main(String[] args) {

Game game = new Game();

game.setVisible(true);

//创建线程每两秒钟更新字母

new Thread(new LitterChange()).start();

}

private Game() {

super();

setTitle("打字母游戏");

getContentPane().setLayout(null);

setResizable(false);

setBounds(100, 100, 450, 120);

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

setLocation(screenSize.width / 2 - getWidth() / 2, screenSize.height / 2 - getHeight() / 2);

JLabel label = new JLabel();

label.setText("显示字母：");

label.setBounds(40, 20, 80, 40);

getContentPane().add(label);

letter = new JLabel();

letter.setText("g");

letter.setFont(new Font("微软雅黑", Font.BOLD, 24));

letter.setForeground(Color.red);

letter.setBounds(110, 15, 40, 40);

getContentPane().add(letter);

JLabel label2 = new JLabel();

label2.setText("输入所显示的字母(回车)");

label2.setBounds(140, 20, 150, 40);

getContentPane().add(label2);

content = new JTextField();

content.setBounds(310, 30, 55, 20);

getContentPane().add(content);

JLabel resultText = new JLabel();

resultText.setText("得分:");

resultText.setBounds(370, 20, 40, 40);

getContentPane().add(resultText);

result = new JLabel();

result.setText("0");

result.setFont(new Font("微软雅黑", Font.BOLD, 20));

result.setForeground(Color.red);

result.setBounds(410, 20, 40, 40);

getContentPane().add(result);

//给输入框添加键盘的监听

content.addKeyListener(new enter());

}

private static class LitterChange implements Runnable {

@Override

public void run() {

try {

while (true) {

//延时2秒

Thread.sleep(2000);

letter.setText(a[(int) (Math.random() \* a.length)]);

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

private static class enter implements KeyListener {

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

//按下回车键触发事件

if (e.getKeyCode() == KeyEvent.VK\_ENTER) {

//判断是否相等

if (content.getText().equals(letter.getText())) {

//分数累加

get += 2;

result.setText(get + "");

}

//重置输入框

content.setText("");

}

}

@Override

public void keyReleased(KeyEvent e) {

}

}

}

第七题

package pers.ycy.test6;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class RabbitGame extends JFrame {

private final JLabel rabbit;

private static int move = 0;

private final Thread run;

public static void main(String[] args) {

new RabbitGame().setVisible(true);

}

private RabbitGame() {

super();

setTitle("聊天窗口");

setBounds(100, 100, 600, 400);

getContentPane().setLayout(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

setLocation((screenSize.width / 2) - (getWidth() / 2), (screenSize.height / 2) - (getHeight() / 2));

rabbit = new JLabel();

rabbit.setBounds(0, 100, 200, 200);

rabbit.setIcon(getImageIcon("src/img/rabbit.png", 80, 80));

// rabbit.setIcon(new ImageIcon("src/img.rabbit.png"));

getContentPane().add(rabbit);

JButton startBtn = new JButton();

startBtn.setText("开始");

startBtn.setBounds(140, 320, 60, 28);

getContentPane().add(startBtn);

JButton stopBtn = new JButton();

stopBtn.setText("停止");

stopBtn.setBounds(400, 320, 60, 28);

getContentPane().add(stopBtn);

startBtn.addActionListener(new startListener());

stopBtn.addActionListener(new stopListener());

run = new Thread(new rabbitRun());

}

private class startListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

run.start();

}

}

private ImageIcon getImageIcon(String file, int w, int h) {

ImageIcon icon = new ImageIcon(file);

icon.setImage(icon.getImage().getScaledInstance(w, h, Image.SCALE\_DEFAULT));

return icon;

}

private class stopListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

run.suspend();

}

}

private class rabbitRun implements Runnable {

@Override

public void run() {

while (true) {

try {

Thread.sleep(1000);

if (move < 500) {

move += 50;

rabbit.setBounds(move, 100, 200, 200);

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

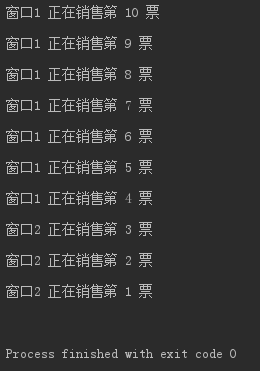
**四、程序运行结果截图及还存在的问题**

单偶数

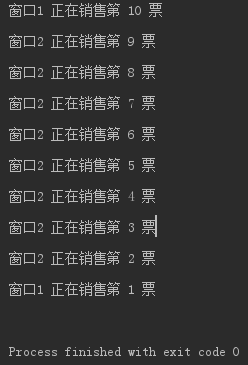
运行结果1

mainPrint

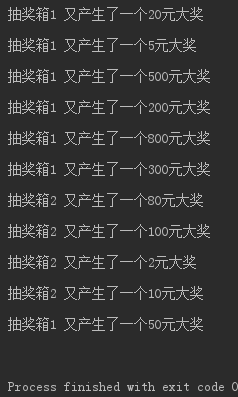
运行结果2



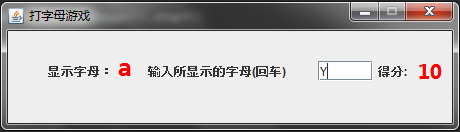
运行结果3



运行结果4



运行结果5



运行结果6



运行结果7