**江西师范大学计算机信息工程学院学生实验报告**

**专业 计算机科学与技术 姓名 李科 学号 202126202107 日期2022 11-9**

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 面向对象 | 实验室名称 | X4313d1 |
| 实验名称 | 实验八 | | |
| 指导教师 | 万中英 | 成绩 |  |

**一、实验目的**（结出本次实验所涉及并要求掌握的知识点）

1. 掌握线程的多种构造方法
2. 掌握继承Thread类与实现Runnable接口创建线程的区别
3. 掌握java线程互斥机制

**二、实验内容**（结出实验内容具体描述）

1. 用四种方法创建线程，输出50以内奇偶数
2. 用多线程计算累加和
3. 借助互斥机制，实现多人竞争打电话
4. 借助线程实现多个用户对同一个账户汇款
5. 类 BufferArea 用于产生一个[1..100]的随机整数，并可取出此数。存数线  
   程类 ThreadPutNumber 将数存入缓冲区（BufferArea）。读数线程类  
   ThreadGetNumber 从缓冲区读取数并判断是否是素数。要求 main 线程最后  
   结束。

**三、算法描述及实验步骤**（用适当的形式表达算法设计思想与算法实现步骤）

1、

方法一：

**class** MyThread **extends** Thread {  
 **boolean** isOdd;  
  
 **public** MyThread(**boolean** isOdd) {  
 **this**.isOdd = isOdd;  
 }  
  
 **public void** run() {  
 **for** (**int** i = 1; i <= 50; i++) {  
 **if** (isOdd && i % 2 == 1) {  
 System.*out*.println("奇数：");  
 System.*out*.println(i + " "); //输出奇数  
 }  
 **if** (!isOdd && i % 2 == 0) {  
 System.*out*.println("偶数：");  
 System.*out*.println(i + " "); //输出偶数  
 }  
 }  
 }  
}

首先创建一个类，然后该类继承Thread类，有一个成员变量isOdd用来判断是否输出奇数还是偶数，构造方法用来传参，然后重写run()方法，使用一个for循环，循环内部判断该类为奇数还是偶数，并输出

方法2：

**class** MyRunnable **implements** Runnable {  
  
 **boolean** isOdd;  
  
 **public** MyRunnable(**boolean** isOdd) {  
 **this**.isOdd = isOdd;  
 }  
  
 @Override  
 **public void** run() {  
 **for** (**int** i = 1; i <= 50; i++) {  
 **if** (isOdd && i % 2 == 1) {  
 System.*out*.println("奇数：");  
 System.*out*.println(i + " "); //输出奇数  
 }  
 **if** (!isOdd && i % 2 == 0) {  
 System.*out*.println("偶数：");  
 System.*out*.println(i + " "); //输出偶数  
 }  
 }  
 }  
}

实现Runnable接口，实现run方法

方法三：

**class** MyThread3 {  
 Thread t;  
  
 **public** MyThread3(**boolean** isOdd) {  
 MyRunnable m = **new** MyRunnable(isOdd);  
 t = **new** Thread(m);  
  
 }  
  
 **public void** start() {  
 t.start();  
 }  
  
 **class** MyRunnable **implements** Runnable {  
 **boolean** isOdd;  
  
 **public** MyRunnable(**boolean** isOdd) {  
 **this**.isOdd = isOdd;  
 }  
  
 @Override  
 **public void** run() {  
 **for** (**int** i = 1; i <= 50; i++) {  
 **if** (isOdd && i % 2 == 1) {  
 System.*out*.println("奇数：");  
 System.*out*.println(i + " "); //输出奇数  
 }  
 **if** (!isOdd && i % 2 == 0) {  
 System.*out*.println("偶数：");  
 System.*out*.println(i + " "); //输出偶数  
 }  
 }  
 }  
 }  
  
}

创建一个类，该类中定义一个内部类MyRunnable去实现Runnable接口，实现run方法，与方法二中一样，然后在外部类中定义一个start方法，其中创建MyRunnable对象，并传参给t=new Thread()中，并且t.start()启动线程。

方法四：

**class** MyThread4 **implements** Runnable {  
 Thread t;  
 **boolean** isOdd;  
  
 **public** MyThread4(**boolean** isOdd) {  
 **this**.isOdd = isOdd;  
 t = **new** Thread(**this**);  
 }  
  
 **public void** start() {  
 t.start();  
 }  
  
 @Override  
 **public void** run() {  
 **for** (**int** i = 1; i <= 50; i++) {  
 **if** (isOdd && i % 2 == 1) {  
 System.*out*.println("奇数：");  
 System.*out*.println(i + " "); //输出奇数  
 }  
 **if** (!isOdd && i % 2 == 0) {  
 System.*out*.println("偶数：");  
 System.*out*.println(i + " "); //输出偶数  
 }  
 }  
 }  
}

创建一个类实现Runnable接口，实现run()方法，在构造方法中创建Thread类对象，定义一个start()方法，其中t.start()启动线程

实验二：

package like.test2;  
  
import java.util.TreeMap;  
  
public class App {  
 public static void main(String[] args) throws InterruptedException {  
 final int max = 1000;  
 int[] a = new int[max];  
 int sum1 = 0, sum2 = 0, step = 10, num = 4;  
 //每个线程每次至多累加 10 个数，共有 4 个线程  
 for (int i = 0; i < max; i++) {//先计算出正确的累加和，以方便对比  
 a[i] = i;  
 sum1 = sum1 + a[i];  
 }  
 Pos p=new Pos(max,step);  
 sum2=SumByThread(p, a,num,step);  
 System.*out*.print("顺序："+sum1+"，并发："+sum2);  
 }  
  
 public static int SumByThread(Pos p,int[] a,int num,int step) throws InterruptedException {  
  
 MyThread[] t = new MyThread[4];  
 for(int i=0;i<num;i++) {  
 t[i] = new MyThread(p,step,a);  
 t[i].start();  
 }  
// for(Thread ti :t) {  
// ti.join();  
// }  
 Thread.sleep(1000);  
 return MyThread.*sum*;  
 }  
}  
  
class Pos {  
 private int max, step,pos;  
 public Pos(int max,int step) {  
 this.max = max;  
 this.step = step;  
 pos = 0;  
 }  
  
 public int getPos() {  
 if(pos < max) {  
 return pos;  
 }  
 return -1;  
 }  
 public void setPos(int pos) {  
 this.pos = pos;  
 }  
  
}  
class MyThread extends Thread {  
 static int *sum*=0;  
 private int step;  
 private Pos p;  
 private int[] a;  
 private boolean flag;  
 public MyThread(Pos p,int step,int[] a) {  
 this.p = p;  
 this.step = step;  
 this.a = a;  
 flag = false;  
 }  
 public void run() {  
 while(true) {  
 for(int i=0;i<step;i++) {  
 synchronized (p) {  
 if(p.getPos() != -1) {  
 *sum*+=a[p.getPos()];  
 System.*out*.println(Thread.currentThread().getName()+" num:"+a[p.getPos()]+ " sum:"+*sum*);  
 p.setPos(p.getPos()+1);  
 } else {  
 return;  
 }  
  
 }  
 }  
 try {  
 this.wait();  
 this.notifyAll();  
 } catch (Exception e) {  
  
 }  
 }  
 }  
  
}

实验三：

**class** DianHuaTing {  
 **private final** String dht = "电话亭";  
}

创建一个电话亭类共享资源，定义一个常量dht

**class** Talkers **extends** Thread {  
  
 **private** DianHuaTing dht;  
 **private** String[] sentences;  
 **private** String name;  
 **public** Talkers(DianHuaTing dht,String name,String[] sentences) {  
 **this**.dht = dht;  
 **this**.sentences = sentences;  
 **this**.name = name;  
 }  
  
 **public void** run() {  
  
 **synchronized**(dht) {  
 System.*out*.print(name +" say: ");  
 **for**(String s: sentences){  
 System.*out*.print( s + " ");  
 }  
 System.*out*.println();  
 }  
  
 }  
  
}

创建一个Talkers类继承Thread类，有三个成员变量，电话亭类对象

，sentence字符串数组，name字符串名称，

构造方法传入三个参数，一个电话亭对象，一个sentence数组，一个name，并进行赋初值。

重写run方法，每个人说的话使用增强for遍历并且输出，并使用synchronized()将每个人说的话锁起来，保证说话连续，synchronized()参数是dht共享资源，保证每一个锁对象都为同一个

实验四：

**class** Bank **implements** Runnable {  
 **private double** money;  
 **private final** Object object;  
 **public** Bank(**double** money) {  
 **this**.money = money;  
 object = **new** Object();  
 }  
 @Override  
 **public void** run() {  
 **for**(**int** i=0;i<3;i++) {  
 **synchronized** (object) {  
 System.*out*.println("银行当前余额为："+ money+","+Thread.*currentThread*().getName()+"第"+(i+1)+"次存入100元");  
 money += 100;  
 System.*out*.println("当前银行总额为:" + money);  
 **try** {  
 Thread.*sleep*(10);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

创建一个Bank类实现Runnable接口，实现run方法，有两个成员变量，一个是money代表账户总共余额，另一个object对象代表一个共享资源，run方法中，使用一个for循环，在循环中输出当前余额，并且余额加100。将这段代码使用synchronized锁起来，这样保证代码连续执行，不被打断。Main中创建Bank对象，作为参数传到Thread即可

实验五：

**class** BufferedArea {  
 **int** buffer,num;  
 Random r;  
 **boolean** isEmpty;  
 **public** BufferedArea(**int** num,**boolean** isEmpty) {  
 r = **new** Random();  
 **this**.num = num;  
 **this**.isEmpty = isEmpty;  
 }  
 **public int** getRandomInt() {  
 **return** r.nextInt(100) + 1;  
 }  
 **public void** setBuffer(**int** buffer) {  
 **this**.buffer = buffer;  
 }  
 **public int** getBuffer() {  
 **return** buffer;  
 }  
  
 **public boolean** isPrime(**int** num) {  
 **if**(num == 1) {  
 **return false**;  
 }  
 **int** i;  
 **for** (i = 2; i < num; i++) {  
 **if** (num % i == 0) {  
 **return false**;  
 }  
 }  
 **return true**;  
 }  
  
  
}  
  
**class** ThreadPutNumber **extends** Thread {  
 BufferedArea ba;  
  
 **public** ThreadPutNumber(BufferedArea ba) {  
 **this**.ba = ba;  
 }  
  
 **public void** run() {  
 **while**(ba.num > 0) {  
  
 **synchronized** (ba) {  
 **if**(!ba.isEmpty) {  
 **try** {  
 ba.wait();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }**else** {  
 **try** {  
 Thread.*sleep*(20);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **int** num = ba.getRandomInt();  
 ba.buffer = num;  
 ba.isEmpty = **false**;  
 System.*out*.println("put number is : " + num);  
 ba.notifyAll();  
 }  
  
 }  
  
 }  
 }  
  
}  
  
**class** ThreadGetNumber **extends** Thread {  
 BufferedArea ba;  
  
 **public** ThreadGetNumber(BufferedArea ba) {  
 **this**.ba = ba;  
 }  
 **public void** run() {  
 **while**(ba.num > 0) {  
  
 **synchronized** (ba) {  
 **if**(ba.isEmpty) {  
 **try** {  
 ba.wait();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }**else** {  
 **try** {  
 Thread.*sleep*(20);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **int** num = ba.getBuffer();  
 ba.isEmpty = **true**;  
 **boolean** isPrime = ba.isPrime(num);  
 **if**(isPrime) {  
 System.*out*.println(num + " is a prime!");  
 }**else** {  
 System.*out*.println(num + " is not a prime!");  
 }  
 ba.num--;  
 ba.notifyAll();  
 }  
  
 }  
  
 }  
 }  
}

首先创建一个缓冲区类BufferedArea，有四个成员变量，int buffer，num分别代表缓冲区的数据，以及存取数据的次数。IsEmpty代表缓存区是否有数据。构造方法中传入num,isEmpty参数并赋初值

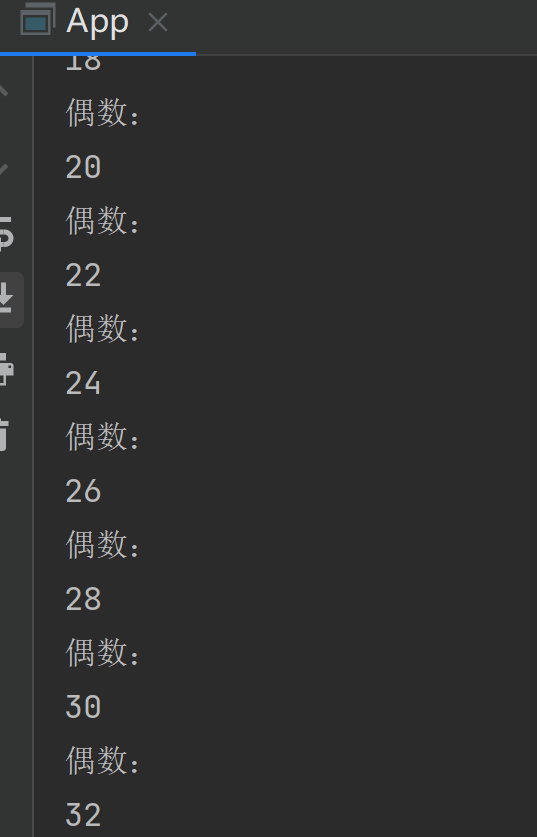
定义获得随机数的方法，存取缓冲区数据的方法，以及判断是否为素数的方法

然后创建ThreadPutNumber类。有一个成员变量BufferedArea，构造方法赋初值。继承Thread类，重写run方法，使用while循环，当ba.num大于0时循环，循环体中使用synchronized(ba)锁对象为ba，如果缓冲区为空，则将产生的随机数放入，并将isEmpty设置为false，使用notifyAll唤醒其他线程，如果缓冲区不为空，则让该线程等待

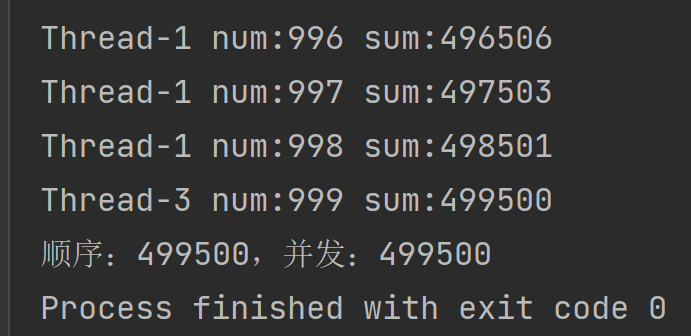
创建ThreadGetNumber类，与ThreadPutNumber类相反，当isEmpty为true等待，false就得到缓冲区数据，ba.num--,唤醒其他线程

**四、调试过程及运行结果**（详细记录在调试过程中出现的问题及解决方法。记录实验执行的结果）

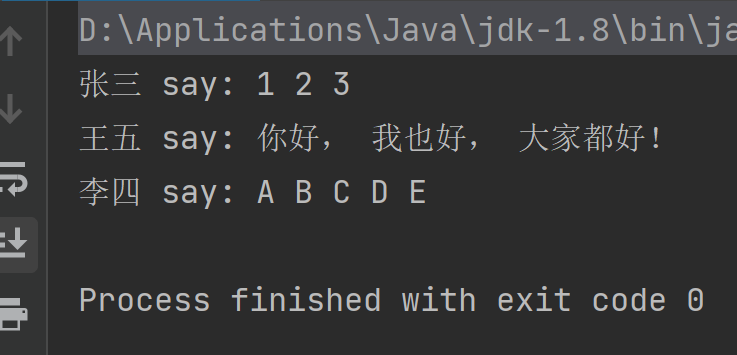
1、

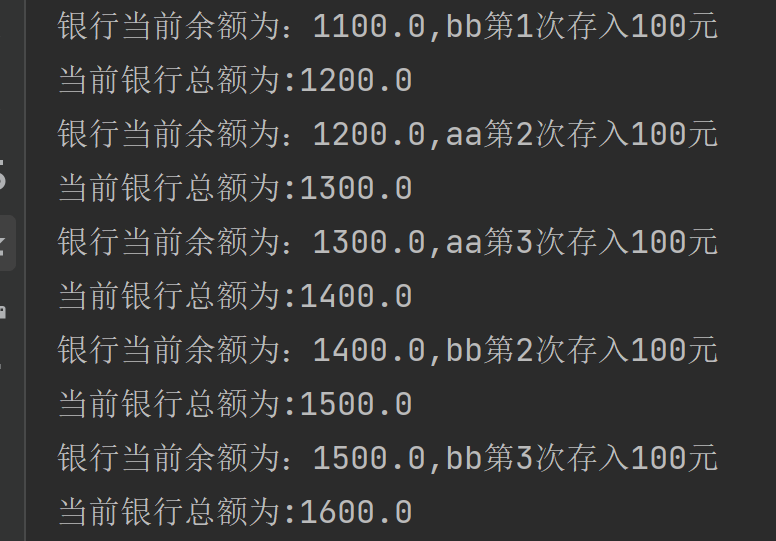


2、



3、



4、

5、

