5-1 利用线程间通信解决单缓冲的生产-消费问题 分数 20

作者 huangcheng 单位 西南石油大学

以下程序模拟了"使用线程间通信解决单缓冲的生产-消费问题"的过程。其中，缓冲区只容纳一个字符，生产者按照 ABCD 的顺序将字符放入缓冲区，消费者从缓冲区取出字符。请阅读程序，并完成填空。

```
//类1：共享缓冲区
class SharedData {
    private char c;//单缓冲(只能放 1 个产品)
    private boolean isProduced=false;//(标志)是否有产品在缓冲
    //方法1：放产品到缓冲区
    public synchronized void push(char c) {
        if(this.isProduced) {
            try {
                System.out.println("产品"+this.c+"还未消费，不能生产");
                wait()  4 分 ;
            } catch ( InterruptedException e  4 分 ) {
               System.out.println ( e ) ;
            } catch ( Exception e ) {
               System.out.println ( e ) ;
            }
        }
        this.c=c;
        this.isProduced=true;
        //此行代码与填空 3 相同
        System.out.println("生产了产品"+c+"，并通知了消费者");
    }
    //方法2：从缓冲区取产品
    public synchronized char get() {
        if(!this.isProduced) {
            try {
                System.out.println("还未生产，不能消费");
                //此行代码与填空 1 相同
            } catch ( //此处代码与填空 2 相同 ) {
            }
        }
        this.isProduced=false;
        notify()  4 分 ;
        System.out.println("消费了产品"+c+"，并通知了生产者");
        return this.c;
    }
}
//类2：消费者(线程)
class Consumer extends Thread {
    private SharedData s;
    public Consumer(SharedData s) {
        this.s = s;
    }
    public void run() {
        char ch;
        do {
            try {
                Thread.sleep((int)Math.random()*3000);//随机延时
            } catch ( //此处代码与填空 2 相同 ) {
            }
            ch = s.get()  4 分 ;//从共享缓冲区取产品消费
        }while(ch!='D');
    }
}
//类3：生产者(线程)
class Producer extends Thread {
    private SharedData s;
    public Producer(SharedData s) {
        this.s = s;
    }
    public void run() {
        for(char ch='A';ch<='D';ch++) {
            try {
                Thread.sleep((int)Math.random()*3000);//随机延时
            } catch ( //此处代码与填空 2 相同 ) {
                e.printStackTrace();
            }
            s.push(ch)  4 分 ;//产品入共享缓冲区
        }
    }
}
//主类：测试程序
public class Main {
    public static void main(String[] args) {
        SharedData s=new SharedData();//共享缓冲区
        Producer p=new Producer(s);
        Consumer c=new Consumer(s);
        p.start();
        c.start();
    }
}
```

答案正确：20 分

5-2 利用线程同步机制执行正确的计数 分数 6

作者 huangcheng 单位 西南石油大学

当用多个线程处理共享变量时，线程中对共享变量的处理代码应用同步机制进行保护，才能保证处理的正确性。

```
import java.util.*;
class BackCounter implements Runnable{
    private int count=100;  //线程共享变量，对它的处理必须用同步机制进行保护
    public int getCount() { return count; }//返回变量值
    //线程体
    public void  run() {
        for(int i=10;i>0;i--) { //变量值递减 10
            synchronized (this)  2 分  { //以下代码在处理共享变量，需要同步机制保护
                if( count<=0 ) break;
                count--;
            }
            try { Thread.sleep(10); } catch ( InterruptedException e ) { }//模拟延时 10 毫秒
        }
    }//线程体结束
}
public class Main {
    public static void main(String[] args) throws InterruptedException {//某些线程方法会抛出检查型异常
        ArrayList<Thread> lt=new ArrayList<Thread>();
        BackCounter bc=new BackCounter();//创建实现类对象
        lt.add(new Thread(bc));//创建线程对象
        lt.add(new Thread(bc));
        for (Thread th:lt)  th.start()  2 分  ; //启动线程
        for (Thread th:lt)  th.join()  2 分  ;  //等待线程结束
        System.out.println(bc.getCount());
    }
}
```