Java 程序设计 LAB10

1. 简答题

简述程序，进程，线程的概念

1．程序是含有指令和数据的文件，被存储在磁盘或其他数据存储设备中，即程序是静态的代码。
2．进程是指运行中的应用程序，每一个进程都有自己独立的内存空间，对一个应用程序可以同时启动多个进程。
3．线程是指进程中的一个执行流程，有时也称为执行情景。一个进程可以由多个线程组成，即在一个线程中可以同时运行多个不同的线程，他们分别执行不同的任务，在同一块地址空间中工作。

2. 简答题

产生死锁的四个条件是什么？

1．互斥使用，即当资源被一个线程使用(占有)时，别的线程不能使用
2．不可抢占，资源请求者不能强制从资源占有者手中夺取资源，资源只能由资源占有者主动释放。
3．请求和保持，即当资源请求者在请求其他的资源的同时保持对原有资源的占有。
4．循环等待，即存在一个等待队列：P1 占有 P2 的资源，P2 占有 P3 的资源，P3 占有 P1 的资源。这样就形成了一个等待环路。

3. 简答题

创建线程的两种方式分别是什么？各有什么优缺点。

方法一：扩展 java.lang.Thread 类
优点：编写简单，在 Tread 类中有一些静态方法和实例方法帮助操作。并且如果需要访问当前线程，无需使用 Thread.currentThread()方法，直接使用 this，即可获得当前线程。
缺点：因为该类继承了 Tread 类，并且一个类只能继承一个类，故该类不能再继承其他类了。
方法二：实现 Runnable 接口
优点：线程类只是实现了该接口，还可以继承其他类。在这种创建方式下，多个线程共享同一个目标对象，适合多个线程处理同一份资源的情况。
缺点：编程较为复杂，如果需要访问当前线程，需使用 Thread.currentThread()方法获得当前线程。

4. 判断题

(1)进程是线程 Thread 内部的一个执行单元，它是程序中一个单一顺序控制流程。
(2)一个进程可以包括多个线程。两者的一个主要区别是：线程是资源分配的单位，而进程是 CPU 调度和执行的单位。
(3)线程可以用 yield 使低优先级的线程运行。
(4)当一个线程进入一个对象的一个 synchronized 方法后，其它线程可以再进入该对象的其它同步方法执行。
(5) notify 是唤醒所在对象 wait pool 中的第一个线程。

(1)错，应该是线程是进程 Thread 内部的一个执行单元。

(2)错，无论是进程还是线程，都是轮流获得 CUP 的使用权。
(3)错，yield()方法会让当前运行的线程进入可运行池中，并使处于就绪状态的并且具有相同或更高优先级的其他线程运行；若没有相同或更高优先级的线程，yield()方法什么也不做。
(4)错，其它线程试图执行带有 synchronized 标记的代码块时，它必须获得对应的锁。而此时这个锁被占用，那么这个线程会被放到锁池中，并进入阻塞状态。所以其它线程不可以再进入该对象的其它同步方法执行。
(5)错，notify()唤醒的是其中任意一个线程。

5. 程序输出题

(1)请写出上述程序的输出
(2)用 synchronized 修饰 run()的作用是什么？
(3)标号【1】处 sleep 的作用是什么？如果改为 wait(100);输出会发生改变吗，为什么？

(1)

```
SyncThread1:0
SyncThread1:1
SyncThread1:2
SyncThread1:3
SyncThread1:4
SyncThread2:5
SyncThread2:6
SyncThread2:7
SyncThread2:8
SyncThread2:9
```

(2)为了保证这两个线程能正常执行原子操作，用 synchronized 修饰 run()的原因是将 run()方法看作一个原子操作，作为同步代码块。当一个线程在执行这个 run()方法时，若另一个线程试图执行带有 synchronized 标记的 run()时，它必须获得对应的锁。而此时这个锁被占用，那么这个线程会被放到锁池中，并进入阻塞状态，等待另一个执行完成。
(3)sleep()方法的作用是导致此线程暂停执行指定时间，试图给其他线程机会，，但是监控状态依然保持，到规定的时间后会自动恢复，即调用 sleep 不会释放对象锁。因为用 synchronized 修饰 run()方法，所以还是只能由该线程继续执行。若改为 wait(100);输出会发生改变，如下：

```
SyncThread1:0
SyncThread2:1
SyncThread1:2
SyncThread2:3
SyncThread2:4
SyncThread1:5
SyncThread1:6
SyncThread2:7
SyncThread2:8
SyncThread1:9
```

这是因为线程在执行同步代码块的过程中，执行了锁所属对象的 wait()方法，这个线程会释放锁，进入的对象的等待池中，而另一个线程就会获得锁，执行同步代码块。

6. 程序补全题

(1)补全标号处的代码

(2)简述上述程序的功能

(1)

```
1.new Thread(st, "售票口" + i).start();
2.while (ticket > 0)
3.synchronized (this)
4.Thread.sleep(500);// 线程休眠 0.5 秒
```

(2)上述代码模拟了售票情况，一共有 100 张票要售出，共有六个售票点同时售票（六个线程模拟），直到票全部售完为止。

7. 程序补全题

(1)补全标号处的代码

(2)详细说明上述程序的功能**(三线程交替打印 ABC)**

(3)主函数 main 中的 Thread.*sleep*(100)语句不能省略，请简述原因。

(4)主函数 main 中的 Thread.*sleep*(100)语句全部去掉后程序可能出现死锁吗？试举例说明。

(1)

```
1.self.notifyAll();//【1】
2.break;//【2】
3.prev.wait();//【3】
```

(2)程序的功能是交替打印 ABC。为了实现这一目的，这个问题转化为三线程间的同步唤醒操作，主要的目的就是 ThreadA->ThreadB->ThreadC->ThreadA 循环执行三个线程，确定唤醒、等待的顺序，所以每个线程必须拥有两个对象锁进行相关的 notifyAll()和 wait()操作。在如下这段代码中：

```
synchronized (prev){
    synchronized (self){
        System.out.print(name);
        count--;
        self.notifyAll();//【1】
    }
    try{
        if(count==0)
            break;//【2】
        else
         prev.wait();//【3】
    }catch (InterruptedException e){
        e.printStackTrace();
    }
}
```

threadA 的执行：首先 threadA 线程依次获得 prev 锁和 self 锁，即 c 和 a 对象的锁，并进入 a 对象的同步代码块，打印 A，更新 count 的值，执行 a 对象的 notifyAll()方法，唤醒在 a 对象等待池中的所有线程（此时的 a 对象锁并没有释放，并且由于此时等待池中没有任何线程，所以该方法什么也不执行），执行完 a 对象同步代码块后，释放 a 对象锁。线程

仍然在执行 c 对象的同步代码块，当线程执行到 prev.wait()时(c.wait())，线程释放 c 对象锁并且进入等待池中，然后 Java 虚拟机会在 wait()对象锁的线程中选一个线程获得锁，转到就绪状态。
threadB 的执行（threadA 在 c 对象等待池中）：首先 threadB 线程依次获得 prev 锁和 self 锁，即 a 和 b 对象的锁，并进入 b 对象的同步代码块，打印 B，更新 count 的值，执行 b 对象的 notifyAll()方法，唤醒在 b 对象锁池中的所有线程（此时的 b 对象锁并没有释放，并且由于此时锁池中没有任何线程，所以该方法什么也不执行），执行完 b 对象同步代码块后，释放 b 对象锁。线程仍然在执行 a 对象的同步代码块，当线程执行到 prev.wait()时(a.wait())，线程释放 a 对象锁并且进入等待池中，然后 Java 虚拟机会在 wait()对象锁的线程中选一个线程获得锁，转到就绪状态。
threadC 的执行（threadA 在 c 对象等待池中，threadB 在 a 对象等待池中）：首先 threadC 线程依次获得 prev 锁和 self 锁，即 b 和 c 对象的锁，并进入 c 对象的同步代码块，打印 C，更新 count 的值，执行 c 对象的 notifyAll()方法，唤醒在 c 对象锁池中的所有线程（即唤醒 threadA，此时的 c 对象锁并没有释放），执行完 c 对象同步代码块后，释放 c 对象锁。线程仍然在执行 b 对象的同步代码块，当线程执行到 prev.wait()时(b.wait())，线程释放 b 对象锁并且进入等待池中，然后 Java 虚拟机会在 wait()对象锁的线程中选一个线程获得锁，转到就绪状态。
由此可见 threadA 拥有 a 对象锁和 c 对象锁，并唤醒 a 对象等待池中的 threadB 和使 threadA 在 c 对象等待池中等待；threadB 拥有 a 对象锁和 b 对象锁，并唤醒 b 对象等待池中的 threadC 和使 threadB 在 a 对象等待池中等待；threadC 拥有 b 对象锁和 c 对象锁，并唤醒 c 对象等待池中的 threadA 和使 threadC 在 b 对象等待池中等待；如此重复。
(3)这样可以保证三个线程 threadA，threadB，threadC 的执行顺序，从而保证 ABC 的顺序。
(4)会出现死锁。按照线程竞争的顺序来看，可能会出现在 synchronized (prev)处，线程 threadA 等待 c,threadB 等待 a,threadC 等待 b 的死锁情况。

8. 编程题

oo/08

9. 编程题

oo/09