

Lab 10 Assignment

班级：202111

学号：19241027

姓名：胡峻诚

Question1 简述程序，进程，线程的概念 简答

- 程序是含有指令和数据文件，被存储在磁盘或其他的数据存储设备中，也就是说程序是静态的代码。
- 进程是程序的一次执行过程，是系统运行程序的基本单位，因此进程是动态的。系统运行一个程序即是一个进程从创建，运行到消亡的过程。简单来说，一个进程就是一个执行中的程序，它在计算机中一个指令接着一个指令地执行着，同时，每个进程还占有某些系统资源如CPU时间，内存空间，文件，文件，输入输出设备的使用权等等。换句话说，当程序在执行时，将会被操作系统载入内存中。
- 线程与进程相似，但线程是一个比进程更小的执行单位。一个进程在其执行的过程中可以产生多个线程。与进程不同的是同类的多个线程共享同一块内存空间和一组系统资源，所以系统在产生一个线程，或是在各个线程之间作切换工作时，负担要比进程小得多，也正因为如此，线程也被称为轻量级进程。

Question2 产生死锁的四个条件是什么？ 简答

1. **互斥条件**：指进程对所分配到的资源进行排它性使用，即在一段时间内某资源只由一个进程占用。如果此时还有其它进程请求资源，则请求者只能等待，直至占有资源的进程用毕释放。
2. **请求和保持条件**：指进程已经保持至少一个资源，但又提出了新的资源请求，而该资源已被其它进程占有，此时请求进程阻塞，但又对自己已获得的其它资源保持不放。
3. **不可剥夺条件**：指进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。
4. **环路等待条件**：指在发生死锁时，必然存在一个进程—资源的环形链，即进程集合{P0, P1, P2, ..., Pn}中的P0正在等待一个P1占用的资源；P1正在等待P2占用的资源；.....；Pn正在等待已被P0占用的资源。

Question3 创建线程的两种方式分别是什么？各有什么优缺点 简答

一是继承 `Thread` 类，并重写 `run()` 方法；

二是实现 `Runnable` 接口，覆盖接口中的 `run()` 方法，并把 `Runnable` 接口的实现扔给 `Thread`。

优缺点：

1. 采用继承Thread类方式：
 - 优点：编写简单，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：因为线程类已经继承了Thread类，所以不能再继承其他的父类。
2. 采用实现Runnable接口方式：
 - 优点：线程类只是实现了Runnable接口，还可以继承其他的类。在这种方式下，可以多个线程共享同一个目标对象，所以非常适合多个相同线程来处理同一份资源的情况，从而可以将CPU代码和数据分开，形成清晰的模型，较好地体现了面向对象的思想
 - 缺点：编程稍微复杂，如果需要访问当前线程，必须使用Thread.currentThread()方法。

Question4 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。
x，反了吧？
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。
x，进程不负责执行，引入线程后，进程只负责资源管理。
3. 线程可以用 `yield` 使低优先级的线程运行。
x，显然高优先级先运行。
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。
x，该对象的 `synchronized` 方法不能进入了，但是其他非 `synchronized` 方法仍然可以访问。
5. `notify` 是唤醒所在对象 `wait pool` 中的第一个线程。
x，是唤醒任意一个线程。

Question5 程序输出简答

1. 请写出上述程序的输出

```

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

```

2. 用 `synchronized` 修饰 `run()` 的作用是什么？
给 `run()` 这个方法加锁，使得同一时刻只能有一个线程调用 `syncThread` 的 `run` 方法，防止两个线程交替输出。
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？
`sleep` 的作用是使得该线程等待而停止运行，不释放对象的锁。若改为 `wait`，输出会发生改变，因为 `wait` 释放对象的锁。

```

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

```

Question6 程序补全题

```

1 | public class ThreadPrint {
2 |     public static void main(String[] args) throws InterruptedException {
3 |         Object a = new Object();
4 |         Object b = new Object();
5 |         Object c = new Object();
6 |         Thread8 threadA = new Thread8("A", c, a);
7 |         Thread8 threadB = new Thread8("B", a, b);
8 |         Thread8 threadC = new Thread8("C", b, c);
9 |         new Thread(threadA).start();
10 |         Thread.sleep(100);
11 |         new Thread(threadB).start();
12 |         Thread.sleep(100);
13 |         new Thread(threadC).start();
14 |         Thread.sleep(100);
15 |     }
16 | }
17 |
18 | class Thread8 implements Runnable {
19 |     private String name;
20 |     private Object prev;
21 |     private Object self;
22 |
23 |     public Thread8(String name, Object prev, Object self) {
24 |         this.name = name;
25 |         this.prev = prev;
26 |         this.self = self;
27 |     }
28 |
29 |     @Override
30 |     public void run() {

```

```

31         int count = 10;
32         while (count > 0) {
33             synchronized (prev) {
34                 synchronized (self) {
35                     System.out.print(name);
36                     count--;
37                     self.notify();
38                 }
39             try {
40                 if (count == 0)
41                     break;
42                 else
43                     prev.wait();
44             } catch (InterruptedException e) {
45                 e.printStackTrace();
46             }
47         }
48     }
49 }
50 }

```

1. 补全标号处的代码
2. 详细说明上述程序的功能

输出"ABC"十次

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

省略了不能保证A、B、C的顺序。

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。

有可能会发生死锁，如果三个线程同时进入，那么A获得c的锁，B获得a的锁，C获得b的锁，他们就会一直等待其他线程释放所需要的锁，发生死锁。

Question7 创建两个线程，其中一个输出1-52，另外一个输出A-Z。输出格式要求： `编程`

