1. 题：两个线程交替打印0~100的奇偶数：

偶线程：0

奇线程：1

偶线程：2

奇线程：3

思路解析：控制两个线程的执行顺序，偶线程执行完之后奇数线程执行，这个有点像通知机制，偶线程通知奇线程，奇线程再通知偶线程。用wait和notify对其进行实现：

public class AlternatePrinting {  
 static class PrintThread implements Runnable{  
 static int value = 0;  
  
 @Override  
 public void run() {  
 while (value <= 100){  
 synchronized (PrintThread.class){  
 System.out.println(Thread.currentThread().getName()+":"+value++);  
 PrintThread.class.notify();  
 try {  
 PrintThread.class.wait(1000);  
 }catch (InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 new Thread(new PrintThread(),"偶数").start();  
 new Thread(new PrintThread(),"奇数").start();  
 }  
}

1. 题：通过N个线程顺序循环打印从0至100，如给定N=3则输出:

thread0: 0

thread1: 1

thread2: 2

thread0: 3

thread1: 4

.....

答：

public class ThreadLoopPrint {  
 public static final int THREAD\_NUMBER = 3;  
 public static final int MAX\_COUNT = 100;  
 private static int count = 0;  
 public static void main(String[] args) {  
 Semaphore[] semaphores = new Semaphore[THREAD\_NUMBER];  
 for (int i = 0; i < THREAD\_NUMBER; i++){  
 if (i == 0){  
 semaphores[i] = new Semaphore(1);  
 }else {  
 semaphores[i] = new Semaphore(0);  
 }  
 }  
 for (int i = 0; i < THREAD\_NUMBER; i++){  
 new Thread(new TestThread(semaphores,i)).start();  
 }  
 }  
 public static class TestThread implements Runnable{  
 private Semaphore[] semaphores;  
 private int number = 0;  
 public TestThread(Semaphore[] semaphores, int number) {  
 this.semaphores = semaphores;  
 this.number = number;  
 }  
 @Override  
 public void run() {  
 try {  
 while (true){  
 semaphores[number].acquire();  
 if (count >= MAX\_COUNT){

System.exit(0);  
 break;  
 }  
 System.out.println(Thread.currentThread().getName()+":"+count);  
 count++;  
 int current = number + 1;  
 if (current >= THREAD\_NUMBER){  
 current = 0;  
 }  
 semaphores[current].release();  
 }  
 }catch (InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
 }  
}

排序算法：java的8大排序算法：https://www.cnblogs.com/morethink/p/8419151.html、https://blog.csdn.net/jackesy/article/details/80135033