**Override 和 Overload的区别**：

|  |
| --- |
| Override:跟面向对象有关，是多态的具体体现，注意被覆盖的函数名称，参数类型，返回类型都相同，简单的说就是子类重写了父类的方法。  Overload: 跟面向对象无关，是指编写一个与已有函数同名但是参数列表不一样的函数，其是一种语法规则 |

**进程和线程的区别**：

|  |
| --- |
| 1: 不同的操作系统资源管理方式  进程是系统进行资源分配和调度的一个独立单位，线程是进程的一个实体,是CPU调度和分派的基本单位  2: 内存占用的不同：  进程有独立的地址空间，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间。  3:效率问题：  进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率  4:健壮型问题：  多进程的程序要比多线程的程序健壮，因为单独的内存单元，不容易引起并发性的操作  5:总之：  线程和进程在使用上各有优缺点：线程执行开销小，但不利于资源的管理和保护；而进程正相反。同时，线程适合于在SMP机器上运行，而进程则可以跨机器迁移。  6:形象的例子：（工厂和车间的例子） |

**实现一个栈(stack)**:用数组

|  |
| --- |
| 1: 底层用数组来实现，构造函数包括有参和无参的构造函数  2: 类的成员变量包括，Object[] arr, int top, int maxSize; => 注意数组的类型是object,表示什么类型都能装  3: 实现pop,push的操作都是基于top成员变量的设计，  pop() => top-1  push()=> top+1  但是要注意边界条件的检测，初始化的时候，top为-1. |

**实现一个队列(queue)**:用数组

|  |
| --- |
| 思想与上面类似，不过需要两个index来标记：  front 和 rear,代表队列的第一个元素和最后一个元素，当然操作也复杂一点，  例如超过maxSize的情况下，我们考虑用mod来处理：  this.rear = (this.rear+1) % this.maxSize; => 入队列  this.front = (this.front+1) % this.maxSize; =>出队列 |