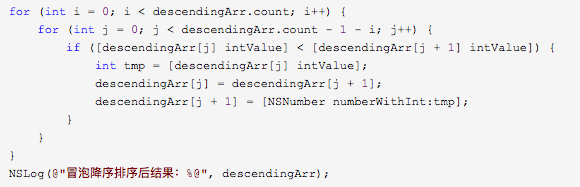
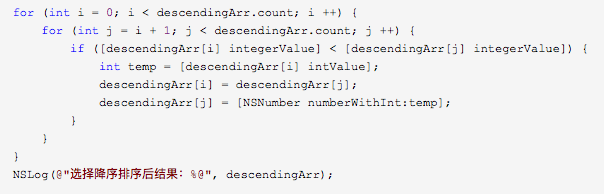
我不会，但我可以学

1）排序算法：

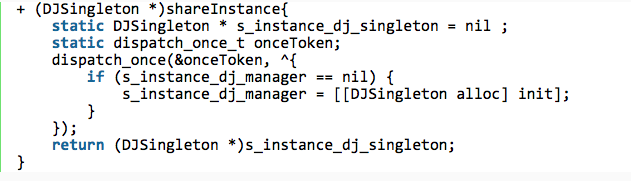
1. 冒泡排序：重复的比较数组中相邻的两个元素。如果一个元素比另一个元素大（小），那么就交换这两个元素的位置。重复这一比较直至最后一个元素，重复n-1趟，每一趟比较n-j次，j是已经排序好的元素个数（j与j+1交换）



2. 选择排序：从数组的第i个元素开始到第n个元素，寻找最小的元素。（具体过程为:先设arr[i]为最小，逐一比较，若遇到比之小的则交换）（i与j交换）



2）单例：一个类只有一个实例对象，并且提供一个全局的访问入口访问这个实例（例如,一个数据库,只需要全局统一的读取,写入操作.不要多个实例去读写）

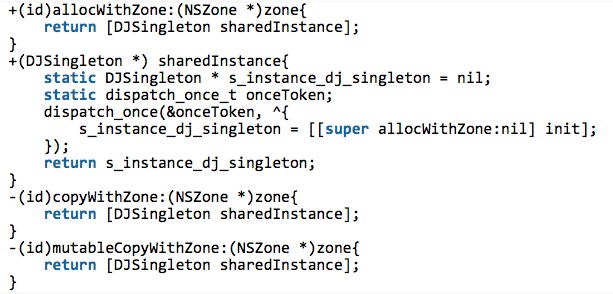


上面的方法还不是很完美，当使用alloc的方式创建对象。这样会导致外面使用的时候，依旧可能创建多个实例。

由Objective-C 在对象创建的时候，无论是alloc还是new，都会调用到 allocWithZone方法。在通过拷贝的时候创建对象时，会调用到-(id)copyWithZone:(NSZone \*)zone，

-(id)mutableCopyWithZone:(NSZone \*)zone方法。因此，可以重写这些方法(遵守NSCoping和NSMutableCopying)，让创建的对象唯一。

[[super allocWithZone:nil] init] 而不是 [[super alloc] init]



销毁单例进行重建：dispatch\_once\_t参数的初始值就是0l，只需要重置dispatch\_once\_t参数(=0l)及实例参数(=nil)

3）多肽

条件：子类继承父类、子类有重写父类的方法、父类指针指向子类对象

原理：动态类型识别，动态绑定，动态加载

4）KVO（Key-Value Observing 键值监听）的实现原理与具体应用

指定一个被观察对象(例如A类)，当对象某个属性(例如A中的字符串name)发生更改时，对象会获得通知，并作出相应处理。

- (void)observeValueForKeyPath:(NSString \*)keyPath ofObject:(id)object change:(NSDictionary<NSKeyValueChangeKey,id> \*)change context:(void \*)context

一般调用setter方法，首先会通过对象的isa指针找到对象，然后在对象中找到setter方法，然后找方法对应的实现

5）iOS开发中的几种设计模式: 代理模式、观察者模式、MVC模式、单例模式、策略模式、工厂模式、MVVM

1.代理模式：tableview的 数据源delegate、和协议进行搭配使用

2.观察者模式：Notification通知中心、kvo：键值对改变通知的观察者

3.MVC模式：

4.单例：确保程序运行期某个类只有一个实例，[UIApplication sharedApplication]、[ NSBundle mainBunle]、[NSFileManager defaultManager]、[NSNotificationCenter defaultCenter]、[NSUserDefaults defaultUserDefaults]

5.策略模式：

6）Runtime运行时

runtime是一套底层的c语言API，实际上,平时我们编写的oc代码,底层都是基于runtime实现的;

只有在程序运行时，才会去确定对象的类型，并调用类与对象相应的方法。利用runtime机制让我们可以在程序运行时动态修改类的具体实现、包括类中的所有私有属性、方法（动态特性）。 runtime就是OC辛苦的幕后工作人员。

runtime的几种使用方法：

a:获取 类的所有变量；

b:获取 类的所有方法；

c:改变 类的私有变量 的值；

d:为 category类增加新属性；

e:为 类添加一个方法；

f:交换 类的2个方法的功能；

7）NSRunLoop的实现机制,及在多线程中如何使用

NSRunLoop是IOS消息机制的处理模式,

1.主要作用: 控制NSRunLoop里面线程的执行和休眠，在有事情做的时候使当前NSRunLoop控制的线程工作，没有事情做让当前NSRunLoop的控制的线程休眠。

2. NSRunLoop 就是一直在循环检测, 从线程start到线程end，检测

inputsource(如点击，双击等操作)异步事件，检测timesource同步事件，检测到输入源会执行处理函数，首先会产生通知，corefunction向线程添加runloop observers来监听事件，意在监听事件发生时来做处理。

3. runloopmode是一个集合，包括监听：事件源，定时器，以及需通知的runloop observers

4.只有在为你的程序创建次线程的时候，才需要运行runloop。对于程序的主线程而言，runloop是关键部分。Cocoa提供了运行主线程runloop的代码同时也会自动运行runloop。IOS程序UIApplication中的run方法在程序正常启动的时候就会启动runloop。如果你使用xcode提供的模板创建的程序，那你永远不需要自己去启动runloop

5.在多线程中，你需要判断是否需要runloop。如果需要runloop，那么你要负责配置runloop并启动。你不需要在任何情况下都去启动runloop。比如，你使用线程去处理一个预先定义好的耗时极长的任务时，你就可以毋需启动runloop。Runloop只在你要和线程有交互时才需要 。

8) Block

1.block其实就是一个代码块，把你想要执行的代码封装在这个代码块里，等到需要的时候再去调用

2.block本身是像对象一样可以retain，和release。但是，block在创建的时候，它的内存是分配在栈(stack)上，而不是在堆(heap)上。他本身的作于域是属于创建时候的作用域，一旦在创建时候的作用域外面调用block将导致程序崩溃。为了能够在block的声明域外使用,所以要把block拷贝（copy）到堆.

3.使用retain也可以，但是block的retain行为默认是用copy的行为实现的，

4. 在Block的{}体内，是不可以对外面的变量进行更改的,使用\_\_block就能改变block块中变量的值

5. \_\_block和\_\_weak区别:

\_\_block不管是ARC还是MRC模式下都可以使用，可以修饰对象，还可以修饰基本数据类型。 \_\_weak只能在ARC模式下使用，也只能修饰对象（NSString），不能修饰基本数据类型（int）。 \_\_block对象可以在block中被重新赋值，\_\_weak不可以。

总结：1.当block里面会有b类相关的参数要回调回去的时候，属性用copy修饰，将其拷贝到堆里面，这样即便栈释放掉了，b类的指针也在堆中存在，能够成功的回调回去。

2. 如果语法块仅仅是执行而不再回调回去了， 比如操作某个数据库，修改某个单利类的属性，发送某个通知之类的，则可以用weak来修饰。这样在block执行后就可以马上释放掉， 节约内存