<http://blog.csdn.net/c349126799/article/details/38413843>

共90题

1.Difference between shallow copy and deep copy? 浅复制和深复制的区别？  答案：浅层复制：只复制指向对象的指针，而不复制引用对象本身。 深层复制：复制引用对象本身。 意思就是说我有个A对象，复制一份后得到A\_copy对象后，对于浅复制来说，A和A\_copy指向的是同一个内存资源，复制的只不过是是一个指针，对象本身资源 还是只有一份，那如果我们对A\_copy执行了修改操作,那么发现A引用的对象同样被修改，这其实违背了我们复制拷贝的一个思想。深复制就好理解了,内存中存在了 两份独立对象本身。 用网上一哥们通俗的话将就是： 浅复制好比你和你的影子，你完蛋，你的影子也完蛋 深复制好比你和你的克隆人，你完蛋，你的克隆人还活着。

2.What is advantage of categories? What is difference between implementing a category and inheritance? 类别的作用？继承和类别在实现中有何区别？ 答案：category 可以在不获悉，不改变原来代码的情况下往里面添加新的方法，只能添加，不能删除修改。 并且如果类别和原来类中的方法产生名称冲突，则类别将覆盖原来的方法，因为类别具有更高的优先级。

类别主要有3个作用：

 (1)将类的实现分散到多个不同文件或多个不同框架中。

 (2)创建对私有方法的前向引用。

 (3)向对象添加非正式协议。

继承可以增加，修改方法，并且可以增加属性。

3.Difference between categories and extensions? 类别和类扩展的区别。 答案：category和extensions的不同在于 后者可以添加属性。另外后者添加的方法是必须要实现的。 extensions可以认为是一个私有的Category。

4.Difference between protocol in objective c and interfaces in java? obc中的协议和java中的接口概念有何不同？  答案：OBC中的代理有2层含义，官方定义为 formal和informal protocol。前者和Java接口一样。 informal protocol中的方法属于设计模式考虑范畴，不是必须实现的，但是如果有实现，就会改变类的属性。 其实关于正式协议，类别和非正式协议我很早前学习的时候大致看过，也写在了学习教程里 “非正式协议概念其实就是类别的另一种表达方式“这里有一些你可能希望实现的方法，你可以使用他们更好的完成工作”。 这个意思是，这些是可选的。比如我门要一个更好的方法，我们就会申明一个这样的类别去实现。然后你在后期可以直接使用这些更好的方法。 这么看，总觉得类别这玩意儿有点像协议的可选协议。" 现在来看，其实protocal已经开始对两者都统一和规范起来操作，因为资料中说“非正式协议使用interface修饰“， 现在我们看到协议中两个修饰词：“必须实现(@requied)”和“可选实现(@optional)”。

5.What are KVO and KVC?  答案：kvc:键 - 值编码是一种间接访问对象的属性使用字符串来标识属性，而不是通过调用存取方法，直接或通过实例变量访问的机制。 很多情况下可以简化程序代码。apple文档其实给了一个很好的例子。 kvo:键值观察机制，他提供了观察某一属性变化的方法，极大的简化了代码。 具体用看到嗯哼用到过的一个地方是对于按钮点击变化状态的的监控。 比如我自定义的一个button [cpp]  [self addObserver:self forKeyPath:@"highlighted" options:0 context:nil];      #pragma mark KVO    - (void)observeValueForKeyPath:(NSString \*)keyPath ofObject:(id)object change:(NSDictionary \*)change context:(void \*)context  {   if ([keyPath isEqualToString:@"highlighted"] ) {   [self setNeedsDisplay];   }  } 对于系统是根据keypath去取的到相应的值发生改变，理论上来说是和kvc机制的道理是一样的。 对于kvc机制如何通过key寻找到value： “当通过KVC调用对象时，比如：[self valueForKey:@”someKey”]时，程序会自动试图通过几种不同的方式解析这个调用。首先查找对象是否带有 someKey 这个方法，如果没找到，会继续查找对象是否带有someKey这个实例变量（iVar），如果还没有找到，程序会继续试图调用 -(id) valueForUndefinedKey:这个方法。如果这个方法还是没有被实现的话，程序会抛出一个NSUndefinedKeyException异常错误。   (cocoachina.com注：Key-Value Coding查找方法的时候，不仅仅会查找someKey这个方法，还会查找getsomeKey这个方法，前面加一个get，或者\_someKey以及\_getsomeKey这几种形式。同时，查找实例变量的时候也会不仅仅查找someKey这个变量，也会查找\_someKey这个变量是否存在。）   设计valueForUndefinedKey:方法的主要目的是当你使用-(id)valueForKey方法从对象中请求值时，对象能够在错误发生前，有最后的机会响应这个请求。这样做有很多好处，下面的两个例子说明了这样做的好处。“ 来至cocoa，这个说法应该挺有道理。 因为我们知道button却是存在一个highlighted实例变量.因此为何上面我们只是add一个相关的keypath就行了， 可以按照kvc查找的逻辑理解，就说的过去了。

6.What is purpose of delegates? 代理的作用？ 答案：代理的目的是改变或传递控制链。允许一个类在某些特定时刻通知到其他类，而不需要获取到那些类的指针。可以减少框架复杂度。 另外一点，代理可以理解为java中的回调监听机制的一种类似。

7.What are mutable and immutable types in Objective C? obc中可修改和不可以修改类型。  答案：可修改不可修改的集合类。这个我个人简单理解就是可动态添加修改和不可动态添加修改一样。 比如NSArray和NSMutableArray。前者在初始化后的内存控件就是固定不可变的，后者可以添加等，可以动态申请新的内存空间。

8.When we call objective c is runtime language what does it mean? 我们说的obc是动态运行时语言是什么意思？  答案：多态。 主要是将数据类型的确定由编译时，推迟到了运行时。 这个问题其实浅涉及到两个概念，运行时和多态。 简单来说，运行时机制使我们直到运行时才去决定一个对象的类别，以及调用该类别对象指定方法。 多态：不同对象以自己的方式响应相同的消息的能力叫做多态。意思就是假设生物类（life）都用有一个相同的方法-eat; 那人类属于生物，猪也属于生物，都继承了life后，实现各自的eat，但是调用是我们只需调用各自的eat方法。 也就是不同的对象以自己的方式响应了相同的消息（响应了eat这个选择器）。 因此也可以说，运行时机制是多态的基础？~~~

9.what is difference between NSNotification and protocol? 通知和协议的不同之处？  答案：协议有控制链(has-a)的关系，通知没有。 首先我一开始也不太明白，什么叫控制链（专业术语了~）。但是简单分析下通知和代理的行为模式，我们大致可以有自己的理解 简单来说，通知的话，它可以一对多，一条消息可以发送给多个消息接受者。 代理按我们的理解，到不是直接说不能一对多，比如我们知道的明星经济代理人，很多时候一个经济人负责好几个明星的事务。 只是对于不同明星间，代理的事物对象都是不一样的，一一对应，不可能说明天要处理A明星要一个发布会，代理人发出处理发布会的消息后，别称B的 发布会了。但是通知就不一样，他只关心发出通知，而不关心多少接收到感兴趣要处理。 因此控制链（has-a从英语单词大致可以看出，单一拥有和可控制的对应关系。

10.What is push notification? 什么是推送消息？

11.Polymorphism？ 关于多态性  答案：多态，子类指针可以赋值给父类。 这个题目其实可以出到一切面向对象语言中， 因此关于多态，继承和封装基本最好都有个自我意识的理解，也并非一定要把书上资料上写的能背出来。 最重要的是转化成自我理解。

12.Singleton? 对于单例的理解  答案：11，12题目其实出的有点泛泛的感觉了，可能说是编程语言需要或是必备的基础。 基本能用熟悉的语言写出一个单例，以及可以运用到的场景或是你编程中碰到过运用的此种模式的框架类等。 进一步点，考虑下如何在多线程访问单例时的安全性。

13.What is responder chain? 说说响应链  答案： 事件响应链。包括点击事件，画面刷新事件等。在视图栈内从上至下，或者从下之上传播。 可以说点事件的分发，传递以及处理。具体可以去看下touch事件这块。因为问的太抽象化了 严重怀疑题目出到越后面就越笼统。

14.Difference between frame and bounds? frame和bounds有什么不同？  答案:frame指的是：该view在父view坐标系统中的位置和大小。（参照点是父亲的坐标系统） bounds指的是：该view在本身坐标系统中 的位置和大小。（参照点是本身坐标系统）

15.Difference between method and selector? 方法和选择器有何不同？  答案：selector是一个方法的名字，method是一个组合体，包含了名字和实现. 详情可以看apple文档。

16.Is there any garbage collection mechanism in Objective C.? OBC的垃圾回收机制？  答案： OBC2.0有Garbage collection，但是iOS平台不提供。 一般我们了解的objective-c对于内存管理都是手动操作的，但是也有自动释放池。 但是差了大部分资料，貌似不要和arc机制搞混就好了。 求更多~~

17.NSOperation queue?  答案：存放NSOperation的集合类。 操作和操作队列，基本可以看成java中的线程和线程池的概念。用于处理ios多线程开发的问题。 网上部分资料提到一点是，虽然是queue，但是却并不是带有队列的概念，放入的操作并非是按照严格的先进现出。 这边又有个疑点是，对于队列来说，先进先出的概念是Afunc添加进队列，Bfunc紧跟着也进入队列，Afunc先执行这个是必然的， 但是Bfunc是等Afunc完全操作完以后，B才开始启动并且执行，因此队列的概念离乱上有点违背了多线程处理这个概念。 但是转念一想其实可以参考银行的取票和叫号系统。 因此对于A比B先排队取票但是B率先执行完操作，我们亦然可以感性认为这还是一个队列。 但是后来看到一票关于这操作队列话题的文章，其中有一句提到 “因为两个操作提交的时间间隔很近，线程池中的线程，谁先启动是不定的。” 瞬间觉得这个queue名字有点忽悠人了，还不如pool~ 综合一点，我们知道他可以比较大的用处在于可以帮组多线程编程就好了。

18.What is lazy loading?  答案：懒汉模式，只在用到的时候才去初始化。 也可以理解成延时加载。 我觉得最好也最简单的一个列子就是tableView中图片的加载显示了。 一个延时载，避免内存过高，一个异步加载，避免线程堵塞。

19.Can we use two tableview controllers on one viewcontroller? 是否在一个视图控制器中嵌入两个tableview控制器？  答案：一个视图控制只提供了一个View视图，理论上一个tableViewController也不能放吧， 只能说可以嵌入一个tableview视图。当然，题目本身也有歧义，如果不是我们定性思维认为的UIViewController， 而是宏观的表示视图控制者，那我们倒是可以把其看成一个视图控制者，它可以控制多个视图控制器，比如TabbarController 那样的感觉。

20.Can we use one tableview with two different datasources? How you will achieve this? 一个tableView是否可以关联两个不同的数据源？你会怎么处理？  答案：首先我们从代码来看，数据源如何关联上的，其实是在数据源关联的代理方法里实现的。 因此我们并不关心如何去关联他，他怎么关联上，方法只是让我返回根据自己的需要去设置如相关的数据源。 因此，我觉得可以设置多个数据源啊，但是有个问题是，你这是想干嘛呢？想让列表如何显示，不同的数据源分区块显示？

21.Object－c的类可以多重继承么？可以实现多个接口么？重写一个类的方式用继承好还是分类好？为什么？ Objective-c只支持单继承，如果要实现多继承的话，可以通过类别和协议的方式来实现，cocoa 中所有的类都是NSObject 的子类，多继承在这里是用protocol 委托代理 来实现的。

22.#import 跟#include 又什么区别 ＃import<> 跟 #import""又什么区别？ 答案：@class一般用于头文件中需要声明该类的某个实例变量的时候用到，在m文 件中还是需要使用#import而#import比起#include的好处就是不会引起交叉编译。

23.类变量的@protected ,@private,@public,@package声明各有什么含义？

24.id 声明的对象有什么特性？ 答案：id是个很重要的类型，是个可以指向任何类型的指针或者可以理解为指向任何未知类型的指针。

25.MVC是什么？有什么特性？为什么在iPhone上被广泛运用？ 答案：MVC设计模式考虑三种对象：模型对象、视图对象、和控制器对象。模型对象代表 特别的知识和专业技能，它们负责保有应用程序的数据和定义操作数据的逻辑。视图对象知道如何显示应用程序的模型数据，而且可能允许用户对其进行编辑。控制 器对象是应用程序的视图对象和模型对象之间的协调者。

26.对于语句NSString\* testObject = [[NSData alloc] init];testObject 在编译时和运行时分别是什么类型的对象？

27.什么是安全释放？

28.为什么有些4.0独有的objective－c 函数在3.1上运行时会报错.而4.0独有的类在3.1上分配内存时不会报错？分配的结果是什么？

29.为什么4.0独有的c函数在3.1的机器上运行不会报错（在没有调用的情况下？）而4.0独有的类名在3.1的机器上一运行就报错？

30.异常exception 怎么捕获？不同的CPU结构上开销怎样？C中又什么类似的方法？

31.property中属性retain,copy,assgin的含义分别是什么？有什么区别？将其转换成get／set方法怎么做？有什么注意事项？

32.委托是什么？委托的property声明用什么属性？为什么？

34.Cocoa中有虚基类的概念么？怎么简洁的实现？

35.自动释放池跟GC（垃圾回收）有什么区别？iPhone上有GC么？［pool release］ 和［pool drain］有什么区别？ iPhone上没有GC。iPhone开发的时候没有垃圾回收机制。 在垃圾回收环境中，release是一个空操作。因此，NSAutoreleasePool提供了drain方法，在引用计数环境中，该方法的作用等同于调用release，但在垃圾回收环境中，它会触发垃圾回收（如果自上次垃圾回收以来分配的内存大于当前的阈值）。因此，在通常情况下，您应该使用drain而不是release来销毁自动释放池。

36.

for(int index = 0; index < 20; index ++){ NSString \*tempStr = @”tempStr”; NSLog(tempStr); NSNumber \*tempNumber = [NSNumber numberWithInt:2]; NSLog(tempNumber); } 这段代码有什么问题.？会不会造成内存泄露（多线程）？在内存紧张的设备上做大循环时自动释放池是写在循环内好还是循环外好？为什么？

37.内存管理的几条原则时什么？按照默认法则.那些关键字生成的对象需要手动释放？在和property结合的时候怎样有效的避免内存泄露？

38.在一个对象释放前.如果他被加到了notificationCenter 中.不在notificationcenter中remove这个对象可能会出现什么问题？

39.怎样实现一个 singleton的类.给出思路。

40.什么是序列化或者Acrchiving,可以用来做什么,怎样与copy结合,原理是什么?.

41. 线程与进程的区别和联系? 答案: 进程和线程都是由操作系统所体会的程序运行的基本 单元，系统利用该基本单元实现系统对应用的并发性。 程和线程的主要差别在于它们是不同的操作系统资源 管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，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，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一 些。但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。

42.在iphone上有两件事情要做,请问是在一个线程里按顺序做效率高还是两个线程里做效率高？为什么？

43.runloop是什么？在主线程中的某个函数里调用了异步函数，怎么样block当前线程,且还能响应当前线程的timer事件，touch事件等.

44.ios平台怎么做数据的持久化?coredata和sqlite有无必然联系？coredata是一个关系型数据库吗？ iOS中可以有四种持久化数据的方式： 属性列表、对象归档、SQLite3和Core Data；core data可以使你以图形界面的方式快速的定义app的数据模型，同时在你的代码中容易获取到它。core data提供了基础结构去处理常用的功能，例如保存，恢复，撤销和重做，允许你在app中继续创建新的任务。在使用core data的时候，你不用安装额外的数据库系统，因为core data使用内置的sqlite数据库。core data将你app的模型层放入到一组定义在内存中的数据对象。core data会追踪这些对象的改变，同时可以根据需要做相反的改变，例如用户执行撤销命令。当core data在对你app数据的改变进行保存的时候，core data会把这些数据归档，并永久性保存。 mac os x中sqlite库，它是一个轻量级功能强大的关系数据引擎，也很容易嵌入到应用程序。可以在多个平台使用，sqlite是一个轻量级的嵌入式sql数据库编程。与core data框架不同的是，sqlite是使用程序式的，sql的主要的API来直接操作数据表。 Core Data不是一个关系型数据库，也不是关系型数据库管理系统(RDBMS)。虽然Core Dta支持SQLite作为一种存储类型，但它不能使用任意的SQLite数据库。Core Data在使用的过程种自己创建这个数据库。Core Data支持对一、对多的关系。

45.阐述一个nil对象从interface bulider产生，到载入程序运行空间，最后被释放时所经历的生命周期.

46.notification是同步还是异步? kvo是同步还是异步？notification是全进程空间的通知吗？kvo呢？

~~47.kvc是什么?kvo是什么?有什么特性？~~

~~48.响应者链是什么？~~

49.unix上进程怎么通信？

   UNIX主要支持三种通信方式：

   1. 基本通信：主要用来协调进程间的同步和互斥

   (1)锁文件通信

     通信的双方通过查找特定目录下特定类型的文件(称锁文件)来完成进程间  对临界资源访问时的互斥；例如进程p1访问一个临界资源，首先查看是否有一个特定类型文件，若有，则等待一段时间再查找锁文件。

   (2)记录锁文件

   2. 管道通信：适应大批量的数据传递

   3. IPC：适应大批量的数据传递

50.timer的间隔周期准吗？为什么？怎样实现一个精准的timer?

51.UIscrollVew用到了什么设计模式？还能再foundation库中找到类似的吗？（答案众多，不知道哪个是对的～～）

模板(Template)模式，所有datasource和delegate接口都是模板模式的典型应用，

组合模式composition，所有的containerview都用了这个模式

观察者模式observer，所有的UIResponder都用了这个模式。

52如果要开发一个类似eclipse的软件，支持插件结构。且开放给第三方开发。你会怎样去设计它？（大概思路）

53. main()

{

  int a[5]={1,2,3,4,5};

  int \*ptr=(int \*)(&a+1);

  printf("%d,%d",\*(a+1),\*(ptr-1));

}

答：2,5

    \*(a+1)就是a[1]，\*(ptr-1)就是a[4],执行结果是2.5 ，&a+1不是首地址+1，系统会认为加一个a数组的偏 移，是偏移了一个数组的大小（本例是5个int，int \*ptr=(int \*)(&a+1); 则ptr实际 是&(a[5]),也就是a+5 原因如下：

　　&a是数组指针，其类型为 int (\*)[5]; 而指针加1要根据指针类型加上一定的值，不同类型的指针+1之后增加的大小不同。a是长度为5的int数组指针，所以要加 5\*sizeof(int)所以ptr实际是a[5]，但是prt与(&a+1)类型是不一样的(这点很重要)，所以prt-1只会减去sizeof(int\*)，a,&a的地址是一样的，但意思不一样，a是数组首地址，也就是a[0]的地址，&a是对象（数组）首地址，a+1是数组下一元素的地址，即a[1],&a+1是下一个对象的地址，即a[5].

54. 以下为Windows NT下的32位C++程序，请计算sizeof的值  voidFunc ( char str[100] )   {     sizeof( str ) =?   }   void \*p = malloc( 100 );  sizeof ( p ) = ?

答案：这题 很常见了,Func( char str[100] )函数中数组名作为函数形参时，在函数体内，数组名失去了本身的内涵，仅仅只是一个指针；在失去其内涵的同时，它还失去了其常量特性，可以作自增、自减等操作，可以被修改。Windows NT 32位平台下，指针的长度（占用内存的大小）为4字节，故sizeof( str ) 、sizeof( p ) 都为4。

55.- （void）\*getNSString(const NSString \*inputString)

{inputString =@"This is a main test\n";

     return ;}

-main(void)

{NSString \*a=@"Main";

NSString \*aString = [NSStringstringWithString:@"%@",getNSString(a)];

NSLog(@"%@\n", aString);}

最后问输出的字符串:NULL,output在 函数返回后，内存已经被释放。

56.列举几种进程的同步机制，并比较其优缺点。

答案：  原子操作信号量机制    自旋锁    管程，会合，分布式系统

进程之间通信的途径

答案：共享存储系统消息传递系统管道：以文件系统为基础

进程死锁的原因

答案：资源竞争及进程推进顺序非法

死锁的4个必要条 件

答案：互斥、请求保持、不可剥夺、环路

死锁的处理

答案：鸵鸟策略、预防策略、避免策略、检测与解除死锁

57.堆和栈的区别

管理方式：对于栈来讲，是由编译器自动管理，无需我们手工控制；对于堆来说，释放工作由程序员控制，容易产生memory leak。

申请大小： 栈：在Windows下,栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶的地址和栈的最大容量是系统预先规定好的，在WINDOWS下，栈的大小是2M（也有的说是1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示overflow。因此，能从栈获得的空间较小。 堆：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。

碎片问题：对于堆来讲，频繁的new/delete势必会造成内存空间的不连续，从而造成大量的碎片，使程序效率降低。对于栈来讲，则不会存在这个 问题，因为栈是先进后出的队列，他们是如此的一一对应，以至于永远都不可能有一个内存块从栈中间弹出

分配方式：堆都是动态分配的，没有静态分配的堆。栈有2种分配方式：静态分配和动态分配。静态分配是编译器完成的，比如局部变量的分配。动态分配由alloca函数进行分配，但是栈的动态分配和堆是不同的，他的动态分配是由编译器进行释放，无需我们手工实现。

分配效率：栈是机器系统提供的数据结构，计算机会在底层对栈提供支持：分配专门的寄存器存放栈的地址，压栈出栈都有专门的指令执行，这就决定了栈的效率比较高。堆则是C/C++函数库提供的，它的机制是很复杂的。

58.什么是键-值,键路径是什么?

模型的性质是通过一个简单的键（通常是个字符串）来指定的。视图和控制器通过键来查找相应的属性值。在一个给定的实体中，同一个属性的所有值具有相同的数据类型。键-值编码技术用于进行这样的查找—它是一种间接访问对象属性的机制。

键路径是一个由用点作分隔符的键组成的字符串，用于指定一个连接在一起的对象性质序列。第一个键的 性质是由先前的性质决定的，接下来每个键的值也是相对于其前面的性质。键路径使您可以以独立于模型 实现的方式指定相关 对象的性质。通过键路径，您可以指定对象图中的一个任意深度的路径，使其指向相 关对象的特定属性。

59.c和obj-c如何混用

1）obj-c的编译器处理后缀为m的文件时，可以识别obj-c和c的代码，处理mm文件可以识别obj-c,c,c++代码，但cpp文件必须只能用c/c++代码，而且cpp文件include的头文件中，也不能出现obj- c的代码，因为cpp只是cpp 2) 在mm文件中混用cpp直接使用即可，所以obj-c混cpp不是问题 3）在cpp中混用obj- c其实就是使用obj-c编写的模块是我们想要的。 如果模块以类实现，那么要按照cpp class的标准写类的定义，头文件中不能出现obj-c的东西，包括#import cocoa的。实现文件中，即类的实现代码中可以使用obj-c的东西，可以import,只是后缀是mm。 如果模块以函数实现，那么头文件要按 c的格式声明函数，实现文件中，c++函数内部可以用obj-c，但后缀还是mm或m。  总结：只要cpp文件和cpp include的文件中不包含obj-c的东西就可以用了，cpp混用obj-c的关键是使用接口，而不能直接使用实现代码，实际上cpp混用的是 obj-c编译后的o文件，这个东西其实是无差别的，所以可以用。obj-c的编译器支持cpp.

60.目标-动作机制

目标是动作消息的接收者。一个控件，或者更为常见的是它的单元，以插座变量（参见"插座变量"部分） 的形式保有其动作消息的目标。

动作是控件发送给目标的消息，或者从目标的角度看，它是目标为了响应动作而实现的方法。

程序需要某些机制来进行事件和指令的翻译。这个机制就是目标-动作机制。

61.cocoatouch框架

iPhoneOS 应用程序的基础Cocoa Touch 框架重用了许多 Mac 系统的成熟模式，但是它更多地专注于触摸的接口和优化。UIKit 为您提供了在 iPhone OS 上实现图形，事件驱动程序的基本工具，其建立在和 Mac OS X 中一样的 Foundation 框架上，包括文件处理，网络，字符串操作等。

CocoaTouch 具有和 iPhone 用户接口一致的特殊设计。有了 UIKit，您可以使用 iPhone OS 上的独特的图形接口控件，按钮，以及全屏视图的功能，您还可以使用加速仪和多点触摸手势来控制您的应用。

各色俱全的框架 除了UIKit 外，Cocoa Touch 包含了创建世界一流iPhone 应用程序需要的所有框架，从三维图形，到专业音效，甚至提供设备访问 API 以控制摄像头，或通过 GPS 获知当前位置。Cocoa Touch 既包含只需要几行代码就可以完成全部任务的强大的 Objective-C 框架，也在需要时提供基础的 C 语言 API 来直接访问系统。这些框架包括：

CoreAnimation

通过 CoreAnimation，您就可以通过一个基于组合独立图层的简单的编程模型来创建丰富的用户体验。

CoreAudio

CoreAudio 是播放，处理和录制音频的专业技术，能够轻松为您的应用程序添加强大的音频功能。

CoreData

提供了一个面向对象的数据管理解决方案，它易于使用和理解，甚至可处理任何应用或大或小的数据模型。

功能列表：框架分类

下面是 CocoaTouch 中一小部分可用的框架：

•   音频和视频

•   Core Audio

•   OpenAL

•   Media Library

•   AV Foundation

•   数据管理

•   Core Data

•   SQLite

•   图形和动画

•   Core Animation

•   OpenGL ES

•   Quartz 2D

•   网络/li>

•   Bonjour

•   WebKit

•   BSD Sockets

•   用户应用

•   Address Book

•   Core Location

•   Map Kit

•   Store Kit

62.objc的内存管理

如果您通过分配和初始化（比如[[MyClass alloc] init]）的方式来创建对象，您就拥 有这个对象，需要负责该对象的释放。这个规则在使用NSObject的便利方法new 时也同样适用.

如果您拷贝一个对象，您也拥有拷贝得到的对象，需要负责该对象的释放.

如果您保持一个对象，您就部分拥有这个对象，需要在不再使用时释放该对象。

如果您从其它对象那里接收到一个对象，则您不拥有该对象，也不应该释放它（这个规则有少数 的例外，在参考文档中有显式的说明）。

63.自动释放池是什么,如何工作

当您向一个对象发送一个autorelease消息时，Cocoa就会将该对 象的一个引用放入到最新的自动释放池。它仍然是个正当的对象，因此自动释放池定义的作用域内的其它对象可以向它发送消息。当程序执行到作用域结束的位置时，自动释放池就会被释放，池中的所有对象也就被释放。

1. ojc-c 是通过一种"referringcounting"(引用计数)的方式来管理内存的, 对象在开始分配内存(alloc)的时候引用计数为一,以后每当碰到有copy,retain的时候引用计数都会加一, 每当碰到release和autorelease的时候引用计数就会减一,如果此对象的计数变为了0, 就会被系统销毁. 2. NSAutoreleasePool 就是用来做引用计数的管理工作的,这个东西一般不用你管的. 3. autorelease和release没什么区别,只是引用计数减一的时机不同而已,autorelease会在对象的使用真正结束的时候才做引用计数减一.

64.类工厂方法是什么

类工厂方法的实现是为了向客户提供方便，它们将分配和初始化合在一个步骤中， 返回被创建的对象，并 进行自动释放处理。这些方法的形式是+ (type)className...（其中 className不包括任何前缀）。

工厂方法可能不仅仅为了方便使用。它们不但可以将分配和初始化合在一起，还可以为初始化过程提供对 象的分配信息。

类工厂方法的另一个目的是使类（比如NSWorkspace）提供单件实例。虽 然init...方法可以确认一 个类在每次程序运行过程只存在一个实例，但它需要首先分配一个“生的”实例，然后还必须释放该实例。 工厂 方法则可以避免为可能没有用的对象盲目分配内存。

65.单件实例是什么

Foundation和Application Kit 框架中的一些类只允许创建单件对象，即这些类在当前进程中的唯一实例。举例来说，NSFileManager和NSWorkspace 类在使用时都是基于进程进行单件对象的实例化。当向这些类请求实例的时候，它们会向您传递单一实例的一个引用，如果该实例还不存在，则首先进行实例的分配和初始化。 单件对象充当控制中心的角色，负责指引或协调类的各种服务。如果类在概念上只有一个实例（比如 NSWorkspace），就应该产生一个单件实例，而不是多个实例；如果将来某一天可能有多个实例，您可 以使用单件实例机制，而不是工厂方法或函数。

66.动态绑定—在运行时确定要调用的方法

动态绑定将调用方法的确定也推迟到运行时。在编译时，方法的调用并不和代码绑定在一起，只有在消实发送出来之后，才确定被调用的代码。通过动态类型和动态绑定技术，您的代码每次执行都可以得到不同的结果。运行时因子负责确定消息的接 收者和被调用的方法。运行时的消息分发机制为动态绑定提供支持。当您向一个动态类型确定了的对象发送消息时，运行环境系统会通过接收者的isa指针定位对象的类，并以此为起点确定被调用的方法，方法和消息是动态绑定的。而且，您不必在Objective-C 代码中做任何工作，就可以自动获取动态绑定的好处。您在每次发送消息时，

 特别是当消息的接收者是动态类型已经确定的对象时，动态绑定就会例行而 透明地发生。

67.obj-c的优缺点

objc优点：

1)Cateogies

2) Posing

3) 动态识别

4) 指标计算

5）弹性讯息传递

6) 不是一个过度复杂的 C 衍生语言

7) Objective-C 与 C++ 可混合编程

缺点:

1) 不支援命名空間

2) 不支持运算符重载

3） 不支持多重继承

4） 使用动态运行时类型，所有的方法都是函数调用，所以很多编译时优化方法都用不到。（如内联函数等），性能低劣。

68.readwrite，readonly，assign，retain，copy，nonatomic 属性的作用？

@property是 一个属性访问声明，扩号内支持以下几个属性： 1，getter=getterName，setter=setterName，设置setter与 getter的方法名 2，readwrite,readonly，设置可供访问级别 2，assign，setter方法直接赋值，不进行 任何retain操作，为了解决原类型与环循引用问题 3，retain，setter方法对参数进行release旧值再retain新值，所有实现都是这个顺序(CC上有相关资料) 4，copy，setter方法进行Copy操作，与retain处理流程一样，先旧值release，再 Copy出新的对象，retainCount为1。这是为了减少对上下文的依赖而引入的机制。 5，nonatomic，非原子性访问，不加同步， 多线程并发访问会提高性能。注意，如果不加此属性，则默认是两个访问方法都为原子型事务访问。锁被加到所属对象实例级(我是这么理解的…)。

~~69.ViewController的 didReceiveMemoryWarning 是在什么时候被调用的？~~（87题）

70.谈谈你对ARC 的认识和理解？

71.  ObjC中，与alloc语义相反的方法是dealloc还是release？与retain语义相反的方法是dealloc还是release，为什么？需要与alloc配对使用的方法是dealloc还是release，为什么？

alloc与dealloc语意相反，alloc是创建变量，dealloc是释放变量。 retain 对应release,retain 保留一个对象。调用之后，变量的计数加1。或许不是很明显，在这有例为证：

- (void)setName : (NSString\*) name {

    [name retain];

    [myname release];

    myname = name; }

我们来解释一下：设想，用户在调用这个函数的时候，他注意了内存的管理，所以他小心的写了如下代码：

NSString\* newname = [[NSString alloc] initWithString: @"John"];

[aClasssetName: newname];

[newnamerelease];

我们来看一看newname的计数是怎么变化的。首先，它被alloc，count = 1; 然后，在setName中，它被retain，count = 2; 最后，用户自己释放newname，count= 1，myname指向了newname。这也解释了为什么需要调用[myname release]。我们需要在给myname赋新值的时候，释放掉以前老的变量。retain 之后直接dealloc对象计数器没有释放。alloc 需要与release配对使用，因为alloc 这个函数调用之后，变量的计数加1。所以在调用alloc 之后，一定要调用对应的release。另外，在release一个变量之后，他的值仍然有效，所以最好是后面紧接着再var = nil。

72. 在一个对象的方法里面:

self.name<http://self.name/> = “object”;

和

name ＝”object”

有什么不同?

73. 这段代码有什么问题:

@implementation Person

- (void)setAge:(int)newAge {

self.age = newAge;

}

@end

74. 什么是retaincount?

75.以下每行代码执行后，person对象的retain count分别是多少

Person \*person = [[Person alloc] init];

[person retain];

[person release];

[person release];

76.为什么很多内置类如UITableViewController的delegate属性都是assign而不是retain的？

77.定义属性时，什么情况使用copy，assign，和retain？

assign用于简单数据类型，如NSInteger,double,bool,retain和copy用户对象，copy用于当 a指向一个对象，b也想指向同样的对象的时候，如果用assign，a如果释放，再调用b会crash,如果用copy 的方式，a和b各自有自己的内存，就可以解决这个问题。retain 会使计数器加一，也可以解决assign的问题。另外：atomic和nonatomic用来决定编译器生成的getter和setter是否为原子操作。在多线程环境下，原子操作是必要的，否则有可能引起错误的结果。加了atomic，setter函数会变成下面这样：

if(property != newValue) {

[propertyrelease];

property = [newValueretain];

}

78. autorelease的对象是在什么时候被release的？

答：autorelease实际上只是把对release的调用延迟了，对于每一个Autorelease，系统只是把该Object放入了当前的Autoreleasepool中，当该pool被释放时，该pool中的所有Object会被调用Release。对于每一个Runloop，系统会隐式创建一个Autoreleasepool，这样所有的releasepool会构成一个象CallStack一样的一个栈式结构，在每一个Runloop结束时，当前栈顶的Autoreleasepool会被销毁，这样这个pool里的每个Object（就是autorelease的对象）会被release。那什么是一个Runloop呢？一个UI事件，Timer call，delegate call， 都会是一个新的Runloop。那什么是一个Runloop呢？一个UI事件，Timer call，delegate call， 都会是一个新的Runloop。

79. 这段代码有什么问题,如何修改

for (int i = 0; i < someLargeNumber; i++)

{

NSString \*string = @”Abc”;

string = [string lowercaseString];

string = [stringstringByAppendingString:@"xyz"];

NSLog(@“%@”, string);

}

~~80. autorelease和垃圾回收机制(gc)有什么关系？IPhone OS有没有垃圾回收（gc）？~~

81. Objective C中的protocal是什么？

@protocal obj里面的协议就如同java里面的interface。

82. Objective C中的selector 是什么？

你可以理解 @selector()就是取类方法的编号,他的行为基本可以等同C语言的中函数指针,只不过C语言中，可以把函数名直接赋给一个函数指针，而Objective-C的类不能直接应用函数指针，这样只能做一个@selector语法来取.它的结果是一个SEL类型。这个类型本质是类方法的编号(函数地址)。

83. Objective C中的category是什么?

Objective-C提供了一个非常灵活的类(Class)扩展机制－类别(Category)。类别用于对一个已经存在的类添加方法(Methods)。你只需要知道这个类的公开接口，不需要知道类的源代码。需要注意的是，类别不能为已存在的类添加实例变量(Instance Variables)。

子类(subclassing)是进行类扩展的另一种常用方法。与子类相比，类别最主要的优点是：系统中已经存在的类可以不需修改就可使用类别的扩展功能。

类别的第二大优点是实现了功能的局部化封装。类别定义可以放在一个已存在的类(类A)的定义文件中(.h)。这意味着这个类别只有在类A被引用的前提下才会被外部看到。如果另一个类(类B)不需要用到类A的功能(没有包含类A的.h文件)，也就不会看到依附类A存在的类别。iOS SDK中广泛运用这种类别定义法来封装功能。

84. 什么是Notification？什么时候用delegate，什么时候用Notification？

观察者模式，controller向defaultNotificationCenter添加自己的notification，其他类注册这个notification就可以收到通知，这些类可以在收到通知时做自己的操作（多观察者默认随机顺序发通知给观察者们，而且每个观察者都要等当前的某个观察者的操作做完才能轮到他来操作，可以用NotificationQueue的方式安排观察者的反应顺序，也可以在添加观察者中设定反映时间，取消观察需要在viewDidUnload跟dealloc中都要注销）。

delegate针对one-to-one关系，并且reciever可以返回值给sender，notification可以针对one-to-one/many/none,reciever无法返回值给sender.所以，delegate用于sender希望接受到reciever的某个功能反馈值，

notification用于通知多个object某个事件。

85. ~~什么是KVC和KVO？~~Notification和KVO有什么不同？KVO在ObjC中是怎么实现的？

86. ViewController 的 loadView,viewDidLoad, viewDidUnload 分别是在什么时候调用的？在自定义ViewController的时候这几个函数里面应该做什么工作？

答：viewDidLoad在view 从nib文件初始化时调用，loadView在controller的view为nil时调用。此方法在编程实现view时调用,view 控制器默认会注册memory warning notification,当view controller的任何view 没有用的时候，viewDidUnload会被调用，在这里实现将retain 的view release,如果是retain的IBOutlet view 属性则不要在这里release,IBOutlet会负责release 。

87. ViewController 的didReceiveMemoryWarning 是在什么时候被调用的？UIViewController类中didReceiveMemoryWarning默认的操作是什么？

答：默认调用[superdidReceiveMemoryWarning]

88. UITableViewController 中，创建UITableViewCell时，initWithSytle:resuseIdentifier中，reuseIdentifier有什么用？简述UITableViewCell的复用原理.

复用队列的元素增加：只有在cell被滑动出界面的时候，此cell才会被加入到复用队列中。每次在创建cell的时候，程序会首先通过调用dequeueReusableCellWithIdentifier:cellType方法，到复用队列中去寻找标示符为“cellType”的cell，如果找不到，返回nil，然后程序去通过调用[[[UITableViewCell alloc]initWithStyle:style reuseIdentifier:cellType] autorelease]来创建标示符为“cellType”的cell。

89. UIView 和CALayer 有什么区别?

两者最大的区别是，图层不会直接渲染到屏幕上。

90. 以 UIView 类animateWithDuration:animations: 为例，简述UIView动画原理。

0

<http://blog.csdn.net/c349126799/article/details/38413713>

总共6题

Object-C有多继承吗？没有的话用什么代替？

cocoa 中所有的类都是NSObject 的子类 多继承在这里是用protocol 委托代理 来实现的 你不用去考虑繁琐的多继承 ,虚基类的概念. ood的多态特性 在 obj-c 中通过委托来实现.

Object-C有私有方法吗？私有变量呢？

objective-c– 类里面的方法只有两种, 静态方法和实例方法. 这似乎就不是完整的面向对象了,按照OO的原则就是一个对象只暴露有用的东西. 如果没有了私有方法的话, 对于一些小范围的代码重用就不那么顺手了. 在类里面声名一个私有方法 @interface Controller : NSObject { NSString \*something; } + (void)thisIsAStaticMethod; – (void)thisIsAnInstanceMethod; @end @interface Controller (private) - (void)thisIsAPrivateMethod; @end

@private可以用来修饰私有变量 在Objective‐C中，所有实例变量默认都是私有的，所有实例方法默认都是公有的

关键字const什么含义

const意味着”只读”，下面的声明都是什么意思？const int a; int const a; const int \*a; int \* const a; int const \* a const;

前两个的作用是一样，a是一个常整型数。第三个意味着a是一个指向常整型数的指针（也就是，整型数是不可修改的，但指针可以）。第四个意思a是一个指向整型数的常指针（也就是说，指针指向的整型数是可以修改的，但指针是不可修改的）。最后一个意味着a是一个指向常整型数的常指针（也就是说，指针指向的整型数是不可修改的，同时指针也是不可修改的）。

结论： •; 关键字const的作用是为给读你代码的人传达非常有用的信息，实际上，声明一个参数为常量是为了告诉了用户这个参数的应用目的。如果 你曾花很多时间清理其它人留下的垃圾，你就会很快学会感谢这点多余的信息。（当然，懂得用const的程序员很少会留下的垃圾让别人来清 理的。） •; 通过给优化器一些附加的信息，使用关键字const也许能产生更紧凑的代码。 •; 合理地使用关键字const可以使编译器很自然地保护那些不希望被改变的参数，防止其被无意的代码修改。简而言之，这样可以减少bug的出现。

欲阻止一个变量被改变，可以使用 const 关键字。在定义该 const 变量时，通常需要对它进行初 始化，因为以后就没有机会再去改变它了； （2）对指针来说，可以指定指针本身为 const，也可以指定指针所指的数据为 const，或二者同时指 定为 const； （3）在一个函数声明中，const 可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值； （4）对于类的成员函数，若指定其为 const 类型，则表明其是一个常函数，不能修改类的成员变量； （5）对于类的成员函数，有时候必须指定其返回值为 const 类型，以使得其返回值不为“左值”。

关键字volatile有什么含义？并给出三个不同例子？

一个定义为volatile的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到 这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。下面是volatile变量的几个例子： • 并行设备的硬件寄存器（如：状态寄存器） • 一个中断服务子程序中会访问到的非自动变量(Non-automatic variables) • 多线程应用中被几个任务共享的变量

• 一个参数既可以是const还可以是volatile吗？解释为什么。 • 一个指针可以是volatile 吗？解释为什么。

下面是答案： • 是的。一个例子是只读的状态寄存器。它是volatile因为它可能被意想不到地改变。它是const因为程序不应该试图去修改它。 • 是的。尽管这并不很常见。一个例子是当一个中服务子程序修该一个指向一个buffer的指针时。

static作用？

函数体内 static 变量的作用范围为该函数体，不同于 auto 变量，该变量的内存只被分配一次， 因此其值在下次调用时仍维持上次的值； （2）在模块内的 static 全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问； （3）在模块内的 static 函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明 它的模块内； （4）在类中的 static 成员变量属于整个类所拥有，对类的所有对象只有一份拷贝； （5）在类中的 static 成员函数属于整个类所拥有，这个函数不接收 this 指针，因而只能访问类的static 成员变量。

#import和#include的区别，@class代表什么？

@class一般用于头文件中需要声明该类的某个实例变量的时候用到，在m文件中还是需要使用#import 而#import比起#include的好处就是不会引起重复包含

线程和进程的区别？

进程和线程都是由操作系统所体会的程序运行的基本单元，系统利用该基本单元实现系统对应用的并发性。 进程和线程的主要差别在于它们是不同的操作系统资源管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，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，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。

堆和栈的区别？

管理方式：对于栈来讲，是由编译器自动管理，无需我们手工控制；对于堆来说，释放工作由程序员控制，容易产生memory leak。 申请大小： 栈：在Windows下,栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶的地址和栈的最大容量是系统预先规定好的，在WINDOWS下，栈的大小是2M（也有的说是1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示overflow。因此，能从栈获得的空间较小。 堆：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。 碎片问题：对于堆来讲，频繁的new/delete势必会造成内存空间的不连续，从而造成大量的碎片，使程序效率降低。对于栈来讲，则不会存在这个问题，因为栈是先进后出的队列，他们是如此的一一对应，以至于永远都不可能有一个内存块从栈中间弹出 分配方式：堆都是动态分配的，没有静态分配的堆。栈有2种分配方式：静态分配和动态分配。静态分配是编译器完成的，比如局部变量的分配。动态分配由alloca函数进行分配，但是栈的动态分配和堆是不同的，他的动态分配是由编译器进行释放，无需我们手工实现。 分配效率：栈是机器系统提供的数据结构，计算机会在底层对栈提供支持：分配专门的寄存器存放栈的地址，压栈出栈都有专门的指令执行，这就决定了栈的效率比较高。堆则是C/C++函数库提供的，它的机制是很复杂的。

Object-C的内存管理？

1.当你使用new,alloc和copy方法创建一个对象时,该对象的保留计数器值为1.当你不再使用该对象时,你要负责向该对象发送一条release或autorelease消息.这样,该对象将在使用寿命结束时被销毁. 2.当你通过任何其他方法获得一个对象时,则假设该对象的保留计数器值为1,而且已经被设置为自动释放,你不需要执行任何操作来确保该对象被清理.如果你打算在一段时间内拥有该对象,则需要保留它并确保在操作完成时释放它. 3.如果你保留了某个对象,你需要(最终)释放或自动释放该对象.必须保持retain方法和release方法的使用次数相等.

为什么很多内置的类，如TableViewController的delegate的属性是assign不是retain？

循环引用 所有的引用计数系统，都存在循环应用的问题。例如下面的引用关系：     •    对象a创建并引用到了对象b.     •    对象b创建并引用到了对象c.     •    对象c创建并引用到了对象b. 这时候b和c的引用计数分别是2和1。当a不再使用b，调用release释放对b的所有权，因为c还引用了b，所以b的引用计数为1，b不会被释放。b不释放，c的引用计数就是1，c也不会被释放。从此，b和c永远留在内存中。 这种情况，必须打断循环引用，通过其他规则来维护引用关系。比如，我们常见的delegate往往是assign方式的属性而不是retain方式的属性，赋值不会增加引用计数，就是为了防止delegation两端产生不必要的循环引用。如果一个UITableViewController 对象a通过retain获取了UITableView对象b的所有权，这个UITableView对象b的delegate又是a， 如果这个delegate是retain方式的，那基本上就没有机会释放这两个对象了。自己在设计使用delegate模式时，也要注意这点。

定义属性时，什么情况使用copy、assign、retain？

assign用于简单数据类型，如NSInteger,double,bool, retain和copy用于对象， copy用于当a指向一个对象，b也想指向同样的对象的时候，如果用assign，a如果释放，再调用b会crash,如果用copy 的方式，a和b各自有自己的内存，就可以解决这个问题。 retain 会使计数器加一，也可以解决assign的问题。 另外：atomic和nonatomic用来决定编译器生成的getter和setter是否为原子操作。在多线程环境下，原子操作是必要的，否则有可能引起错误的结果。 加了atomic，setter函数会变成下面这样： if (property != newValue) { [property release]; property = [newValue retain]; }

对象是什么时候被release的？

引用计数为0时。 autorelease实际上只是把对release的调用延迟了，对于每一个Autorelease，系统只是把该Object放入了当前的Autorelease pool中，当该pool被释放时，该pool中的所有Object会被调用Release。对于每一个Runloop， 系统会隐式创建一个Autoreleasepool，这样所有的releasepool会构成一个象CallStack一样的一个栈式结构，在每一个Runloop结束时，当前栈顶的Autorelease pool会被销毁，这样这个pool里的每个Object（就是autorelease的对象）会被release。那什么是一个Runloop呢？ 一个UI事件，Timer call， delegate call， 都会是一个新的Runloop

iOS有没有垃圾回收？

Objective-C 2.0也是有垃圾回收机制的，但是只能在Mac OS X Leopard 10.5 以上的版本使用。

tableView的重用机制？

　查看UITableView头文件，会找到NSMutableArray\*  visiableCells，和NSMutableDictnery\* reusableTableCells两个结构。visiableCells内保存当前显示的cells，reusableTableCells保存可重用的cells。

　　TableView显示之初，reusableTableCells为空，那么tableViewdequeueReusableCellWithIdentifier:CellIdentifier返回nil。开始的cell都是通过[[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier]来创建，而且cellForRowAtIndexPath只是调用最大显示cell数的次数。

　　比如：有100条数据，iPhone一屏最多显示10个cell。程序最开始显示TableView的情况是：

　　1. 用[[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier]创建10次cell，并给cell指定同样的重用标识(当然，可以为不同显示类型的cell指定不同的标识)。并且10个cell全部都加入到visiableCells数组，reusableTableCells为空。

　　2. 向下拖动tableView，当cell1完全移出屏幕，并且cell11(它也是alloc出来的，原因同上)完全显示出来的时候。cell11加入到visiableCells，cell1移出visiableCells，cell1加入到reusableTableCells。

3. 接着向下拖动tableView，因为reusableTableCells中已经有值，所以，当需要显示新的cell，cellForRowAtIndexPath再次被调用的时候，tableView dequeueReusableCellWithIdentifier:CellIdentifier，返回cell1。cell1加入到visiableCells，cell1移出reusableTableCells；cell2移出visiableCells，cell2加入到reusableTableCells。之后再需要显示的Cell就可以正常重用了。

 ViewController 的loadView、viewDidLoad、viewDidUnload分别是什么时候调用的，在自定义ViewCointroller时在这几个函数中应该做什么工作？

由init、loadView、viewDidLoad、viewDidUnload、dealloc的关系说起 init方法 在init方法中实例化必要的对象（遵从LazyLoad思想） init方法中初始化ViewController本身

loadView方法 当view需要被展示而它却是nil时，viewController会调用该方法。不要直接调用该方法。 如果手工维护views，必须重载重写该方法 如果使用IB维护views，必须不能重载重写该方法

loadView和IB构建view 你在控制器中实现了loadView方法，那么你可能会在应用运行的某个时候被内存管理控制调用。 如果设备内存不足的时候， view 控制器会收到didReceiveMemoryWarning的消息。 默认的实现是检查当前控制器的view是否在使用。 如果它的view不在当前正在使用的view hierarchy里面，且你的控制器实现了loadView方法，那么这个view将被release, loadView方法将被再次调用来创建一个新的view。

viewDidLoad方法 viewDidLoad 此方法只有当view从nib文件初始化的时候才被调用。 重载重写该方法以进一步定制view 在iPhone OS3.0及之后的版本中，还应该重载重写viewDidUnload来释放对view的任何索引 viewDidLoad后调用数据Model

viewDidUnload方法 当系统内存吃紧的时候会调用该方法（注：viewController没有被dealloc） 内存吃紧时，在iPhone OS3.0之前didReceiveMemoryWarning是释放无用内存的唯一方式，但是OS 3.0及以后viewDidUnload方法是更好的方式 在该方法中将所有IBOutlet（无论是property还是实例变量）置为nil（系统release view时已经将其release掉了）

在该方法中释放其他与view有关的对象、其他在运行时创建（但非系统必须）的对象、在viewDidLoad中被创建的对象、缓存数据等release对象后，将对象置为nil（IBOutlet只需要将其置为nil，系统release view时已经将其release掉了）

一般认为viewDidUnload是viewDidLoad的镜像，因为当view被重新请求时，viewDidLoad还会重新被执行

viewDidUnload中被release的对象必须是很容易被重新创建的对象（比如在viewDidLoad或其他方法中创建的对象），不要release用户数据或其他很难被重新创建的对象

dealloc方法 viewDidUnload和dealloc方法没有关联，dealloc还是继续做它该做的事情

 ViewController的didReceiveMemoryWarning是在什么时候调用的？默认的操作是什么？

当程序接到内存警告时View Controller将会收到这个消息：didReceiveMemoryWarning

从iOS3.0开始，不需要重载这个函数，把释放内存的代码放到viewDidUnload中去。

这个函数的默认实现是:检查controller是否可以安全地释放它的view(这里加粗的view指的是controller的view属性)，比如view本身没有superview并且可以被很容易地重建（从nib或者loadView函数）。

如果view可以被释放，那么这个函数释放view并调用viewDidUnload。

你可以重载这个函数来释放controller中使用的其他内存。但要记得调用这个函数的super实现来允许父类（一般是UIVIewController）释放view。

如果你的ViewController保存着view的子view的引用，那么，在早期的iOS版本中，你应该在这个函数中来释放这些引用。而在iOS3.0或更高版本中，你应该在viewDidUnload中释放这些引用。

列举Cocoa中常见的集中多线程的实现，并谈谈多线程安全的几种解决办法，一般什么地方会用到多线程？

NSOperationNSThread @sychonized

怎么理解MVC，在Cocoa中MVC是怎么实现的？

MVC设计模式考虑三种对象：模型对象、视图对象、和控制器对象。模型对象代表特别的知识和专业技能，它们负责保有应用程序的数据和定义操作数据的逻辑。视图对象知道如何显示应用程序的模型数据，而且可能允许用户对其进行编辑。控制器对象是应用程序的视图对象和模型对象之间的协调者。 ViewCotroller Xib

delegate和notification区别，分别在什么情况下使用？

KVC(Key-Value-Coding) KVO（Key-Value-Observing） 理解KVC与KVO（键-值-编码与键-值-监看）

当通过KVC调用对象时，比如：[self valueForKey:@”someKey”]时，程序会自动试图通过几种不同的方式解析这个调用。首先查找对象是否带有 someKey 这个方法，如果没找到，会继续查找对象是否带有someKey这个实例变量（iVar），如果还没有找到，程序会继续试图调用 -(id) valueForUndefinedKey:这个方法。如果这个方法还是没有被实现的话，程序会抛出一个NSUndefinedKeyException异常错误。

(Key-Value Coding查找方法的时候，不仅仅会查找someKey这个方法，还会查找getsomeKey这个方法，前面加一个get，或者\_someKey以及\_getsomeKey这几种形式。同时，查找实例变量的时候也会不仅仅查找someKey这个变量，也会查找\_someKey这个变量是否存在。)

设计valueForUndefinedKey:方法的主要目的是当你使用-(id)valueForKey方法从对象中请求值时，对象能够在错误发生前，有最后的机会响应这个请求。

 self.跟self什么区别？

 id、nil代表什么？

id

id和void \*并非完全一样。在上面的代码中，id是指向struct objc\_object的一个指针，这个意思基本上是说，id是一个指向任何一个继承了Object（或者NSObject）类的对象。需要注意的是id是一个指针，所以你在使用id的时候不需要加星号。比如id foo=nil定义了一个nil指针，这个指针指向NSObject的一个任意子类。而id \*foo=nil则定义了一个指针，这个指针指向另一个指针，被指向的这个指针指向NSObject的一个子类。

nil

nil和C语言的NULL相同，在objc/objc.h中定义。nil表示一个Objctive-C对象，这个对象的指针指向空（没有东西就是空）。

内存管理 Autorelease、retain、copy、assign的set方法和含义？

1，你初始化(alloc/init)的对象，你需要释放(release)它。例如：

　　NSMutableArray aArray = [[NSArray alloc] init];

　　后，需要

　　[aArray release];

　　2，你retain或copy的，你需要释放它。例如：

　　[aArray retain]

　　后，需要

　　[aArray release];

　　3，被传递(assign)的对象，你需要斟酌的retain和release。例如：

　　obj2 = [[obj1 someMethod] autorelease];

　　对象2接收对象1的一个自动释放的值，或传递一个基本数据类型(NSInteger，NSString)时： 你或希望将对象2进行retain，以防止它在被使用之前就被自动释放掉。但是在retain后，一定要在适当的时候进行释放。

　　关于索引计数(Reference Counting)的问题

　　retain值 = 索引计数(ReferenceCounting)

　　NSArray对象会retain(retain值加一)任何数组中的对象。当NSArray被卸载(dealloc)的时候，所有数组中的对象会被执行一次释放(retain值减一)。不仅仅是NSArray，任何收集类(CollectionClasses)都执行类似操作。例如NSDictionary，甚至UINavigationController。

　　Alloc/init建立的对象，索引计数为1。无需将其再次retain。

　　[NSArray array]和[NSDate date]等“方法”建立一个索引计数为1的对象，但是也是一个自动释放对象。所以是本地临时对象，那么无所谓了。如果是打算在全Class中使用的变量(iVar)，则必须retain它。

　　缺省的类方法返回值都被执行了“自动释放”方法。(\*如上中的NSArray)

　　在类中的卸载方法“dealloc”中，release所有未被平衡的NS对象。(\*所有未被autorelease，而retain值为1的)

类别的作用？

有时我们需要在一个已经定义好的类中增加一些方法，而不想去重写该类。比如，当工程已经很大，代码量比较多，或者类中已经包住很多方法，已经有其他代码调用了该类创建对象并使用该类的方法时，可以使用类别对该类扩充新的方法。

  注意：类别只能扩充方法，而不能扩充成员变量。

委托（举例）

委托代理（degegate），顾名思义，把某个对象要做的事情委托给别的对象去做。那么别的对象就是这个对象的代理，代替它来打理要做的事。反映到程序中，首先要明确一个对象的委托方是哪个对象，委托所做的内容是什么。 委托机制是一种设计模式，在很多语言中都用到的，这只是个通用的思想，网上会有很多关于这方面的介绍。 那么在苹果开发过程中，用到委托的程序实现思想如下，我主要拿如何在视图之间传输信息做个例子。 譬如：在两个页面（UIIview视图对象）实现传值，用委托（delegate）可以很好做到！ 方法： 类A @interface A：UIView         id transparendValueDelegate;        @property(nomatic, retain) idtransparendValueDelegate; @end  @implemtion A @synthesize transparendValueDelegate -(void)Function {        NSString\* value = @"hello";       //让代理对象执行transparendValue动作       [transparendValueDelegate transparendValue:value]; } @end  类B @interface B：UIView       NSString\* value; @end  @implemtion B -(void)transparendValue:(NSString\*)fromValue {       value = fromValue;       NSLog(@"the value is %@",value);  } @end  //下面的设置A代理委托对象为B //在定义A和B类对象处：  A\* a = [[A alloc] init]; B\* b = [[B alloc] init]; a. transparendValueDelegate = b;//设置对象a代理为对象b  这样在视图A和B之间可以通过委托来传值！  下面这个例子委托有两类： 1、一个视图类对象的代理对象为父视图，子视图用代理实现让父视图显示别的子视图 2、同一父视图下的一个子视图为另一个子视图的代理对象，让另一个子视图改变自身背景色为给定的颜色 ＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝ 规范格式如下： @protocol TransparendValueDelegate;  @interface A：UIView id< TransparendValueDelegate > m\_dTransparendValueDelegate; @property(nomatic, retain) id m\_dTransparendValueDelegate; @end //代理协议的声明 @protocol TransparendValueDelegat<NSObject> {       -(void)transparendValue:(NSString\*)fromValue; }

retainCount？

1.objective-c中的数字对象都有哪些，简述它们与基本数据类型的区别是什么

2.用NSLog函数输出一个浮点类型，结果四舍五入，并保留一位小数

3.截取字符串”20｜http://www.621life.com“ 中 ‘｜’字符前面及后面的数据，分别输出它们

4.objective-c中的词典对象、可变词典对象是哪个，初始化一个含有两个键值对的可变词典对象，并动态的添加和删除一条记录，输出第一条记录

5.获取项目根路径，并在其下创建一个名称为userData的目录。

6.在一个对象的方法里面:self.name = “object”;和name ＝”object”有什么不同吗?

7.定义属性时，什么情况使用copy，assign，和retain

第3题：

NSRange range = [responseString rangeOfString:@"|"];

int location = range.location;

NSString \*str1 = [responseString substringToIndex:location];

NSString \*str2 = [responseString substringFromIndex:location+1];

第4题：

NSDictionary NSMutableDictionary

NSMutableDictionary \*dic = [NSMutableDictionary dictionaryWithObjectsAndKeys:@"value1",@"key1",@"value2",@"key2",nil];

[dic setObject:@"value3" forKey:@"key3"];

[dic removeObjectForKey:@"key3"];

[dic objectForKey:@"key1"];

第5题：

// 获取根路径

NSArray \*paths =NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,NSUserDomainMask, YES);

NSString \*documentsDirectory = [paths objectAtIndex:];

// 创建文件系统管理器

NSFileManager \*fileManager = [[NSFileManager alloc] init];

// 判断userData目录是否存在

if(![fileManager fileExistsAtPath:[NSString stringWithFormat:@"%@/userData", documentsDirectory]]) {

// 不存在,创建一个userData目录

[fileManager createDirectoryAtPath:[NSString stringWithFormat:@"%@/userData", documentsDirectory]withIntermediateDirectories:false attributes:nil error:nil];

}

第6题：

self.name = “object”会调用对象的setName()方法，

name = “object”会直接把object赋值给当前对象的name 属性。

并且 self.name 这样retainCount会加１,而name就不会。

第 7题：

assign用于简单数据类型，如NSInteger,double,bool,retain 和copy用户对象，copy用于当 a指向一个对象，b也想指向同样的对象的时候，如果用assign，a如果释放，再调用b会crash,如果用copy 的方式，a和b各自有自己的内存，就可以解决这个问题。retain 会使计数器加一，也可以解决assign的问题。另外：tomic和nonatomic用来决定编译器生成的getter和setter是否为原子操作。在多线程环境下，原子操作是必要的，否则有可能引起错误的结果。

外企英文面试题汇总: 40道。

<http://blog.csdn.net/dumengqiu/article/details/42059487>

1.Difference between shallow copy and deep copy?

2.What is advantage of categories? What is difference between implementing a category and inheritance?

3.Difference between categories and extensions?

4.Difference between protocol in objective c and interfaces in java?

5.What are KVO and KVC?

6.What is purpose of delegates?

7.What are mutable and immutable types in Objective C?

8.When we call objective c is runtime language what does it mean?

9.what is difference between NSNotification and protocol?

10.What is push notification?

11.Polymorphism?

12.Singleton?

13.What is responder chain?

14.Difference between frame and bounds?

15.Difference between method and selector?

16.Is there any garbage collection mechanism in Objective C.?

17.NSOperation queue?

18.What is lazy loading?

19.Can we use two tableview controllers on one viewcontroller?

20.Can we use one tableview with two different datasources? How you will achieve this?

21.What is advantage of using RESTful webservices?

22.When to use NSMutableArray and when to use NSArray?

23.What is the difference between REST and SOAP?

24.Give us example of what are delegate methods and what are data source methods of uitableview.

25.How many autorelease you can create in your application? Is there any limit?

26.If we don’t create any autorelease pool in our application then is there any autorelease pool already provided to us?

27.When you will create an autorelease pool in your application?

28.When retain count increase?

29.Difference between copy and assign in objective c?

30.What are commonly used NSObject class methods?

31.What is convenience constructor?

32.How to design universal application in Xcode?

33.What is keyword atomic in Objective C?

34.What are UIView animations?

35.How can you store data in iPhone applications?

36.What is coredata?

37.What is NSManagedObject model?

38.What is NSManagedobjectContext?

39.What is predicate?

40.What kind of persistence store we can use with coredata?