[iphone笔试题收集](http://blog.csdn.net/edward0004/article/details/7264816)

1.**写一个NSString类的实现**

**+ (id)initWithCString:(const char\*)nullTerminatedCString encoding:(NSStringEncoding)encoding{**

**NSString\*obj;**

**obj= [self allocWithZone: NSDefaultMallocZone()];**

**obj= [obj initWIthCString: nullTerminatedCString encoding: encoding];**

**returnAUTORELEASE(obj);**

**}**

**2.写一个委托的interface**

**@protocol MyDelegate;**

**@interface MyClass: NSObject**

**{**

**id<MyDelegate> delegate;**

**}**

**// 委托方法**

**@protocol MyDelegate**

**- (void)didJobs:(NSArray \*)args;**

**@end**

**3.obj-c有私有方法么?私有变量呢?**

**答：在Objective-C中没有私有方法，只有静态方法和实例方法。但是可以用@private修饰私有变量。**

**4.#import跟#include的区别?**

答：前者不会引起交叉编译的问题。因为在Objective-C中会存在C/C++和Object-C混编的问题，如果用include引入头文件，会导致交叉编译。

**5.请写出你对MVC模式的理解**

**答：MVC模式考虑三种对象：模型对象、视图对象和控制器对象。模型对象负责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；控制器对象是M与V之间的协调者.**

**6.什么是键-值,键路径是什么**

**答：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.**

**7.什么是目标-动作机制**

**答：http://www.apple.com.cn/developer/mac/library/documentation/Cocoa/Conceptual/CocoaFundamentals/CommunicatingWithObjects/chapter\_6\_section\_5.html**

    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"](http://www.apple.com.cn/developer/mac/library/documentation/Cocoa/Conceptual/CocoaFundamentals/CommunicatingWithObjects/chapter_6_section_3.html" \l "//apple_ref/doc/uid/TP40002974-CH7-SW3)部分）的形式保有其动作消息的目标。虽然目标可以是任何实现恰当的动作方法的Cocoa对象，但通常是您的定制类的一个实例。

    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。控件，或者更为常见的是它的单元，将动作存储为SEL类型的实例变量。SEL是一种Objective-C的数据类型，用于指定消息的签名。动作消息必须有一个简单而清楚的签名，消息调用的方法没有返回值，且只有一个类型为id的参数。

    目标和动作机制的连接：您可以在程序中设置单元和控件的目标和动作，也可以在Interface Builder中进行。对于大多数的开发者和大多数的使用场合，Interface Builder是更好的方法。当您使用Interface Builder设置控件和目标时，它可以提供视觉上的确认，并且可以支持锁定连接，以及将连接归档到nib文件中。

8.请列举你熟悉cocoatouch框架(至少三个)

答：

Core Animation

通过 Core Animation，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。

Core Audio

Core Audio 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。

Core Data

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 Cocoa Touch 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 Bonjour WebKit BSD Sockets
* 用户应用 Address Book Core Location Map Kit Store Kit

9.**请写出obj-c的内存管理代码**

**答：**

**方法描述**

**-retain**

**将一个对象的reference数量增加1。**

**-release**

**将一个对象的reference数量减少1。**

**-autorelease**

**在未来某些时候将reference数量减少1.**

**-alloc**

**为一个对象分配内存，并设置保留值数量(retain count)为1。**

**-copy**

**复制一个对象，并将其做为返回值。同时设置保留值数量(retain count)为1。**

**保留值数量规则**

**1 在一定的代码段中，使用-copy，-alloc和-retain的次数应该和-release，-autorelease保持一致。**

**2 使用便利构造方法创建的对象(比如NSString的stringWithString)可以被认为会被自动释放。(autoreleased)**

**3 在使用你自己的参数实例时，需要实现-dealloc方法来释放。**

**ojc-c 是 通过一种"referring counting"(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.**

**10.自动释放池是什么,如何工作**

**答：当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。**

**11.obj-c的优缺点**

**答：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) 不支援命名空间    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。

**12.readwrite，readonly，assign，retain，copy，nonatomic属性的作用**

**答：@property是一个属性访问声明，扩号内支持以下几个属性： 答：**

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。

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

答：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。

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

**会引起循环引用**

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。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。

15.什么是Notification？

答：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）。参考链接：http://useyourloaf.com/blog/2010/6/6/delegation-or-notification.html

16.什么时候用delegate，什么时候用Notification？

答：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。

17.什么是KVC和KVO？

答：KVC(Key-Value-Coding)内部的实现：一个对象在调用setValue的时候，（1）首先根据方法名找到运行方法的时候所需要的环境参数。（2）他会从自己isa指针结合环境参数，找到具体的方法实现的接口。（3）再直接查找得来的具体的方法实现。KVO（Key-Value-Observing）：当观察者为一个对象的属性进行了注册，被观察对象的isa指针被修改的时候，isa指针就会指向一个中间类，而不是真实的类。所以isa指针其实不需要指向实例对象真实的类。所以我们的程序最好不要依赖于isa指针。在调用类的方法的时候，最好要明确对象实例的类名。

18.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 。

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

答：

1. //iPhone的内存有限，操作系统会通过先调用
2. DidReceiveMemoryWarning()方法，
3. //释放可用内存供给应用程序，并及时发出警告提示，

调用[super didReceiveMemoryWarning].

20.OOP

OOP 是一种用于解决问题的编程方法或通用方法。与之相反，算法是用于解决特定问题的特定方法。OOP 天生是一种强有力的方法；它往往使过程型和函数型编程方法与该问题较少相关，并且除非它与过程型和函数型编程方法的混合对其极其有益，否则它们之间不会很好地结合在一起。

    通常，有三个语言特性是 OOP 编程语言所必需的。它们是继承、多态性和封装。

   有两种**继承**：单一继承和多重继承。 单一继承要求子对象只有一个父对象，而 多重继承更自由（与实际生活一样，在编程过程中具有两个以上的父对象会导致混乱并使子对象难以工作，因此，不要过多使用多重继承）。尽管两个或两个以上的父对象实际上很少见，但 Perl 支持多重继承。

**多态性**（来自希腊语，表示“很多形态”）是使一个对象被看成另一个对象的技术。这有点复杂，那么我举个例子。比方说，您有一个绵羊牧场，里面有四只绵羊（绵羊属），但是您刚刚买了两只山羊（山羊属）和一只德国牧羊犬（犬科犬属）。您一共有多少动物？您得把所有的绵羊、山羊和狗加起来，结果是 7 只。其实您刚刚应用了多态性，即为了计算，把三种不同种类的动物当成一种通用类型（“动物”）对待。如果您把绵羊、山羊和狗当成哺乳动物看待，这就是一个简单的信心飞跃。生物学家每天都以这种方式使用多态性，而程序员则以从其它科学领域“窃用”（我是指“重用”）好主意闻名。  **封装**指的是以这样一种方式包含对象行为和特性：除非对象作者允许，否则用户无法访问该对象的行为和特性。在这种方式下，对象用户无法做不准他们做的事，无法访问不准他们访问的数据，并且通常是有害数据。

21.

Category准确的定义是这样的：Category拥有一个名字，它不需要使用子类（Subclassing）机制，就允许向一个类文件中添加新的方法声明，并且在类实现的文件中的同一个名字下定义这些方法。其语法举例如下：

|  |
| --- |
| #import "ClassName.h" |
|  |
| @interface ClassName ( CategoryName ) |
| // method declarations |
| @end |

不过到现在为止，Category这个名字看起来仍然让人摸不着头脑——Category的中文是分类和范畴的意思——即便这个动态机制很强大，跟分类有什么关系呢？

这是因为利用这个机制，程序员可以把一堆方法分门别类，分成若干组，每组方法用一个Category名字加以命名，定义在同一个文件里。这个就是为什么把这个机制叫做Category的原因。

注意Category只能用于方法，不能用于成员变量。

理解了Category，Extension就不难理解了。Extension是Category的一个特例，其名字为匿名（为空），并且新添加的方法一定要予以实现。（Category没有这个限制）

**Categories**

catgory 允许你为一个已经存在的类增加方法----甚至是一个你没有source的类。

Categories是一种强大的特性，它允许你直接扩展类的功能，而不需要使用子类的方法来扩展。

使用categories，你可以把你自己的类的实现方法分布在几个不同的文件里。

Class extensions与此相似，但是它允许在@ implementation代码块额外的增加自己需要的APIs，而不是在原始类的@interface代码块里。

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

答：Objective-C 对存取权限的设定。也是变量的作用域。

protected —Methods defined in the class and any subclasses can directly access the instance variables that follow.This is the default case. 该类和所有的子类中的方法可以直接访问这样的变量，这是默认的。

private —Methods defined in the class can directly access the instance variables that follow, but subclasses cannot. 该类中的方法可以访问这样的变量，子类不可以。

public —Methods defined in the class and any other classes or modules can di- rectly access the instance variables that follow. 除了自己和子类中的方法外，也可以被其他类或者其他模块中的方法所访问。开放性最大。

package —For 64-bit images, the instance variable can be accessed anywhere within the image that implements the class. 对于64位图像，这样的成员变量可以在实现这个类的图像中随意访问。

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

[Objective-C教程](http://www.ibokanwisdom.com/)：Objective-C中，类的实例化变量的范围有@private、@protected、@public。他们代表的意思和C++中相同，只是前面添加了一个@符号。下面介绍一下他们代表的范围：

下面是官方的图：

具体的可以参考：

http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Chapters/ocDefiningClasses.html#//apple\_ref/doc/uid/TP30001163-CH12-SW1

下面是一个例子，来说明他们之间的关系：

实现一个Boss类：

实现一个Worker类继承Boss类：

在Worker的.m文件中添加：

 ---------------------------------------------------------------------------------------------------------------------------------------

**23.委托是什么？**

**答：委托是一种简单而强大的模式。在此模式中，程序中的一个对象代表另外一个对象执行某个动作，或者与之相互协作共同完成某个任务。发布委托的对象持有其他对象（委托）的引用。在适当的时候，它会向委托发送消息。消息用于通知委托对象发布委托的对象将要处理或者已经处理某个事件。作为响应，委托对象会更新外观或者更新自身或应用程序其他对象的状态。在某些情况下，委托对象也可以返回一个值，通过它来影响事件（即将被处理的事件）的处理方式。通过委托，我们可以以某个对象为中心，轻松定制周围的数个对象。**

24．什么时候用到深浅拷贝？

答：如果对一个不可变对象复制，copy是指针复制，即浅拷贝；而mutableCopy则是对象复制，即深拷贝。

下面举几个例子说明。 **非容器类对象**如NSString，NSNumber等一类对象

示例1：

1. // 非容器类对象     NSString \*str = @"origin string";

2.    NSString \*strCopy = [str copy];

3.    NSMutableString \*mstrCopy = [str mutableCopy];

4.    [mstrCopy appendString:@"??"];

查看内存可以发现，str和strCopy指向的是同一块内存区域，我们称之为弱引用（weak reference）。而mstrCopy是真正的复制，系统为其分配了新内存空间，保存从str复制过来的字符串值。从最后一行代码中修改这些值而不影响str和strCopy中可证明。

示例2：

1. NSMutableString \*mstr = [NSMutableString stringWithString:@"origin"];

2.    NSString \*strCopy = [mstr copy];

3.    NSMutableString \*mstrCopy = [mstr copy];

4.    NSMutableString \*mstrMCopy = [mstr mutableCopy];

5.    //[mstrCopy appendString:@"1111"];  //error

6.    [mstr appendString:@"222"];

7.    [mstrMCopy appendString:@"333"];

以上四个对象所分配的内存都是不一样的。而且对于mstrCopy，它所指向的其实是一个imutable对象，是不可改变的，所以会出错。这点要注意，好好理解。

24.cocoa中有虚基类的概念吗？怎么简洁的实现？

答：Obj-c语言里面没有指明有虚基类, 不过实现虚基类更简单, 只要建立一个类, 子类从它派生. 运行时环境会自动确定子类型. 你也可以主动通过[obj isKindOfClass:];方法鉴定对象类型.  如果不希望基类生成的对象方法被调用, 只要 - (NSString \*)myMethod

25.[pool release]和[pool drain]区别？

答：In a garbage collected environment, release is a no-op. NSAutoreleasePool therefore provides a [drain](http://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSAutoreleasePool_Class/Reference/Reference.html" \l "//apple_ref/occ/instm/NSAutoreleasePool/drain) method that in a reference-counted environment behaves the same as calling release, but which in a garbage collected environment triggers garbage collection (if the memory allocated since the last collection is greater than the current threshold). Typically, therefore, you should use drain rather than release to dispose of an autorelease pool. 在一个garbage collected环境里，release不做任何操作。 NSAutoreleasePool因此提供了一个 [drain](http://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSAutoreleasePool_Class/Reference/Reference.html" \l "//apple_ref/occ/instm/NSAutoreleasePool/drain) 方法，它在reference-counted环境中的行为和调用release一样, 但是在一个garbage collected环境中则触发garbage collection动作 (if the memory allocated since the last collection is greater than the current threshold)。 因此通常你应该使用drain而不是release去释放一个autorelease pool {     [self doesNotRecognizeSelector:\_cmd];     return nil; } 就可以了.

26.自动释放池写在循环内好,还是写在循环外好?

答：如果新建自己的pool可根据情况选择放的位置，循环次数少可放在外面，次数多放在里面，这样不用等循环结束就可以进行释放了。如果觉得pool每次alloc和dealloc不舒服，可以每循环几次做一下。如果用系统本身的pool，就把变量尽量放在外面。

27.[Iphone 实现Singleton(单例)模式](http://blog.csdn.net/zypsg/article/details/4314976)?

答：

Singleton模式经常来做应用程序级别的共享资源控制, 应该说这个模式的使用频率非常高, 现在来看看在Objective-C里面的实现方法. 要实现一个Singleton Class, 至少需要做以下四个步骤: 1. 为Singleton Object实现一个静态实例, 初始化, 然后设置成nil. 2. 实现一个实例构造方法(通常命名为 sharedInstance 或者 sharedManager)检查上面声名的静态实例是否为nil, 如果是则新建并返回一个本类实例. 3. 重写 allocWithZone: 方法来保证当其他人直接使用 alloc 和 init 试图获得一个新实例的时候不会产生一个新的实例. 4. 适当的实现 copyWithZone:, release, retain, retainCount 和 autorelease.

http://blog.csdn.net/zypsg/article/details/4314976

28。什么是序列化或者Archiving？可以用来做什么？怎么跟copy结合？

答：https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Archiving/Archiving.html

29

.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].

30.以下为Windows NT下的32位C++程序，请计算sizeof的值

void Func ( char str[100] )

{

  sizeof( str ) = ?

}

void \*p = malloc( 100 );

sizeof ( p ) = ?

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。

31.还是考指针,不过我对cocoa的代码还是不太熟悉

大概是这样的

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

{

[inputString = @"This](mailto:*output=@"This) is a main test\n";

     return ;

}

-main(void)

{

NSString [\*a=@"Main](mailto:*a=@"Main)";

 NSString \*aString = [NSString stringWithString:@"%@",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。

32 .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。

33.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。

34.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。

35.KVO是同步还是异步？notification是同步还是异步？

答：1。 不知道。  2。Notification center使用同步的传递一个notification给它的观察者们，你也可以使用一个notification queue来发送一个异步的notification。

36.ios对象的生命周期？

答：**对象生命周期**

对象的生命周期—即对象在运行时从创建到销毁的过程—取决于对象接收到的各种消息。当程序显式分配并初始化一个对象（或者复制另外一个对象）之后，对象就诞生了。当您从档案字节流解码出某个对象的时候，对象就开始其运行时的生命。如果对象从 nib 文件里面解档而来，当 nib 文件所有对象被加载到内存后，该对象会接收到一条[awakeFromNib](http://developer.apple.com/iphone/library/documentation/UIKit/Reference/NSObject_UIKitAdditions/Introduction/Introduction.html" \l "/apple_ref/occ/instm/NSObject/awakeFromNib)消息。

在创建和初始化完成后，只要对象保持数量大于零，就会停留在内存。程序中的其他对象可向其发送 [retain](http://developer.apple.com/iphone/library/documentation/Cocoa/Reference/Foundation/Protocols/NSObject_Protocol/Reference/NSObject.html" \l "/apple_ref/occ/intfm/NSObject/retain)消息或者复制该对象，以此来拥有该对象。稍后它们可以向对象发送[release](http://developer.apple.com/iphone/library/documentation/Cocoa/Reference/Foundation/Protocols/NSObject_Protocol/Reference/NSObject.html" \l "/apple_ref/occ/intfm/NSObject/release) 消息，这样就可以撤销拥有权。当对象接收到最后一条release消息后，其保持数量会下降至零，这时，对象的[dealloc](http://developer.apple.com/iphone/library/documentation/Cocoa/Reference/Foundation/Classes/NSObject_Class/Reference/Reference.html" \l "/apple_ref/occ/instm/NSObject/dealloc)被调用。该方法将会释放对象持有的全部对象以及所分配的内存，然后这个对象会被销毁。