　　1. Object-c的类可以多重继承么?可以实现多个接口么?Category是什么?重写一个类的方式用继承好还是分类好?为什么?

答： Object-c的类不可以多重继承;可以实现多个接口，通过实现多个接口可以完成C++的多重继承;Category是类别，一般情况用分类好，用Category去重写类的方法，仅对本Category有效，不会影响到其他类与原有类的关系。

　　2. #import 跟#include 又什么区别，@class呢, #import<> 跟 #import”” 又什么区别?

　　答：#import是Objective-C导入头文件的关键字，#include是C/C++导入头文件的关键字,使用#import头文件会自动只导入一次，不会重复导入，相当于#include和#pragma once;@class告诉编译器某个类的声明，当执行时，才去查看类的实现文件，可以解决头文件的相互包含;#import<>用来包含系统的头文件，#import”” 用来包含用户头文件。

　　3. 属性readwrite，readonly，assign，retain，copy，nonatomic 各是什么作用，在那种情况下用?

　　1. readwrite 是可读可写特性;需要生成getter方法和setter方法时

　　2. readonly 是只读特性 只会生成getter方法 不会生成setter方法 ;不希望属性在类外改变

　　3. assign 是赋值特性，setter方法将传入参数赋值给实例变量;仅设置变量时;

　　4. retain 表示持有特性，setter方法将传入参数先保留，再赋值，传入参数的retaincount会+1;

　　5. copy 表示赋值特性，setter方法将传入对象复制一份;需要完全一份新的变量时。

　　6. nonatomic 非原子操作，决定编译器生成的setter getter是否是原子操作，atomic表示多线程安全，一般使用nonatomic

　　4.写一个setter方法用于完成@property (nonatomic,retain)NSString \*name,写一个setter方法用于完成@property(nonatomic，copy)NSString \*name

　　- (void) setName:(NSString\*) str

　　{

　　[str retain];

　　[name release];

　　name = str;

　　}

　　- (void)setName:(NSString \*)str

　　{

　　id t = [str copy];

　　[name release];

　　name = t;

　　}

　　5.对于语句NSString\*obj = [[NSData alloc] init]; obj在编译时和运行时分别时什么类型的对象?

　　编译时是NSString的类型;运行时是NSData类型的对象

　　6.常见的object-c的数据类型有那些， 和C的基本数据类型有什么区别?如：NSInteger和int

　　object-c的数据类型有NSString，NSNumber，NSArray，NSMutableArray，NSData等等，这些都是class，创建后便是对象，而C语言的基本数据类型int，只是一定字节的内存空间，用于存放数值;NSInteger是基本数据类型，并不是NSNumber的子类，当然也不是NSObject的子类。NSInteger是基本数据类型Int或者Long的别名(NSInteger的定义typedef long NSInteger)，它的区别在于，NSInteger会根据系统是32位还是64位来决定是本身是int还是Long。

　　7.id 声明的对象有什么特性?

　　Id 声明的对象具有运行时的特性，即可以指向任意类型的objcetive-c的对象;

　　8.Objective-C如何对内存管理的,说说你的看法和解决方法?

　　Objective-C的内存管理主要有三种方式ARC(自动内存计数)、手动内存计数、内存池。

　　1. (Garbage Collection)自动内存计数：这种方式和java类似，在你的程序的执行过程中。始终有一个高人在背后准确地帮你收拾垃圾，你不用考虑它什么时候开始工作，怎样工作。你只需要明白，我申请了一段内存空间，当我不再使用从而这段内存成为垃圾的时候，我就彻底的把它忘记掉，反正那个高人会帮我收拾垃圾。遗憾的是，那个高人需要消耗一定的资源，在携带设备里面，资源是紧俏商品所以iPhone不支持这个功能。所以“Garbage Collection”不是本入门指南的范围，对“Garbage Collection”内部机制感兴趣的同学可以参考一些其他的资料，不过说老实话“Garbage Collection”不大适合适初学者研究。

　　解决: 通过alloc – initial方式创建的, 创建后引用计数+1, 此后每retain一次引用计数+1, 那么在程序中做相应次数的release就好了.

　　2. (Reference Counted)手动内存计数：就是说，从一段内存被申请之后，就存在一个变量用于保存这段内存被使用的次数，我们暂时把它称为计数器，当计数器变为0的时候，那么就是释放这段内存的时候。比如说，当在程序A里面一段内存被成功申请完成之后，那么这个计数器就从0变成1(我们把这个过程叫做alloc)，然后程序B也需要使用这个内存，那么计数器就从1变成了2(我们把这个过程叫做retain)。紧接着程序A不再需要这段内存了，那么程序A就把这个计数器减1(我们把这个过程叫做release);程序B也不再需要这段内存的时候，那么也把计数器减1(这个过程还是release)。当系统(也就是Foundation)发现这个计数器变成了0，那么就会调用内存回收程序把这段内存回收(我们把这个过程叫做dealloc)。顺便提一句，如果没有Foundation，那么维护计数器，释放内存等等工作需要你手工来完成。

　　解决:一般是由类的静态方法创建的, 函数名中不会出现alloc或init字样, 如[NSString string]和[NSArray arrayWithObject:], 创建后引用计数+0, 在函数出栈后释放, 即相当于一个栈上的局部变量. 当然也可以通过retain延长对象的生存期.

　　3. (NSAutoRealeasePool)内存池：可以通过创建和释放内存池控制内存申请和回收的时机.

　　解决:是由autorelease加入系统内存池, 内存池是可以嵌套的, 每个内存池都需要有一个创建释放对, 就像main函数中写的一样. 使用也很简单, 比如[[[NSString alloc]initialWithFormat:@”Hey you!”] autorelease], 即将一个NSString对象加入到最内层的系统内存池, 当我们释放这个内存池时, 其中的对象都会被释放.

　　9. 原子(atomic)跟非原子(non-atomic)属性有什么区别?

　　1. atomic提供多线程安全。是防止在写未完成的时候被另外一个线程读取，造成数据错误

　　2. non-atomic:在自己管理内存的环境中，解析的访问器保留并自动释放返回的值，如果指定了 nonatomic ，那么访问器只是简单地返回这个值。

　　10. 看下面的程序,第一个NSLog会输出什么?这时str的retainCount是多少?第二个和第三个呢? 为什么?

　　=======================================================

　　NSMutableArray\* ary = [[NSMutableArray array] retain];

　　NSString \*str = [NSString stringWithFormat:@"test"];

　　[strretain];

　　[aryaddObject:str];

　　NSLog(@”%@%d”,str,[str retainCount]);

　　[strretain];

　　[strrelease];

　　[strrelease];

　　NSLog(@”%@%d”,str,[str retainCount]);

　　[aryremoveAllObjects];

　　NSLog(@”%@%d”,str,[str retainCount]);

　　=======================================================

　　str的retainCount创建+1，retain+1，加入数组自动+1 3

　　retain+1，release-1，release-1 2

　　数组删除所有对象，所有数组内的对象自动-1 1

　　11. 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?

　　谁申请，谁释放

　　遵循Cocoa Touch的使用原则;

　　内存管理主要要避免“过早释放”和“内存泄漏”，对于“过早释放”需要注意@property设置特性时，一定要用对特性关键字，对于“内存泄漏”，一定要申请了要负责释放，要细心。

　　关键字alloc 或new 生成的对象需要手动释放;

　　设置正确的property属性，对于retain需要在合适的地方释放，

　　12.如何对iOS设备进行性能测试?

　　Profile-> Instruments ->Time Profiler

　　13. Object C中创建线程的方法是什么?如果在主线程中执行代码，方法是什么?如果想延时执行代码、方法又是什么?

　　线程创建有三种方法：使用NSThread创建、使用GCD的dispatch、使用子类化的NSOperation,然后将其加入NSOperationQueue;在主线程执行代码，方法是performSelectorOnMainThread，如果想延时执行代码可以用performSelector:onThread:withObject:waitUntilDone:

　　14.描述一下iOS SDK中如何实现MVC的开发模式

　　MVC是模型、试图、控制开发模式，对于iOS SDK，所有的View都是视图层的，它应该独立于模型层，由视图控制层来控制。所有的用户数据都是模型层，它应该独立于视图。所有的ViewController都是控制层，由它负责控制视图，访问模型数据。

　　15 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。

　　16. 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，修改或者删除方法，并且可以增加属性。

　　17. 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。

　　18. oc中的协议和java中的接口概念有何不同?

　　答案：OC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)”。

　　19. 什么是KVO和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

　　[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。

　　20. 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。

　　21. oc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。

　　22. 我们说的oc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。

　　23. 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。

　　24. 是推送消息?

　　这是cocoa上的答案。

　　其实到不是说太简单，只是太泛泛的一个概念的东西。就好比说，什么是人。

　　推送通知更是一种技术。

　　简单点就是客户端获取资源的一种手段。

　　普通情况下，都是客户端主动的pull。

　　推送则是服务器端主动push。 测试push的实现可以查看该博文。

　　25. 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。

　　26. 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。

　　27. 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。

　　可以从责任链模式，来讲通过事件响应链处理，其拥有的扩展性

　　28. 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)

　　29. 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。

　　30. OC的垃圾回收机制?

　　答案： OC2.0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。

　　31. 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。

　　32. 什么是延迟加载?

　　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。

　　33. 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

　　34. 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?

　　35. 什么时候使用NSMutableArray，什么时候使用NSArray?

　　答案：当数组在程序运行时，需要不断变化的，使用NSMutableArray，当数组在初始化后，便不再改变的，使用NSArray。需要指出的是，使用NSArray只表明的是该数组在运行时不发生改变，即不能往NSAarry的数组里新增和删除元素，但不表明其数组內的元素的内容不能发生改变。NSArray是线程安全的，NSMutableArray不是线程安全的，多线程使用到NSMutableArray需要注意。

　　36. 给出委托方法的实例，并且说出UITableVIew的Data Source方法

　　答案：CocoaTouch框架中用到了大量委托，其中UITableViewDelegate就是委托机制的典型应用，是一个典型的使用委托来实现适配器模式，其中UITableViewDelegate协议是目标，tableview是适配器，实现UITableViewDelegate协议，并将自身设置为talbeview的delegate的对象，是被适配器，一般情况下该对象是UITableViewController。

UITableVIew的Data Source方法有

-(NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section;

　　- (UITableViewCell\*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath;

　　37. 在应用中可以创建多少autorelease对象，是否有限制?

　　答案：无

　　38. 如果我们不创建内存池，是否有内存池提供给我们?

　　答案:界面线程维护着自己的内存池，用户自己创建的数据线程，则需要创建该线程的内存池

　　39. 什么时候需要在程序中创建内存池?

　　答案：用户自己创建的数据线程，则需要创建该线程的内存池

　　40. 类NSObject的那些方法经常被使用?

　　答案：NSObject是Objetive-C的基类，其由NSObject类及一系列协议构成。

　　其中类方法alloc、class、description 对象方法init、dealloc、– performSelector:withObject:afterDelay:等经常被使用

　　41. 什么是简便构造方法?

　　答案：简便构造方法一般由CocoaTouch框架提供，如NSNumber的 + numberWithBool: + numberWithChar: + numberWithDouble: + numberWithFloat: + numberWithInt:

　　Foundation下大部分类均有简便构造方法，我们可以通过简便构造方法，获得系统给我们创建好的对象，并且不需要手动释放。

　　42. 如何使用Xcode设计通用应用?

　　答案：使用MVC模式设计应用，其中Model层完成脱离界面，即在Model层，其是可运行在任何设备上，在controller层，根据iPhone与iPad(独有UISplitViewController)的不同特点选择不同的viewController对象。在View层，可根据现实要求，来设计，其中以xib文件设计时，其设置其为universal。

　　43. UIView的动画效果有那些?

答案：有很多，如

UIViewAnimationOptionCurveEaseInOut

UIViewAnimationOptionCurveEaseIn

UIViewAnimationOptionCurveEaseOut

UIViewAnimationOptionTransitionFlipFromLeft UIViewAnimationOptionTransitionFlipFromRight UIViewAnimationOptionTransitionCurlUpUIViewAnimationOptionTransitionCurlDown

　　44. 在iPhone应用中如何保存数据?

　　答案：有以下几种保存机制：

　　1.通过web服务，保存在服务器上

　　2.通过NSCoder固化机制，将对象保存在文件中

　　3.通过SQlite或CoreData保存在文件数据库中

　　45. 什么是coredata?

　　答案：coredata是苹果提供一套数据保存框架，其基于SQlite

　　46. 什么是NSManagedObject模型?

　　答案：NSManagedObject是NSObject的子类 ，也是coredata的重要组成部分，它是一个通用的类,实现了core data 模型层所需的基本功能，用户可通过子类化NSManagedObject，建立自己的数据模型。

　　47. 什么是NSManagedobjectContext?

　　答案：NSManagedobjectContext对象负责应用和数据库之间的交互。

　　48. 什么是谓词?

　　答案：谓词是通过NSPredicate，是通过给定的逻辑条件作为约束条件，完成对数据的筛选。

　　predicate = [NSPredicate predicateWithFormat:@"customerID == %d",n];

　　a = [customers filteredArrayUsingPredicate:predicate];

　　49. 和coredata一起有哪几种持久化存储机制?

　　答案：存入到文件、 存入到NSUserDefaults(系统plist文件中)、存入到Sqlite文件数据库

　　50. 谈谈对Block 的理解?并写出一个使用Block执行UIVew动画?

　　答案：Block是可以获取其他函数局部变量的匿名函数，其不但方便开发，并且可以大幅提高应用的执行效率(多核心CPU可直接处理Block指令)

　　[UIView transitionWithView:self.view

　　duration:0.2

　　options:UIViewAnimationOptionTransitionFlipFromLeft

　　animations:^{ [[blueViewController view] removeFromSuperview]; [[self view] insertSubview:yellowViewController.view atIndex:0]; }

　　completion:NULL];

　　51. 写出上面代码的Block的定义。

　　答案：

　　typedef void(^animations) (void);

　　typedef void(^completion) (BOOL finished);

　　52. 试着使用+ beginAnimations:context:以及上述Block的定义，写出一个可以完成

+(void)transitionWithView:(UIView\*)viewduration:(NSTimeInterval)duration options:(UIViewAnimationOptions)optionsanimations:(void(^)(void))animations completion:(void (^)(BOOL finished))completion NS\_AVAILABLE\_IOS(4\_0);操作的函数执行部分

　　53. 做过的项目是否涉及网络访问功能，使用什么对象完成网络功能?

　　答案：ASIHTTPRequest与NSURLConnection

　54.简单介绍下NSURLConnection类及+ sendSynchronousRequest:returningResponse:error:

与– initWithRequest:delegate:两个方法的区别?

答案: NSURLConnecti主要用于网络访问，其中+ sendSynchronousRequest:returningResponse:error:是同步访问数据，即当前线程会阻塞，并等待request的返回的response，而– initWithRequest:delegate:使用的是异步加载，当其完成网络访问后，会通过delegate回到主线程，并其委托的对象。

　　55. 多线程是什么

　　多线程是个复杂的概念，按字面意思是同步完成多项任务，提高了资源的使用效率，从硬件、操作系统、应用软件不同的角度去看，多线程被赋予不同的内涵，对于硬件，现在市面上多数的CPU都是多核的，多核的CPU运算多线程更为出色;从操作系统角度，是多任务，现在用的主流操作系统都是多任务的，可以一边听歌、一边写博客;对于应用来说，多线程可以让应用有更快的回应，可以在网络下载时，同时响应用户的触摸操作。在iOS应用中，对多线程最初的理解，就是并发，它的含义是原来先做烧水，再摘菜，再炒菜的工作，会变成烧水的同时去摘菜，最后去炒菜。

　　56. iOS 中的多线程

　　iOS中的多线程，是Cocoa框架下的多线程，通过Cocoa的封装，可以让我们更为方便的使用线程，做过C++的同学可能会对线程有更多的理解，比如线程的创立，信号量、共享变量有认识，Cocoa框架下会方便很多，它对线程做了封装，有些封装，可以让我们创建的对象，本身便拥有线程，也就是线程的对象化抽象，从而减少我们的工程，提供程序的健壮性。

　　GCD是(Grand Central Dispatch)的缩写 ，从系统级别提供的一个易用地多线程类库，具有运行时的特点，能充分利用多核心硬件。GCD的API接口为C语言的函数，函数参数中多数有Block，关于Block的使用参看这里，为我们提供强大的“接口”，对于GCD的使用参见本文

　　NSOperation与Queue

　　NSOperation是一个抽象类，它封装了线程的细节实现，我们可以通过子类化该对象，加上NSQueue来同面向对象的思维，管理多线程程序。具体可参看这里：一个基于NSOperation的多线程网络访问的项目。

　　NSThread

　　NSThread是一个控制线程执行的对象，它不如NSOperation抽象，通过它我们可以方便的得到一个线程，并控制它。但NSThread的线程之间的并发控制，是需要我们自己来控制的，可以通过NSCondition实现。

　　参看 iOS多线程编程之NSThread的使用

　　其他多线程

　　在Cocoa的框架下，通知、Timer和异步函数等都有使用多线程，(待补充).

　　57. 在项目什么时候选择使用GCD，什么时候选择NSOperation?

　　项目中使用NSOperation的优点是NSOperation是对线程的高度抽象，在项目中使用它，会使项目的程序结构更好，子类化NSOperation的设计思路，是具有面向对象的优点(复用、封装)，使得实现是多线程支持，而接口简单，建议在复杂项目中使用。

　　项目中使用GCD的优点是GCD本身非常简单、易用，对于不复杂的多线程操作，会节省代码量，而Block参数的使用，会是代码更为易读，建议在简单项目中使用。

　　58. 什么是block

　　对于闭包(block),有很多定义，其中闭包就是能够读取其它函数内部变量的函数，这个定义即接近本质又较好理解。对于刚接触Block的同学，会觉得有些绕，因为我们习惯写这样的程序main(){ funA();} funA(){funB();} funB(){…..}; 就是函数main调用函数A，函数A调用函数B… 函数们依次顺序执行，但现实中不全是这样的，例如项目经理M，手下有3个程序员A、B、C，当他给程序员A安排实现功能F1时，他并不等着A完成之后，再去安排B去实现F2，而是安排给A功能F1，B功能F2，C功能F3，然后可能去写技术文档，而当A遇到问题时，他会来找项目经理M，当B做完时，会通知M，这就是一个异步执行的例子。在这种情形下，Block便可大显身手，因为在项目经理M，给A安排工作时，同时会告诉A若果遇到困难，如何能找到他报告问题(例如打他手机号)，这就是项目经理M给A的一个回调接口，要回掉的操作，比如接到电话，百度查询后，返回网页内容给A，这就是一个Block，在M交待工作时，已经定义好，并且取得了F1的任务号(局部变量)，却是在当A遇到问题时，才调用执行，跨函数在项目经理M查询百度，获得结果后回调该block。

　　59. block 实现原理

　　Objective-C是对C语言的扩展，block的实现是基于指针和函数指针。

　　从计算语言的发展，最早的goto，高级语言的指针，到面向对象语言的block，从机器的思维，一步步接近人的思维，以方便开发人员更为高效、直接的描述出现实的逻辑(需求)。

　　下面是两篇很好的介绍block实现的博文

　　iOS中block实现的探究

　　谈Objective-C Block的实现

　　3 block的使用

　　使用实例

　　cocoaTouch框架下动画效果的Block的调用

　　使用typed声明block

　　typedef void(^didFinishBlock) (NSObject \*ob);

　　这就声明了一个didFinishBlock类型的block，

　　然后便可用

　　@property (nonatomic,copy) didFinishBlock finishBlock;

　　声明一个blokc对象，注意对象属性设置为copy，接到block 参数时，便会自动复制一份。

　　\_\_block是一种特殊类型，

　　使用该关键字声明的局部变量，可以被block所改变，并且其在原函数中的值会被改变。

59.为什么retaincount不能用在发布的代码中

--retaincount受到时间和框架影响太大，不能准确反映内存的引用计数

--使用alloc和release等的时候要一一对应，不需要retaincount

60.请解释一下method swizzling，并说出你一般什么时候会用到

MethodSwizzling 是一种在运行时修改方法名与方法实现映射关系的技术，常用于在系统内部方法的基础上增加新的处理流程。

61.当一个空指针调用了一个方法的时候会发生什么

不会有所影响，这是oc自带的消息机智,nil也能发送消息，不会报错

62.查找或者解决内存泄露的处理过程

利用Xcode的工具

分析各种提示信息

Profile跟进一些具体的内存点

看一些block,delegate部分

使用instruments作为动态分析的手段，Xcode的静态内存分析

63.解释下自动回收池（qutorelease pool）在程序运行时如何运作

Xcode为开发者写的代码外层包了一层NSAutoreleasePool。建立一个回收池堆栈每次对象发送autorelease消息时,对象的引用计数并不是真正的变化，而是向pool中添加一条记录,记下对象的这种要求。最后当pool发送drain或release消息时，池中的多有对象要求被一一执行。

64．解释代码签名是如何运作的

1.代码签名也叫APP签名，是依靠IOS证书来进行的。保证了APP的合法、完整、安全。

2.代码签名的核心是：证书、公钥、私钥。

3.签名过程在底层是由命令行工具codesign来完成

4.代码签名的验证主要是通过配置文件 Provisioning Profile（PP文件）与个人私钥 Private Key 共同完成。

65.OC中的posing指什么

冒充！Posing存在的目的在于子类可以冒充父类，使得后续代码无需修改父类，就可以很方便的让父类表现为子类的行为。

一个类可能只对作为其直接或间接的父类之一。

冒充类绝不能定义任何新的实例变量目标类（虽然它可能定义或重写方法）不存在。

目标类可能没有收到任何消息之前冒充。

冒充类可以通过super调用覆盖的方法，从而将的实现目标类。

冒充类可以覆盖类别中定义的方法

66.列举5个IOS App的状态

程序开始，程序挂起(暂停)，程序进入后台，程序进入前台，程序重新激活，程序意外暂停

首次启动应用程序：程序开始，程序再次激活

按home建退出：程序暂停，程序进入后台

再次进入程序：程序进入前台，程序再次激活

IOS核心框架：

CoreAnimation

直接作用在CALayer上的(并非UIView),夸平台的动画处理API，动画执行过程都在后台，不会阻塞线程。

CoreGraphics

基于C的绘图API

CoreLocation

定位API

IOS 支持三种检测当前位置的方式：手机基站、Wi-Fi、和GPS，其中GPS是经度最高的，同时也是最耗费手机电量的。一般情况下在室内是无法通过GPS获 取位置信息的，通过Wi-Fi获取位置的原理是通过网络提供商的IP地址信息来获取位置，经度不是很高，最后是通过手机基站获取位置，手机开机后会连接附 近的基站塔获取信号，通过基站可以得到手机所在的位置信息，基站越密集，所获取的位置信息经度就越高。

AVFoundation

视频流处理

Foundation

提供WEB上的UI、控件等等

我们通常称为“Cocoa框架”，事实上Cocoa本身是一个框架的集合，它包含了众多子框架，其中最重要的要数“Foundation”和“UIKit”。前者是框架的基础，和界面无关，其中包含了大量常用的API；后者是基础的UI类库。

67. 什么情况使用weak关键字，相比assign有什么不同

1.在ARC中，在有可能出现循环引用的时候，往往要用过让其中一端使用weak来解决，比如代理属性。

2.自身已经对它进行了一次强引用，没有必要再引用一次。也会使用weak,自定义IBOutlet属性一般也用weak,也可用strong

区别：

1.weak此特质表明该属性定义了一种“非拥有关系”(nonowning relationship)。

为这种属性设置新值时，设置方法既不保留新值，也不释放旧值。此特质同assign类似,

然而在属性所指的对象遭到摧毁时，属性值也会清空(nil out)。而assign的“设置方法”只会执行针对“纯量类型”(scalar type,例如CGFloat或NSlnteger等)的简单赋值操作。

2.assigin 可以用非 OC 对象,而 weak 必须用于 OC 对象

68．Copy的使用

NSString,NSArray,NSDictionary等经常使用copy,因为它们有对应的可变类型：

NSMutableString、NSMtableArray、NSMutableDictionary。Block也经常使用。

69.这个写法会出现什么问题 @property (copy) NsmutableArray \*array;

1.添加，删除，修改数组内容的时候程序会因为找不到对应的方法二崩溃，因为copy就是复制一个不变Array对象。

2.使用了atomic属性会严重影响性能。

70.如何让自己的类用copy修饰符？如何重写带copy关键字的setter

若想令自己所写的对象具有拷贝功能，则需实现 NSCopying 协议。如果自定义的对象分为可变版本与不可变版本，那么就要同时实现 NSCopying 与 NSMutableCopying 协议。

具体步骤：

1. 声明该类遵从NSCopying协议
2. 实现NSCopying协议 - (id)copyWithZone:(NSZone \*)zone;

注意：一提到让自己的类用 copy 修饰符，我们总是想覆写copy方法，其实真正需要实现的却是 “copyWithZone” 方法。

71.@property本质是什么

@property = ivar + getter + setter

property有两大概念：ivar（实例变量）、存取方法（access method ＝ getter + setter)。

72. @protocol和category中如何使用@property

1.在protocol中使用property只会生成setter和getter方法声明,我们使用属性的目的,是希望遵守我协议的对象能实现该属性

2.category 使用@property也是只会生成setter和getter方法的声明,如果我们真的需要给category增加属性的实现,需要借助于运行时的两个函数：

objc\_setAssociatedObject

objc\_getAssociatedObject

73. runtime如何实现weak属性

runtime对注册的类,会进行布局,对于weak对象会放入一个has 表中。用wek指向的对象内存地址作为key，当此对象的引用计数为0的时候会dealloc，假如weak指向的对象内存地址是a，那么就会以a为键， 在这个weak表中搜索，找到所有以a为键的weak对象，从而设置为nil。

74. weak属性需要在dealloc中置nil么？

不需要，在ARC环境无论是强指针还是弱指针都无需设置nil, ARC会自动处理。

75. @synthesize和@dynamic分别有什么作用？

@property有两个对应的词，一个是@synthesize，一个是@dynamic。如果@synthesize和@dynamic都没写，那么默认的就是@syntheszie var = \_var;

1.@synthesize 的语义是如果你没有手动实现setter方法和getter方法，那么编译器会自动为你加上这两个方法。

2.@dynamic告诉编译器：属性的 setter 与 getter 方法由用户自己实现，不自动生成。（当然对于readonly的属性只需提供getter即可）。假如一个属性被声明为 @dynamic var，然后你没有提供 @setter方法和 @getter 方法，编译的时候没问题，但是当程序运行到 instance.var = someVar，由于缺setter方法会导致程序崩溃；或者当运行到someVar = var时，由于缺getter方法同样会导致崩溃。编译时没问题，运行时才执行相应的方法，这就是所谓的动态绑定。

76. 用@property声明的NSString（或NSArray，NSDictionary）经常使用copy关键字，为什么？如果改用strong关键字，可能造成什么问题？

1. 因为父类指针可以指向子类对象,使用copy的目的是为了让本对象的属性不受外界影响,使用copy无论给我传入是一个可变对象还是不可对象,我本身持有的就是一个不可变的副本.

如果我们使用是strong,那么这个属性就有可能指向一个可变对象,如果这个可变对象在外部被修改了,那么会影响该属性.

1. copy此特质所表达的所属关系与strong类似。然而设置方法并不保留新值，而是将其“拷贝”(copy)。 当属性类型为NSString时，经常用此特质来保护其封装性，因为传递给设置方法的新值有可能指向一个NSMutableString类的实例。这个类是NSString的子类，表示一种可修改其值的字符串，此时若是不拷贝字符串，那么设置完属性之后，字符串的值就可能会在对象不知情的情况下遭人更改。所以，这时就要拷贝一份“不可变”(immutable)的字符串，确保对象中的字符串值不会无意间变动。只要实现属性所用的对象是“可变的”(mutable)，就应该在设置新属性值时拷贝一份。

77. 什么时候会报unrecognized selector的异常？

当调用该对象上某个方法,而该对象上没有实现这个方法的时候.

78. 下面的代码输出什么？

@implementation Son : Father

- (id)init

{

self = [super init];

if (self) {

NSLog(@"%@", NSStringFromClass([self class]));

NSLog(@"%@", NSStringFromClass([super class]));

}

return self;

}

@end

答案：都输出 Son

NSStringFromClass([self class]) = Son

NSStringFromClass([super class]) = Son

这个题目主要是考察关于Objective-C中对self和super的理解。

我们都知道：self是类的隐藏参数，指向当前调用方法的这个类的实例。那super呢？很多人会想当然的认为“super和self类似，应该是指向父类的指针吧！”。这是很普遍的一个误区。其实super是一个Magic Keyword,它本质是一个编译器标示符，和self是指向的同一个消息接受者！他们两个的不同点在于：super会告诉编译器，调用class这个方法时，要去父类的方法，而不是本类里的。

上面的例子不管调用[self class]还是[super class]，接受消息的对象都是当前 Son ＊xxx 这个对象。

当使用self调用方法时，会从当前类的方法列表中开始找，如果没有，就从父类中再找；而当使用super时，则从父类的方法列表中开始找。然后调用父类的这个方法。这也就是为什么说“不推荐在init方法中使用点语法”，如果想访问实例变量iVar应该使用下划线（\_iVa ），而非点语法（self.iVar）。

点语法（self.iVar）的坏处就是子类有可能覆写setter。

79. objc中的类方法和实例方法有什么本质区别和联系？

类方法：

类方法是属于类对象的

类方法只能通过类对象调用

类方法中的self是类对象

类方法可以调用其他的类方法

类方法中不能访问成员变量

类方法中不定直接调用对象方法

实例方法：

实例方法是属于实例对象的

实例方法只能通过实例对象调用

实例方法中的self是实例对象

实例方法中可以访问成员变量

实例方法中直接调用实例方法

实例方法中也可以调用类方法(通过类名)

80. 能否向编译后得到的类中增加实例变量？能否向运行时创建的类中添加实例变量？为什么？

不能向编译后得到的类中增加实例变量；

能向运行时创建的类中添加实例变量；

解释：

因为编译后的类已经注册在 runtime 中，类结构体中的 objc\_ivar\_list 实例变量的链表和 I nstance\_size 实例变量的内存大小已经确定，同时runtime 会调用 class\_setIvarLayout 或 class\_setWeakIvarLayout 来处理 strong weak 引用。所以不能向存在的类中添加实例变量；

运行时创建的类是可以添加实例变量，调用 class\_addIvar 函数。但是得在调用 objc\_allocateClassPair 之后，objc\_registerClassPair 之前，原因同上。

81. runloop和线程有什么关系？

总的说来,Run loop，正如其名,loop表示某种循环，和run放在一起就表示一直在运行着的循环。实际上，run loop和线程是紧密相连的，可以这样说run loop是为了线程而生，没有线程，它就没有存在的必要。Run loops是线程的基础架构部分，Cocoa和CoreFundation都提供了run loop对象方便配置和管理线程的run loop（以下都以Cocoa为例）。每个线程，包括程序的主线程（main thread）都有与之相应的run loop对象。

1. 主线程的run loop默认是启动的
2. 对其它线程来说，run loop默认是没有启动的，如果你需要更多的线程交互则可以手动配置和启动，如果线程只是去执行一个长时间的已确定的任务则不需要。
3. 在任何一个 Cocoa 程序的线程中，都可以通过以下代码来获取到当前线程的run loop。

NSRunLoop \*runloop = [NSRunLoop currentRunLoop];

82. runloop的mode作用是什么？

model 主要是用来指定事件在运行循环中的优先级的，分为：

NSDefaultRunLoopMode（kCFRunLoopDefaultMode）：默认，空闲状态

UITrackingRunLoopMode：ScrollView滑动时

UIInitializationRunLoopMode：启动时

NSRunLoopCommonModes（kCFRunLoopCommonModes）：Mode集合

苹果公开提供的 Mode 有两个：

NSDefaultRunLoopMode（kCFRunLoopDefaultMode）

NSRunLoopCommonModes（kCFRunLoopCommonModes）

83. objc使用什么机制管理对象内存？

通过retainCount的机制来决定对象是否需要释放。 每次runloop的时候，都会检查对象的retainCount，如果retainCount为0,说明该对象没有地方需要继续使用了，可以释放掉了。

84. 不手动指定autoreleasepool的前提下，一个autorealese对象在什么时刻释放？（比如在一个vc的viewDidLoad中创建）

1.手动干预释放时机--指定autoreleasepool 就是所谓的：当前作用域大括号结束时释放。

2.系统自动去释放--不手动指定autoreleasepool

Autorelease对象出了作用域之后，会被添加到最近一次创建的自动释放池中，并会在当前的 runloop 迭代结束时释放。

85. BAD\_ACCESS在什么情况下出现？

访问了野指针，比如对一个已经释放的对象执行了release、访问已经释放对象的成员变量或者发消息。 死循环

86. 苹果是如何实现autoreleasepool的

1. objc\_autoreleasepoolPush

2.objc\_autoreleasepoolPop

3.objc\_autorelease

87. 使用block时什么情况会发生引用循环，如何解决？

一个对象中强引用了block，在block中又使用了该对象，就会发射循环引用。 解决方法是将该对象使用\_\_weak或者\_\_block修饰符修饰之后再在block中使用。

1.id \_\_weak weakSelf = self; 或者

Weak \_\_typeof(&\*self) weakSelf = self该方法可以设置宏

2.id \_\_block weakSelf = self;

88. 在block内如何修改block外部变量？

默认情况下，在block中访问的外部变量是复制过去的，即：写操作不对原变量生效。但是你可以加上\_\_block来让其写操作生效，示例代码如下:

\_\_block int a = 0;

void (^foo)(void) = ^{

a = 1;

}

f00();

//这里，a的值被修改为1

89. GCD的队列（dispatch\_queue\_t）分哪两种类型？

1. 串行队列Serial Dispatch Queue

2. 并发队列Concurrent Dispatch Queue

90. 如何用GCD同步若干个异步调用？（如根据若干个url异步加载多张图片，然后在都下载完成后合成一张整图）

使用Dispatch Group追加block到Global Group Queue,这些block如果全部执行完毕，就会执行Main Dispatch Queue中的结束处理的block。

dispatch\_queue\_t queue = dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0);

dispatch\_group\_t group = dispatch\_group\_create();

dispatch\_group\_async(group, queue, ^{ /\*加载图片1 \*/ });

dispatch\_group\_async(group, queue, ^{ /\*加载图片2 \*/ });

dispatch\_group\_async(group, queue, ^{ /\*加载图片3 \*/ });

dispatch\_group\_notify(group, dispatch\_get\_main\_queue(), ^{

// 合并图片

});

91. dispatch\_barrier\_async的作用是什么？

在并行队列中，为了保持某些任务的顺序，需要等待一些任务完成后才能继续进行，使用 barrier 来等待之前任务完成，避免数据竞争等问题。 dispatch\_barrier\_async 函数会等待追加到Concurrent Dispatch Queue并行队列中的操作全部执行完之后，然后再执行 dispatch\_barrier\_async 函数追加的处理，等 dispatch\_barrier\_async 追加的处理执行结束之后，Concurrent Dispatch Queue才恢复之前的动作继续执行。

92. 苹果为什么要废弃dispatch\_get\_current\_queue

容易造成死锁

93. 以下代码运行结果如何？

- (void)viewDidLoad

{

[super viewDidLoad];

NSLog(@"1");

dispatch\_sync(dispatch\_get\_main\_queue(), ^{

NSLog(@"2");

});

NSLog(@"3");

}

只输出：1发生主线程锁死。

94. KVC的keyPath中的集合运算符如何使用？

1. 必须用在集合对象上或普通对象的集合属性上

2. 简单集合运算符有@avg， @count ， @max ， @min ，@sum，

3. 格式 @"@sum.age"或 @"集合属性.@max.age"

95. lldb（gdb）常用的调试命令

1．breakpoint 设置断点定位到某一个函数

2. n 断点指针下一步

3. po打印对象