1.请谈一谈你对面向对象的理解？
答：面向对象关注的是解决问题需要哪些对象，面向对象编程简称OOP。OC中的面向对象:OC中的类相当于图纸，用来描述一类事物。OC利用类来创建对象，对象是类的具体存在。因此，OC面向对象解决问题应该是先考虑需要设计哪些类，再利用类创建多少个对象。
2.面向对象的三大特性？
答：面向对象的三大特性：封装，继承，多态。(1)封装的好处：过滤掉不合理的值，屏蔽内部的赋值细节，让外界不必关注内部的细节，更加接近人类的思考方式，只需要关注对象，不需要关注步骤。(2)继承：当两个类拥有相同属性和方法的时候，就可以将相同的抽取到一个父类中，这两个类作为子类；当A类完全拥有B类中的部分属性和方法时，可以考虑让B类继承A类。继承的注意:OC是单继承，父类必须声明在子类的前面，子类不能拥有和父类相同的成员变量，调用某个方法时，优先在当前类中找，如果找不到，去父类中找（就近），子类可以重写父类方法，子类重新实现父类中的某个方法，覆盖父类以前的做法。继承的好处：可以在不改变原来模型的基础上，扩充方法，建立了类与类之间的联系，子类可以拥有父类中的所有成员变量和方法，抽取了公共重复代码。继承的坏处：耦合性太强。(3)多态：没有继承就没有多态，OC对象具有多态性，代码的体现：父类类型的指针指向子类对象，父类指针访问对应的属性和方法。多态的好处：用父类接收参数，节省代码，如果函数\方法参数中使用的父类类型，可以传入父类、子类对象。多态的局限性：父类类型的指针变量不能直接调用子类特有的方法，必须强转为子类类型后，才能直接调用子类特有的方法。多态的细节：在运行时动态绑定检测真实对象。
3.分类Category？
答：在不改变原来类模型的前提下，给类扩充一些方法有2种方式：继承、分类。分类的好处：①一个庞大的类可以分模块开发，一个庞大的类可以由多个人来编写，更有利于团队合作，可以给系统自带的类添加分类，扩充方法，可以不需修改就可使用分类的扩展功能；②实现了功能的局部化封装。分类的注意：分类方法实现可以访问原始类的成员变量，但不能添加变量，只能添加方法，如果想添加变量，可以考虑通过继承创建子类。分类可以重新实现原始类的方法，但是它是直接替换掉原来的方法，导致再也不能访问原来的方法，分类的优先级最高。多个分类中如果实现了相同的方法，只有最后一个参与编译的才会有效，最后编译的会覆盖以前编译的。方法调用的优先级：分类—原始类—父类。
4.协议Protocol？

答：基本用途：可以用来声明一大堆方法（不能声明成员变量），只要某个类遵守了这个协议，就相当于拥有这个协议中的所有方法声明，只要父类遵守了某个协议，就相当于子类也遵守了。协议中的两个关键字：@required（默认）：这个方法必须要实现，@optional：这个方法不一定要实现。一个协议可以遵守其他多个协议，多个协议之间用逗号隔开，一个协议遵守了其他协议，就相当于拥有了其他协议中的方法声明。同 NSObject 基类一样，NSObject 也是一个基协议，是最根本最基本的协议，建议每个新的协议都要遵守 NSObject 协议。

5.构造方法？

答：(1)完整的创建一个可用对象，先分配存储空间+alloc 方法，再初始化-init 方法，[[类名 alloc] init]创建对象。(2)构造方法：用来初始化对象的方法，是个对象方法，重写构造方法的目的，为了让对象创建出来，成员变量就会有一些固定的值。(3)重写构造方法-(id)init 的注意点：先调用回父类的构造方法 self=[super init]，再进行子类内部成员变量的初始化，返回一个已经初始化完毕的对象。

6.点语法和 id 类型（万能指针）？

答：(1)点语法的本质还是方法调用，当使用点语法时，编译器会自动展开相应的方法。(2)id 类型：任何 OC 对象，id 类型不能使用点语法。id 万能指针，能指向\操纵任何 OC 对象，相当于 NSobject *。使用：id 后面不能带*，局限性：调用一个不存在的方法，会马上报错。

7.类的本质，+load 和+initialize 方法？

答：(1)类的本质：类也是一个对象，是 Class 类型的对象，简称类对象，类名就代表着类对象，每个类只有一个类对象。(2)+load 方法：在程序启动的时候会加载所有的类和分类，并调用所有类和分类的+load 方法，先加载父类，调用父类的+load 方法，再加载子类，调用子类的+load 方法，再加载分类，调用分类的+load 方法，不管程序运行过程中有没有用到这个类，都会调用+load 方法。(3)+initialize 方法：在第一次使用某个类时，就会调用一次+initialize 方法，一个类只会调用一次+initialize 方法，先调用父类的，再调用子类的。获取类对象，某个类的+class 方法，某个对象的-class 方法。

8.手动内存管理（非 ARC）？

答：(1)移动设备的内存极其有限，每个 app 所能占用的内存是有限制的，在 OC 中，任何继承了 NSObject 的对象，我们都应该对其进行内存管理。(2)每个 OC 对象都有自己的引用计数器，表示对象被引用的次数，占用 4 个字节的存储空间。当使用 alloc、new 或者 copy 创建一个对象时，新对象的引用计数器默认就是 1，当一个对象的引用计数器值为 0 时，对象所占用的内存就会被系统回收。给对象发送

一条 retain 消息，对象的引用计数器值+1 (retain 方法返回对象本身)，给对象发送一条 release 消息，对象的引用计数器值-1 (release 方法没有返回值)，给对象发送 retainCout 消息获得当前引用计数器的值。(3)对象的销毁：当一个对象的引用计数器值为 0 时，那么它将被销毁，其占用的内存被系统回收；当一个对象被销毁时，系统自动会向对象发送一条 dealloc 消息，一般会重写 dealloc 方法，验证对象是否被回收，一旦重写 dealloc 方法，就必须调用[super dealloc]，并且放在最后面调用；一旦对象被回收了，它占用的内存就不在可用，坚持使用会导致程序崩溃 (野指针错误)。(4)内存管理原则：只要还在用某个对象，那么这个对象就不会被回收，只要你想用这个对象，就让对象的计数器+1，当你不再使用这个对象时，就让对象的计数器-1。谁创建，谁 release，如果你通过 alloc、new 或 copy 来创建一个对象，那么你必须调用(auto)release，谁 retain，谁 release，只要你调用了 retain，都要调用 release，有始有终，有加就有减，曾经让对象的计数器+1，就必须在最后让对象计数器-1。(5)set 方法的内存管理代码规范：基本数据类型，直接赋值；OC 对象类型，先判断是不是新传进来的对象，对旧对象做一次 release，对新对象做一次 retain。(6)dealloc 方法的代码规范：一定要调用[super dealloc]，并且放在最后面，对 self (当前) 所拥有的其他对象做一次 release。(6)autorelease：给对象发送一条 autorelease 消息，就会将这个对象加到一个自动释放池中，当自动释放池销毁时，会给池子里面的所有对象发送一条 release 消息，调用 autorelease 方法时并不会改变对象的计数器，并且会返回对象本身，autorelease 实际上只是把对 release 的调用延迟了。好处：不用再关心对象释放的时间，不用再关心什么时候用 release。一般来说，除了 alloc、new 或 copy 之外的方法创建的对象都被声明了 autorelease。

9.自动引用记数 (ARC) ？

答：(1)ARC 是自 iOS5 之后增加的新特性，完全消除了手动管理内存的繁琐，编译器会自动在适当的地方插入适当的 retain、autorelease 语句，ARC 是编译器特性，因此 ARC 和手动管理内存性能是一样的，有时还能更加快速，因为编译器还可以执行某些优化。(2)ARC 的判断规则：只要没有强指针指向对象，就会释放对象。(3)强指针：默认情况下，所有的指针都是强指针，__strong；弱指针：弱指针指向的对象被释放时，弱指针会自动变成 nil 指针，__weak。(4)使用注意：不能调用 retain、(auto)release、retainCount；可以重写 dealloc，但是不能调用[super dealloc]。

10.@property 参数？

答：(1)@property 可以同时生成 setter 和 getter 的方法声明和实现，

默认情况下，setter 和 getter 方法中的实现会去访问下划线_开头的成员变量，如果不存在，会自动创建@private 的成员变量_xxx。(2)控制 set 方法的内存管理：strong 成员变量是强指针，适用于 OC 对象类型，weak 成员变量是弱指针，适用于 OC 对象类型，assign 适用于非 OC 对象类型，基本数据类型。循环引用：一端使用 strong，另一端使用 weak。(3)控制需不需生成 set 方法 :readwrite 可读可写，同时生成 setter 和 getter，readonly 只读，只会生成 get 方法。(4)多线程管理：nonatomic，性能高。

11.@class、#import、@protocol？

答：(1)使用@class 类名;就可以引用一个类，说明一下它是一个类；使用@protocol 协议名称;就可以引用一个协议，说明一下它是一个协议。(2)@class 和#import 的区别：#import 方式会包含被引用类的所有信息，包括被引用类的变量和方法，@class 方式只是告诉编译器只是类的声明；如果上百个头文件都#import 了同一个文件，或者这些文件依次被#import，那么一旦最开始的头文件稍有改动，后面引用到这个文件的所有类都需要重新编译，用@class 方式就不会出现这种问题（关乎性能）。(3)开发中，引用一个类和协议的规范：在.h 文件中用@class 声明一个类，用@protocol 声明一个协议，在.m 文件中用#import 来包含类的所有东西。

12.NSLog 和 printf 的区别？

答：NSLog 接收 OC 字符串作为参数，printf 接收 C 语言字符串作为参数，NSLog 输出后会自动换行，printf 输出后不会自动换行，使用 NSLog 需要#import<Foundation/Foundation.h>，使用 printf 需要#include<stdio.h>。

13.#import 和#include 的区别？

答：#import 和#include 都用来拷贝某个文件的内容，#import 可以自动防止文件内容被拷贝多次，不像#include 需加入预处理指令。

14.方法和函数？

答：(1)方法：方法都是以-\+开头，方法的声明必须写在@interface 和@end 之间，方法的实现必须写在@implementation 和@end 之间，方法只能由对象或者类来调用，方法归类\对象所有。(2)函数能写在文件的任意位置，函数归文件所有，函数调用不依赖于对象，函数内部不能直接通过成员变量名访问某个对象的成员变量。

15.C 语言数组和 OC 数组？

答：(1)数组就是相同类型的数据的集合，数组只能存放一种类型的数据，数据称为元素，元素有顺序之分，每个元素都有一个唯一的下标，从 0 开始。(2)OC 数组用来存储对象的有序列表（任意类型的对象），只能存储 Objective-C 的对象，不能存储 C 语言的基本数据类

型，同时也不能存储 nil。