1. 回答person的retainCount值，并解释为什么

```
Person * per = [[Person alloc] init];
```
此时person 的retainCount的值是1

```
self.person = per;
```
在self.person 时，如果是assign，person的 retainCount的值不变，仍为1

若是：retain person的retainCount的值加1，变为2

若是：copy person的retainCount值不变，仍为1

2、 这段代码有什么问题吗:

```
@implementation Person
- (void)setAge:(int)newAge {
  self.age = newAge;
}
@end
```
会死循环，会重复调用自己！self.age 改为_age即可;

并且书写不规范：setter方法中的newAge应该为age

3、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 = [string stringByAppendingString:@"xyz"];
NSLog(@“%@”, string);
}
```
会出现内存泄露

修改之后：

```
for(int i = 0; i<1000;i++){
        NSAutoreleasePool * pool1 = [[NSAutoreleasePool alloc] init];
        NSString *string = @"Abc";
        string = [string lowercaseString];
        string = [string stringByAppendingString:@"xyz"];
        NSLog(@"%@",string);
        //释放池
        [pool1 drain];
    }
```
延伸：堆栈的区别：

(1)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。

(2)申请大小：能从栈获得的空间较小,堆是向高地址扩展的数据结构，是不连续的内存区域。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。

(3)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

(4)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。

(5)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。

4、写一个便利构造器

```
+(id)Person
{
    Person *person=[Person alloc]init];
     return [person autorelease];
}
```
备注：ARC时不用 autorelease

5、截取字符串”20 | http://www.baidu.com”中，”|”字符前面和后面的数据，分别输出它们。

```
NSString * str = @"20 | http://www.baidu.com";
NSArray *array = [str  componentsSeparatedByString:@"|"];
//这是分别输出的截取后的字符串
for (int i = 0; i<[array count]; ++i) {
    NSLog(@"%d=%@",i,[array objectAtIndex:i]);
    }
```

6、用obj-c写一个冒泡排序

```
-(void)mySort:(NSMutableArray *)mutArray
{
    id tmpObj = nil;
    unsigned long flag = mutArray.count-1;//flag :最大脚标
    while (flag > 0) {
        int k = flag;
        flag = 0;
        for (int j = 0 ; j < k ; j++) {
            int order = NSOrderedAscending;// 或 NSOrderedDescending
            if ([[mutArray[j] description] compare:[mutArray[j+1] description]] == -order) {
                tmpObj = mutArray[j];
                mutArray[j] = mutArray[j+1];
                mutArray[j+1] = tmpObj;
                flag = j;
            }
        }
    }
}
```

延伸：C语言的冒泡排序：

(1)冒泡法对一维数组中的元素进行排序

```
void sort(int arr[],int arr_len)
{
    for(int  i=0;i<arr_len-1;i++)//外层循环
    {
        for(int  j=0;j<arr_len-1-i;j++)//借助j实现一趟的次数
        {
            if(arr[j]>arr[j+1])
            {
                int temp=0;
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
}
```

(2)用指针数组给字符串排序

```
void sort_a(char * name[],int n)
{
    char *temp;
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-1-i;j++)
        {
            if(strcmp(name[j], name[j+1])>0)
```

```
            {
                temp=name[j];
                name[j]=name[j+1];
                name[j+1]=temp;
            }
        }
    }
}
```

(3)给字符串数组排序

```
void rang(char str[][20],int  n)
{
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-1-i;j++)
        {
            int result=strcmp(str[j],str[j+1]);
            if(result>=0)
            {
                char temp[20];
                strcpy(temp, str[j]);
                strcpy(str[j], str[j+1]);
                strcpy(str[j+1], temp);
            }
        }
    }
}
```

7、简述你对UIView、UIWindow和CALayer的理解

UIView：属于UIkit.framework框架，负责渲染矩形区域的内容，为矩形区域添加动画，响应区域的触摸事件，布局和管理一个或多个子视图

UIWindow：属于UIKit.framework框架，是一种特殊的UIView，通常在一个程序中只会有一个UIWindow，但可以手动创建多个UIWindow，同时加到程序里面。UIWindow在程序中主要起到三个作用：

1、作为容器，包含app所要显示的所有视图

2、传递触摸消息到程序中view和其他对象

3、与UIViewController协同工作，方便完成设备方向旋转的支持

CAlayer：属于QuartzCore.framework，是用来绘制内容的，对内容进行动画处理依赖与UIView来进行显示，不能处理用户事件。UIView和CALayer是相互依赖的，UIView依赖CALayer提供内容，CALayer依赖UIView一共容器显示绘制内容。

延伸：

UIViewController：管理视图的几成熟，每个视图控制器都有一个自带的视图，并且负责这个视图相关的一切事务。方便管理视图中的子视图，负责model与view的通信；检测设备旋转以及内存警告；是所有视图控制类的积累，定义了控制器的基本功能。

UIResponder的那张图

8、写一个完整的代理，包括声明，实现

```
//创建
@protocol BeforeMarriedDelagate <NSObject>
@required
-(void)doCook:(NSString *)foodName;
-(void)doHomework;
@optional
-(void)driveCar;
-(void)makeMoney;
@end
```

```
//声明
 @interface Boy : NSObject< BeforeMarriedDelagate>
-(void)doCook:(NSString *)foodName;
-(void)doHomework;
-(void)makeMoney;
//实现
 @implementation Boy
-(void)doCook:(NSString *)foodName
{    NSLog(@"做饭：%@！",foodName);}
-(void)doHomework
{    NSLog(@"今天洗衣服！");}
-(void)makeMoney
{    NSLog(@"Coding！！");}
@end
```

9、分析json、xml的区别？json、xml解析方式的底层是如何处理的？

区别：

(1)可读性方面：基本相同，xml的可读性比较好

(2)可扩展性方面：都具有很好的扩展性

(3)编码难度方面：相对而言：JSON的编码比较容易

(4)解码难度：json的解码难度基本为零，xml需要考虑子节点和父节点

(5)数据体积方面：json相对于xml来讲，数据体积小，传递的速度跟快些

(6)数据交互方面：json与JavaScript的交互更加方面，更容易解析处理，更好的数据交互

(7)数据描述方面：xml对数据描述性比较好

(8)传输速度方面：json的速度远远快于xml

JSON底层原理：遍历字符串中的字符，最终根据格式规定的特殊字符，比如{}号，[]号，：号 等进行区分，{}号是一个字典的开始，[]号是一个数组的开始，：号是字典的键和值的分水岭，最终乃是将json数据转化为字典，字典中值可能是字典，数组，或字符串而已。

XML底层原理： XML解析常用的解析方法有两种：DOM解析和SAX解析。DOM 采用建立树形结构的方式访问 XML 文档，而 SAX 采用的事件模型。 。DOM 解析把 XML 文档转化为一个包含其内容的树，并可以对树进行遍历。使用 DOM 解析器的时候需要处理整个 XML 文档，所以对性能和内存的要求比较高。SAX在解析 XML 文档的时候可以触发一系列的事件，当发现给定的tag的时候，它可以激活一个回调方法，告诉该方法制定的标签已经找到。SAX 对内存的要求通常会比较低，因为它让开发人员自己来决定所要处理的tag。特别是当开发人员只需要处理文档中所包含的部分数据时，SAX 这种扩展能力得到了更好的体现。

延伸：SAX与DOM的区别：

1、SAX处理的优点非常类似于流媒体的优点。分析能够立即开始，而不是等待所有的数据被处理。而且由于应用程序只是在读取数据时检查数据，因此不需要将数据存储在内存中。这对于大型文档来说是个巨大的优点。事实上，应用程序甚至不必解析整个文档；它可以在某个条件得到 满足时停止解析。一般来说，SAX 还比它的替代者 DOM 快许多。另一方面，由于应用程序没有以任何方式存储数据，使用 SAX 来更改数据或在数据流中往后移是不可能的。

2、DOM 以及广义的基于树的处理具有几个优点。首先，由于树在内存中是持久的，因此可以修改它以便应用程序能对数据和结构作出更改。它还可以在任何时候在树中上下 导航，而不是像 SAX 那样是一次性的处理。DOM 使用起来也要简单得多。另一方面，在内存中构造这样的树涉及大量的开销。大型文件完全占用系统内存容量的情况并不鲜见。此外，创建一棵 DOM 树可能是一个缓慢的过程。

3、选择 DOM 还是选择 SAX，这取决于下面几个因素:

应用程序的目的：如果打算对数据作出更改并将它输出为 XML，那么在大多数情况下，DOM 是适当的选择。并不是说使用 SAX 就不能更改数据，但是该过程要复杂得多，因为您必须对数据的一份拷贝而不是对数据本身作出更改。

数据容量： 对于大型文件，SAX 是更好的选择。数据将如何使用：如果只有数据中的少量部分会被使用，那么使用 SAX 来将该部分数据提取到应用程序中可能更好。 另一方面，如果您知道自己以后会回头引用已处理过的大量信息，那么 SAX 也许不是恰当的选择。

对速度的需要：SAX 实现通常要比 DOM 实现更快。

SAX 和 DOM 不是相互排斥的，记住这点很重要。您可以使用 DOM 来创建 SAX 事件流，也可以使用 SAX 来创建 DOM 树。事实上，用于创建 DOM 树的大多数解析器实际上都使用 SAX 来完成这个任务！

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

当程序接到内存警告时ViewController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，如果view可以被释放，那么这个函数释放view并

调用viewDidUnload。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。

11、面向对象的三大特征，并作简单的介绍。

面向对象的三个基本特征是：封装、继承、多态。
封装是面向对象的特征之一，是对象和类概念的主要特性。 封装，也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。隐藏对象的属性和实现细节，仅对外公开接口，提高代码安全性，封转程度越高，独立性越强，使用越方便。
继承是指这样一种能力：它可以使用现有类的所有功能，并在无需重新编写原来的类的情况下对这些功能进行扩展。 通过继承创建的新类称为“子类”或“派生类”。 被继承的类称为“基类”、“父类”或“超类”
多态性：允许你将父对象设置成为和一个或更多的他的子对象相等的技术，赋值之后，父对象就可以根据当前赋值给它的子对象的特性以不同的方式运作。简单的说，就是一句话：允许将子类类型的指针赋值给父类类型的指针

12、我们说的obc是动态运行时语言是什么意思？

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

13、readwrite，readonly，assign，retain，copy，nonatomic 、strong、weak属性的作用？并区别strong(强引用)、 weak(弱引用)？什么情况使用copy，assign，和retain?

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

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

会引起循环引用----若是retain，在alloc一次之后，若release一次，会导致内训泄漏，若release两次会导致两个对象的dealloc嵌套执行，结果就是都没有执行成功，最后崩溃！
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、ObjC中，与retain配对使用的方法是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？

与retain配对使用的方法是release，因为retain使retainCount计数加1，release使retainCount计数减1；与retain语义相反的是release。
与alloc配对使用的是release，因为：alloc是使retainCount计数加1，，使retainCount计数减1。与alloc语义相反

的是dealloc，因为：alloc是创建一个对象，，dealloc是销毁一个对象。

16、重写一个NSStrng类型的，retain方式声明name属性的setter和getter方法

{ NSString * _name; } @property NSString *name;----加上这些是为了避免错误

setter方法：

```
-(void)setName:(NSString*)name
{
{[_name release];
_name=[name retain];
}
}
```

getter方法

```
-(NSString*)name
{return _name if(_name!=name)
}
```

17、分别描述内存管理要点、autorelease、release、NSAutoreleasePool? 并说明autorelease是什么时候被release的? 简述什么时候由你负责释放对象，什么时候不由你释放？［NSAutoreleasePool release］和［NSAutoreleasePool drain］有什么区别?

内存管理要点：

Objective-C 使用引用计数机制(retainCount)来管理内存。内存每被引用一次，该内存的引用计数+1，每被释放一次引用计数-1。当引用计数 = 0 的时候，调用该对象的 dealloc 方法，来彻底从内存中删除该对象。

alloc，allocWithZone，new(带初始化)时：该对象引用计数 +1；

retain：手动为该对象引用计数 +1；

copy：对象引用计数 +1；

mutableCopy：生成一个新对象，新对象引用计数为 1；

release：手动为该对象引用计数 -1；

autorelease：把该对象放入自动释放池，当自动释放池释放时，其内的对象引用计数 -1。

NSAutoreleasePool：

NSAutoreleasePool是通过接收对象向它发送的autorelease消息，记录该对象的release消息，当自动释放池被销毁时，会自动向池中的对象发送release消息。

autorelease 是在自动释放池被销毁，向池中的对象发送release

只能释放自己拥有的对象，

区别是：在引用计数环境下（在不使用ARC情况下），两者基本一样，在GC环境下，release 是一个no-op（无效操作），所以无论是不是gc都使用drain

18、IPhone OS有没有垃圾回收? autorelease 和垃圾回收制（gc）有什么关系?

没有。autorelease只是延迟释放，gc是每隔一段时间询问程序，看是否有无指针指向的对象，若有，就将它回收。他们两者没有什么关系。

19、**drawRect和layoutSubviews的区别**

两个方法都是异步执行的，layoutSubviews方便数据计算，drawRect方便视图重绘。

layoutSubviews对subviews重新布局

layoutSubviews方法调用先于drawRect

20、简述NotificationCenter、KVC、KVO、Delegate? 并说明它们之间的区别?

Notification 是观察者模式的实现，KVO是观察者模式的OB-C底层实现。

NOtification 通过 Notifydcation addobserver 和 remove observer 工作。

KVO是键值监听，键值观察机制，提供了观察某一属性变化的方法

KVC是键值编码，是一种间接访问对象的属性，使用字符串来标示属性(例如：setValue：forKey：)

Delegate: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。

区别：

21、线程与进程的区别和联系?

线程是进程的基本单位

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。

22、简述多线程的作用以及什么地方会用到多线程? OC实现多线程的方法有哪些? 谈谈多线程安全问题的几种解决方案? 何为线程同步，如何实现的? 分线程回调主线程方法是什么，有什么作用?

(1)、多线程的作用：可以解决负载均衡问题，充分利用cpu资源 。为了提高CPU的使用率，采用多线程的方式去同时完成几件事情而互不干扰，

(2)、大多情况下，要用到多线程的主要是需要处理大量的IO操作时或处理的情况需要花大量的时间等等，比如：读写文件、视频图像的采集、处理、显示、保存等。

(3)、ios有三种主要方法：1、NSThread。2、NSOperation。3、GCD。

(4)解决方案：使用锁：锁是线程编程同步工具的基础。锁可以让你很容易保护代码中一大块区域以便你可以确保代码的正确性。使用POSIX互斥锁；使用NSLock类；使用@synchronized指令等。

(5)回到主线程的方法： dispatch_async(dispatch_get_main_queue(), ^{

});

作用：主线程是显示UI界面，子线程多数是进行数据处理

23、http和scoket通信的区别？socket连接相关库，TCP,UDP的连接方法，HTTP的几种常用方式？

http和scoket通信的区别：

http是客户端用http协议进行请求，发送请求时候需要封装http请求头，并绑定请求的数据，服务器一般有web服务器配合（当然也非绝对）。 http请求方式为客户端主动发起请求，服务器才能给响应，一次请求完毕后则断开连接，以节省资源。服务器不能主动给客户端响应（除非采取http长连接技术）。iphone主要使用类是NSUrlConnection。

scoket是客户端跟服务器直接使用socket“套接字”进行连接，并没有规定连接后断开，所以客户端和服务器可以保持连接通道，双方都可以主动发送数据。一般在游戏开发或股票开发这种要求即时性很强并且保持发送数据量比较大的场合使用。主要使用类是CFSocketRef。

UDP：是用户数据报协议:主要用在实时性要求高以及对质量相对较弱的地方，但面对现在高质量的线路不是容易丢包除非是一些拥塞条件下 ，如流媒体

TCP：是传输控制协议：是面连接的，那么运行环境必然要求其可靠性不可丢包有良好的拥塞控制机制如http ftp telnet等

http的常用方式：get，post

24、What is lazy loading?

就是懒汉模式，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。

25、你连接服务器用的是什么方法，如果请求过程中，网络出了问题这么办？

NSUrlConnection 连接后，有一系列委托方法来接受来自服务器的响应和数据，其中接受相应的方法回得到服务器要传回的数据有多大，接受数据的方法会反复调用来不断接受服务器数据，如果网络出了问题了，会调用一个方法让你来做相关处理。

26、OC有多继承吗？没有的话可以用什么方法替---多继承即一个子类可以有多个父类，它继承了多个父类的特性。

Object-c的类没有多继承,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,OC类似于多继承，是在用protocol委托代理来实现的；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。

27、什么是Protocol？什么是代理？写一个委托的interface？委托的property声明用什么属性？为什么？

Protocol：一个方法签名的列表，在其中可以定义若干个方法。根据配置，遵守协议的类，会实现这个协议中的若干个方法。

代理：实现这个协议中的方法的类

委托的interface:声明一个某协议的属性delagate

用assgin属性，原因是，为了避免循环引用。

28、分别描述类别（categories）和延展（extensions）是什么？以及两者的区别？继承和类别在实现中有何区别？为什么Category只能为对象添加方法，却不能添加成员变量？

类别：在没有原类.m文件的基础上，给该类添加方法;

延展：一种特殊形式的类别，主要在一个类的.m文件里声明和实现延展的作用，就是给某类添加私有方法或是私有变量。

两个的区别:延展可以添加属性并且它添加的方法是必须要实现的。延展可以认为是一个私有的类目。

继承和类别在实现中的区别：类别可以在不获悉，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。

Category只能为对象添加方法，却不能添加成员变量的原因：如果可以添加成员变量，添加的成员变量没有办法初始化----这是语言规则

29、写一个NSString类的实现+ (id)initWithCString:(constchar *)nullTerminatedCString encoding:(NSStringEncoding)encoding;

{ NSString *obj;

```
    obj = [self allocWithZone: NSDefaultMallocZone()];
    obj = [obj initWithCString: nullTerminatedCString encoding: encoding];
  return [obj autorelease];
}
```

30、Objective-C有私有方法么？私有变量呢？如多没有的话，有没有什么代替的方法？

objective-c类里面的方法只有两种，静态方法和实例方法.但是可以通过把方法的声明和定义都放在.m文件中来实现一个表面上的私有方法。有私有变量，可以通过@private来修饰，或者把声明放到.m文件中。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

31、#import、#include和@class有什么区别

@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，它只是声明了一个类名，关于这个类的内部实现都没有告诉编译器，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。

32、谈谈你对MVC的理解？为什么要用MVC？在Cocoa中MVC是怎么实现的？你还熟悉其他的OC设计模式或别的设计模式吗？

MVC就是Model-View-Controller的缩写，M指的是业务模型，V指的是用户页面，C指的是控制器。MVC是架构模式，是讲M和V的代码分离，从而使同那个一个程序可以使用不同的表现形式。

M：表示数据和业务规则，V是用户看到的并与之交互的页面，C是接受用户的输入并调用M和V取完成用户需求的

单例，代理，观察者，工厂模式等

单例模式：说白了就是一个类不通过alloc方式创建对象，而是用一个静态方法返回这个类的对象。系统只需要拥有一个的全局对象，这样有利于我们协调系统整体的行为；

代理模式：代理模式给某一个对象提供一个代理对象，并由代理对象控制对源对象的引用.比如一个工厂生产了产品，并不想直接卖给用户，而是搞了很多代理商，用户可以直接找代理商买东西，代理商从工厂进货.常见的如QQ的自动回复就属于代理拦截，代理模式在iphone中得到广泛应用.

观察者模式： 当一个物体发生变化时，会通知所有观察这个物体的观察者让其做出反应。实现起来无非就是把所有观察者的对象给这个物体，当这个物体的发生改变，就会调用遍历所有观察者的对象调用观察者的方法从而达到通知观察者的目的；

33、如监测系统键盘的弹出

```
- (id) init
{
    self = [super init];
    if (self)
    {
  NSNotificationCenter *center = [NSNotificationCenter defaultCenter];
        [center    addObserver:self selector:@selector(keyboardDidShow) name:UIKeyboardDidShowNotification  object:nil];
    [center    addObserver:self    selector:@selector(keyboardDidHide) name:UIKeyboardWillHideNotification object:nil];
        _keyboardIsVisible = NO;
    }
    return self;
}
- (void)keyboardDidShow
{
    _keyboardIsVisible = YES;
}
- (void)keyboardDidHide
{
    _keyboardIsVisible = NO;
}
- (BOOL)keyboardIsVisible
{
    return _keyboardIsVisible;
}
```

34、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

词典NSDictionary，可变词典NSMutableDictionary，

```
//初始化一个可变词典，带有2个键值对
NSMutableDictionary *dic = [NSMutableDictionary
dictionaryWithObjectsAndKeys:@"value1",@"key1",@"value2",@"key2",nil];
//添加
[dic setObject:@"value3" forKey:@"key3"];
//删除
[dic removeObjectForKey:@"key3"];
//获取（按key获取）
[dic objectForKey:@"key1"];
```

35、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。

36、举出5个以上你所熟悉的ios sdk库有哪些和第三方库有哪些?

ios-sdk:

Foundation.framework,CoreGraphics.framework,UIKit.framework， MediaPlayer.framework, CoreAudio.framework

第三方库： 1.json编码解码；2.ASIHTTPRequest等相关协议封装；3.EGORefreshTableHeaderView下拉刷新代码；4.AsyncImageView 异步加载图片并缓存；5.SDWebImage——简化网络图片处理

37、objc优缺点----**OC的特点：id类型，动态**

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。

38、UITableView的重用机制? 如何在一个view上显示多个tableView,tableView要求不同的数据源以及不同的样式（要求自定义cell)，如何组织各个tableView的delegate和dataSource?请说说实现思路? 在一个tableView中需要自定义多种样式的cell(两种或三种),通常你如何实现,说说思路即可? UITableView的那些元素是可以自定义的?

UITableView的重用机制：

查看UITableView头文件，会找到NSMutableArray*visiableCells，和NSMutableArray*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

tableView dequeueReusableCellWithIdentifier: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。

39、一个tableView是否可以关联两个不同的数据源？你会怎么处理？

首先从代码来看，数据源如何关联上的，其次是在数据源关联的代理方法里实现的。

```
- (UITableViewCell *)cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
  if(indexPath.section == 0)
  {}
  if(indexPath.section == 1)
  {}
}
```

40、OC中有哪些数据存储方式，各有什么区别？

四种存储方式： 1.NSUserDefaults，用于存储配置信息2.SQLite，用于存储查询需求较多的数据3.CoreData，用于规划应用中的对象4.使用基本对象类型定制的个性化缓存方案.

NSUserDefaults：对象中储存了系统中用户的配置信息，开发者可以通过这个实例对象对这些已有的信息进行修改，也可以按照自己的需求创建新的配置项。

SQLite擅长处理的数据类型其实与NSUserDefaults差不多，也是基础类型的小数据，只是从组织形式上不同。开发者可以以关系型数据库的方式组织数据，使用SQL DML来管理数据。一般来说应用中的格式化的文本类数据可以存放在数据库中，尤其是类似聊天记录、Timeline等这些具有条件查询和排序需求的数据。

CoreData是一个管理方案，它的持久化可以通过SQLite、XML或二进制文件储存。它可以把整个应用中的对象建模并进行自动化的管理。从归档文件还原模型时CoreData并不是一次性把整个模型中的所有数据都载入内存，而是根据运行时状态，把被调用到的对象实例载入内存。框架会自动控制这个过程，从而达到控制内存消耗，避免浪费。

无论从设计原理还是使用方法上看，CoreData都比较复杂。因此，如果仅仅是考虑缓存数据这个需求，CoreData绝对不是一个优选方案。CoreData的使用场景在于：整个应用使用CoreData规划，把应用内的数据通过CoreData建模，完全基于CoreData架构应用。

使用基本对象类型定制的个性化缓存方案：从需求出发分析缓存数据有哪些要求：按Key查找，快速读取，写入不影响正常操作，不浪费内存，支持归档。这些都是基本需求，那么再进一步或许还需要固定缓存项数量，支持队列缓存，缓存过期等。

41、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

coredata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。coredata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coredata的时候，你不用安装额外的数据库系统，因为coredata使用内置的sqlite数据库。coredata将你app的模型层放入到一组定义在内存中的数据对象。coredata会追踪这些对象的改变，同时可以根据需要做相应的改变，例如用户执行撤销命令。当coredata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coredata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。

42、OC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

Objective-C中的数字对象NSNumber；

Objective-C中的基本类型和C语言中的基本类型一样.主要有:int,long,float,double,char,void,bool等.

对于基本类型变量，不需要用指针，也不用手动回收，方法执行结束会自动回收.数字对象需要指针，也需要手动回收内存。

43、什么是动态识别，动态绑定？**延展--程序的编译过程**

44、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。

45、如何将产品进行多语言发布？

程序国际化；

比如：本地化应用程序名称

(1、选中工程，Info—Localizations点击“+”添加要国际化的语言。

(2、在InfoPlist.strings右边会多出一个三角形，点击展开可看到InfoPlish.strings(english)和InfoPlish.strings(chinese)两个版本的文件；

(3、在InfoPlish.strings(english)文件中加入：

CFBundleDisplayName ="Program";
其中“Program”为英文应用程序名称，同理在InfoPlish.strings(chinese)文件中加入：
CFBundleDisplayName ="应用程序";
其中“应用程序”为中文名称，注意：CFBundleDisplayName加不加双引号都行；
(4、编辑Info.plist，添加一个新的属性Application has localized display name，设置其类型为boolean，并将其value设置为YES即可。

46、什么是动态链接库和静态链接库？调用一个类的静态方法需不需要release？ **程序的编译过程--链接---**

静态连接库就是把(lib)文件中用到的函数代码直接链接进目标程序，程序运行的时候不再需要其它的库文件；动态链接就是把调用的函数所在文件模块 （DLL）和调用函数在文件中的位置等信息链接进目标程序，程序运行的时候再从DLL中寻找相应函数代码，因此需要相应DLL文件的支持。

静态链接库和动态链接库的另外一个区别在于静态链接库中不能再包含其他的动态链接库或者静态库，而在动态链接库中还可以再包含其他的动态或静态链接库。

动态的是：运行时才加载到内存中，静态：编译时就加载到内存中

静态方法也就是类方法，不需要release

47、声明一个静态方法和一个实例方法？

解释：就是类方法和对象方法，

48、什么是push？远程推送？

第一步：UIApplication向APNS注册push notification服务

1、应用程序 要支持 推送服务（在网页里配置）

（1）https://developer.apple.com/devcenter/ios/index.action

（2）登录 苹果开发者账号（注意是收费账号，$99或$299）

3）下载push证书（主要是给程序签名，push服务只有收费开发者才具备。所以需要签名验证），如果没有push证书，创建一个push证书（App ID->钥匙串程序生成request->push证书）**注意事项**：App ID的Bundle ID必须和程序plist文件里的Bundle identifier一致。App ID一旦生成，将不可修改。

（4）把证书安装到钥匙串里（双击证书文件）

（5）生成 编译程序 用的描述文件（网页里进行）

2、向APNS注册push服务（UIApplication的registerForRemoteNotificationTypes:方法）

第二步 获取APNS分配的DeviceToken（64位16进制串）

- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken

第三步 把DeviceToken发送给自己的后台服务器，服务器记录每台设备的DeviceToken以便日后推送信息给客户端。（需要有一个网络接口，让客户端发送DeviceToken）

第四步 服务器推送信息给客户端

1、服务器除了需要有客户端的DeviceToken之外，还要有push证书，对push的内容进行签名。（苹果为了防止 恶意向客户端（比如DeviceToken泄露了）发送消息，每次推送消息，都需要证书进行签名，从而避免黑客恶意攻击用户手机。）

2、如果你的服务器是java写的，可以直接使用钥匙串导出的p12文件（证书和密钥一起导出）。如果你的服务器是php写的，因为php语言不支持p12文件类型，需要转换为pem文件。

3、将p12转换为pem文件：终端 先找到你p12所在的目录 openssl pkcs12 -in CertificateName.p12 -outCertificateName.pem -nodes

4、服务器发送信息给APNS，APNS自动将信息推送给客户端

第五步 客户端处理收到的信息

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo

注意事项：

1、测试版的push证书仅仅用于开发期间测试，发布版的程序需要生成一个发布版的push证书。

2、测试版APNS的ssl地址和发布版的ssl地址不同

3、测试版的DeviceToken和发布版的DeviceToken也不同

4、php文件要喝pem文件放在同一目录。

5、除了alert sound和badge之外，json串里还可以包含自定义信息。

6、推送的信息最大255字节

7、推送的信息受网络影响较大，有可能造成延迟甚至丢失，重要信息的传递不应该使用push通知，应该有专门的后台接口。

8、借助push推送，两个客户端可以实现即时通信，工程里面存放我们的p12文件，客户端自己组织json串，发送请求到APNS。

49、什么是沙箱模型？哪些操作是属于私有api范畴？

某个iphone工程进行文件操作有此工程对应的指定的位置，不能逾越。

iphone常见私有api的应用（比如直接发送短信，访问沙箱之外的磁盘文件）.

50、如何将敏感字变成** **延展---几个字变为几个*如何实现？**

调用stringByReplacingOccurrencesOfString:withString:给定字符串，指定替换字，替换＊＊

51、iphone阅读器，如果要读取一个文本文件，请问你是如何处理编码问题的？另外像pdf格式的文件，你如何读取？

首先检测文本编码格式（只需读取小部分用来判断），

iphone手机阅读器中对于PDF格式的阅读，可以直接用UIWebView控件显示，也可以从网上下到很多直接读取pdf格式的代码

直接从pdf中得到数据。复杂表格动画- (void)insertRowsAtIndexPaths:(NSArray *)indexPaths withRowAnimation:(UITableViewRowAnimation)animation;

-(void)deleteRowsAtIndexPaths:(NSArray *)indexPaths withRowAnimation:(UITableViewRowAnimation)animation;- (void)reloadRowsAtIndexPaths:(NSArray *)indexPaths withRowAnimation:(UITableViewRowAnimation)animation;

52、在开发大型项目的时候，如何进行内存泄露检测的？内存泄露怎么处理？

如何检测内存泄露：

可以通过xcode的自带工具run---start with performance tool里有instruments下有个leaks工具，启动此工具后，运行项目，工具里可以显示内存泄露的情况，双击可找到源码位置，可以帮助进行内存泄露的处理。

如何处理：先定位到具体位置， 再解决之。

53、iphone app为什么会被打回来，如何制止？

app的设置界面、按钮使用了类似iphone的操作方式以及icon的圆角设计 -> 重新设计…

app的年龄设置太低 -> 改了年龄…

app里有实物奖励 -> 免责声明，和苹果无关…

app描述里提了后续版本的功能的字样 -> 删除…

app有打分的功能 -> 有reject的，也有通过的…

app需要使用location，没有提示用户 -> 加了提示，允许用户拒绝…

app没提供测试账号 -> 提供…

app里有私有api -> 修改…

应用内含有有某公司LOGO的图片，没有该公司授权文件，被拒-> 修改…

第三方静态库包含私有api的调用(联系第三方技术支持，更新静态库);

包含潜在的色情，暴力等内容(调整应用年龄限制等级，并加入举报功能)

做浏览器的，分级必须选17+

54、iphone应用程序的项目基本结构？

- Classes -> 源程序文件（.h、.m）
- Other Sources-> main.m 等，不需要程序员修改 -Prefix.pch
- Resources -> 界面文件（.xib）、配置文件-info.plist
- Frameworks -> 链接的库 • Targets -> 项目的不同Target(资源、编译配置不同)
- Executables -> 项目中所有的可执行文件

-Prefix.pch:_Prefix为所有的项目程序文件预先配置运行环境的前缀标头，在程序运行之前，引入所需框架中的(.h)头文件。这样可以减少每个头文件对程序编译做出相同的定义，在巨型的应用程序项目开发中节省大量的时间，例如，程序有100个根文件需要定义abc.h，只需要在_Preix.pch文件下建立一个对象，所有的根文件便可以重复地对程序编译做出定义。

55、请写出代码,用blocks来取代上例中的protocol,并比较两种方法的优势。实际应用部分？请写出代码，用blocks取代协议或回调方法

56、你做iphone开发时候，有哪些传值方式，view和view之间是如何传值的？

block， target-action ，代理，属性，

57、给定的一个字符串，判断字符串中是否还有png，有就删除它？

```
NSMutableString *mstr=[NSMutableString stringWithFormat:@"ccc"];
    NSRange substr = [mstr rangeOfString:@"png"];            //字符串查找,可以判断字符串中是否有
```

```
if (substr.location != NSNotFound) {
    [mstr deleteCharactersInRange:substr];
}
```

58、编译语言和解释语言的区别

区别：C语言，OC语言属于编译语言；解释语言：也可以理解为脚本文件，不需要编译，

编译型语言写的程序执行之前，需要一个专门的编译过程，把程序编译成为机器语言的文件，比如exe文件，以后要运行的话就不用重新翻译了，直接使用编译的结果就行了（exe文件），因为翻译只做了一次，运行时不需要翻译，所以编译型语言的程序执行效率高，但也不能一概而论，部分解释型语言的解释器通过在运行时动态优化代码，甚至能够使解释型语言的性能超过编译型语言。　解释则不同，解释性语言的程序不需要编译，省了道工序，解释性语言在运行程序的时候才翻译，比如解释性basic语言，专门有一个解释器能够直接执行basic程序，每个语句都是执行的时候才翻译。这样解释性语言每执行一次就要翻译一次，效率比较低。解释是一句一句的翻译。

59、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?

编译时是NSString，运行时是NSDate

60、给用户推送的通知的伪代码

61、ViewController的 loadView, viewDidLoad,viewWillAppear,viewDidUnload,dealloc、init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?

1、viewDidLoad 此方法只有当view从nib文件初始化的时候才被调用

2、viewDidUnload当系统内存吃紧的时候会调用该方法,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）

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

流程应该是这样：

(loadView/nib文件)来加载view到内存 -->viewDidLoad函数进一步初始化这些view -->内存不足时，调用viewDidUnload函数释放views -->当需要使用view时有回到第一步

如此循环

4、viewWillAppear方法,视图即将过渡到屏幕上时调用,(一般在返回需要刷新页面时,我都选择使用代理,所以很少用到)

5、viewWillDisappear方法,这个A->B之后,A在B之后的操作

62、描述程序启动的顺序

1、main.m是程序的入口

2、UIApplicationMain()创建应用程序对象，并且为此对象指定委托，检测程序的执行，同时开启事件循环，处理程序接收到的事件

3、UIApplicationDelegate方法的执行

4、加载window

5、指定根视图控制器

6、在指定的试图控制器中添加控件，实现应用程序界面

63、OC中是所有对象间的交互是如何实现的?

通过指针实现的

64、objective-c中的类型转换分为哪几类?

可变与不可变之间的转化;

可变与可变之间的转化;

不可变与不可变之间。

65、获取项目根路径，并在其下创建一个名称为userData的目录?

```
// 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];
}
```

66、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。
参考target-action

67、浅复制和深复制的区别?

浅层复制（copy）：只复制指向对象的指针，而不复制引用对象本身。//通过对象的指针来访问这个对象----只赋值地址
深层复制（mutableCopy）：复制引用对象本身---再创建一个对象
意思就是有个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。//当修改A时，A copy不变。

68、obc中可修改和不可以修改类型

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。

69、什么是安全释放?

[_instance release],_instance = nil;

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

变量的作用域不同，@protected 该类和所有子类中 的方法可以直接访问这样的变量，这是默认的;
@private 该类中的方法可以访问这样的变量，子类不可以;
@public除了自己和子类方法外，也可以被其他类或者其他模块中的方法访问;
@package 目前尚未得出结论

71、OC中异常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?

了解一下异常捕获
CPU的开销:

72、关于Objective-C++中的异常处理，可以相互捕获到吗?

不可以;

73、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?

74、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?.

序列化就是：归档

75、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.---NSRunLoop，NSTimer需要自己实现----GCD的返回主线程的方法，看前面的题

run loop，正如其名称所示，是线程进入和被线程用来响应事件以及调用事件处理函数的地方。需要在代码中使用控制语句实现run loop的循环，也就是说，需要代码提供while 或者 for循环来驱动run loop。在这个循环中，使用一个runloop对象[NSRunloop currentRunloop]执行接收消息，调用对应的处理函数。

76、描述下拉刷新的实现机制?

获取数据，刷新页面

77、什么是沙盒? 沙盒包含哪些文件，描述每个文件的使用场景。如何获取这些文件的路径? 如何获取应用程序包中文件的路径?

沙盒是某个iphone工程进行文件操作有此工程对应的指定的位置，不能逾越。
包括：四个文件夹：documents，tmp，app，Library。
手动保存的文件在documents文件里。
Nsuserdefaults保存的文件在tmp文件夹里。
Documents 目录：您应该将所有de应用程序数据文件写入到这个目录下。这个目录用于存储用户数据或其它应该定期备份的信息。AppName.app 目录：这是应用程序的程序包目录，包含应用程序的本身。由于应用程序必须经过签名，所以您在运行时不能对这个目录中的内容进行修改，否则可能会使应用程序无法启动。Library 目录：这个目录下有两个子目录：Caches 和 PreferencesPreferences 目录包含应用程序的偏好设置文件。您不应该直接创建偏好设置文件，而是应该使用NSUserDefaults类来取得和设置应用程序的偏好.Caches 目录用于存放应用程序专用的支持文件，保存应用程序再次启动过程中需要的信息。tmp 目录：这个目录用于存放临时文件，保存应用程序再次启动过程中不需要的信息。

获取这些目录路径的方法：
1，获取家目录路径的函数：
NSString *homeDir = NSHomeDirectory();
2，获取Documents目录路径的方法：
NSArray *paths =NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
NSString *docDir = [paths objectAtIndex:0];
3，获取Caches目录路径的方法：
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSCachesDirectory, NSUserDomainMask, YES);
NSString *cachesDir = [paths objectAtIndex:0];
4，获取tmp目录路径的方法：
NSString *tmpDir = NSTemporaryDirectory();
5，获取应用程序程序包中资源文件路径的方法：
例如获取程序包中一个图片资源（apple.png）路径的方法：
NSString *imagePath = [[NSBundle mainBundle] pathForResource:@”apple” ofType:@”png”];
UIImage *appleImage = [[UIImage alloc] initWithContentsOfFile:imagePath];
代码中的mainBundle类方法用于返回一个代表应用程序包的对象。

78、介绍一下XMPP？有什么优缺点吗？

XMPP（Extensible Messaging and Presence Protocol，前称）是一种以XML为基础的开放式实时通信协议，是经由互联网工程工作小组（IETF）通过的互联网标准。简单的说，XMPP就是一种协议，一种规定。就是说，在网络上传东西，要建立连接，TCP/IP连接，建立后再传东西，而XMPP就是规定你传的东西的格式。XMPP是基于XML的协议。

优点

开放：
　XMPP协议是自由、开放、公开的，并且易于了解。 而且在客户端 、 服务器 、 组件 、 源码库等方面，都已经各自有多种实现。
标准：
　互联网工程工作小组（ IETF ）已经将Jabber的核心XML流协议以XMPP之名，正式列为认可的实时通信及Presence技术。而XMPP的技术规格已被定义在RFC 3920及RFC 3921 。 任何IM供应商在遵循XMPP协议下，都可与Google Talk实现连接。
证实可用：
　第一个Jabber(现在XMPP)技术是Jeremie Miller在1998年开发的，现在已经相当稳定；数以百计的开发者为XMPP技术而努力。 今日的互联网上有数以万计的XMPP服务器运作着，并有数以百万计的人们使用XMPP实时传讯软件。
分散式：
　XMPP网络的架构和电子邮件十分相像；XMPP核心协议通信方式是先创建一个stream，XMPP以TCP传递XML数据流，没有中央主服务器。 任何人都可以运行自己的XMPP服务器，使个人及组织能够掌控他们的实时传讯体验。
安全：
　任何XMPP协议的服务器可以独立于公众XMPP网络（例如在企业内部网络中），而使用SASL及TLS等技术的可靠安全性，已自带于核心XMPP技术规格中。
可扩展：
　XML 命名空间的威力可使任何人在核心协议的基础上建造定制化的功能；为了维持通透性，常见的扩展由XMPP标准基金会 。
弹性佳：
　XMPP除了可用在实时通信的应用程序，还能用在网络管理、内容供稿、协同工具、文件共享、游戏、远程系统监控等。
多样性：
　用XMPP协议来建造及布署实时应用程序及服务的公司及开放源代码计划分布在各种领域；用XMPP技术开发软件，资源及支持的来源是多样的，使得使你不会陷于被“绑架”的困境。

缺点

数据负载太重：
　随着通常超过70%的XMPP协议的服务器的数据流量的存在和近60%的被重复转发，XMPP协议目前拥有一个大型架空中存在的数据提供给多个收件人。 新的议定书正在研究，以减轻这一问题。
没有二进制数据：
XMPP协议的方式被编码为一个单一的长的XML文件，因此无法提供修改二进制数据。 因此， 文件传输协议一样使用外部的HTTP。 如果不可避免，XMPP协议还提供了带编码的文件传输的所有数据使用的Base64 。 至于其他二进制数据加密会话（encrypted conversations）或图形图标（graphic icons）以嵌入式使用相同的方法。

79、谈谈对性能优化的看法，如何做？

控制好内存，不用的内存实时释放；冗余代码；用户体验度；耗时操作，开线程进行处理

80、写一个递归方法：计算N的阶乘，然后将计算结果进行存储。以便应用退出后下次启动可直接获取该值。

开启一个线程，在线程种实现递归的方法，将结果存到本地，下次运行时先看本地，没有在运行这个递归方法。--

---用代码实现----

81、简述应用程序按Home键进入后台时的生命周期，和从后台回到前台时的生命周期?

应用程序：

-[AppDelegate application:willFinishLaunchingWithOptions:]
-[AppDelegate application:didFinishLaunchingWithOptions:]
-[AppDelegate applicationDidBecomeActive:]
退到后台:
-[AppDelegate applicationWillResignActive:]
-[AppDelegate applicationDidEnterBackground:]
回到前台:
-[AppDelegate applicationWillEnterForeground:]
-[AppDelegate applicationDidBecomeActive:]

ViewController之间

加载页面:
-[mainViewController viewDidLoad]
-[mainViewController viewWillAppear:]
-[mainViewController viewWillLayoutSubviews]
-[mainViewController viewDidLayoutSubviews]
-[mainViewController viewDidAppear:]
退出当前页面:
-[mainViewController viewWillDisappear:]
-[mainViewController viewDidDisappear:]
返回之前页面:
-[mainViewController viewWillAppear:]
-[mainViewController viewWillLayoutSubviews]
-[mainViewController viewDidLayoutSubviews]
-[mainViewController viewDidAppear:]

82、简述值传递和引用传递的区别?

所谓值传递，就是说仅将对象的值传递给目标对象，就相当于copy；系统将为目标对象重新开辟一个完全相同的内存空间。
所谓引用传递，就是说将对象在内存中的地址传递给目标对象，就相当于使目标对象和原始对象对应同一个内存存储空间。此时，如果对目标对象进行修改，内存中的数据也会改变。

83、NSArray和NSMutableArray的区别，多线程操作哪个更安全?

NSArray更安全，当同时被访问时，NSArray是不可改变

84、当前有一个数组，里面有若干重复的数据，如何去除重复的数据？（会几个写几个）

可以由数组，到集合。。。。

85、isKindOfClass、isMemberOfClass、selector作用分别是什么

isKindOfClass，作用是，某个对象属于某个类型，包括继承的类型---
isMemberOfClass：某个对象确切属于某个类型，是不是具体的实例
selector：通过方法名，获取在内存中的函数的入口地址

86、写出下面程序段的输出结果

```
NSDictionary *dict = [NSDictionary dictionaryWithObject:@"a string value" forKey:@"akey"];
NSLog(@"%@", [dict objectForKey:@"akey"]);
[dict release];
```

打印输出 a string value，然后崩溃----原因：便利构造器创建的对象，之后的release，会造成过度释放

87、请写出以下代码的执行结果

```
NSString  * name = [ [ NSString alloc] init ];
name = @”Habb”;
[ name  release];
```

打印输出结果是： Habb，在[name release]前后打印均有输出结果
---会造成内存泄露---原先指向的区域变成了野指针，之后的释放，不能释放之前创建的区域

88、请分别写出SEL、id、@的意思?

SEL是“selector”的一个类型，表示一个方法的名字-------就是一个方法的入口地址
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，所以在使用id的时候不需要加*。

@：OC中的指令符

89、以.mm为拓展名的文件里，可以包含的代码有哪些?

.mm是oc和C++混编类型文件后缀，给编译器识别的。

90、说说如何进行后台运行程序?

答：判断是否支持多线程

```
UIDevice* device = [UIDevice currentDevice];
BOOL backgroundSupported = NO;
if ([device respondsToSelector:@selector(isMultitaskingSupported)])
backgroundSupported = device.multitaskingSupported;
```

声明你需要的后台任务Info.plist中添加UIBackgroundModes键值，它包含一个或多个string的值，包括audio:在后台提供声音播放功能，包括音频流和播放视频时的声音 location：在后台可以保持用户的位置信息 voip：在后台使用VOIP功能

前面的每个value让系统知道你的应用程序应该在适当的时候被唤醒。例如，一个应用程序，开始播放音乐，然后移动到后台仍然需要执行时间，以填补音频输出缓冲区。添加audio键用来告诉系统框架，需要继续播放音频，并且可以在合适的时间间隔下回调应用程序；如果应用程序不包括此项，任何音频播放在移到后台后将停止运行。除了添加键值的方法，IOS还提供了两种途径使应用程序在后台工作：

Task completion—应用程序可以向系统申请额外的时间去完成给定的任务

Local notifications—应用程序可以预先安排时间执行local notifications 传递实现长时间的后台任务：应用程序可以请求在后台运行以实现特殊的服务。这些应用程序并不连续的运行，但是会被系统框架在合适的时间唤醒，以实现这些服务

91、你了解svn,cvs等版本控制工具么?

版本控制 svn,cvs 是两种版控制的器,需要配套相关的svn，cvs服务器。scm是xcode里配置版本控制的地方。版本控制的原理就是a和b同时开发一个项目，a写完当天的代码之后把代码提交给服务器，b要做的时候先从服务器得到最新版本，就可以接着做。 如果a和b都要提交给服务器，并且同时修改了同一个方法，就会产生代码冲突，如果a先提交，那么b提交时，服务器可以提示冲突的代码，b可以清晰的看到，并做出相应的修改或融合后再提交到服务器。