1.什么是类

```
类是对一些具有相通对属性和行为,并且存在的事物的一种描述统称
```

2.面向对象的三大特性

```
1.封装
    a.函数的实现现在.c文件中,函数的声明写在.h文件中.当我们调用这个函数的时候只需要调用这个.h文件，不需要知道如何实现
    b.方法的实现写在.m文件中，方法的声明写在.h文件中.不需要知道实现,知道如何调用即可
    c.对于类中的属性，进行setter 和getter方法的封装.
2.继承
    类A中有若干属性，我希望类B创建出来就拥有类A中的所有属性和方法.这就叫做继承
    单根性－－传递性
3.多态
    同一种行为,不同的事物有不同的表现方式
    子类重写父类的方法,通过父类指针指向子类对象，  调用的方法 ，就是子类重写之后的方法.
```

3.属性的访问修饰符(修饰后子类能否继承,能否访问)

```
访问修饰符
1.@public
    a.在类和子类对外部可以通过->访问，也可以通过get / set访问
    b.在类和子类对内部可以通过_属性名访问，也可以通过->访问，也可以通过get / set访问
2.@protect
    a.在类和子类对外部不可以通过->访问，可以通过get / set访问
    b.在类和子类对内部可以通过_属性名，也可以通过 ->访问，可以通过get/set访问
3.@private
    a.在类和子类对外部不可以通过->访问，可以通过get / set访问
    b.在当前类可以通过_属性名访问，也可以通过->访问，可以通过get / set访问
    c.在子类对内部不可以通过->访问，但能通过set / get方法访问.
```

4.多态 （编译检查 和运行检查）

```
1.多态
    同一种行为,不同的事物有不同的表现方式
    子类重写父类的方法,通过父类指针指向子类对象，  调用的方法 ，就是子类重写之后的方法.
2.编译检查
    检查的是指针对应的类型的类里面有没有对应的方法
3.运行检查
    检查的是指针指向的对象,对象的类里面有没有对应的方法.
```

5.点语法

```
本质是对方法的调用
eg:如果是取值 p1.age  等同于 [p1 age];   如果是赋值p1.age = 10;等同于 [p1 setAge:10] ;
如果这里age不是一个get方法而是自定义的一个方法比如 -(void)age{ nsl@"hahaha"}
这里就是输出hahaha，所以可以看出点语法的本质是对方法对调用.
```

6.@property的作用

```
1>在类的.h生成属性xxx对应的标准的set/get的声明
2>在类的.m中帮我们生成_xxx这个属性的定义
3>在类的.m中帮我们生成_xxx这个属性的set/get的实现
```

7.@property到参数有哪些?

```
1.MRC中
    生成setter/getter方法的参数    assign retain（1.非oc对象(不全部是)  2.oc对象带内存管理）
2.ARC中
    生成setter/getter方法的参数   assign strong （1.非oc对象(不全部是)  2.oc对象只是简单的赋值）
3.是否生成只读和读写封装    readonly    readwrite（1.只读    2.读写封装）
4.多线程 atomic nonatomic （1.安全效率低    2.不安全效率高）
5.getter  和 setter  修改对应的名字
```

8.MRC下@property的参数有什么用?(copy retain)XXX

```
reatin 本质是生成setter的方法带了判断
-(void)setCar:(LWCar*)car{
    if(_car != car){
    [_car release];
    _car = [car release];
    }
}
```

9.ARC下@property的参数(copy strong)XXX

```
strong 本质只是赋值
-(void)setCar:(LWCar*)car{
    _car = car;
}
weak（可以替换strong）   本质是让_car指向的指针变为一个弱指针(循环引用才回使用！)
-(void)setCar:(LWCar*)car{
    __weak  _car = car; //弱指针
}
```

10.分类和延展的作用?

```
1.一个类分成多个模块，一个本类，多个分类.本质还是一个类.分类不能定义属性，只有方法声明和方法的实现,分类的.h在main函数中导入.
    给系统的类写分类叫做非自定义
2.一个匿名分类.用来私有化成员.可以定义属性.
    匿名分类100%使用在本类的.m文件中.
    定义的属性是私有属性.(手写getter / setter方法)
    定义的方法是私有方法.只能内部调用
    @property的属性也是私有属性.(生成的getter / setter声明在.m里);
    "延展头文件只能包含在本类的.m文件中."
    "扩充":延展单独的.h文件,被本类.m包含.哪么本类对象可以通过getter / setter方法在类外部调用延展中的带getter / setter方法带属性.
```

11.block的作用

```
block是一个数据类型.用来存储一段代码.可以有返回值，可以有参数.
block 不可以修改局部变量的值,局部变量被__block修饰过后可以.
block可以作为函数的参数.传递到函数到内部执行.
blcok可以作为方法参数.将一段代码传递到方法中执行.

block 和函数到相同点:都是将一段代码封装起来.实现一个功能
不同点:block是一个数据类型      函数是一个函数
    block可以做参数      函数不可以直接做参数.可以通过函数指针做参数.本质传递到是地址
    block可以做返回值.不推荐使用
```

12.协议的作用,修饰符

```
协议是专门用来声明方法的.遵守类协议的类.就相当于有了这些类的声明.还是要实现的.
@required 修饰的协议必须要在遵守的类中实现.否则会报警告
@optional 修饰的协议可以实现也可以不实现.不会报警告
协议可以继承.:协议A继承协议B哪么协议A就拥有协议B的所有方法的声明
协议可以多继承.: 协议A可以同时继承协议B和协议C
类可以多遵守: 类A可以同时遵守多个协议
所有的协议都直接或间接的继承NSObject这个协议（基协议）
协议的类型限制.id<协议> a; a指向的对象必须遵守协议.否则⚠警告
```

13.字符串的截取

```

```

14.字符串的删除

```

```

15.字典的插入

```

```

16.深拷贝和浅拷贝？

```
1.
```