```
MRC项目 转 ARC项目
                                                   ARC
                                                 选中项目跟节点 -> 选中target -> Build
Phases->Compile Sources (1 item) ->在需要
           ARC与MRC【掌握】
                               MRC 与 ARC 共存
                                                 设置为不使用ARC文件所在的行与complier
                                                 Flags的列的交汇处双击,写上命令 -fno-objc-
                          【掌握】作用:扩充原有类的功能
                                         名称:原有类的名称(分类名称)
                          【掌握】定义
                                         文件名称:原有类的名称 + 分配名称
                                        代码模块化
                          【了解】优点
                                        团队协作
                                        可以扩充系统自带类的功能
                                        在分类中可以声明和实现新的方法
                          【掌握】使用
                                        在分类的方法中可以访问原有类的成员变量
                                        给NSString类增加一个计算字符串中有多少数
                                        字的功能
           分类【掌握】
                          【掌握】练习
                                                                   给NSObject扩展分类,应该非常谨慎,因为他
                                        给NSObject方法增加一个run方法
                                                                   会作用到所的类上面
                                          分类只能扩充功能不能扩充属性
                                          在分类使用@property 只能生成方法的声明,
                                          不能生成方法的实现, 更不能生成成员变量
                                          分类中不能有与原来类同名的方法,不同分类
                                          中也不能有同名的方法
                          【掌握】注意点
                                          【理解】如果分类中如果存在与原来类中的同
                                          名函数,那么分类中的方法会覆盖原有类中的
                                          方法(哪怕你没有导入这个分类)
                                          【理解】如果多个分类中有相同的方法,最后
                                          将使用最后编译的那个分类中的方法
                           用在.m文件中,给类增加私有的成员变量和方
            【掌握】类扩展
                           类扩展中小括号中没有内容
                                      它是一种代表着方法的签名数据类型,我们也
成为它使一个选择器
                       概念【掌握】
                                   【掌握】定义普通的变量
                                                                               【掌握】判断一个对象是否实现了某个方法
                                                                                                               respondsToSelector:(SEL)
                                                                                                       performSelector:(SEL)
                                                                                                      performSelector:(SEL) withObject:
                                                                               [了解]执行对象上某个的方法
                                                                                                       (id)
                                   【理解】作为方法实参
                                                      与NSObject 相关方法配合使用
           SEL[掌握]
                       使用
                                                                                                      performSelector:(SEL) withObject:
                                                                                                      (id) withObject:(id)
                                                                                                           performSelector:(SEL) withObject:
(id) afterDelay:(NSTimeInterval)]
                                                                               【了解】延迟执行某个对象上的方法
                                  [了解] 方法形参
                                                  makeObject:(id) performSelector:(SEL)
                                  【了解】作为方法的返回值
                        【了解】OC中的方法调用过程
                           【掌握】作用: 封装代码
                           【掌握】优点:消除了函数不能能定义在方法
内部的缺点,使用起来更加灵活
                                          【掌握】定义变量
                                                           返回值类型 (^变量名称) (形参列表)
                                                                                              ^{ 语句... };
                                                                               无参无返回值
                                                                                              ^(参数列表){语句...}
                                          【掌握】定义功能
                                                           ^返回值类型 (形参列表)
                                                                               有参无返回值
                                                                               有参有返回值
                                                                                              ^(参数列表){语句...}
                           【理解】使用
                                                             返回值类型(^)(形参列表)
day02
                                          【掌握】作为函数参数
            【掌握】block
                                                             技巧:参数列表中写上参数的名称,方便以后
                                                             使用
                                                                 typedef 返回值类型 (^block类型名称) (形参
                                          【理解】typedef 来定义类型
                                            程序员的一天
                                            当一个方法中前面一部分是确定的后面一部分
也是确定的,只有中间一部分经常变化,那么
中间的这部分就可以提取成为一个block
                           【理解】使用场景
                                           block内部不可以改变局部变量的值
                           【掌握】注意点
                                           若想改变局部变量值,就需要使用__block 来修饰这个变量
                          【掌握】概念:两个或两个以上对象之间约定
                                        @protocol 协议名称 <NSObject>
                                                                     中间只写方法声明
                                        //声明方法
                                                                     需要遵守协议NSObject
                                                                     以@protocol 开头以@end 结尾
                                        @end
                          【掌握】定义
                                                                @protocol 协议名称 <协议1,协议2, 协议3 ...>
                                        一个协议也可以遵守多个协议
                                                                //声明方法
                                                                @end
                                               第一步: 定义协议
                                                                                                  @interface 类名: 父类名称 <协议>
                                                                在声明类的时候,在类名后面的通过<协议名称
                                                                >来遵守协议
                                                                                                  @end
            【掌握】协议
                                               第二步: 遵守协议
                                                                                   @interface 类名() <协议>
                          【掌握】协议的基本使用
                                                                在类扩展中遵守协议
                                                                                   @end
                                               第三步: 实现协议中的方法
                                                                                             @interface 类名: 父类名称 <协议1,协议2,协议3...>
                                                【掌握】注意点
                                                               一个类可以遵守多个协议中间以逗号隔开
                                                                                             @end
                                                                  通过这种方式定义出来的指针变量,只能指向 遵循了该协议的对象
                          【掌握】id 类型与协议
                                              id<协议名称> 变量名称
                                                                  通过这方式定义出来的指针变量,可以指向遵
                                                                  循该协议的任意对象
                                                          表示遵守该协议的类必须要实现的方法
                                            @required
                          【掌握】两个关键字
                                                          表示遵守协议的类可以选择实现的方法
                                            @optional
                                           当A类对象发生了什么事,想通知B类对象的时候(让B类对象成为A类对象的代理)
                               使用场景
                                           当A类对象有些事情,自己无法处理,需要B类对象帮助它处理(让B类对象成为A类对象的代理)
                                           当B类对象想要监听A类对象中发生的事情的时候(让B类对象成为A类对象的代理)
                                                           该协议名称为ADelegate,其中声明一些希望B
类实现的方法
                                           A类中定义协议
                                           A类中添加一个类型 id<ADelegate> delegate的属性
                                                                                       注意在调用这个方法之前首先要判断delegate
                                           A类中当发生了什么事的时候,调用delegate相应的方法
                                                                                       是否确实实现了这个方法,只有delegate实现
这个方法才能调用这个方法
                               实现步骤
                                                                 @interface B() <ADelegate>
            【掌握】代理设计模式
                                           B类遵守ADelegate的协议
                                                                         注意: 这里实现的方法可以是某一个或某几个
                                           B类中实现ADelegate协议中声明的方法
                                                                          或是全部,取决B对象希望监听A对象的什么事
                                           将B类对象赋值给A类对象的delegate属性
                                                        婴儿想吃了要通知保姆,让保姆成为婴儿代理
                                                        保姆想知道婴儿什么时候想吃了,让保姆成为
                               使用场景
                                           婴儿与保姆
                                                        婴儿自己不会吃,需要保姆帮忙,让保姆成为
```

婴儿的代理

Edit -> Refactor-> Convert to Objective-C