- 思维导图
  - 栈与堆
    - 栈【掌握】
      - 内存由系统管理
      - 局部变量保存在栈中
      - 当变量离开了其所在的代码块就会被回收
    - 堆
      - 需要程序员自己管理
      - OC中的对象保存在堆中【掌握】
      - 内存管理不当造成的两个问题【掌握】
        - 内存泄露
          - 不在需要的对象没有释放，导致内存泄露，内存泄露会导致应用闪退
        - 野指针
          - 正在使用的对象被释放了，导致野指针，访问野指针导致程序崩溃
      - MRC 手动内存管理【理解】
        - 引用计数
          - 每一个对象上都有一个引用计数器
          - 当对象出生的时候引用计数器为1
          - 当调用对象的 retain方法时候计数器+1
          - 当给对象发送一条release消息的时候，计数器-1
          - 当一个对象的引用计数为0的时候，这个对象立即被回收
        - 内存管理原则
          - 谁给一个对象发送 alloc,copy,retain消息，就必须在适当时候给这个对象发送相应的release或 autorelease消息
          - 当你需要一个对象的时候就给这个对象的引用器+1，当你不在需要这个对象了，就将该对象引用计数器-1
        - 多个对象内存管理
          - setter方法
            - 当设置新对象的时候需要对新对象做一次retain操作
            - 当原来的对象不需要了，需要对原来的对象做一次release操作
            - 如果是同一个对象不需要上面的操作
            - QQ游戏的房间
          - dealloc方法
            - 当一个对象即将被销毁的时候会调用这个方法
            - 它就相当于对象的遗言，在这里释放其成员所占用的内存
            - 在释放内存之前首先要调用[super dealloc]
          - 循环引用
            - 什么是循环引用
              - 当两个或多个对象间的引用构成一个封闭的环就是循环引用
            - 导致问题
              - 环中的所有对象都释放不了
            - 解决方案
              - 有一端必须是assign的
        - @property 的使用
          - 内存管理
            - 基本数据类型使用assign
            - 普通对象一般使用 retain
            - NSString 及 block 对象使用copy
            - 循环引用一端使用assign
          - 权限控制
            - readonly
              - 只读仅生成getter方法
            - readwrite
              - 读写生成getter方法和setter方法
          - 线程安全
            - `atomic`
              - 线程安全的，会给setter方法加锁,访问速度比较慢
            - `nonatomic`
              - 非线程安全的，不会给setter方法加锁，访问速度快
          - 方法名称定制
            - setter
              - 定制setter方法名称
            - getter
              - 定制getter方法名称
        - 自动释放池
          - 作用
            - 延迟对象的释放
          - 原理
            - 自动释放池栈
          - 使用
            - 当你调用autorelease的时候会该对象放到离最近的自动释放池中
            - 当执行到自动释放池后面的大括号的时候，自动释放池会出栈，此时会该栈中的会所有对象做一次release操作
            - 你调用几次autorelease，当自动释放池被销毁的时候就会调用几次release
          - 优点
            - 你不必时刻关注对象的释放，让内存管理变的简单
          - 缺点
            - 对象不能及时的得到释放，如果对象占用内存很大，使用它会导致你程序闪退
          - 使用场景
            - 当一个对象很小，使用次数也比较少的时候可以使用自动释放
            - 当一个对象占用内存比较大的时候不要使用自动释放出
            - 快捷构造方法
            - 其他需要延长对象生命周期的地方
      - ARC 自动引用计数【掌握】
        - 什么是自动引用计数
          - 他是编译器特性，不是运行时特性
          - 编译器会再适当的时候插入内存管理的代码
        - 强指针
          - 默认所有的指针都是强指针
          - 只要有强指针指向这个对象，这个对象就不会被销毁
          - 只要没有强指针指向这个对象，该对象立即被销毁
          - 关键字：__strong
        - 弱指针
          - 弱指针不影响对象的销毁
          - 关键字 __weak
        - 循环引用
          - 什么是循环引用
            - 当多个对象间的引用出现环的时候就是循环引用
          - 问题
            - 环中的所有对象都是方法释放不了
          - 解决方案
            - 当出现循环引用的时候，一端必须使用__weak
        - @property的使用
          - 内存管理
            - assign
              - 不参与内存管理，用于基本数据类型
            - strong
              - 强引用：用于普通对象
            - copy
              - 复制：用于字符串与block
            - weak
              - 弱引用：防止循环引用
        - 自动释放池
          - 通过类方法创建出来的对象都是放在离它最近的方法池中
          - 当自动释放池销毁的时候才会才能解除自动释放对该对象的引用