内存管理

- 属性修饰符
  - @property / @synthesize / @dynmic
    - @property (nonatomic/atomic, readwrite/readonly, assgin/strong/weak/copy/retain) id iver;
      - ⚠️: 默认 (atomic,readwrite,strong).
      - ⚠️: 默认 @protected (受保护的).
      - @property作用:编译器自动生成
        1. iver 成员变量.
           { id _ivar; }
        2. 声明 ivar 的 getter/setter.
           -(id)ivar;
           -(void)setIvar:(id)ivar;
        3. @synthesize.
    - @synthesize iver = _iver;(关联属性ivar和成员变量_ivar)
      - @synthesize作用:编译器自动生成
        1. 实现 ivar 的 getter/setter.
    - @dynmic ivar;
      - ⚠️: 需要手动添加 getter/setter.否则调用self.语法, 会Crash.
      - @dynmic作用:阻止编译器自动生成 getter/setter 的 声明 和 实现.
  - nanatomic / atomic
    - nonatomic: getter/setter不加锁,线程不安全,性能高.
    - atomic: getter/setter加锁,线程安全,性能低.
  - readonly / readwrite
    - readonly: 只读.
    - readwrite: 读写.
  - assgin / weak / unsafe_unretained
    - ❓问题: 如何选择关键字修饰属性?
      如何选择关键字, 取决于内存何时释放.
  - retain / strong / copy
    - assgin: 简单赋值
      weak:
      unsafe_unretained:
- NSString
- 执行顺序
  - alloc:
    +alloc → +allocWithZone: → class_createInstance -> calloc
  - retainCount:
    -retainCount → __CFDoExternRefOperation → CFBasicHashGetCountOfKey
  - retain:
    -retain → __CFDoExternRefOpertaion → CFBasicHashAddValue
  - release:
    -release → __CFDoExternRefOperation → CFBasicHashRemoveValue
    (CFBasicHashRemoveValue 返回0时, -release调用dealloc)
- 内存分布
  - 栈区(stack) 高地址 ⬇️
    - 函数{ }中的 *局部*变量
      - 系统自动释放(超出作用域后释放)
        - 0x7ffeeda75070 高地址
  - 堆区(heap)
    - ①. alloc/new 初始化的 *Objective-C NSObject 对象*
      ⚠️字符串(类/实例方法 初始化的 .length>=10 非中文 字符串 或 任意长度 中文 字符串) 在堆区⚠️
      - 手动 release, 立即引用计数器-1.
        当 计数器 = 0 时, 调用 dealloc 释放内存.
        ⚠️此刻, 内存为解除分配, 放到可用内存池, 被标记为可重用, 如果该内存没有被覆写, 地址内仍然有数据. 如果指针指向该地址, 会成为野指针⚠️
        - 0x7fc3cde008b0 - 0x6000012c9940
  - 全局(静态)区 (bss/data)
    * -全局/静态 未初始化区
    (bss)-
    * -全局/静态 已初始化区
    (data)-
    - bss:
      ①. *全局 未初始化* 变量
      ②. static 修饰的 未初始化 所有(局部和全局) 变量,
      即 *静态 未初始化* 变量
      - 运行时不释放内存, 程序结束才释放内存
        - 0x109c8c198
    - data:
      ①. *全局 已初始化* 变量
      ②. static 修饰的 未初始化 所有(局部和全局) 变量,
      即 *静态 已初始化* 变量
      - 运行时不释放内存, 程序结束才释放内存
        - 0x109c8c198
  - 常量区(rodata)
    - ①. *字符串 常量*
      * @“”方式 初始化的 字符串
      * 类/实例方法 初始化的 .length<10 且 非中文 字符串(⚠️否则, 在堆区⚠️)
      ②. *宏(#define)中的 常量*
      - 运行时不释放内存, 程序结束才释放内存
        - 0x109c8c198
  - 代码区(text) 低地址
    - *code*
      - 运行时不释放内存, 程序结束才释放内存
        - 0x109c8c198 低地址
- MRC(手动引用计数)
  - 手动添加
    - retain
    - release
      立即释放
    - autorelease
      延迟释放, 注册到 autoreleasepool 中, pool结束时, 自动release.(一次RunLoop结束时,创建新pool,销毁旧pool)
  - 编译器自动添加
    - retain
    - release
      立即释放
- ARC(自动引用计数)
  LLVM编译器进行内存管理,
  在适合的地方插入 retain 和 release.
  - __weak, 弱引用
    避免相互强引用(循环引用)
    避免野指针
  - __strong, 强引用
    默认