Proper Attributes

- 프로퍼티에 적용할 수 있는 설정
  - nullable / nonnull / null_unspecified / null_resettable
  - readwrite / readonly
  - getter / setter
  - retain / unsafe_unretained
  - strong / weak
  - copy / assign
  - atomic / nonatomic

nullable / nonnulla / null_unspecified / null_resettable

- 프로퍼티가 nil이 할당될 수 있는지에 대한 여부
- 기본설정 : null_unspecified(뭐든지 상관없다, nil이던 아니던 어떤값이던 들어가있음)
- 객체타입(인스턴스 nil값이 들어갈수있는)에만 사용
  - nullable : 프로퍼티가 null일 수 있다.
  - nonnull : 프로퍼티가 null일 수 없다.
  - null_unspecified : 프로퍼티의 null 여부가 지정되어있지 않다 - 스위프트
  - null_resettable : 객체가 init되면서 해당 프로퍼티에 기본값을 할당하므로 nil일 수 없다., 처음에는 null값을 넣을수 없지만 나중에는 null값을 넣을 수있다.

readwrite / readonly

- 프로퍼티의 getter와 setter의 생성에 관한 설정
  - readwrite : getter와 setter를 모두 자동생성
  - readonly : getter만 자동생성
- 기본 설정 : readwrite

getter / setter

- 프로퍼티의 getter와 setter를 수동으로 지정
  - getter : getter 메서드의 이름을 수동지정
  - setter : setter 메서드의 이름을 수동지정
- 기본설정
  - getter => 프로퍼티 이름
  - setter => set프로퍼티 이름

Automatic Reference Counting

- 주소(참조) 참조할 경우 어디서 참조를 하였는지 알아야함
- Objective-C의 메모리 관련 용어
  - retain(보관)
  - release(해제)
  - retain count(메모리에 객체가 존재하거나 사라지는 거에 대한 확인사항, count로 확인할수있다.)

메모리 규칙

- alloc을 하면 retain count가 1 증가한다.
- retain을 호출하면 retain count가 1 증가한다.
- release를 호출하면 retain count가 1 감소한다.
- retain count가 0이 되면 객체가 메모리에서 해제된다.
- delloc을 호출하면 메모리에서 해제된다.

ARC 규칙

- retain count를 임의로 조절할 수 없다.
- retain count는 코드 블럭(메서드의 return직전에 release시켜준다)을 벗어나면 1 감소한다.
- strong 참조가 해제되면 retain count가 1 감소한다.
- delloc을 명시적으로 호출할 수 없다.
- ARC는 Objective-C 참조형 객체에만 적용된다. - 구조체와 클래스의 차이 중 하나

ARC를 제대로 이해하지 못하면

- 순환참조문제가 발생할 수 있다.
- 메모리에서 해제되지 못한 객체가 메모리에 남아 프로세스가 종료되기 전까지 좀비로 남는다.
- 좀비 객체가 많아지면 프로세스가 차지하는 메모리가 커진다.
- 결국 시스템과 성능에 지대한 악영향을 끼칠 수 있다.

메모리관련 property attributes

- retain / unsafe_unretained
- strong / weak
- copy / assign

retain / unsafe_unretained

- 프로퍼티의 retain count에 관한 설정
  - retain : 프로퍼티에 세팅할 때 retain count를 1 증가시킴
  - unsafe_unretained : 프로퍼티에 세팅할 때 retain count 변경 없음
- 기본 설정 : retain(ARC가 아닌 환경에서)

strong / weak

- 프로퍼티의 retain count에 관한 설정
  - strong : 프로퍼티에 세팅할 때 retain count를 1 증가시킴
    - nil값을 넣어주면 그순간에 retain count를 1 감소시킴
  - weak : 프로퍼티에 세팅할 때 retain count 변경없음, unsafe_unretained과 차이는 객체가 메모리에서 해제되었을때 자동으로 변수에 nil이 할당된다는 것
- 기본 설정 : strong
- remove 연결을 끊어준다.

copy / assign

- 프로퍼티에 값을 넘기는 방법에 관한 설정
  - copy : 객체 타입에서 사용, 객체를 새로 생성(copy를 할때마다, 딕셔너리가 있다면 그것을 복사하여 그복사한것을 참조하겠다)하여 프로퍼티에 할당
  - assign : 값 타입(integer, float, double등)에서 사용

property attribute

```
@interface Musician : NSObject

@property (nonnull, readonly) NSString *groupName;
@property (null_resettable) NSNumber *memberConut;
@property (nullable, getter=companyName) NSString *company;
@property (null_unspecified) NSString *manager;
```

strong / weak

```
//weak은 약한 연결
//연결이 되어도 retaincount가 증가되거나 감소되지 않는다.
//스토리보드에
@property (weak, nonatomic) IBOutlet UILabel *weaklabel;

//strong은 강한 연결
//연결시 retaincount가 증가 및 감소 한다.
@property (strong, nonatomic) IBOutlet UILabel *stronglabel;

@end

@implementation ViewController

//버튼을 누를 경우 위의 라벨이 둘다 사라지도록 한다.
- (IBAction)removebutton:(id)sender
{
    //uilabel 프로퍼티에게 superview에서 떨어져라
    [self.weaklabel removeFromSuperview];
    [self.stronglabel removeFromSuperview];
}

//버튼을 누를 경우 사라진 라벨을 다시 올린다.
- (IBAction)addbutton:(id)sender
{
    //view에게 uilabel을 다시 올려라
    [self.view addSubview:self.weaklabel];
    [self.view addSubview:self.stronglabel];
    NSLog(@"weak : %@, strong : %@", self.weaklabel, self.stronglabel);
}
```