Use Automatic Reference Counting ARC

- 자동으로 객체들의 메모리를 관리
- 오브젝티브-씨 에서 제공하는 자동메모리 관리기능
- 어플을 개발하는데 있어 Retain(보관) / Release(해제)에 대한 생각으로 개발하는데 있어 퍼포먼스를 낮추는 부분을 해소하기위해 만들어짐
- 컴파일 시간에 메모리에 대한 관리를 추가하여 객체의 생존 시간을 필요한 만큼 보장하는데 있다, 시간이 길지않다

ARC도입이유

- 앱의 비정상 종료 원인 중 많은 부분이 메모리 문제
- 많은 개발자들이 수동적인 Retain(보관) / Release(해제) 메모리 관리로 힘들어함
- Retain(보관) / Release(해제)로 코드 복잡도가 증가

ARC를 위한 규칙

- 더이상 retain / release / autorelease를 코드에 직접 삽입할 필요가 없다, 컴파일러가 알아서 다 해준다. 블록도 object이므로 기존 블록에 사용하던 copy, autorelease는 불필요하다. 메소드의 return에 대한 autorelase도 고려할 필요없다. 전부 컴파일러가 알아서 한다. delloc 메소드는 사용할수는 있지만 이는 인스턴스 변수들의 메모리 해제가 아닌 자원관리 차원에세 허용된다. 또한 개발자에 의해 재정의된 delloc 메소드에서는 [super delloc]을 호출해서는 안된다.
- ARC는 object만을 고려한다. 구조체 내의 object는 ARC가 계산하기 어렵다. 그냥 구조체가 아닌 objective-c class를 사용하라
- core foundation은 objective-c의 object타입이 아니므로 core foundation과 object간의 명시적인 타입 캐스팅이 필요하다, 추가로 core foundation 스타일의 객체에는 CFRetain, CFRelease와 같은 메모리 관리 함수를 사용할 수있다.“Managing Toll-Free Bridging”
- NSAutoreleasePool 사용하지마라, NSAutoreleasePool은 내부적으로 보면 실제로는 object가 아니다. 대신 @autoreleasepool{}을 사용하라. 속도도 더 빠르다.

새로운 지시어

- 프로퍼티 속성 / strong = retain, weak = assign
- 변수지시자 / `_strong` : 디폴트 지시자. 객체의 소유권을 획득한다. (retain count 증가)

`_weak` : 객체의 소유권을 가져오지 않고 참조한다. 참조한 객체가 소멸되면 nil 로 설정된다.

`_unsafe_unretained` : _weak 와 비슷하지만 참조 객체 해제되어도 nil 로 재설정되지 않는다. 변수의 값은 특정 포인터를 계속 가리키고 있으므로 변수 사용에 유의해야 한다. 오류가 발생할 수 있다.

`_autoreleasing` : 보통 메소드의 (id *) 타입 파라미터 인수에 사용하며 함수 리턴시 자동으로 해제된다. Cocoa Framework 의 간편 생성자(Convenience Initializer)역시 _autoreleasing 타입의 객체를 반환함.

_strong, _weak, _autoreleasing 변수는 생성시 자동으로 nil 로 초기화 된다. 변수 선언 형식은 다음과 같다.