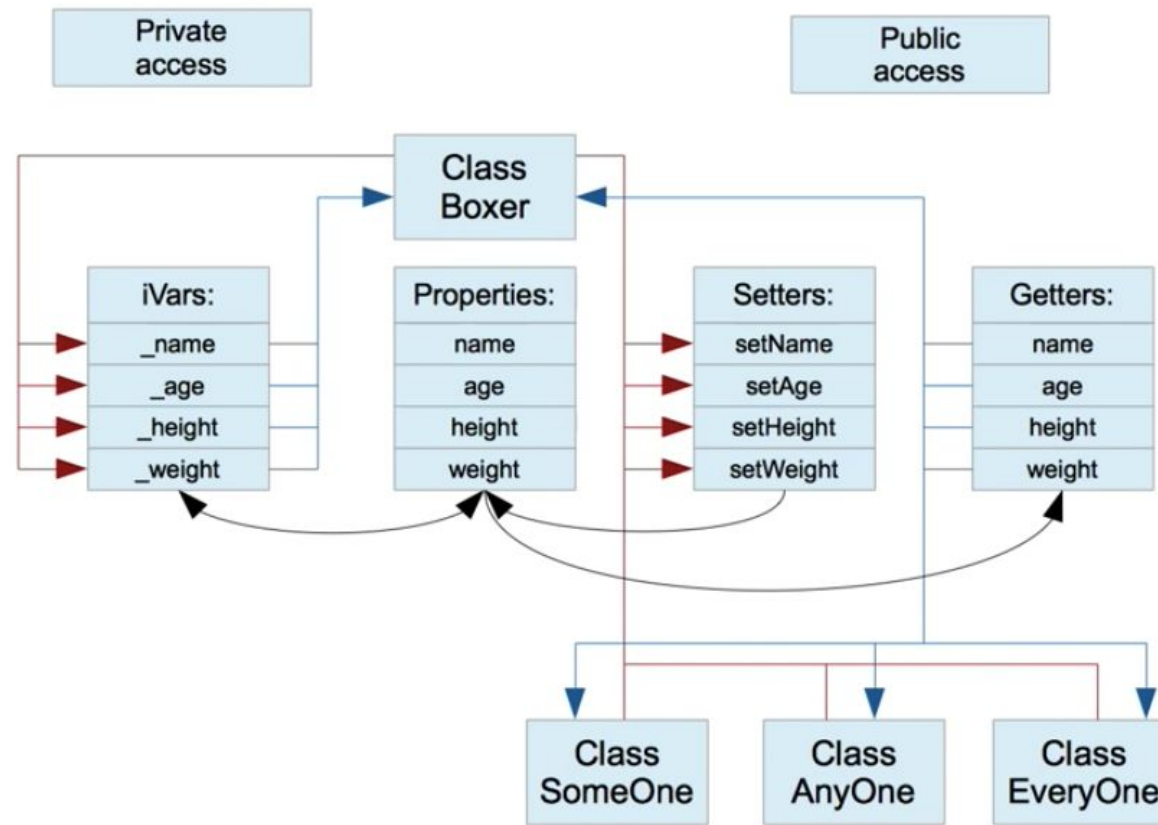


Свойства класса (properties)

Общая схема



Атрибуты свойств классов

- атрибуты доступности (readonly/readwrite);
- атрибуты владения (retain/strong/copy/assign/unsafe_unretained/weak);
- атрибут атомарности (atomic/nonatomic);
- nullability атрибут (null_unspecified/null_resettable/nullable/nonnull) — появился в xcode 6.3.

Атрибуты доступности

- `readwrite` — указывает, что свойство доступно и на чтение, и на запись, то есть будут сгенерированы и сеттер, и геттер. Это значение задается всем свойствам по умолчанию, если не задано другое значение;
- `readonly` — указывает, что свойство доступно только для чтения. Это значение стоит применять в случаях, когда изменение свойства «снаружи» во время выполнения объектом своей задачи нежелательно, либо когда значение свойства не хранится ни в какой переменной, а генерируется исходя из значений других свойств.

Атрибуты владения - RETAIN

- retain (соответствующая переменная должна быть с атрибутом `__strong`) — это значение показывает, что в сгенерированном сеттере счетчик ссылок на присваиваемый объект будет увеличен, а у объекта, на который свойство ссылалось до этого, — счетчик ссылок будет уменьшен. Это значение применимо при выключенном ARC для всех Objective-C классов во всех случаях, когда никакие другие значения не подходят. Это значение сохранилось со времен, когда ARC еще не было и, хотя ARC и не запрещает его использование, при включенном автоматическом подсчете ссылок лучше вместо него использовать `strong`.

Атрибуты владения - STRONG

- strong (соответствующая переменная должна быть с атрибутом __strong) — это значение аналогично retain, но применяется только при включенном автоматическом подсчете ссылок. При использовании ARC это значение используется по умолчанию.

Атрибуты владения - COPY

- copy (соответствующая переменная должна быть с атрибутом `__strong`) — при таком значении атрибута владения в сгенерированном сеттере соответствующей переменной экземпляра присваивается значение, возвращаемое сообщением copy, отправленным присваиваемому объекту. Использование этого значения атрибута владения накладывает некоторые ограничения на класс объекта:

- 1) класс должен поддерживать протокол `NSCopying`;
- 2) класс не должен быть изменяемым (mutable). У некоторых классов есть mutable-подкласс, например, `NSString-NSMutableString`. Если ваше свойство — экземпляр «мутабельного» класса, использование copy приведет к нежелательным последствиям, так как метод copy вернет экземпляр его «немутабельного» сородича. Например, вызов copy у экземпляра `NSMutableString` вернет экземпляр `NSString`.

Атрибуты владения - WEAK

- weak (соответствующая переменная должна быть с атрибутом `__weak`) — это значение аналогично `assign` и `unsafe_unretained`. Разница в том, что особая уличная магия позволяет переменным с таким значением атрибута владения менять свое значение на `nil`, когда объект, на который указывала переменная, уничтожается, что очень хорошо сказывается на устойчивости работы приложения.

Атрибуты владения – UNSAFE_UNRETAINED

- `unsafe_unretained` (соответствующая переменная должна быть с атрибутом `__unsafe_unretained`) — свойство с таким типом владения просто сохраняет адрес присвоенного ему объекта. Методы доступа к такому свойству никак не влияют на счетчик ссылок объекта. Он может удалиться, и тогда обращение к такому свойству приведет к крашу (потому и `unsafe`).

Атрибуты владения – ASSIGN

- assign (соответствующая переменная должна быть с атрибутом `__unsafe_unretained`, но так как атрибуты владения есть только у типов попадающих под ARC, с которыми лучше использовать `strong` или `weak`, это значение вам вряд ли понадобится) — просто присвоение адреса. Без ARC является дефолтным значением атрибута владения. Его стоит применять к свойствам типов, не попадающих под действие ARC (к ним относятся примитивные типы).

Атрибуты атомарности - ATOMIC

- `atomic` — это дефолтное значение для данного атрибута. Оно означает, что акцессор и мутатор будут сгенерированы таким образом, что при обращении к ним одновременно из разных потоков, они не будут выполняться одновременно (то есть все равно сперва один поток сделает свое дело — задаст или получит значение, и только после этого другой поток сможет заняться тем же). Из-за особенностей реализации у свойств с таким значением атрибута атомарности нельзя переопределять только один из методов доступа.

Атрибуты атомарности - NONATOMIC

- nonatomic — значение противоположное atomic — у свойств с таким значением атрибута атомарности методы доступа не обременены защитой от одновременного выполнения в разных потоках, поэтому выполняются быстрее. Это значение пригодно для большинства свойств, так как большинство объектов все-таки используются только в одном потоке, и нет смысла «обвешивать» их лишними фичами.

Nullability атрибут

- Этот атрибут никак не влияет на генерируемые методы доступа. Он предназначен для того, чтобы обозначить, может ли данное свойство принимать значение `nil` или `NULL`.

Кроме того, это значение используется для вывода предупреждений в случае, если ваш код делает что-то, противоречащее заявленному поведению. Например, если вы пытаетесь задать значение `nil` свойству, которое объявлено как `nonnull`. А самое главное, эта информация будет полезна тому, кто будет читать ваш код. На использование этого атрибута есть пара ограничений:

- 1) его нельзя использовать для примитивных типов (им и не нужно, так как они в любом случае не принимают значений `nil` и `NULL`);
- 2) его нельзя использовать для многоуровневых указателей (например, `NSError**`).