Weak versus Unowned

Weak reference

https://cocoacasts.com/what-is-the-difference-between-weak-and-unowned-references-in-swift

**Weak and unowned serve the same purpose to avoid strong reference cycles.**

As the name suggests, a weak reference keeps a **weak** reference to the instance it references. This means that the reference to the instance is not taken into account by ARC. Remember that an instance is deallocated if no other objects have a **strong** reference to the instance.

Unowned References

An unowned reference is always expected to have a value. This is not true for weak references, which are set to nil if the instance they reference is deallocated. When that happens, the reference is set to nil.

Because a weak reference can be set to nil, it is always declared as an optional.

The value of a weak reference needs to be unwrapped before it can be accessed whereas you can directly access the value of an unowned reference.

If the referenced instance of an unowned reference is deallocated, it isn't set to nil. As a result, a fatal error is thrown if the referenced object of the unowned reference is accessed.

Examples

The UITableView class defines a delegate property. Among other things, the delegate is responsible for responding to user interaction. For example, the table view notifies its delegate when the user taps one of its rows.

The table view doesn't keep a strong reference to its delegate.

If the table view would keep a strong reference to its delegate, the delegate would only be deallocated when the table view is deallocated and that is not what you want

Account and plan example

With the code below the vars plan and account both have strong references cycle. Tis can be resolved by making the account property as **unowned var plan: Plan**

class Account {

Image // MARK: - Properties

var plan: Plan

}

class Plan {

// MARK: - Properties

var account: Account

// MARK: - Initialization

init(account: Account) {

self.account = account

}

}