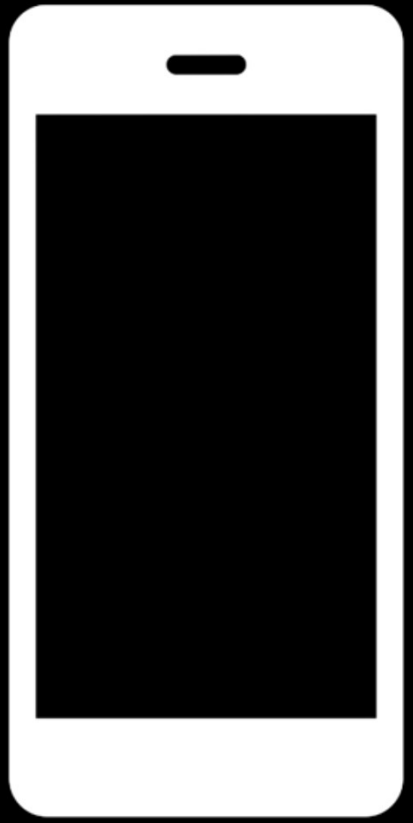
MEMORY

Memory Basics

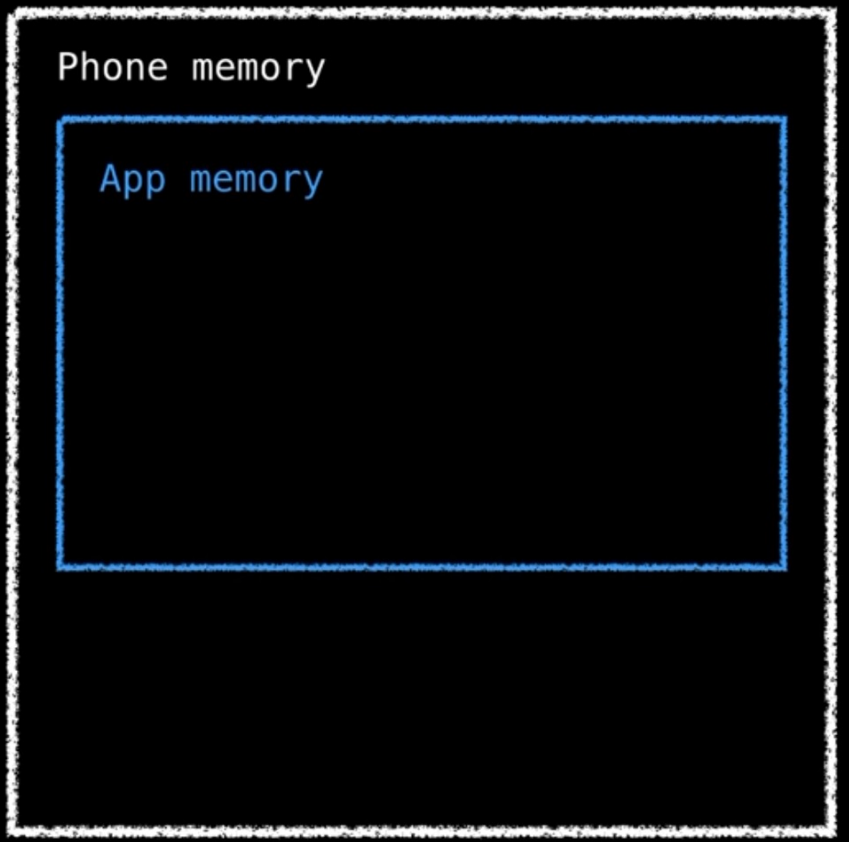


Phone memory

Memory Basics

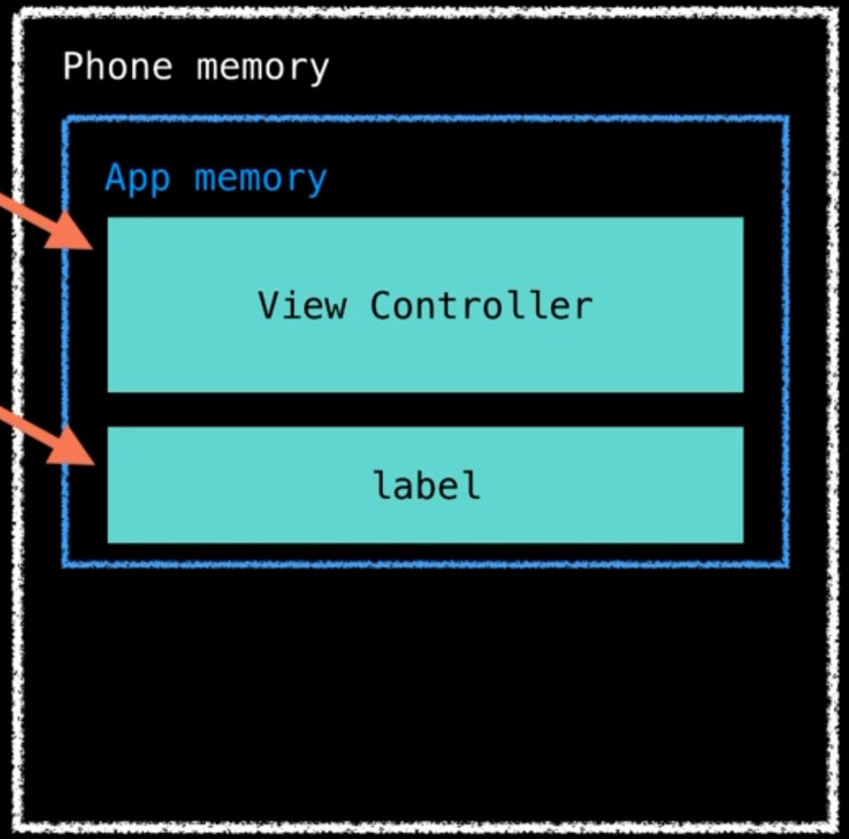
Memory Basics

```
class ViewController: UIViewController {

    @IBOutlet var label: UILabel!
}
```

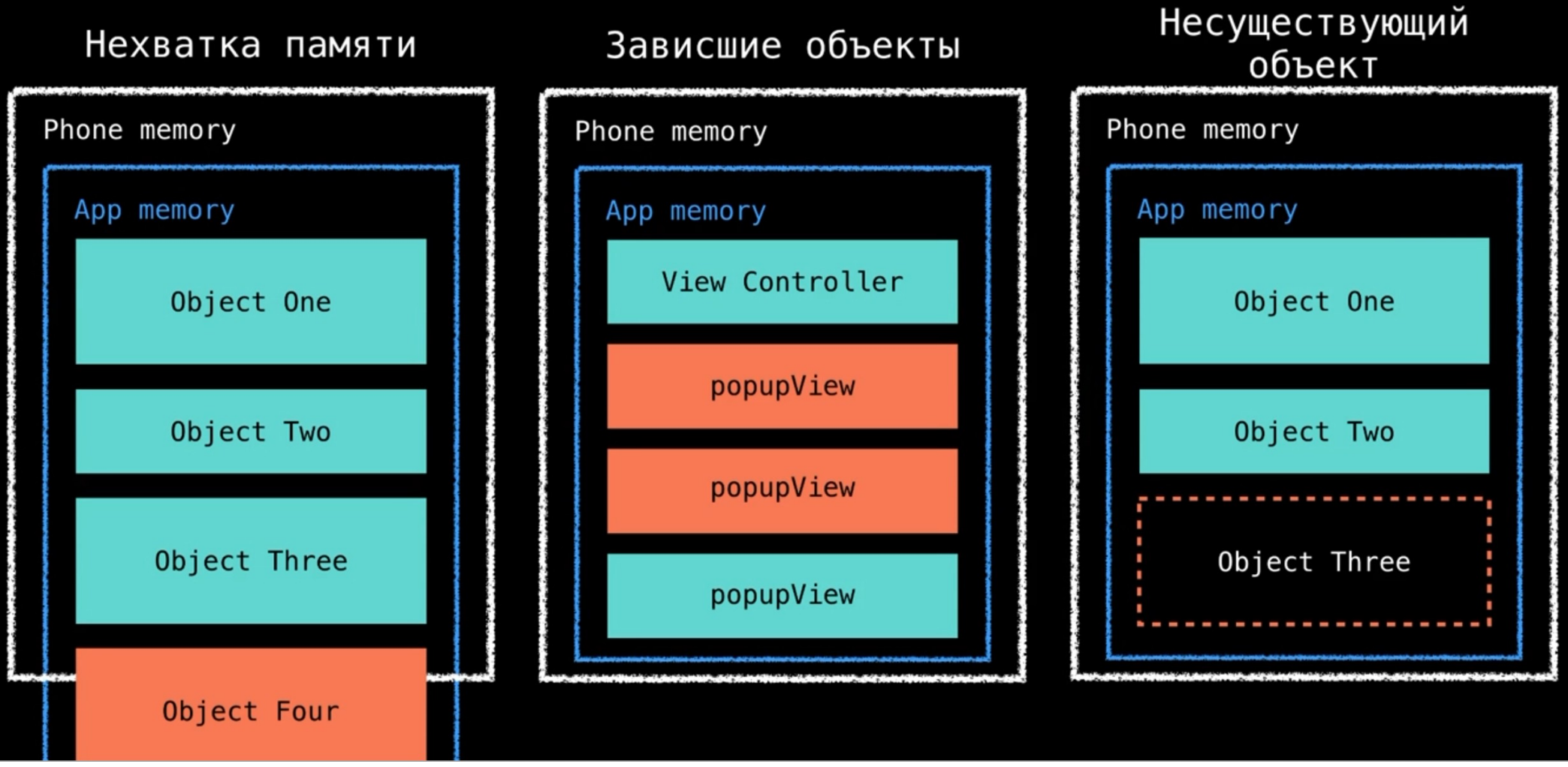


Memory Basics

```
class ViewController: UIViewController {

    @IBOutlet var label: UILabel!
}
```



Memory Problems



Типы значений

Копии экземпляров типов значений являются самостоятельными объектами, каждый из которых занимает отдельное место в памяти устройства

Примеры: Базовые типы данных (String, Int, Bool), структуры, перечисления, массивы, словари и т.д.

Ссылочные типы

0x0

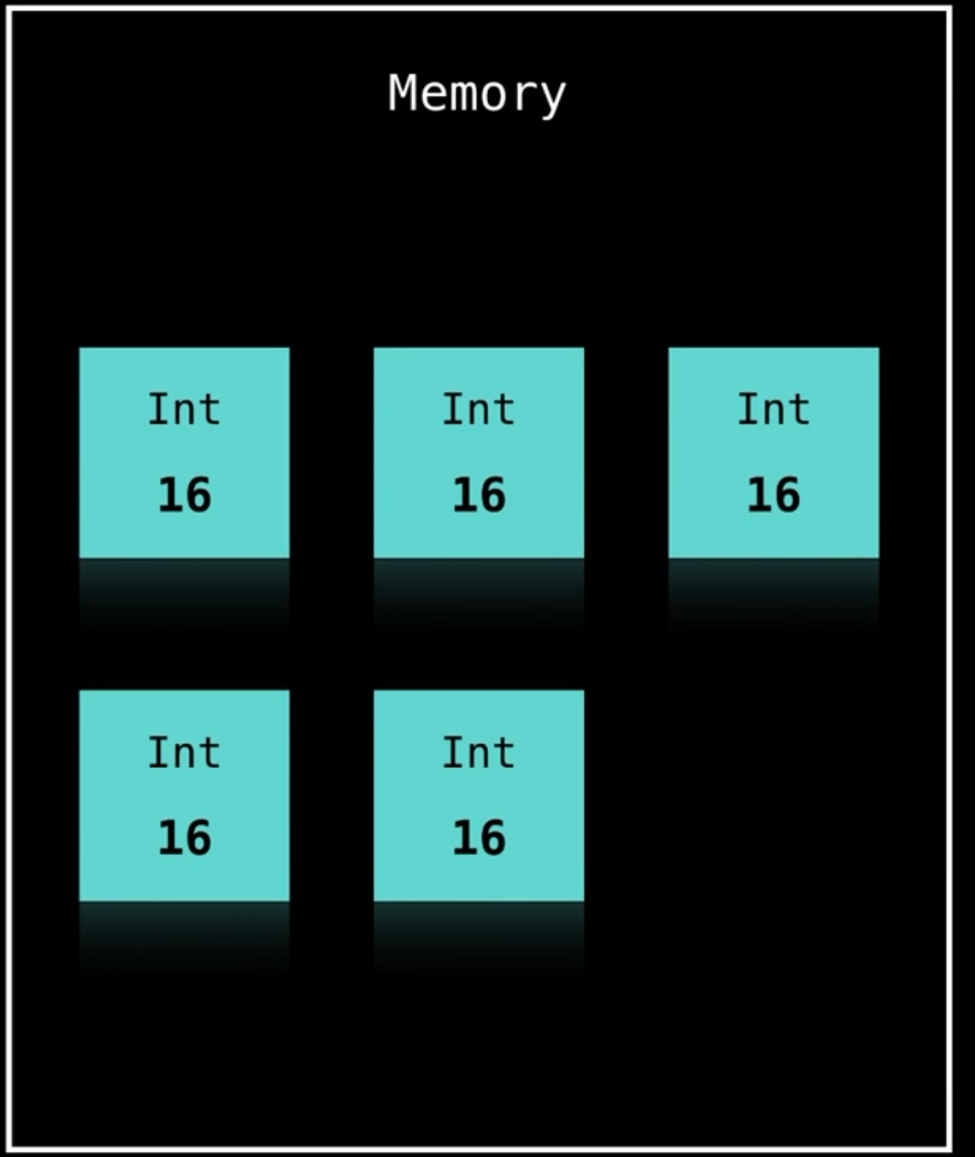
Копии экземпляров ссылочных типов ссылаются но один объект в базе и не являются уникальными

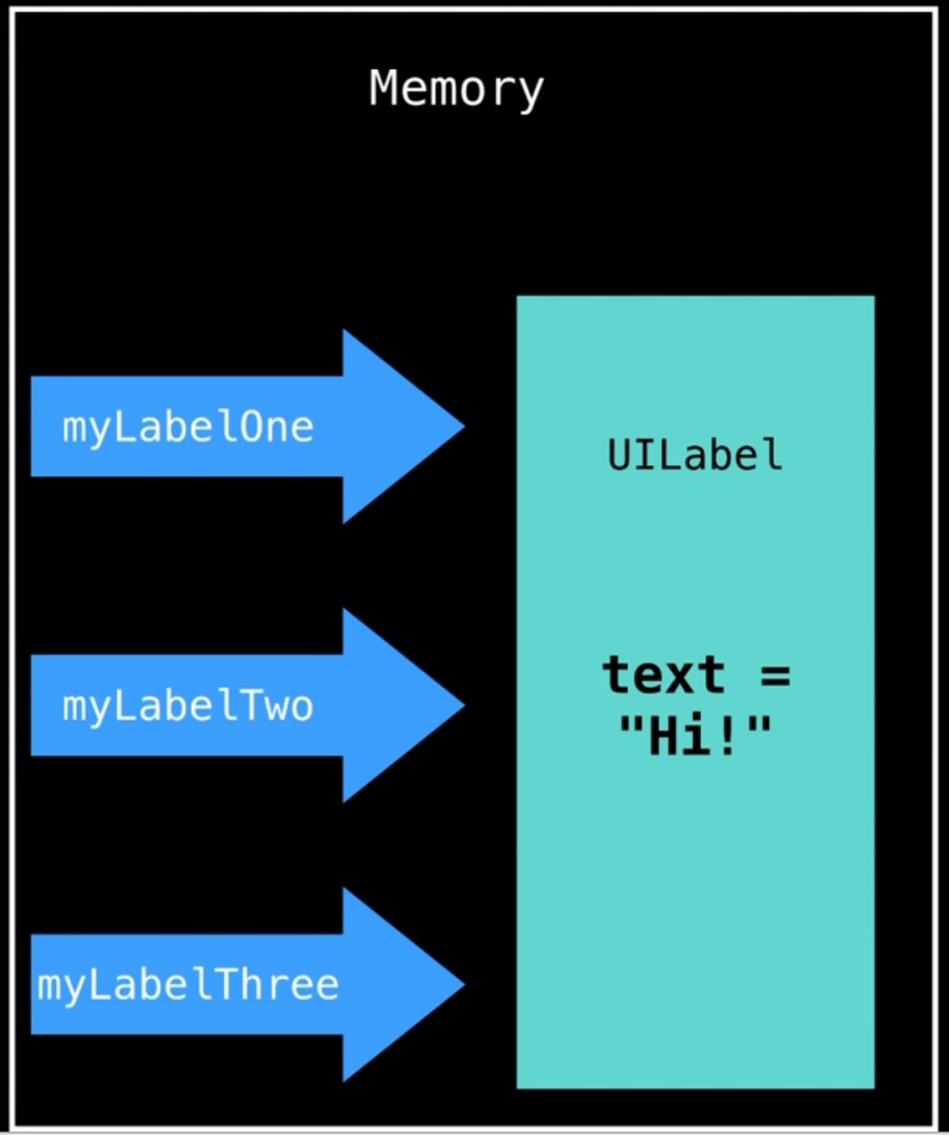
Примеры: все классы являются ссылочными типами данных

Типы значений

Копии экземпляров типов значений являются самостоятельными объектами, каждый из которых занимает отдельное место в памяти устройства

Примеры: Базовые типы данных (String, Int, Bool), структуры, перечисления, массивы, словари и т.д.





Ссылочные типы

0x0

Копии экземпляров ссылочных типов ссылаются но один объект в базе и не являются уникальными

Примеры: все классы являются ссылочными типами данных

Типы значений

Копии экземпляров типов значений является самостоятельными объектами, каждый из которых занимает отдельное место в памяти устройства

Примеры: Базовые типы данных (String, Int, Bool), структуры, перечисления, массивы, словари и т.д.

Memory


```
var firstNumber = 16

var secondNumber = firstNumber

secondNumber = 8

printNumber(number: secondNumber)

func printNumber(number: Int) {
    print(number)
}
```

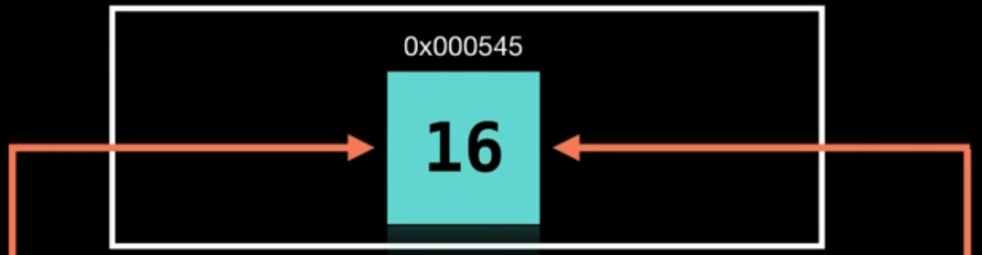
Ссылочные типы

0x0

Копии экземпляров ссылочных типов ссылаются но один объект в базе и не являются уникальными

Примеры: все классы являются ссылочными типами данных

Memory




```
class MyClass {
    var someNumber = 16
}

let first = MyClass()

let second = first

second.someNumber = 8
```

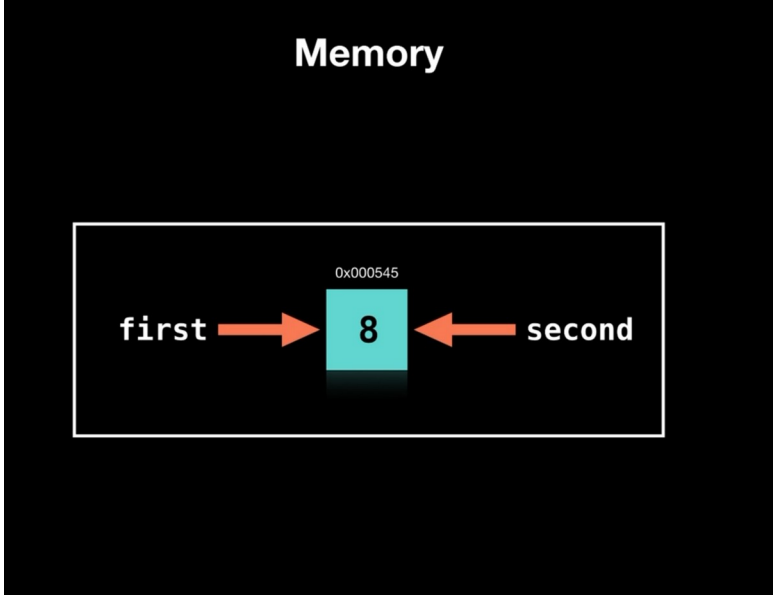
FINDING AND FIXING MEMORY LEAKS

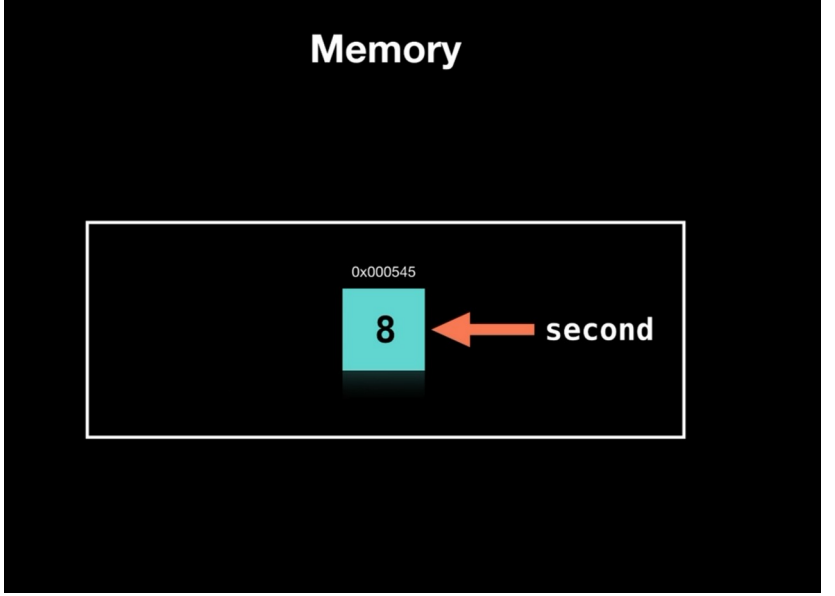
Вопрос:

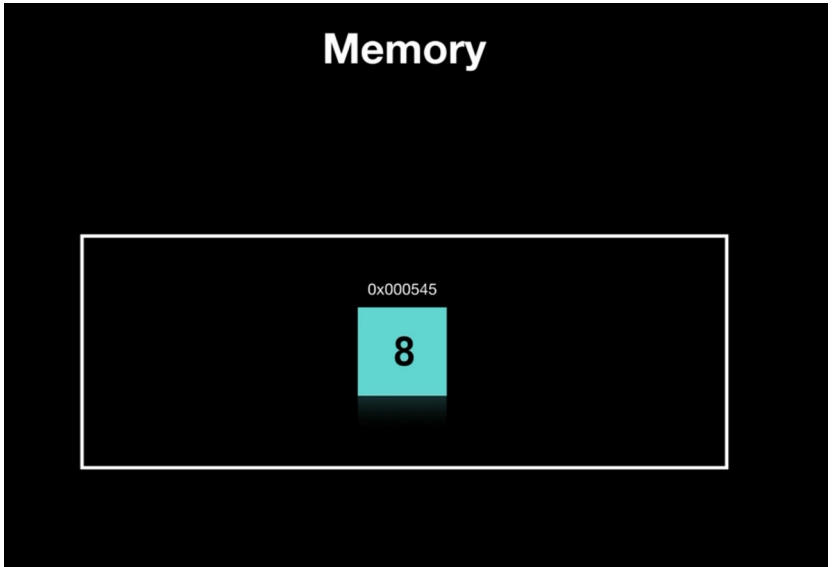
В какой момент Swift удаляет ссылочный тип из памяти?

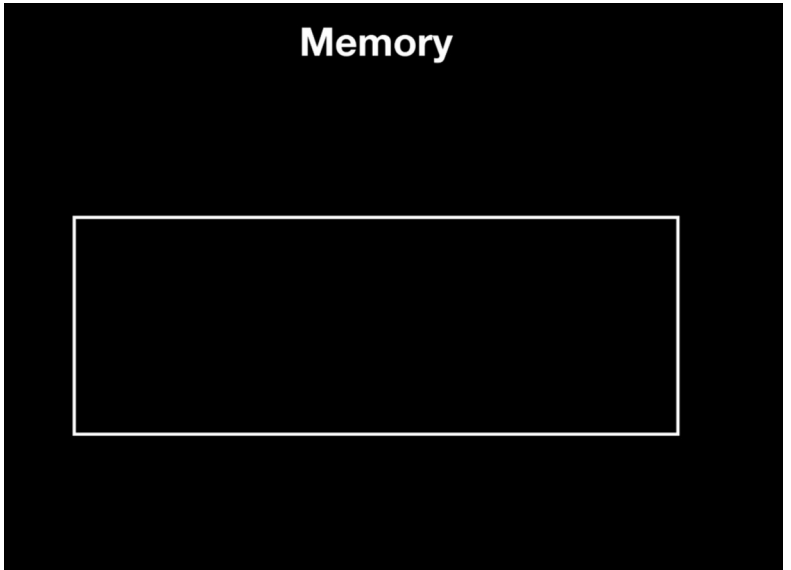
Ответ:

Когда больше нет ни одного свойства, которое ссылается на объект.

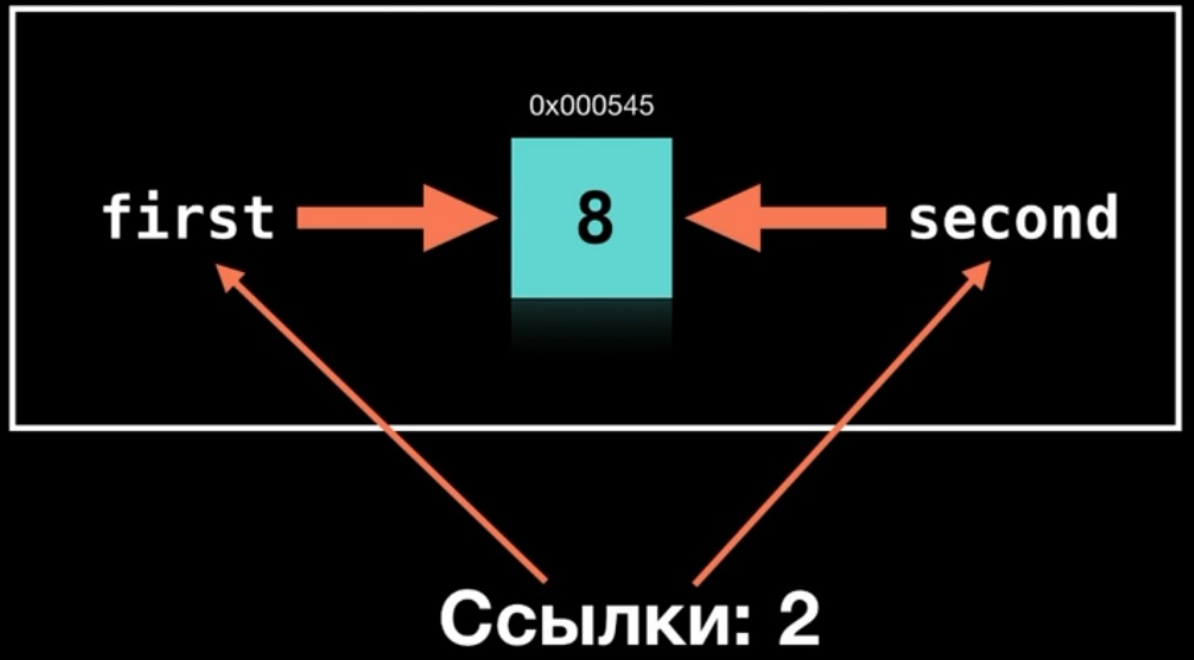

Memory
0x000545
first
8
second


Memory
0x000545
8
second


Memory
0x000545
8


Memory

Automatic Reference Counting (ARC)



Automatic Reference Counting (ARC)

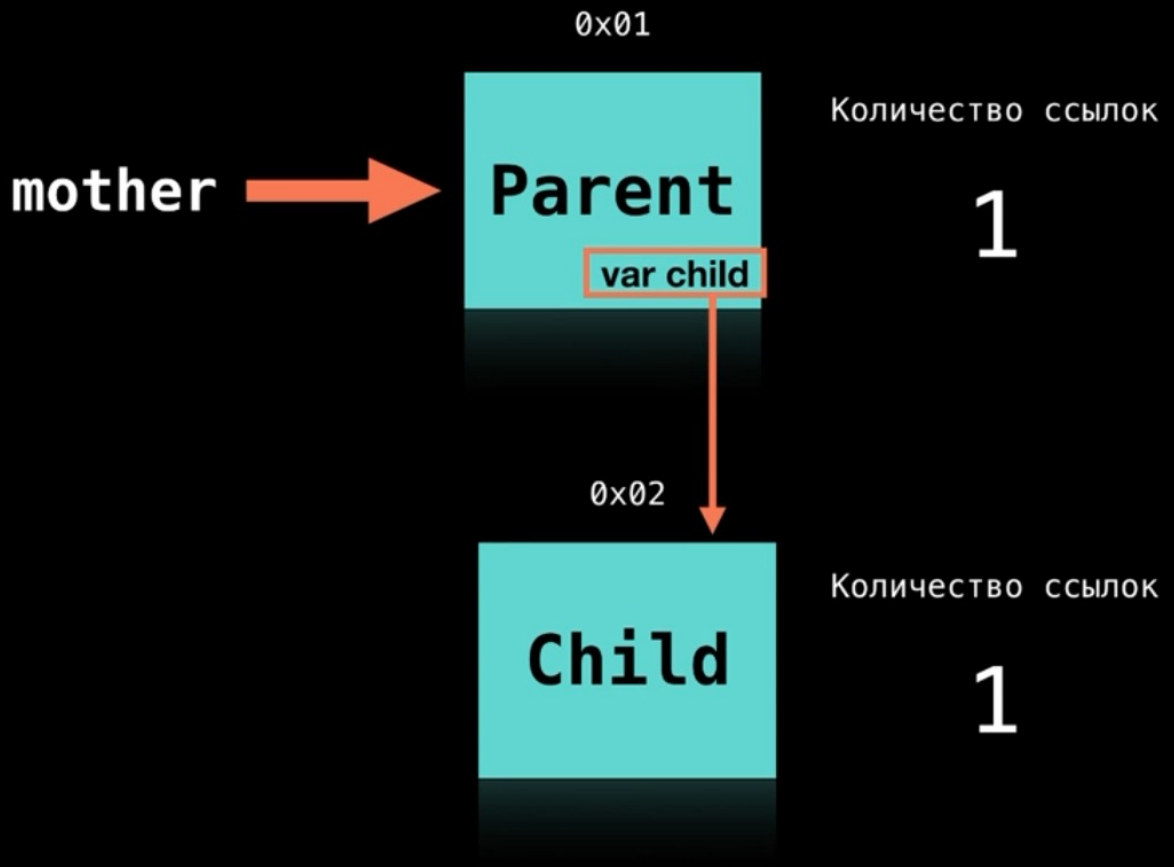
```
class MyClass {
    var someNumber = 16
}

var first: MyClass? = MyClass()
var second = first
var third = first
```

Automatic Reference Counting (ARC)

```
class MyClass {
    var someNumber = 16
}


var first: MyClass? = MyClass()
var second = first
var third = first

first = nil
```

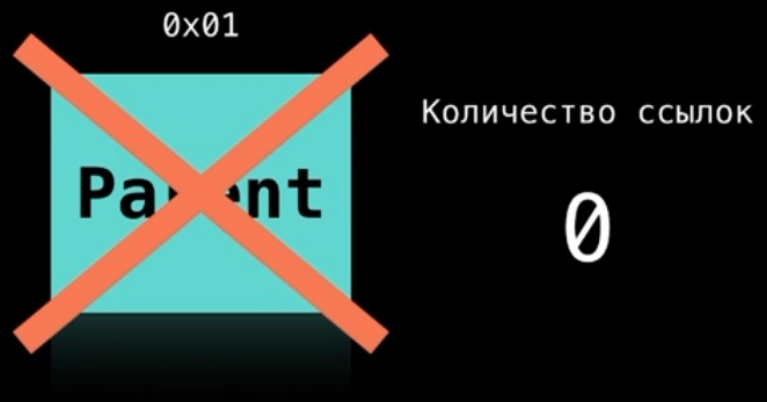
Удаление объектов из памяти

```
class Parent {
    var child = Child()
}

class Child {

}

var mother: Parent? = Parent()

mother = nil
```



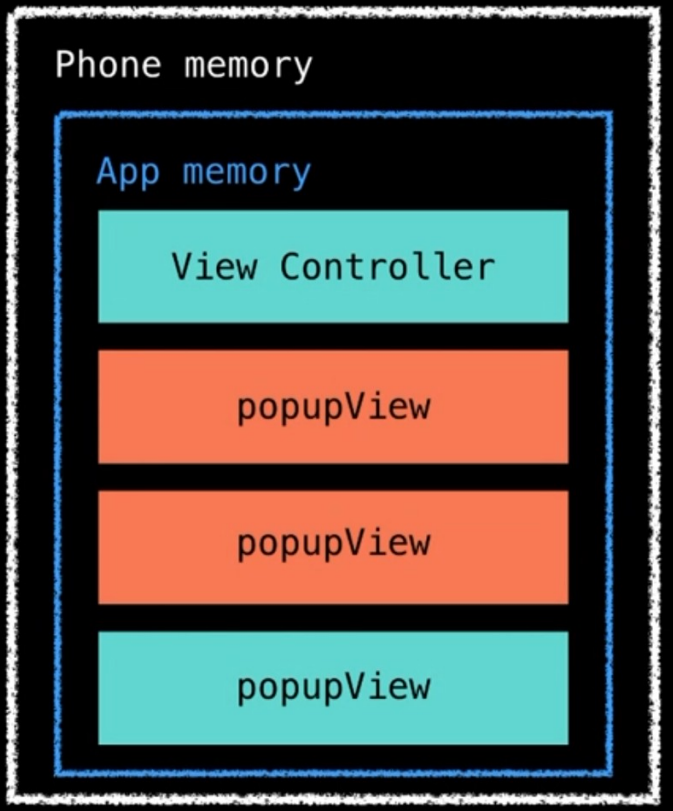
Удаление объектов из памяти

```
class Parent {
    var child = Child()
}

class Child {

}

var mother: Parent? = Parent()

mother = nil
```



Memory Problems

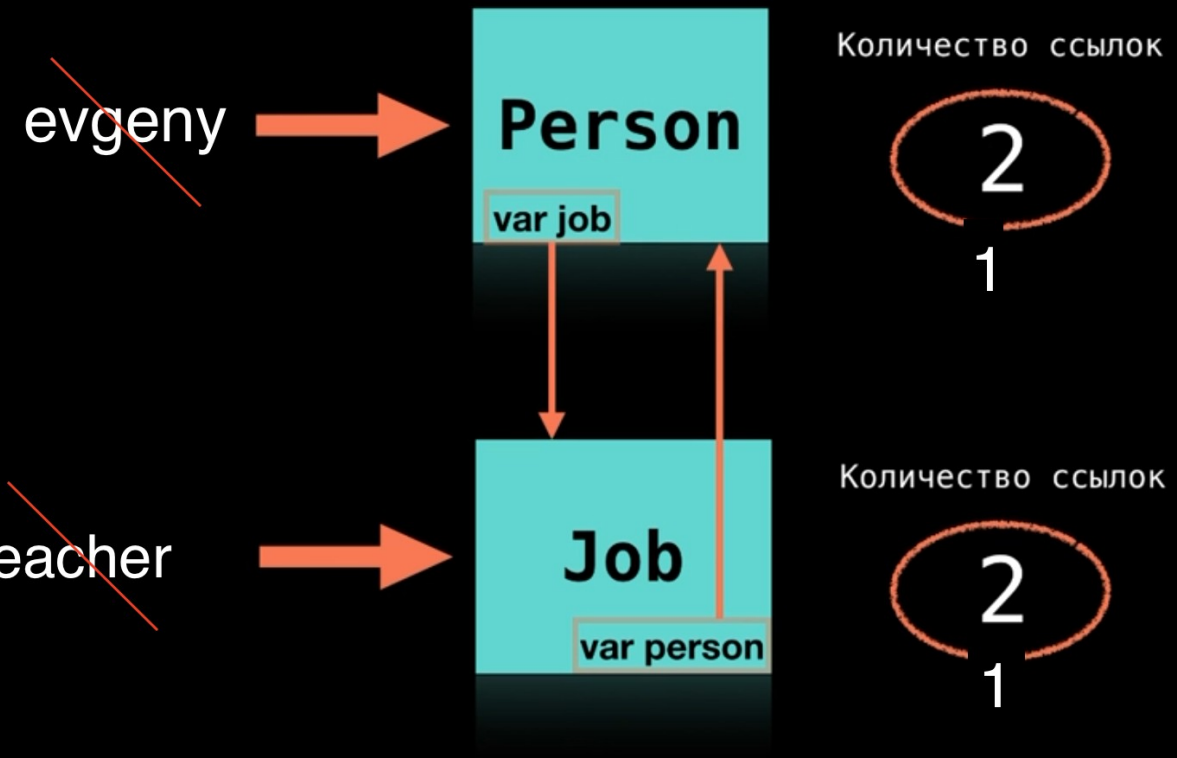
Зависшие объекты



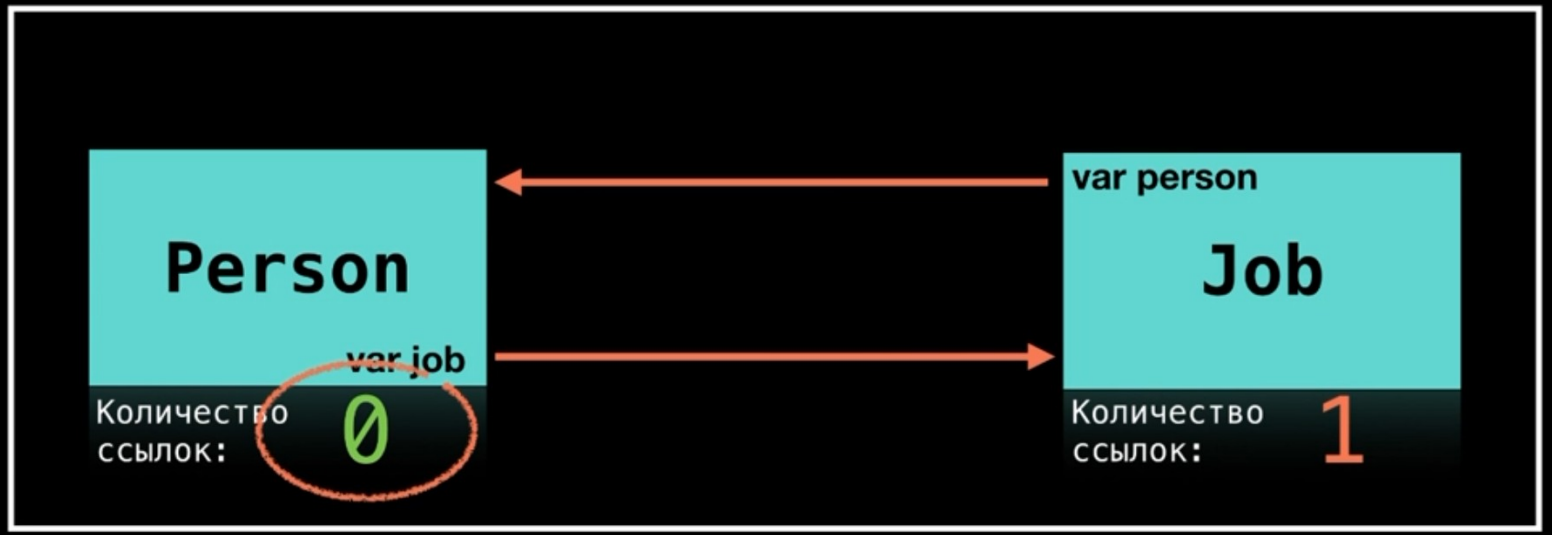
Проблемы с удалением объектов из памяти

```
class Job {
    var person: Person?
}

class Person {
    var job: Job?
}

var evgeny: Person? = Person()

var   teacher   : Job? = Job()

evgeny?.job =    teacher

  teacher   ?.person = evgeny
```

evgeny = nil teacher= nil



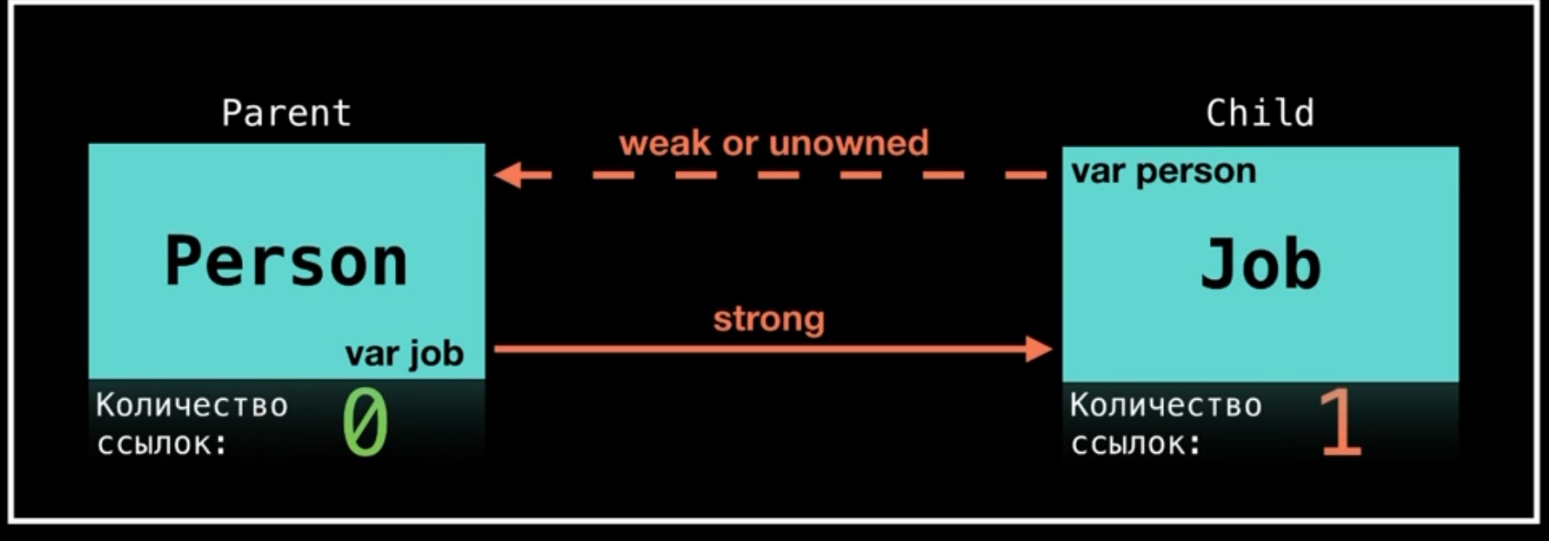
Цикличные ссылки



Как избежать проблемы цикличных ссылок

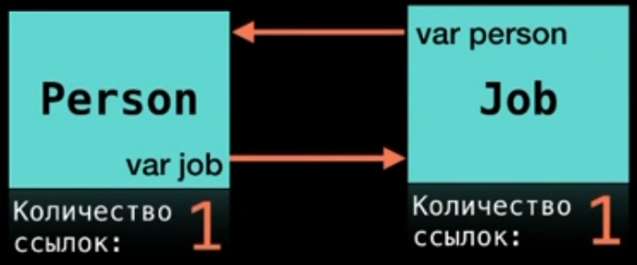
1. Определить иерархическую зависимость между объектами (Владелец — Зависимый)

2. Сделать ссылку от дочернего объекта на родительский с типом weak или unowned



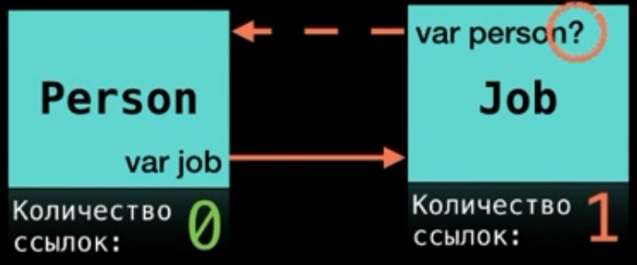
Типы ссылок

Strong (сильная)



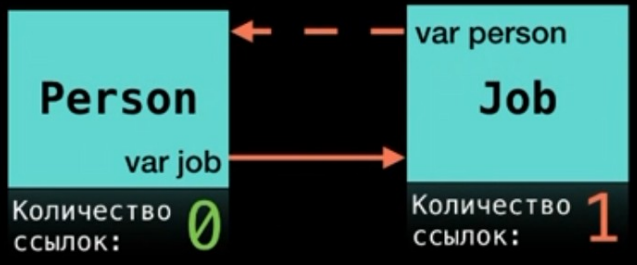
Используется по умолчанию. Зависимый класс существует до тех пока, пока существует класс владелец

Weak (слабая)



**Зависимый объект существует до тех пор, пока существует объект владелец.
Не всегда имеет ссылку на экземпляр, т.к. используются только с опциональными свойствами**

Unowned (бесхозная)



Зависимый объект существует до тех пор, пока существует объект владелец. Всегда имеет ссылку на экземпляр, т.к. не используется с опциональными свойствами.