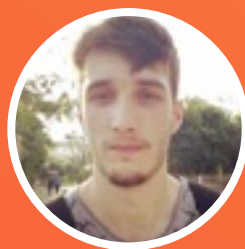




Swift: Gerenciamento de Memória



Txai Wieser



Repel



Recolorindo o Reino



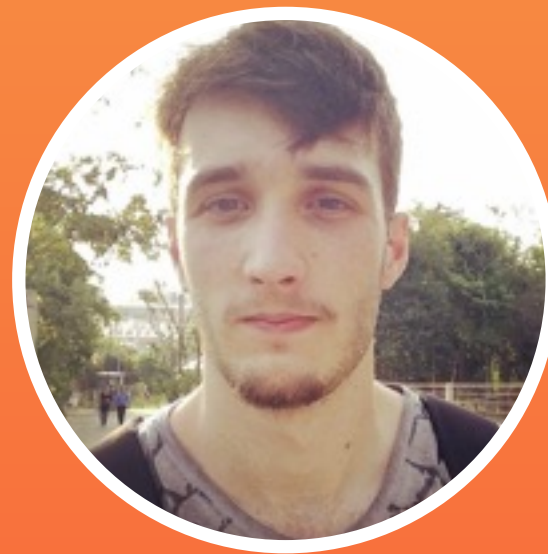
Warren



FityIt



Ballphy



Txai Wieser

- Estudante de engenharia de computação na UFRGS.
- Desenvolvedor iOS



Swift: Gerenciamento de Memória

Gerenciador de Memória

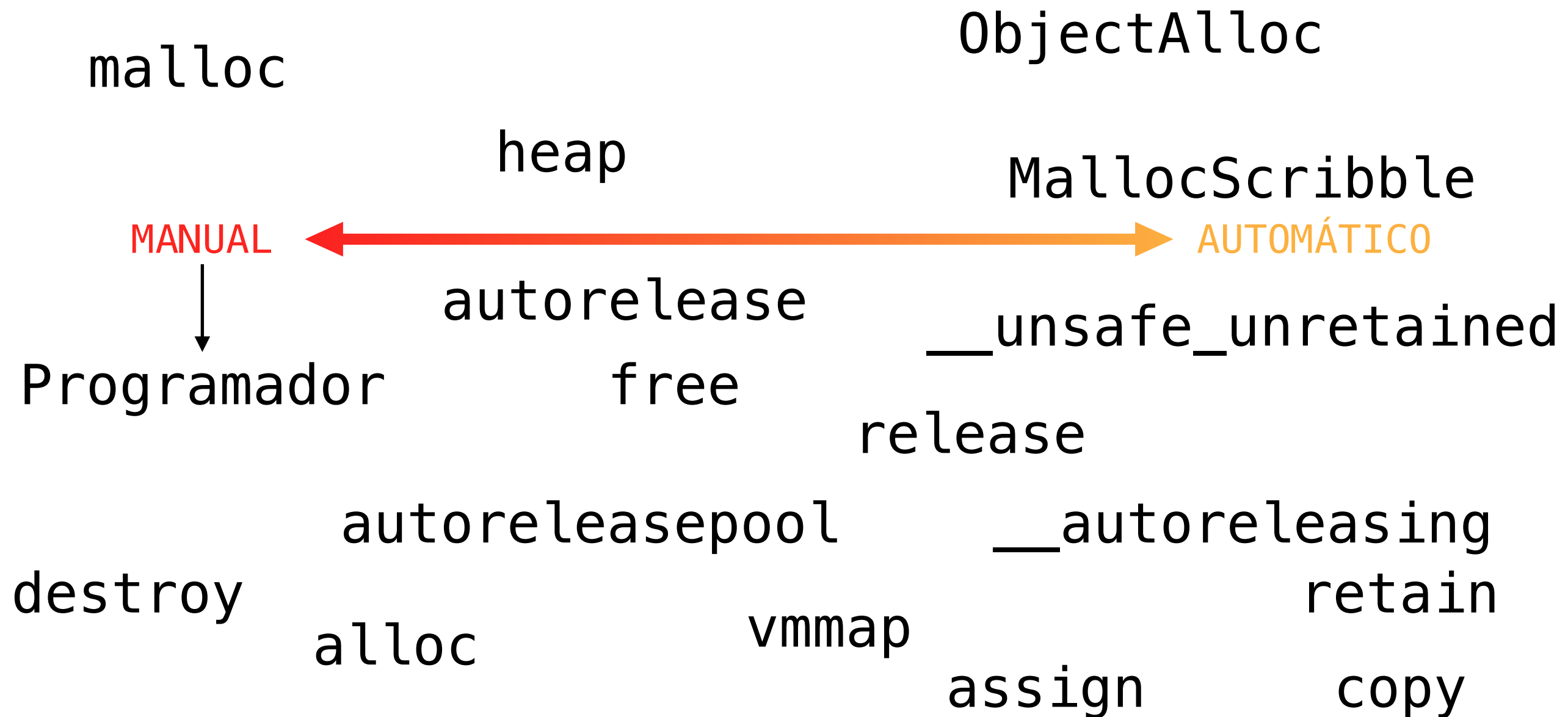
MANUAL ← → AUTOMÁTICO



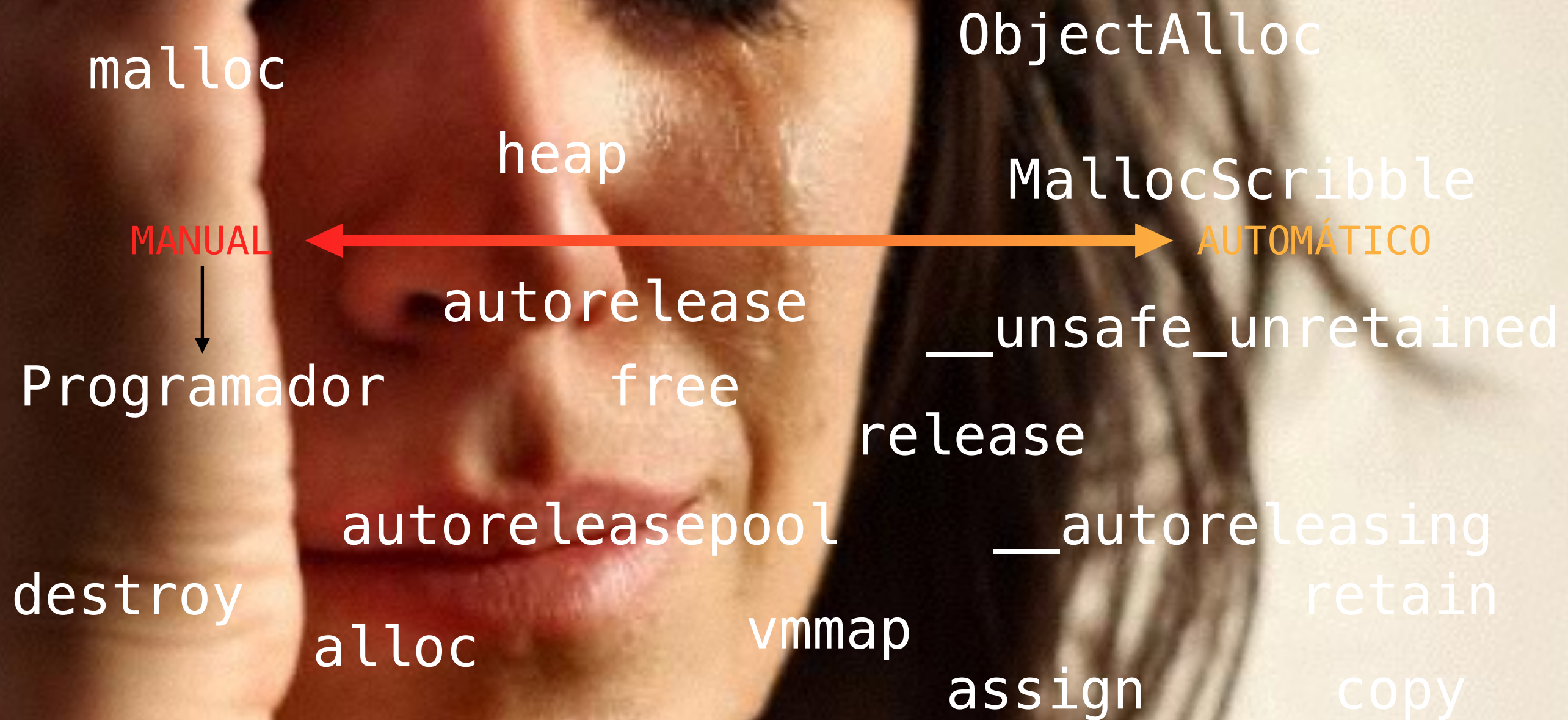
Gerenciador de Memória



Gerenciador de Memória



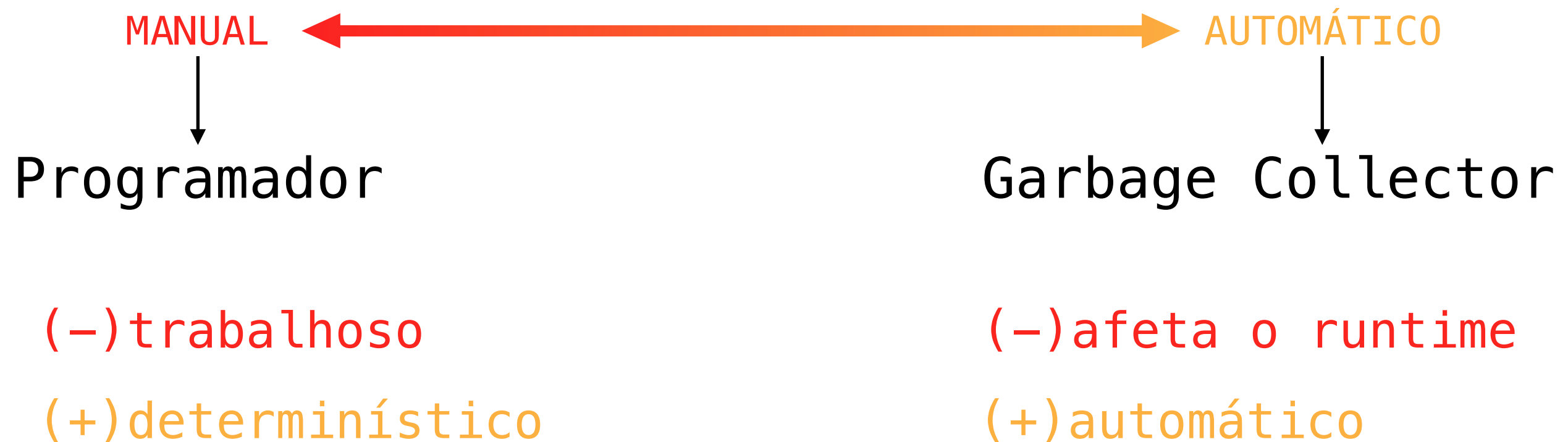
Gerenciador de Memória



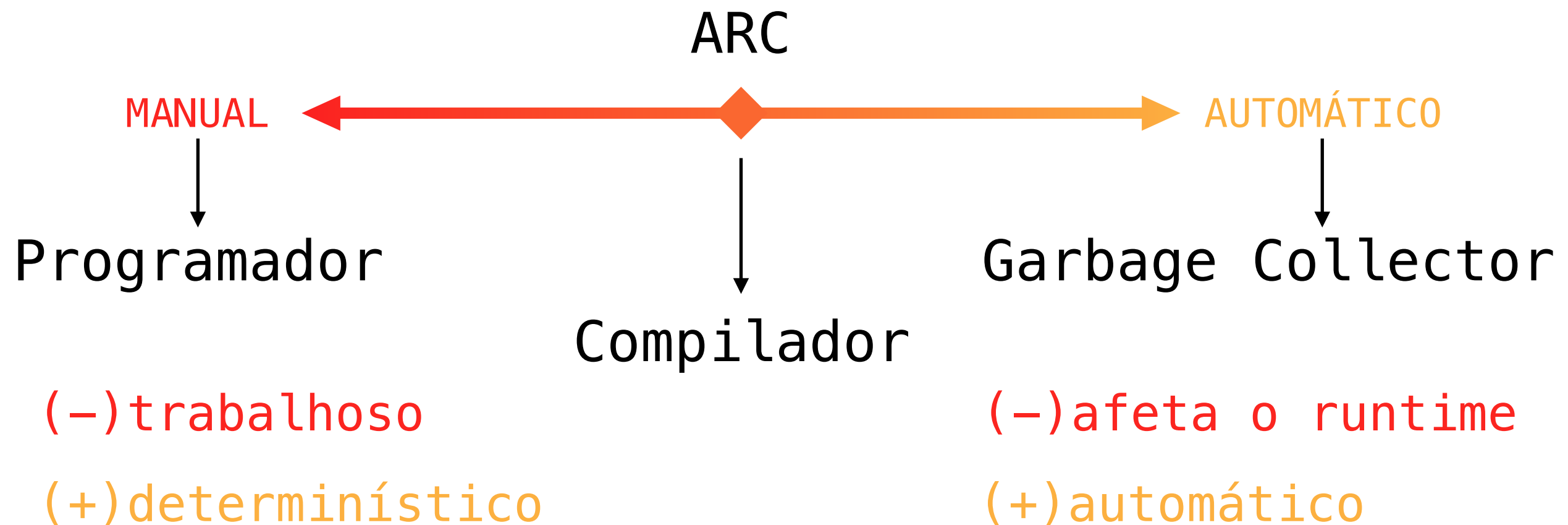
Gerenciador de Memória



Gerenciador de Memória



Gerenciador de Memória



ARC

Reference counting is a technique of storing the number of references, pointers, or handles to a resource such as an object, block of memory, disk space or other resource. It may also refer, more specifically, to a garbage collection algorithm that uses these reference counts to deallocate objects which are no longer referenced.

Garbage Collector

Tracing garbage collection is a form of automatic memory management that consists of determining which objects should be deallocated by tracing which objects are reachable by a chain of references from certain “root” objects, and considering the rest as “garbage” and collecting them.

Automatic Reference Counting

strong

```
class Pessoa { }
```

```
var ref1: Pessoa?
```

ref1



nil

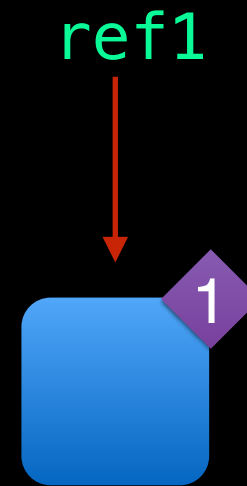
Automatic Reference Counting

strong

```
class Pessoa { }
```

```
var ref1: Pessoa?
```

```
ref1 = Pessoa()
```



Automatic Reference Counting

strong

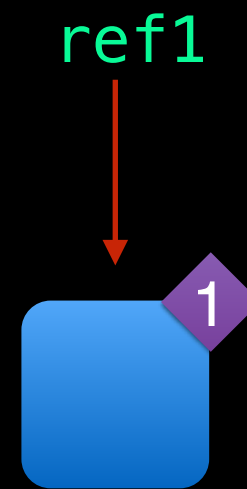
```
class Pessoa { }
```

```
var ref1: Pessoa?
```

```
ref1 = Pessoa()
```

```
var ref2: Pessoa?
```

ref2



Automatic Reference Counting

strong

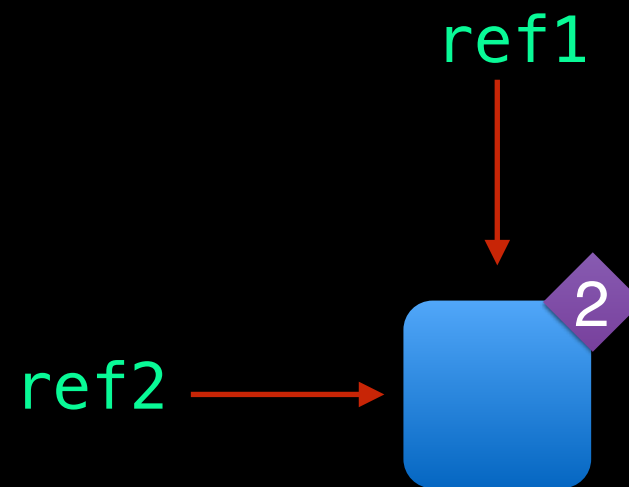
```
class Pessoa { }
```

```
var ref1: Pessoa?
```

```
ref1 = Pessoa()
```

```
var ref2: Pessoa?
```

```
ref2 = ref1
```



Automatic Reference Counting

strong

```
class Pessoa { }
```

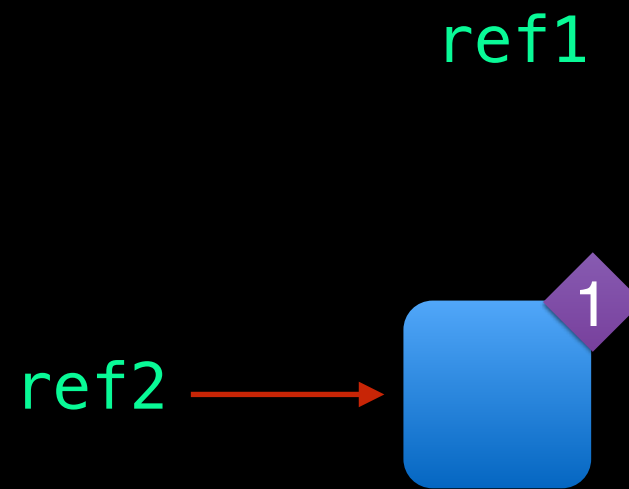
```
var ref1: Pessoa?
```

```
ref1 = Pessoa()
```

```
var ref2: Pessoa?
```

```
ref2 = ref1
```

```
ref1 = nil
```



Automatic Reference Counting

strong

```
class Pessoa { }
```

```
var ref1: Pessoa?
```

```
ref1 = Pessoa()
```

```
var ref2: Pessoa?
```

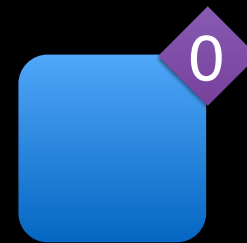
```
ref2 = ref1
```

```
ref1 = nil
```

```
ref2 = nil
```

ref1

ref2



Automatic Reference Counting

strong

```
class Pessoa { }
```

ref1

```
var ref1: Pessoa?
```

```
ref1 = Pessoa()
```

ref2

```
var ref2: Pessoa?
```

```
ref2 = ref1
```

```
ref1 = nil
```

```
ref2 = nil
```



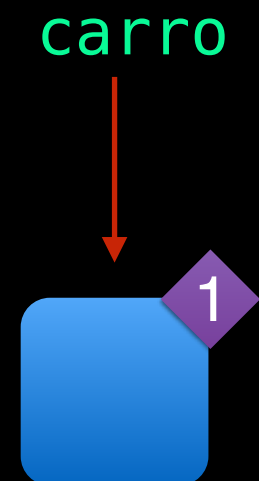
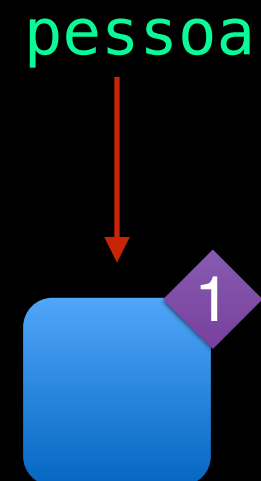
Ciclos de Referências

ARC

(Vida Independente)

weak

```
class Pessoa {  
    var veículo: Carro?  
}  
  
class Carro {  
    var proprietário:Pessoa?  
}  
  
var pessoa: Pessoa? = Pessoa()  
var carro: Carro? = Carro()
```

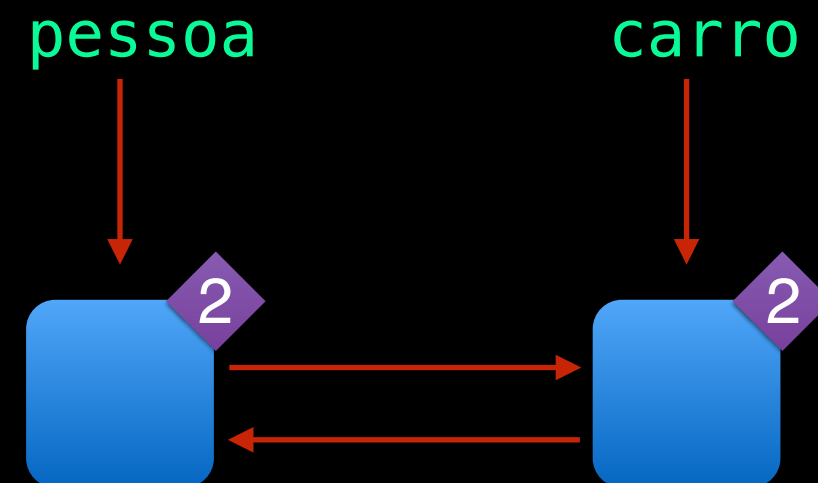


ARC

(Vida Independente)

weak

```
class Pessoa {  
    var veículo: Carro?  
}  
  
class Carro {  
    var proprietário:Pessoa?  
}  
  
var pessoa: Pessoa? = Pessoa()  
var carro: Carro? = Carro()  
  
pessoa!.veículo = carro  
carro!.proprietário = pessoa
```

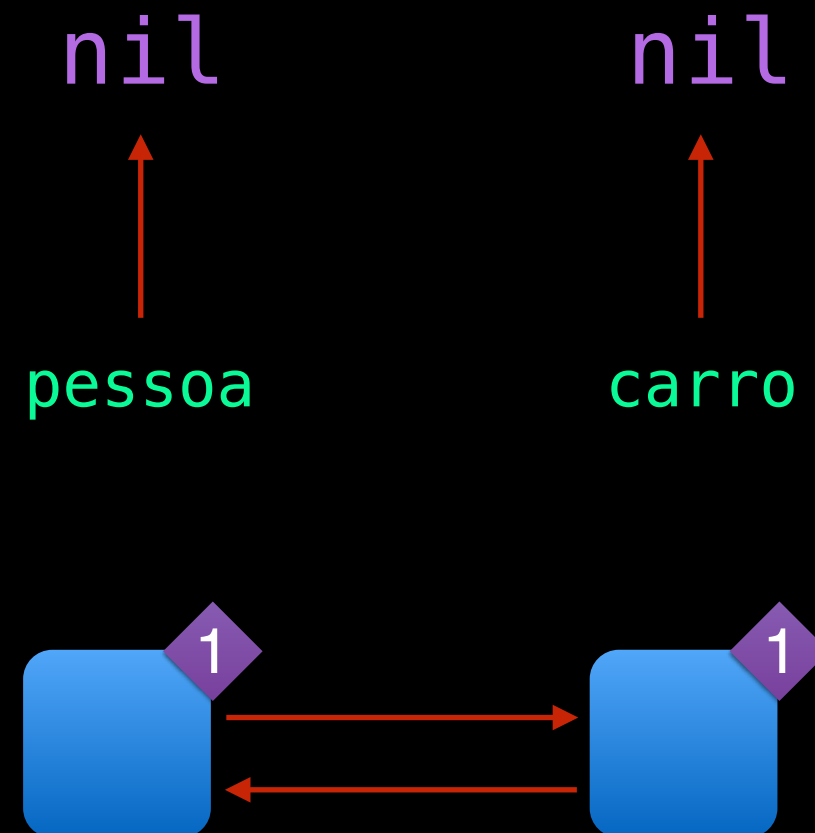


ARC

(Vida Independente)

weak

```
class Pessoa {  
    var veículo: Carro?  
}  
  
class Carro {  
    var proprietário:Pessoa?  
}  
  
var pessoa: Pessoa? = Pessoa()  
var carro: Carro? = Carro()  
  
pessoa!.veículo = carro  
carro!.proprietário = pessoa  
  
carro = nil  
pessoa = nil
```



ARC

(Vida Independente)

weak

```
class Pessoa {  
    var veículo: Carro?  
}  
  
class Carro {  
    var proprietário:Pessoa?  
}  
  
var pessoa: Pessoa? = Pessoa()  
var carro: Carro? = Carro()  
  
pessoa!.veículo = carro  
carro!.proprietário = pessoa  
  
carro = nil  
pessoa = nil
```

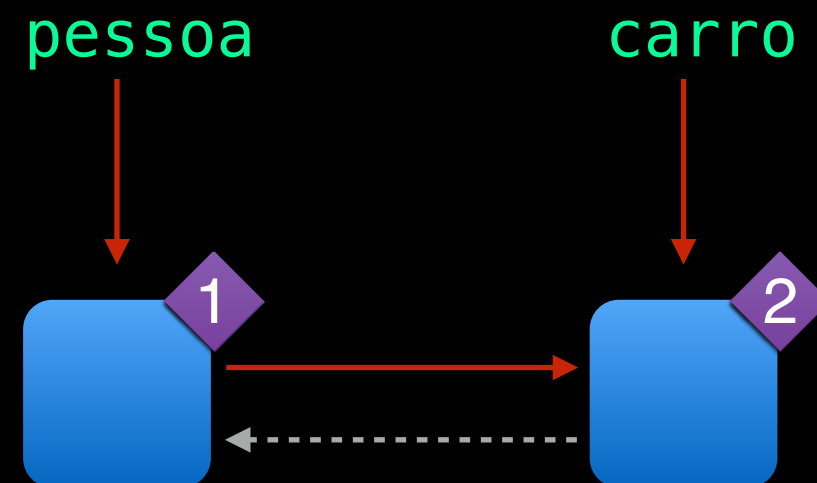


ARC

(Vida Independente)

weak

```
class Pessoa {  
    var veículo: Carro?  
}  
  
class Carro {  
    weak var proprietário: Pessoa?  
}  
  
var pessoa: Pessoa? = Pessoa()  
var carro: Carro? = Carro()  
  
pessoa!.veículo = carro  
carro!.proprietário = pessoa  
  
carro = nil  
pessoa = nil
```



ARC

(Vida Independente)

weak

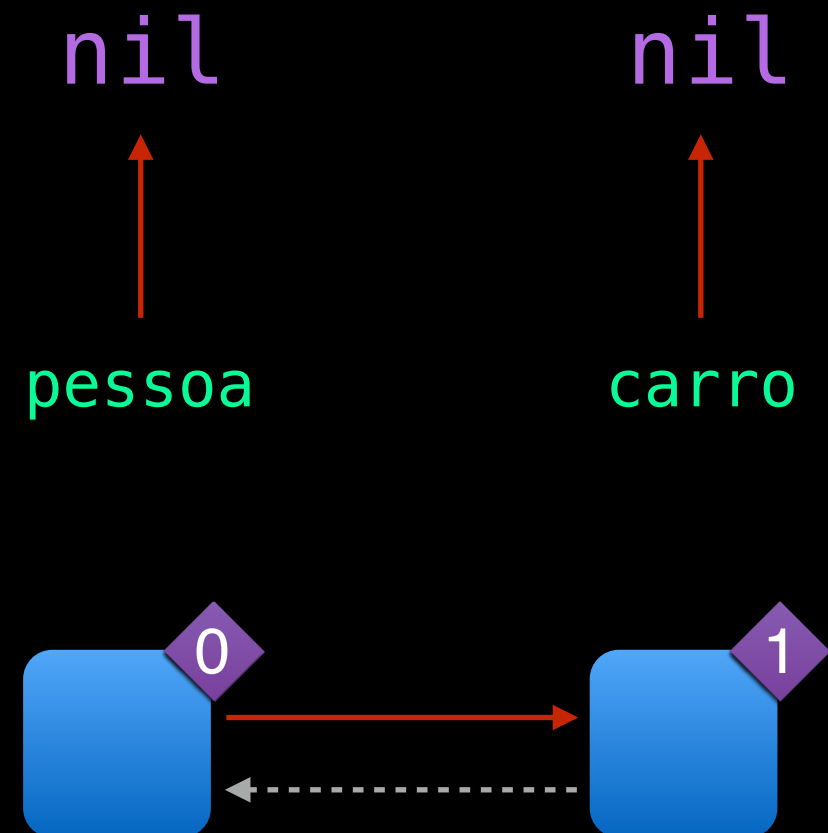
```
class Pessoa {  
    var veículo: Carro?  
}
```

```
class Carro {  
    weak var proprietário: Pessoa?  
}
```

```
var pessoa: Pessoa? = Pessoa()  
var carro: Carro? = Carro()
```

```
pessoa!.veículo = carro  
carro!.proprietário = pessoa
```

```
carro = nil  
pessoa = nil
```



ARC

(Vida Independente)

weak

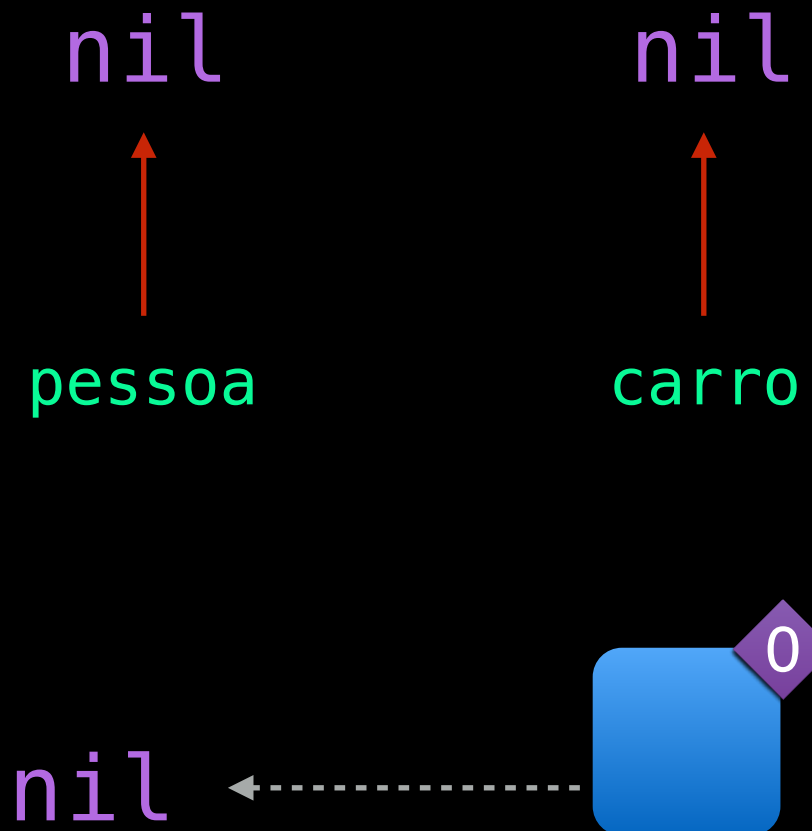
```
class Pessoa {  
    var veículo: Carro?  
}
```

```
class Carro {  
    weak var proprietário: Pessoa?  
}
```

```
var pessoa: Pessoa? = Pessoa()  
var carro: Carro? = Carro()
```

```
pessoa!.veículo = carro  
carro!.proprietário = pessoa
```

```
carro = nil  
pessoa = nil
```



ARC

(Vida Independente)

weak

```
class Pessoa {  
    var veículo: Carro?  
}
```

```
class Carro {  
    weak var proprietário: Pessoa?  
}
```

```
var pessoa: Pessoa? = Pessoa()  
var carro: Carro? = Carro()
```

```
pessoa!.veículo = carro  
carro!.proprietário = pessoa
```

```
carro = nil  
pessoa = nil
```

nil

↑
pessoa

nil

↑
carro



weak

(Vidas Independentes)

ARC

(Vida Dependente)

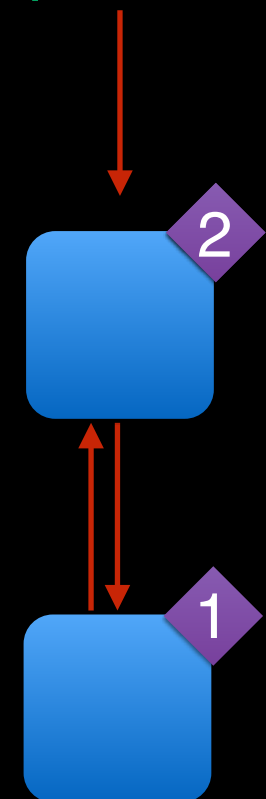
unowned

```
class Pessoa {  
    var habilitação: Habilidade?  
}
```

```
class Habilidade {  
    let titular: Pessoa  
  
    init(titular: Pessoa) {  
        self.titular = titular  
    }  
}
```

```
var pessoa: Pessoa? = Pessoa()  
pessoa!.habilitação = Habilidade(titular: pessoa!)
```

pessoa



ARC

(Vida Dependente)

unowned

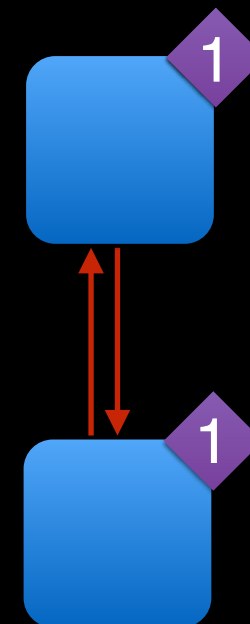
```
class Pessoa {  
    var habilitação: Habilidade?  
}
```

```
class Habilidade {  
    let titular: Pessoa  
  
    init(titular: Pessoa) {  
        self.titular = titular  
    }  
}
```

```
var pessoa: Pessoa? = Pessoa()  
pessoa!.habilitação = Habilidade(titular: pessoa!)
```

```
pessoa = nil
```

nil ← pessoa



ARC

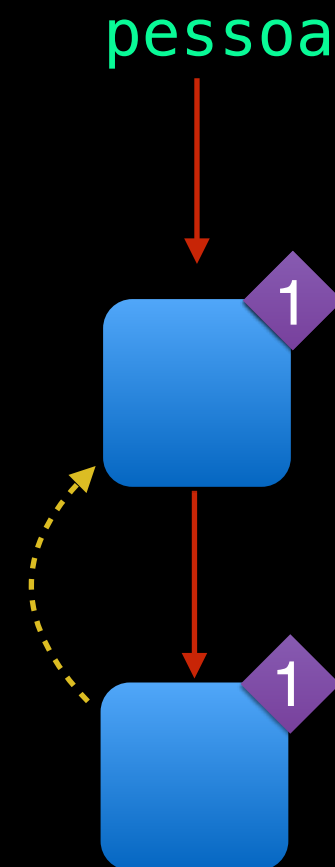
(Vida Dependente)

unowned

```
class Pessoa {  
    var habilitação: Habilidade?  
}
```

```
class Habilidade {  
    unowned let titular: Pessoa  
  
    init(titular: Pessoa) {  
        self.titular = titular  
    }  
}
```

```
var pessoa: Pessoa? = Pessoa()  
pessoa!.habilitação = Habilidade(titular: pessoa!)
```



ARC

(Vida Dependente)

unowned

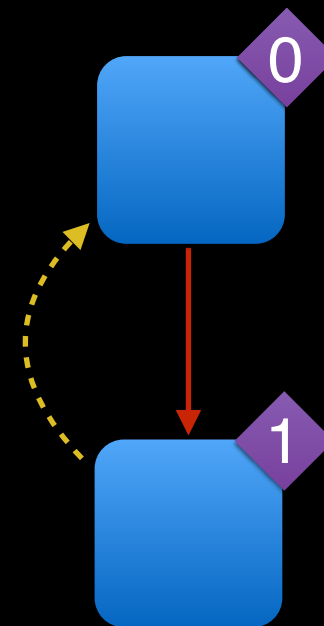
```
class Pessoa {  
    var habilitação: Habilidade?  
}
```

```
class Habilidade {  
    unowned let titular: Pessoa  
  
    init(titular: Pessoa) {  
        self.titular = titular  
    }  
}
```

```
var pessoa: Pessoa? = Pessoa()  
pessoa!.habilitação = Habilidade(titular: pessoa!)
```

```
pessoa = nil
```

nil ← pessoa



unowned

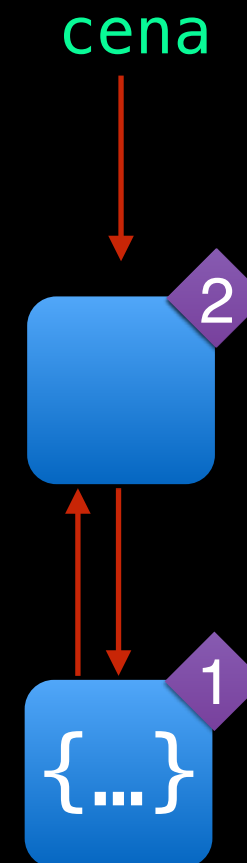
(Vida Dependente)

Closures

```
class Botão {  
    var action: (()->())?  
}
```

```
class Cena {  
    let botão = Botão()  
    var contagem = 0  
  
    init() {  
        botão.action = {  
            contagem += 1  
        }  
    }  
}
```

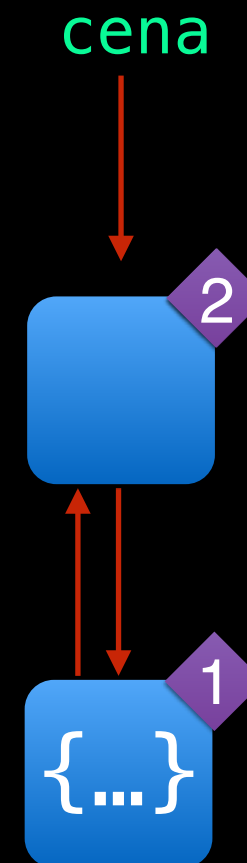
```
var cena:Cena? = Cena()
```



Closures

```
class Botão {  
    var action: (()->())?  
}  
  
class Cena {  
    let botão = Botão()  
    var contagem = 0  
  
    init() {  
        botão.action = {  
            self.contagem += 1  
        }  
    }  
}
```

```
var cena:Cena? = Cena()
```

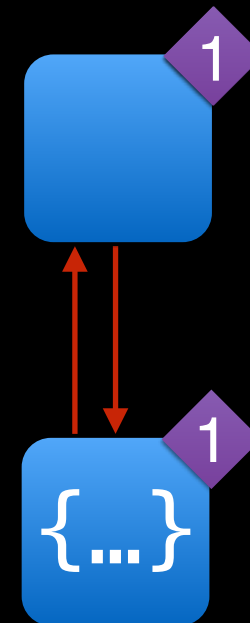


Closures

```
class Botão {  
    var action: (() -> ())?  
}  
  
class Cena {  
    let botão = Botão()  
    var contagem = 0  
  
    init() {  
        botão.action = {  
            self.contagem += 1  
        }  
    }  
}
```

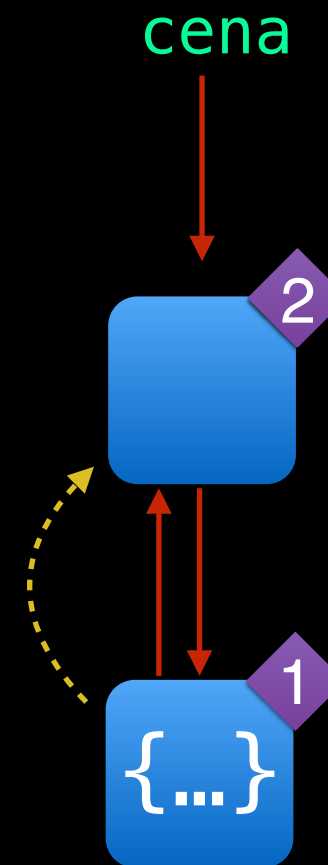
```
var cena:Cena? = Cena()  
  
cena = nil
```

nil ← cena



Closures

```
class Botão {  
    var action: (()->())?  
}  
  
class Cena {  
    let botão = Botão()  
    var contagem = 0  
  
    init() {  
        self.botão.action = { [weak self] in  
            self?.contagem += 1  
        }  
    }  
}  
  
var cena:Cena? = Cena()
```

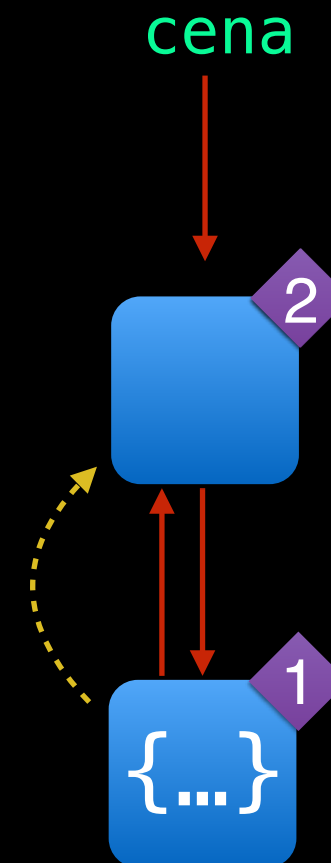


Closures

```
class Botão {  
    var action: (()->())?  
}
```

```
class Cena {  
    let botão = Botão()  
    var contagem = 0  
  
    init() {  
        self.botão.action = { [weak self] in  
            self?.contagem += 1  
        }  
    }  
}
```

```
var cena:Cena? = Cena()
```



Closures

```
let closure: (()->(()))? = { [weak self] in
    if let strongSelf = self {
        strongSelf.view.alpha = 0.0
    }
}
```

Closures

```
let resource = UIViewController()  
  
let closure: (()->())? = { [weak self, weak resource] in  
    if let strongSelf = self {  
        strongSelf.view.alpha = 0.0  
    }  
  
    resource?.view.alpha = 0.0  
}
```

Closures

(Lista de Captura)

Prevenção

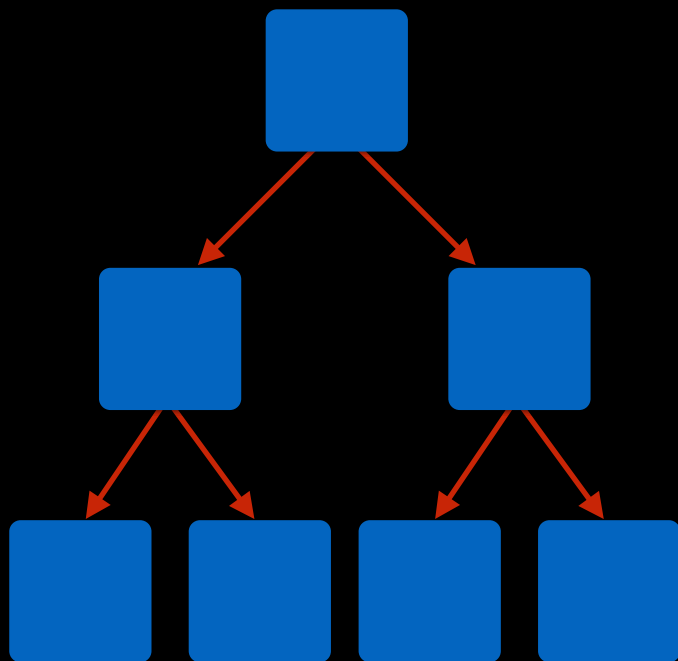
Cuidado

- Relações de Posse
- Referencias para classes
- Closures
- NSTimer
- UIAlertController

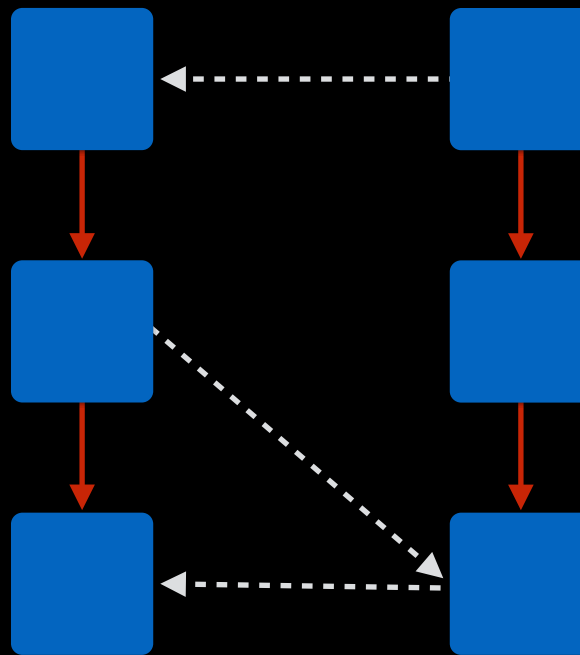
Use logs:

```
deinit {  
    println("object is being deallocated")  
}  
  
-(void)dealloc  
{  
    NSLog(@"object is being deallocated");  
}
```

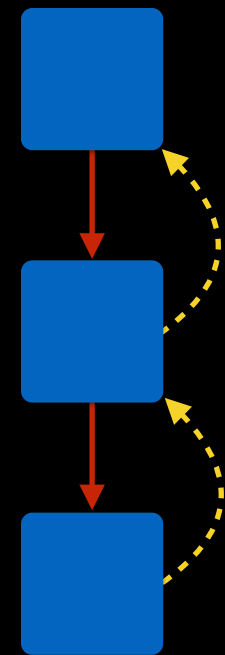
strong



weak



unowned



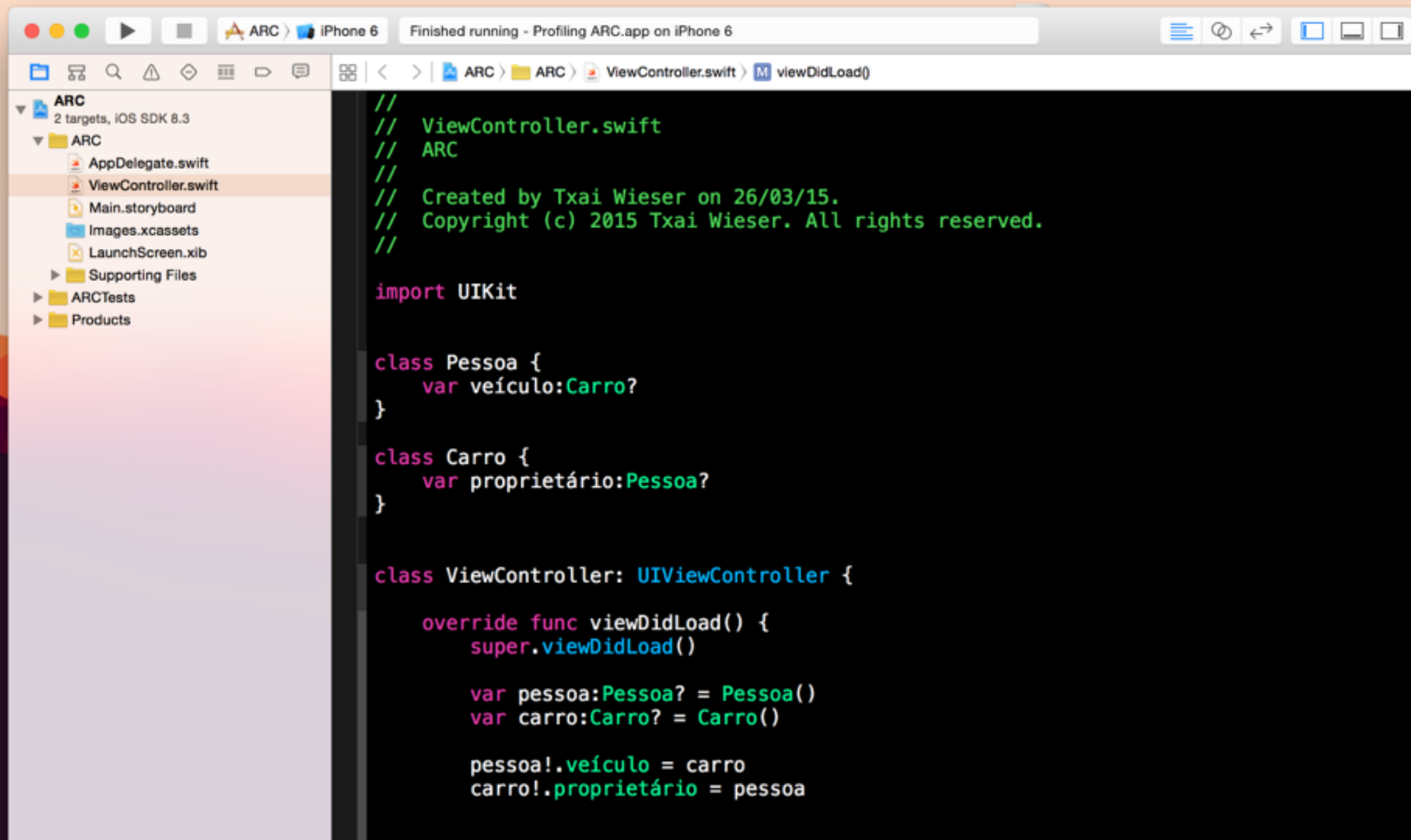


Value Types



Reference Types

debugging



- About Xcode
- Preferences... ⌘,
- Behaviors ▶
- Open Developer Tool ▶
- Services ▶
- Hide Xcode ⌘H
- Hide Others ⌘⇧H
- Show All
- Quit Xcode ⌘Q

- Instruments
- iOS Simulator
- Accessibility Inspector
- FileMerge
- Application Loader
- More Developer Tools...

- AppDelegate.swift
- ViewController.swift
- Main.storyboard
- Images.xcassets
- LaunchScreen.xib
- Supporting Files
- ARCTests
- Products

ashed running - Profiling ARC.app on iPhone 6

ARC > ARC > ViewController.swift > viewDidLoad()

```
ViewController.swift
//
// Created by Txai Wieser on 26/03/15.
// Copyright (c) 2015 Txai Wieser. All rights reserved.
//

import UIKit

class Pessoa {
    var veículo:Carro?
}

class Carro {
    var proprietário:Pessoa?
}

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        var pessoa:Pessoa? = Pessoa()
        var carro:Carro? = Carro()

        pessoa!.veículo = carro
        carro!.proprietário = pessoa
    }
}
```

Choose a profiling template for: iPhone 6 (8.3 Simulator) > ARC.app

Standard Custom Recent



Blank



Activity Monitor



Allocations



Automation



Cocoa Layout



Core Animation



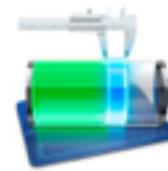
Core Data



Counters



Dispatch



Energy Diagnostics



File Activity



GPU Driver



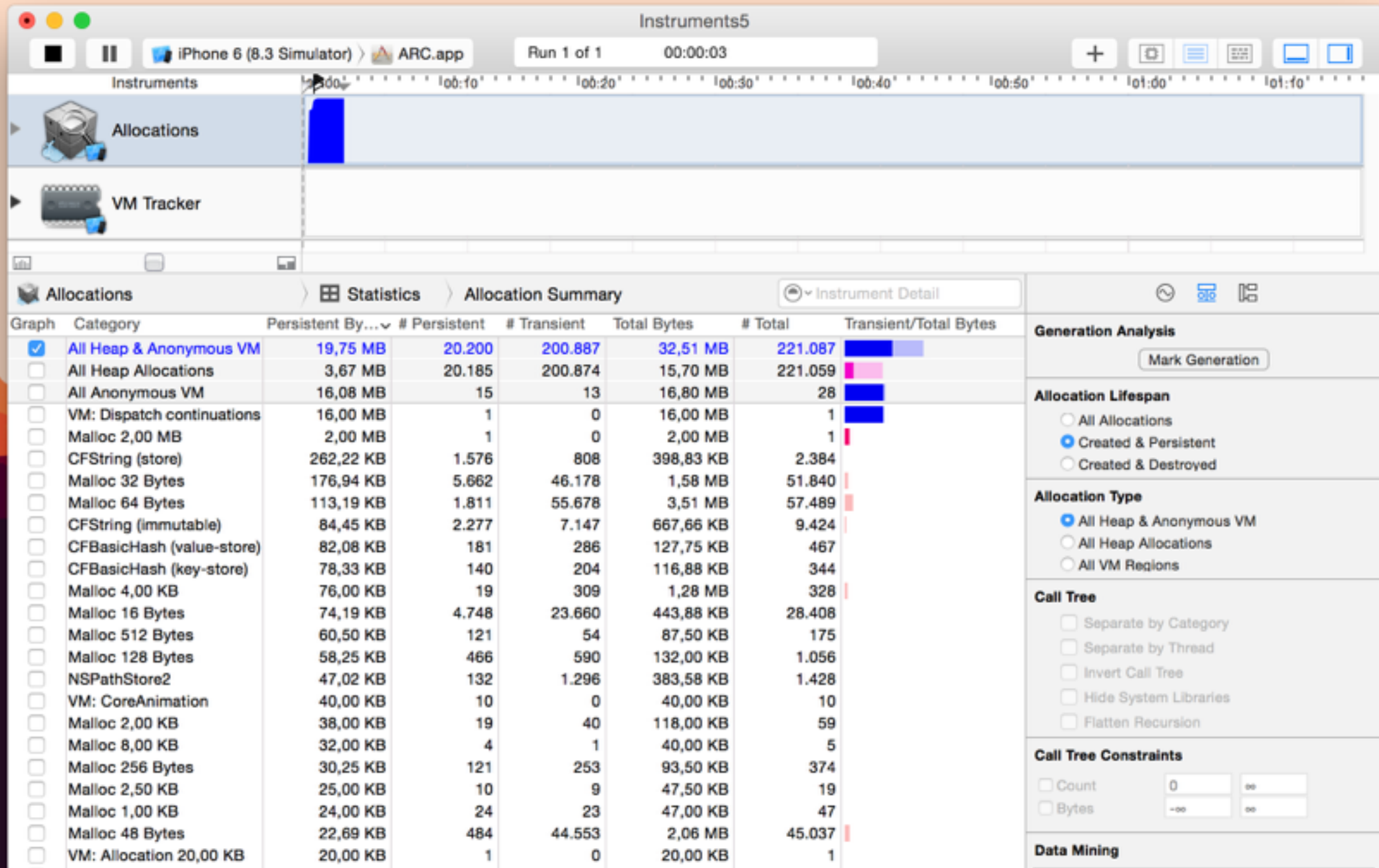
Allocations

Tracks a process' anonymous virtual memory and heap, providing class names and optionally retain/release histories for objects.

Cancel

Choose

The screenshot shows the Instruments application window titled "Instruments5". The top status bar indicates the target is "iPhone 6 (8.3 Simulator)" running "ARC.app", with "Run 0 of 0" and a time of "00:00:00". Below the status bar is a timeline from 00:00' to 01:10'. Two instruments are listed: "Allocations" (described as analyzing memory life-cycle) and "VM Tracker" (described as tracking virtual memory space). The main pane displays the "Allocation Summary" view under the "Statistics" tab. It features a table with columns: Graph, Category, Persistent By..., # Persistent, # Transient, Total Bytes, # Total, and Transient/Total Bytes. The table is currently empty. On the right side, there are configuration panels. The "Launch Configuration for All Allocations" panel has three options: "Discard unrecorded data upon stop" (checked), "Discard events for freed memory" (unchecked), and "Only track VM allocations" (unchecked). The "Launch Configuration for Heap Allocations" panel has three options: "Record reference counts" (unchecked), "Identify virtual C++ objects" (checked), and "Enable NSZombie detection" (unchecked). The "Track Display" panel shows "Style" set to "Current Bytes" and "Type" set to "Overlay". The "Recorded Types" panel contains a table with columns "Type String", "Search", and "Action". It lists four types: "*" (checked, search "is Contai...", action "Record"), "NS" (unchecked, search "is Prefix", action "Ignore"), "CF" (unchecked, search "is Prefix", action "Ignore"), and "Malloc" (unchecked, search "is Prefix", action "Ignore"). At the bottom of this panel, it states "Rules are evaluated top-to-bottom; later rules override earlier ones."



The screenshot shows the Instruments5 application interface. At the top, the title bar reads 'Instruments5'. Below it, a status bar indicates 'iPhone 6 (8.3 Simulator)' and 'ARC.app'. The main timeline at the top shows a blue bar representing memory allocations, starting at 00:00 and ending at 00:10. The left sidebar contains icons for 'Instruments', 'Allocations', and 'VM Tracker'. The 'Allocations' panel is selected, displaying a table with the following columns: Graph, Category, Persistent By..., # Persistent, # Transient, Total Bytes, # Total, and Transient/Total Bytes. A single row is visible for 'ARC.Pessoa' with 32 Bytes of persistent memory, 0 transient bytes, and a total of 32 Bytes. The right sidebar contains several sections: 'Generation Analysis' with a 'Mark Generation' button; 'Allocation Lifespan' with radio buttons for 'All Allocations', 'Created & Persistent' (selected), and 'Created & Destroyed'; 'Allocation Type' with radio buttons for 'All Heap & Anonymous VM' (selected), 'All Heap Allocations', and 'All VM Regions'; 'Call Tree' with checkboxes for 'Separate by Category', 'Separate by Thread', 'Invert Call Tree', 'Hide System Libraries', and 'Flatten Recursion'; 'Call Tree Constraints' with checkboxes for 'Count' and 'Bytes' and input fields for values; and 'Data Mining'.

Graph	Category	Persistent By...	# Persistent	# Transient	Total Bytes	# Total	Transient/Total Bytes
<input type="checkbox"/>	ARC.Pessoa	32 Bytes	1	0	32 Bytes	1	

The screenshot shows the Instruments5 application interface. At the top, the title bar reads 'Instruments5'. Below it, a status bar indicates 'Run 1 of 1' and a duration of '00:00:11'. The main window is divided into several sections. On the left, there are icons for 'Instruments', 'Allocations', and 'VM Tracker'. The 'Allocations' section is currently selected, showing a blue bar representing memory usage over time. Below this, a table titled 'Allocation Summary' is displayed. The table has columns for 'Graph', 'Category', 'Persistent By...', '# Persistent', '# Transient', 'Total Bytes', '# Total', and 'Transient/Total Bytes'. A single row is visible for 'ARC.Pessoa', showing 32 Bytes of persistent memory, 0 transient bytes, and a total of 32 Bytes. To the right of the table, there are several panels for further analysis: 'Generation Analysis' with a 'Mark Generation' button, 'Allocation Lifespan' with radio buttons for 'All Allocations', 'Created & Persistent' (selected), and 'Created & Destroyed', 'Allocation Type' with radio buttons for 'All Heap & Anonymous VM' (selected), 'All Heap Allocations', and 'All VM Regions', 'Call Tree' with checkboxes for 'Separate by Category', 'Separate by Thread', 'Invert Call Tree', 'Hide System Libraries', and 'Flatten Recursion', 'Call Tree Constraints' with checkboxes for 'Count' and 'Bytes' and input fields for values, and 'Data Mining'.

Graph	Category	Persistent By...	# Persistent	# Transient	Total Bytes	# Total	Transient/Total Bytes
<input type="checkbox"/>	ARC.Pessoa	32 Bytes	1	0	32 Bytes	1	

The screenshot shows the Instruments application interface. At the top, the target is 'iPhone 6 (8.3 Simulator)' and the application is 'ARC.app'. The 'Allocations' instrument is selected, showing a blue bar representing memory allocations. The 'Statistics' tab is active, displaying an 'Allocation Summary' for the category 'ARC.Pessoa'. The summary shows 1 persistent allocation of 32 bytes. The right sidebar contains filters for 'Generation Analysis', 'Allocation Lifespan', 'Allocation Type', 'Call Tree', and 'Call Tree Constraints'.

Graph	Category	Persistent Bytes	# Persistent	# Transient	Total Bytes	# Total	Transient/Total Bytes
<input type="checkbox"/>	ARC.Pessoa	32 Bytes	1	0	32 Bytes	1	

The screenshot shows the Instruments5 application interface. At the top, the title bar reads 'Instruments5'. Below it, a status bar indicates 'Run 1 of 1' and a duration of '00:00:11'. The main window is divided into several sections. On the left, there are icons for 'Instruments', 'Allocations', and 'VM Tracker'. The 'Allocations' section is currently selected, showing a summary table. The table has columns for 'Graph', 'Category', 'Persistent By...', '# Persistent', '# Transient', 'Total Bytes', '# Total', and 'Transient/Total Bytes'. A single row is visible for 'ARC.Pessoa', showing 32 Bytes of persistent memory, 0 transient bytes, and a total of 32 Bytes. On the right side of the window, there are several panels: 'Generation Analysis' with a 'Mark Generation' button, 'Allocation Lifespan' with radio buttons for 'All Allocations', 'Created & Persistent' (selected), and 'Created & Destroyed', 'Allocation Type' with radio buttons for 'All Heap & Anonymous VM' (selected), 'All Heap Allocations', and 'All VM Regions', 'Call Tree' with checkboxes for 'Separate by Category', 'Separate by Thread', 'Invert Call Tree', 'Hide System Libraries', and 'Flatten Recursion', 'Call Tree Constraints' with checkboxes for 'Count' and 'Bytes' and input fields for values, and 'Data Mining'.

Graph	Category	Persistent By...	# Persistent	# Transient	Total Bytes	# Total	Transient/Total Bytes
<input type="checkbox"/>	ARC.Pessoa	32 Bytes	1	0	32 Bytes	1	

Instruments5

iPhone 6 (8.3 Simulator) > ARC.app

Run 1 of 1 00:00:11

Instruments

Allocations

VM Tracker

Allocations

Statistics

Allocation Summary

peessoa

Graph	Category	Persistent By...	# Persistent	# Transient	Total Bytes	# Total	Transient/Total Bytes
<input type="checkbox"/>	ARC.Pessoa	32 Bytes	1	0	32 Bytes	1	

Generation Analysis

Mark Generation

Allocation Lifespan

☐ All Allocations
 ☒ Created & Persistent
 ☐ Created & Destroyed

Allocation Type

☒ All Heap & Anonymous VM
 ☐ All Heap Allocations
 ☐ All VM Regions

Call Tree

☐ Separate by Category
 ☐ Separate by Thread
 ☐ Invert Call Tree
 ☐ Hide System Libraries
 ☐ Flatten Recursion

Call Tree Constraints

☐ Count

0

∞

☐ Bytes

-∞

∞

Data Mining

The screenshot shows the Instruments5 application interface. At the top, the title bar reads 'Instruments5'. Below it, a status bar indicates 'iPhone 6 (8.3 Simulator)' and 'ARC.app'. The main window is divided into several sections. On the left, there are icons for 'Instruments', 'Allocations', and 'VM Tracker'. The 'Allocations' section is currently selected, showing a blue bar representing memory usage over time. Below this, the 'Allocation Summary' table is visible, filtered for 'pessoa'. The table has columns for 'Graph', 'Category', 'Persistent By...', '# Persistent', '# Transient', 'Total Bytes', '# Total', and 'Transient/Total Bytes'. The data row shows 'ARC.Pessoa' with 32 Bytes persistent, 0 transient, and 32 Bytes total. On the right side of the window, there are several panels: 'Generation Analysis' with a 'Mark Generation' button, 'Allocation Lifespan' with radio buttons for 'All Allocations', 'Created & Persistent' (selected), and 'Created & Destroyed', 'Allocation Type' with radio buttons for 'All Heap & Anonymous VM' (selected), 'All Heap Allocations', and 'All VM Regions', 'Call Tree' with checkboxes for 'Separate by Category', 'Separate by Thread', 'Invert Call Tree', 'Hide System Libraries', and 'Flatten Recursion', 'Call Tree Constraints' with checkboxes for 'Count' and 'Bytes' and input fields for values, and 'Data Mining'.

Graph	Category	Persistent By...	# Persistent	# Transient	Total Bytes	# Total	Transient/Total Bytes
<input type="checkbox"/>	ARC.Pessoa	32 Bytes	1	0	32 Bytes	1	

The screenshot shows the Instruments5 application interface. At the top, the title bar reads 'Instruments5'. Below it, a status bar indicates 'iPhone 6 (8.3 Simulator)' and 'ARC.app'. A progress bar shows 'Run 1 of 1' and a time of '00:00:18'. The main interface is divided into several sections:

- Instruments:** A timeline at the top with a blue bar representing the duration of the run.
- Allocations:** A section on the left showing a graph of memory allocations over time.
- Statistics:** A section on the left showing a table of allocation statistics.
- Allocation Summary:** A section on the right showing a detailed view of the allocation statistics for the selected category 'pessoa'.

The 'Allocation Summary' table has the following columns: Graph, Category, Persistent By..., # Persistent, # Transient, Total Bytes, # Total, and Transient/Total Bytes. The table contains one row for 'ARC.Pessoa' with the following values:

Graph	Category	Persistent By...	# Persistent	# Transient	Total Bytes	# Total	Transient/Total Bytes
<input type="checkbox"/>	ARC.Pessoa	32 Bytes	1	0	32 Bytes	1	

The right sidebar contains several sections for further analysis:

- Generation Analysis:** Includes a 'Mark Generation' button.
- Allocation Lifespan:** Includes radio buttons for 'All Allocations', 'Created & Persistent' (selected), and 'Created & Destroyed'.
- Allocation Type:** Includes radio buttons for 'All Heap & Anonymous VM' (selected), 'All Heap Allocations', and 'All VM Regions'.
- Call Tree:** Includes checkboxes for 'Separate by Category', 'Separate by Thread', 'Invert Call Tree', 'Hide System Libraries', and 'Flatten Recursion'.
- Call Tree Constraints:** Includes checkboxes for 'Count' and 'Bytes', with input fields for values.
- Data Mining:** A section at the bottom for data mining options.

The screenshot shows the Instruments5 application interface. At the top, the title bar reads 'Instruments5'. Below it, a status bar indicates 'Run 1 of 1' and a duration of '00:00:18'. The main window is divided into several sections. On the left, there are icons for 'Instruments', 'Allocations', and 'VM Tracker'. The 'Allocations' section is currently selected, showing a blue bar representing memory usage over time. Below this, the 'Allocation Summary' table is visible, filtered for 'pessoa'. The table has columns for 'Graph', 'Category', 'Persistent By...', '# Persistent', '# Transient', 'Total Bytes', '# Total', and 'Transient/Total Bytes'. One entry is shown: 'ARC.Pessoa' with a green arrow icon, 32 Bytes persistent, 0 transient, and 32 Bytes total. On the right side of the window, there are several panels: 'Generation Analysis' with a 'Mark Generation' button, 'Allocation Lifespan' with radio buttons for 'All Allocations', 'Created & Persistent' (selected), and 'Created & Destroyed', 'Allocation Type' with radio buttons for 'All Heap & Anonymous VM' (selected), 'All Heap Allocations', and 'All VM Regions', 'Call Tree' with checkboxes for 'Separate by Category', 'Separate by Thread', 'Invert Call Tree', 'Hide System Libraries', and 'Flatten Recursion', 'Call Tree Constraints' with checkboxes for 'Count' and 'Bytes' and input fields for values, and 'Data Mining'.

Graph	Category	Persistent By...	# Persistent	# Transient	Total Bytes	# Total	Transient/Total Bytes
<input type="checkbox"/>	ARC.Pessoa	32 Bytes	1	0	32 Bytes	1	

Instruments5

iPhone 6 (8.3 Simulator)

ARC.app

Run 1 of 1

00:00:32

+

🔍

📄

📊

📌

📁

Instruments

Allocations

VM Tracker

00:00:00

00:00:10

00:00:20

00:00:30

00:00:40

00:00:50

00:01:00

00:01:10

Allocations

Statistics

Allocation Summary

ARC.Pessoa

🔍

pessoa

✕

🕒

📄

📊

#	Address	Category	Timestamp	Live	Size	Responsible Library	Responsible Caller
0	0x7fc3d375...	ARC.Pessoa	00:01.025.762	•	32 Bytes	libswiftCore.dylib	swift_slowAlloc

Generation Analysis

Mark Generation

Allocation Lifespan

☐ All Allocations
 ☒ Created & Persistent
 ☐ Created & Destroyed

Allocation Type

☒ All Heap & Anonymous VM
 ☐ All Heap Allocations
 ☐ All VM Regions

Call Tree

☐ Separate by Category
 ☐ Separate by Thread
 ☐ Invert Call Tree
 ☐ Hide System Libraries
 ☐ Flatten Recursion

Call Tree Constraints

☐ Count

0

∞

☐ Bytes

-∞

∞

Data Mining

Instruments5

iPhone 6 (8.3 Simulator) > ARC.app

Run 1 of 1 00:00:32

Instruments

Allocations

VM Tracker

Allocations

Statistics

Allocation Summary

ARC.Pessoa

peessoa

#	Address	Category	Timestamp	Live	Size	Responsible Library	Responsible Caller
0	0x7fc3d375...	ARC.Pessoa	00:01.025.762		32 Bytes	libswiftCore.dylib	swift_slowAlloc

Generation Analysis

Mark Generation

Allocation Lifespan

☐ All Allocations
 ☒ Created & Persistent
 ☐ Created & Destroyed

Allocation Type

☒ All Heap & Anonymous VM
 ☐ All Heap Allocations
 ☐ All VM Regions

Call Tree

☐ Separate by Category
 ☐ Separate by Thread
 ☐ Invert Call Tree
 ☐ Hide System Libraries
 ☐ Flatten Recursion

Call Tree Constraints

☐ Count
☐ Bytes

Data Mining

Instruments5

iPhone 6 (8.3 Simulator) > ARC.app Run 1 of 1 00:00:32

Instruments

Allocations

VM Tracker

Allocations Statistics Allocation Summary ARC.Pessoa pessoa

#	Address	Category	Timestamp	Live	Size	Responsible Library	Responsible Caller
0	0x7fc3d375...	ARC.Pessoa	00:01.025.762	•	32 Bytes	libswiftCore.dylib	swift_slowAlloc

Generation Analysis

Mark Generation

Allocation Lifespan

☐ All Allocations

☒ Created & Persistent

☐ Created & Destroyed

Allocation Type

☒ All Heap & Anonymous VM

☐ All Heap Allocations

☐ All VM Regions

Call Tree

☐ Separate by Category

☐ Separate by Thread

☐ Invert Call Tree

☐ Hide System Libraries

☐ Flatten Recursion

Call Tree Constraints

☐ Count 0 00

☐ Bytes -00 00

Data Mining

Instruments5

Run 1 of 1 00:00:26

iPhone 6 (8.3 Simulator) > ARC.app

Instruments

Allocations

VM Tracker

#	Address	Category	Timestamp	Live	Size	Responsible Library	Responsible Caller
0	0x7fc3d375...	ARC.Pessoa	00:01.025.762	•	32 Bytes	libswiftCore.dylib	swift_slowAlloc

Allocations

Statistics

Allocation Summary

ARC.Pessoa

peessoa

Generation Analysis

Mark Generation

Allocation Lifespan

☐ All Allocations

☒ Created & Persistent

☐ Created & Destroyed

Allocation Type

☒ All Heap & Anonymous VM

☐ All Heap Allocations

☐ All VM Regions

Call Tree

☐ Separate by Category

☐ Separate by Thread

☐ Invert Call Tree

☐ Hide System Libraries

☐ Flatten Recursion

Call Tree Constraints

☐ Count 0 00

☐ Bytes -00 00

Data Mining

Instruments5

iPhone 6 (8.3 Simulator) > ARC.app

Run 1 of 1 00:00:26

Instruments

Allocations

VM Tracker

Allocations

Statistics > Allocation Summary > ARC.Pessoa

#	Address	Category	Timestamp	Live	Size	Responsible Library	Responsible Caller
0	0x7fc3d375...	ARC.Pessoa	00:01.025.762	•	32 Bytes	libswiftCore.dylib	swift_slowAlloc

Generation Analysis

Mark Generation

Allocation Lifespan

☐ All Allocations

☒ Created & Persistent

☐ Created & Destroyed

Allocation Type

☒ All Heap & Anonymous VM

☐ All Heap Allocations

☐ All VM Regions

Call Tree

☐ Separate by Category

☐ Separate by Thread

☐ Invert Call Tree

☐ Hide System Libraries

☐ Flatten Recursion

Call Tree Constraints

☐ Count 0 00

☐ Bytes -00 00

Data Mining

Instruments5

iPhone 6 (8.3 Simulator)

ARC.app

Run 1 of 1

00:00:26

+

Instruments

Allocations

VM Tracker

Allocations

Statistics

Allocation Summary

ARC.Pessoa

pessoa

#	Address	Category	Timestamp	Live	Size	Responsible Library	Responsible Caller
0	0x7fc3d375...	ARC.Pessoa	00:01.025.762	•	32 Bytes	libswiftCore.dylib	swift_slowAlloc

Generation Analysis

Mark Generation

Allocation Lifespan

☐ All Allocations
 ☒ Created & Persistent
 ☐ Created & Destroyed

Allocation Type

☒ All Heap & Anonymous VM
 ☐ All Heap Allocations
 ☐ All VM Regions

Call Tree

☐ Separate by Category
 ☐ Separate by Thread
 ☐ Invert Call Tree
 ☐ Hide System Libraries
 ☐ Flatten Recursion

Call Tree Constraints

☐ Count

0

∞

☐ Bytes

-∞

∞

Data Mining

The screenshot shows the Instruments5 application with the Allocations instrument selected. The top bar indicates the target is 'iPhone 6 (8.3 Simulator)' and the application is 'ARC.app'. The timeline shows a blue bar representing memory allocations. The bottom pane displays a table of allocation events.

#	Event Type	Δ RefCt	RefCt	Timestamp	Responsible L...	Responsible Caller
0	Malloc	+1	1	00:01.025.762	libswiftCore.d...	swift_slowAlloc

The right sidebar contains several sections for analysis:

- Generation Analysis:** Includes a 'Mark Generation' button.
- Allocation Lifespan:** Includes radio buttons for 'All Allocations', 'Created & Persistent' (selected), and 'Created & Destroyed'.
- Allocation Type:** Includes radio buttons for 'All Heap & Anonymous VM' (selected), 'All Heap Allocations', and 'All VM Regions'.
- Call Tree:** Includes checkboxes for 'Separate by Category', 'Separate by Thread', 'Invert Call Tree', 'Hide System Libraries', and 'Flatten Recursion'.
- Call Tree Constraints:** Includes checkboxes for 'Count' and 'Bytes', with input fields for values (0, -00, 00).
- Data Mining:** (Section header visible).

Instruments5

Run 1 of 1 00:00:40

Instruments

Allocations

VM Tracker

#	Event Type	Δ RefCt	RefCt	Timestamp	Responsible L...	Responsible Caller
0	Malloc	+1	1	00:01.025.762	libswiftCore.d...	swift_slowAlloc

Allocations

Allocat ARC.Pessoa History: 0x7fc3d375aca0

Generation Analysis

Mark Generation

Allocation Lifespan

Allocation Type

Call Tree

Call Tree Constraints

Data Mining

The screenshot shows the Instruments5 application interface. At the top, the title bar reads "Instruments5". Below it, the status bar indicates the target is "iPhone 6 (8.3 Simulator)" and the application is "ARC.app". The "Run 1 of 1" button shows a duration of "00:00:40".

The main area is divided into two sections: "Instruments" and "Allocations". The "Instruments" section shows a timeline with a blue bar representing memory allocation. The "Allocations" section shows a table of memory events.

#	Event Type	Δ RefCt	RefCt	Timestamp	Responsible L...	Responsible Caller
0	Malloc	+1	1	00:01.025.762	libswiftCore.d...	swift_slowAlloc

The right sidebar contains several sections:

- Generation Analysis**: Includes a "Mark Generation" button.
- Allocation Lifespan**: Includes radio buttons for "All Allocations", "Created & Persistent" (selected), and "Created & Destroyed".
- Allocation Type**: Includes radio buttons for "All Heap & Anonymous VM" (selected), "All Heap Allocations", and "All VM Regions".
- Call Tree**: Includes checkboxes for "Separate by Category", "Separate by Thread", "Invert Call Tree", "Hide System Libraries", and "Flatten Recursion".
- Call Tree Constraints**: Includes checkboxes for "Count" and "Bytes", with input fields for values (0, -00, 00).
- Data Mining**: (Section header visible at the bottom).

Referências:

- https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/AutomaticReferenceCounting.html
- <http://www.ibm.com/developerworks/library/mo-ios-memory/>

Thanks ;)

github.com/txaidw

twitter.com/txaidw