

Борьба с утечками памяти: от задачи до победы



Колосов Артём,
VKontakte

Колосов Артём

Разработчик из команды
Core iOS ВКонтакте

- Красил кнопки
- Крутил деревья
- Релизил часы
- Уничтожал утечки

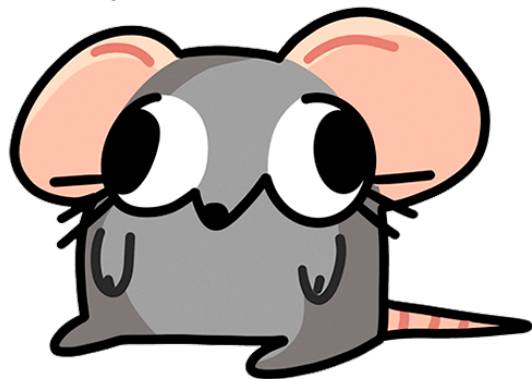




Почему вам не стоит
делать приложение
для Apple Watch

Александр Крайнов

* СКРЫВАЕТ
РЫДАНИЯ *



Что
обсудим?

Что обсудим?

→ Что такое утечка

Что обсудим?

- Что такое утечка
- Графики с пользователями

Что обсудим?

- Что такое утечка
- Графики с пользователями
- Инструменты для индикации

Что обсудим?

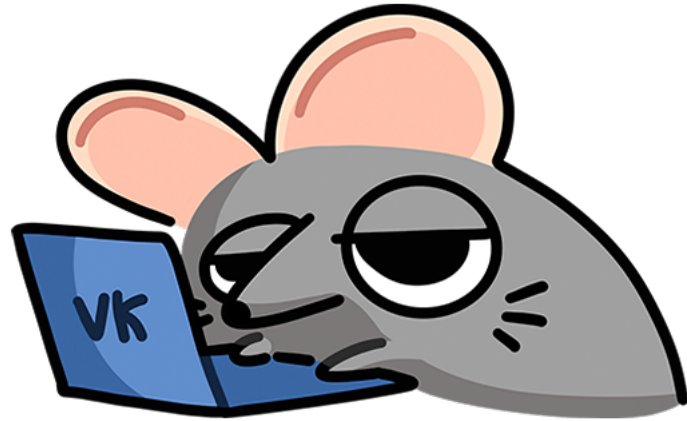
- Что такое утечка
- Графики с пользователями
- Инструменты для индикации
- Особые случаи утечек

Что обсудим?

- Что такое утечка
- Графики с пользователями
- Инструменты для индикации
- Особые случаи утечек
- Автоматизация индикации

Что обсудим?

- Что такое утечка
- Графики с пользователями
- Инструменты для индикации
- Особые случаи утечек
- Автоматизация индикации
- Приобщение коллег к общему благу





ВЫНЮХИВАЮ



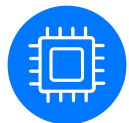




Утечка — наличие в памяти объекта, которого там быть не должно по логике программы

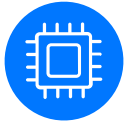
Почему надо бороться с утечками памяти?

Почему надо бороться с утечками памяти?



Сокращение потребляемой оперативной памяти

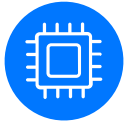
Почему надо бороться с утечками памяти?



Сокращение потребляемой оперативной памяти

Реже OOM

Почему надо бороться с утечками памяти?

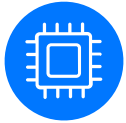


Сокращение потребляемой оперативной памяти

Реже OOM

Система реже выгружает приложение

Почему надо бороться с утечками памяти?



Сокращение потребляемой оперативной памяти

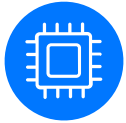
Реже OOM

Система реже выгружает приложение



Поддержание корректного состояния приложения

Почему надо бороться с утечками памяти?



Сокращение потребляемой оперативной памяти

Реже OOM

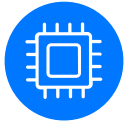
Система реже выгружает приложение



Поддержание корректного состояния приложения

Уменьшение количества ошибок

Почему надо бороться с утечками памяти?



Сокращение потребляемой оперативной памяти

Реже OOM

Система реже выгружает приложение



Поддержание корректного состояния приложения

Уменьшение количества ошибок

Освобождение ресурсов

Анализ работы приложения



MetricKit



MetricKit

Позволяет собирать отчеты по каждому устройству для диагностики сбоев, производительности и других метрик.

[Documentation/MetricKit](#)



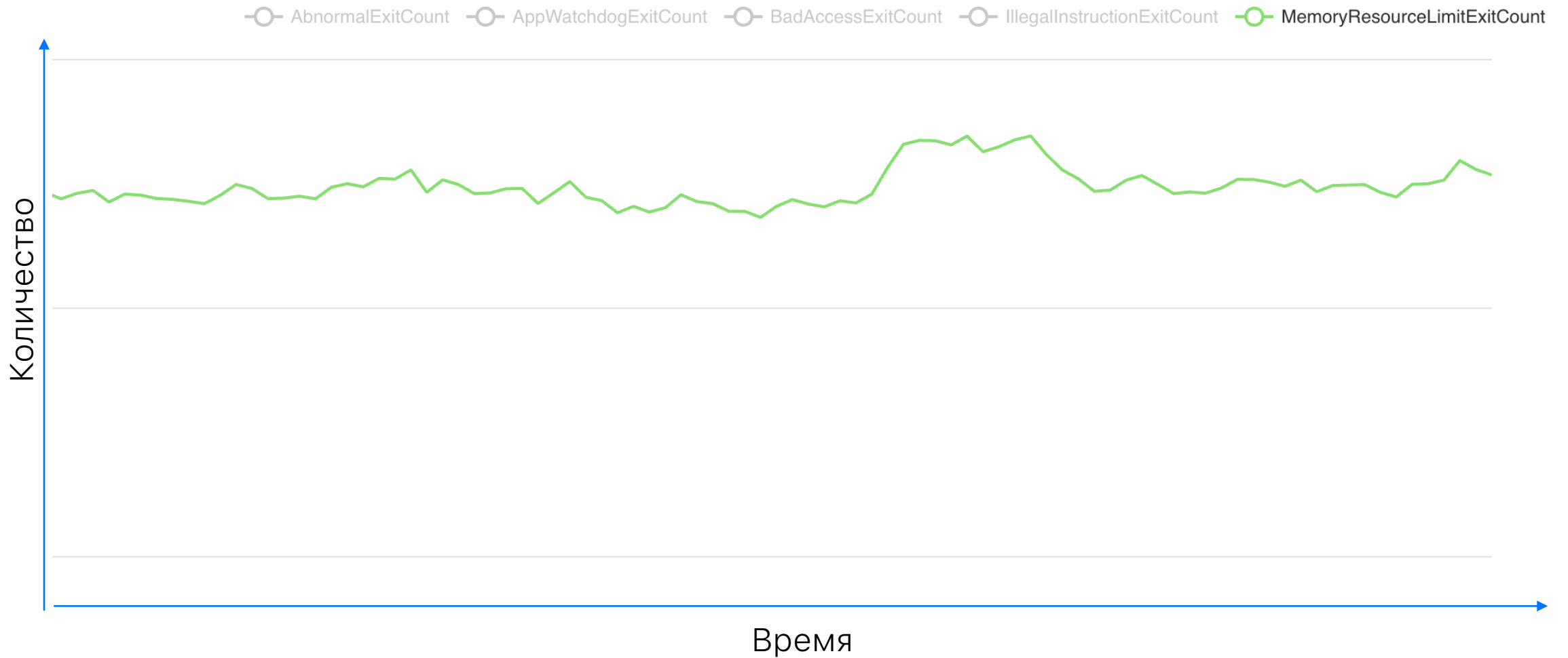
MemoryResourceLimitExitCount

Количество закрытий приложения
из-за нехватки памяти.

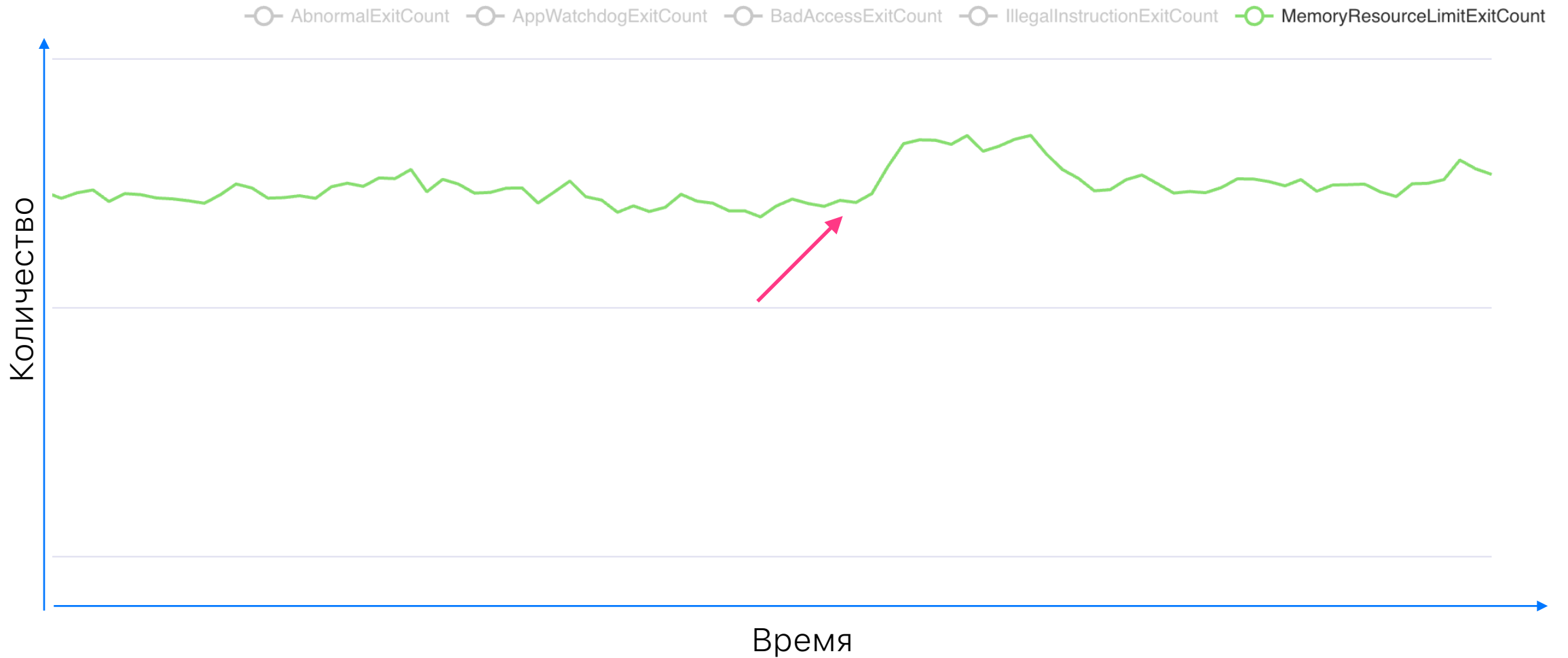
[Documentation/MetricKit/MXAppExitMetric/](#)



Закрүтия по памяти



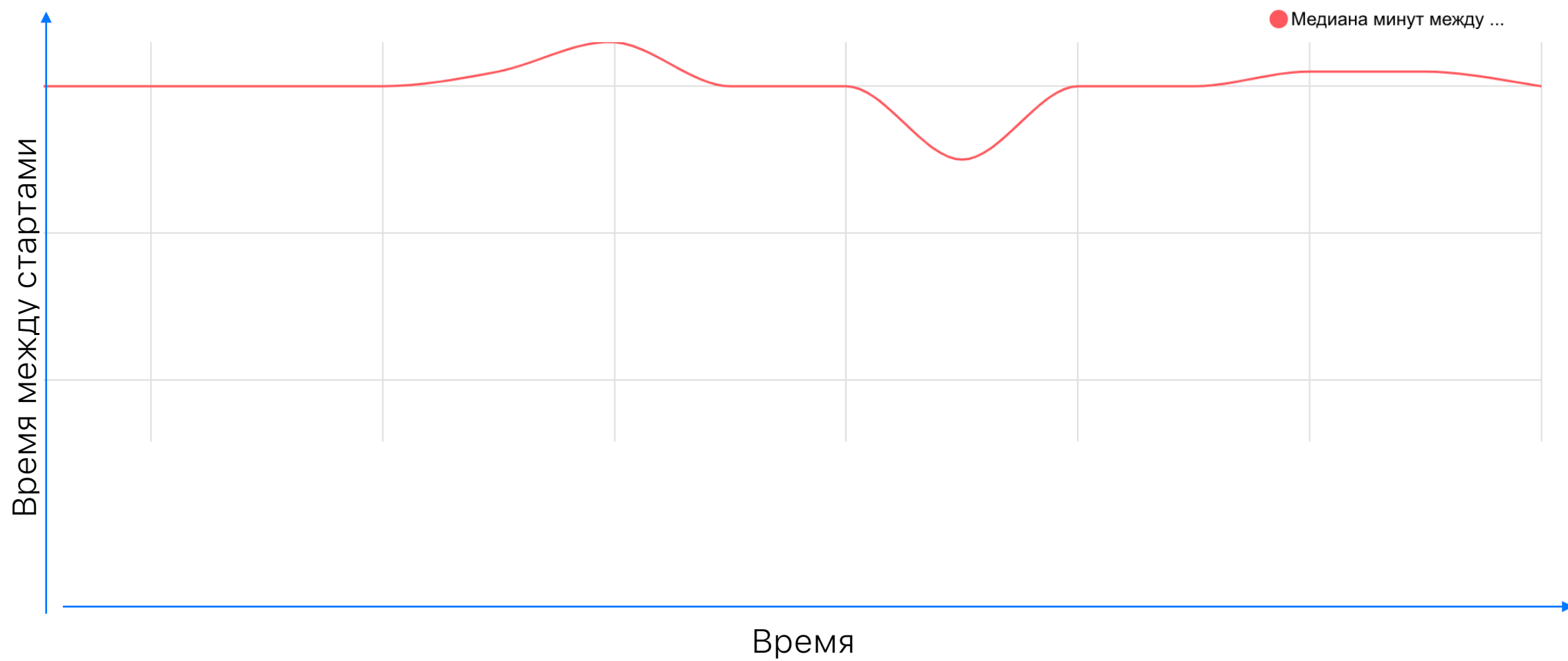
Закрүтия по памяти



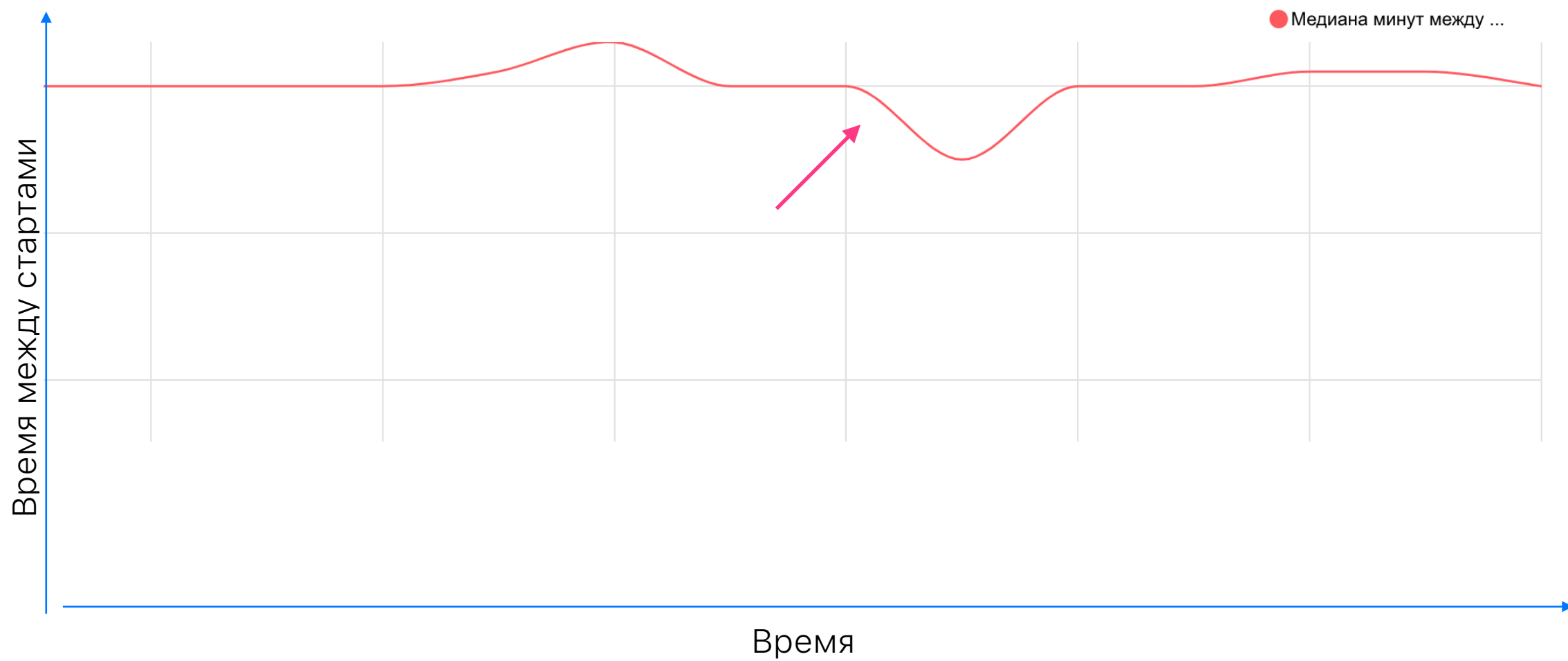
Закрүтия по памяти



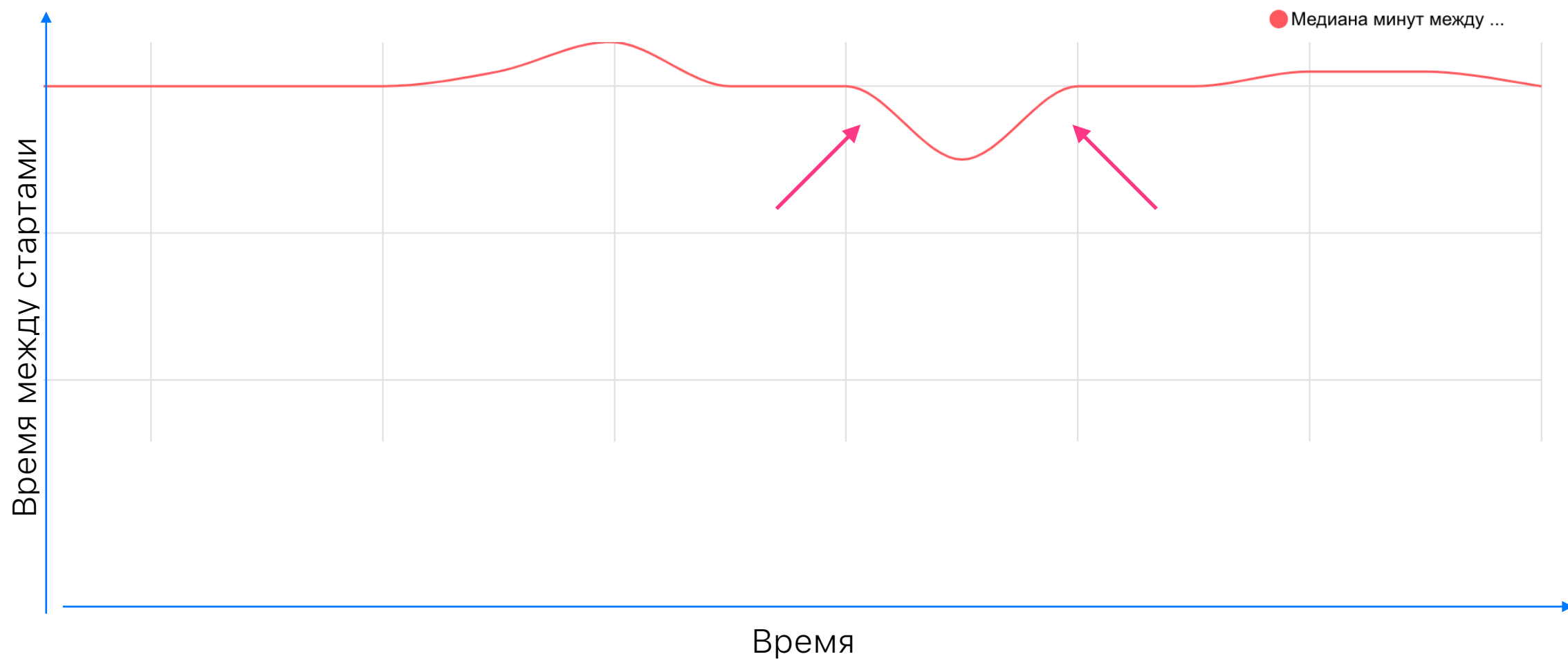
Медиана между холодными стартами



Медиана между холодными стартами



Медиана между холодными стартами





Как сделать,
чтобы не тормозило,
пожалуйста

Евгений Шаповалов



ПЕПЕЦ



Определяем
присутствие
утечки



Как определять
присутствие
утечки?

Как определять присутствие утечки?

- ⚙️ Пользуемся инструментами Xcode и верим, что они определяют

Как определять присутствие утечки?

- ⚙️ Пользуемся инструментами Xcode и верим, что они определяют
- 🔍 Определяем GOD-objects и следим за их количеством самостоятельно
























Как определять присутствие утечки?


- ⚙️ Пользуемся инструментами Xcode и верим, что они определяют
- 🔍 Определяем GOD-objects и следим за их количеством самостоятельно

```
class GodObject {  
    static var objectsCount = 0  
  
    init() {  
        Self.objectsCount += 1  
    }  
  
    deinit {  
        Self.objectsCount -= 1  
    }  
}  
assert(GodObject.objectsCount == 0)
```


Выясняем наличие утечек

All Standard User Recent Filter

 Blank	 Activity Monitor	 Allocations	 Animation Hitches	 App Launch	 Audio System Trace
 Core ML	 CPU Counters	 CPU Profiler	 Data Persistence	 File Activity	 Game Memory
 Game Performance	 Leaks	 Logging	 Metal System Trace	 Network	 SceneKit
 Swift Concurrency	 SwiftUI	 System Trace	 Time Profiler	 Zombies	

 **Leaks**
Measures general memory usage, checks for leaked memory, and provides statistics on object allocations by class as well as memory address histories for all active allocations and leaked blocks.

Пользуемся автоматикой?

```
class A: GodObject {  
    var b: B?  
}  
  
class B {  
    var a: A?  
}  
  
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.a = a  
}  
  
doSomething()
```

```
assert(A.objectsCount == 0) Thread 1: Assertion failed
```

The screenshot shows the JustCode IDE interface. At the top, there's a 'Track Filter' and 'All Tracks' buttons. A timeline at the top right shows a time range from 00:00 to 00:19.036, with a blue highlight from 00:00 to 00:06.700. Below the timeline, there are two main sections: 'Allocations' (All Heap & Anonymous VM) and 'Leaks' (Leak Checks). The 'Leaks' section shows a red diamond icon with a plus sign and a tooltip that says '2 new leaks'. At the bottom, there's a table titled 'Leaked Object' with columns for 'Leaked Object', 'Count', 'Address', and 'Size'.

Leaked Object	Count	Address	Size
B	1	0x6000002d4120	32 Bytes
A	1	0x6000002d40c0	32 Bytes

Instruments

Resolved Issues

- Fixed: Leaks instrument never detects leaks. (116020104)



Как определять присутствие утечки?

- ⚙️ Пользуемся инструментами Xcode и верим, что они определяют
- 🔍 Определяем GOD-objects и следим за их количеством самостоятельно

```
class GodObject {  
    static var objectsCount = 0  
  
    init() {  
        Self.objectsCount += 1  
    }  
  
    deinit {  
        Self.objectsCount -= 1  
    }  
}  
assert(GodObject.objectsCount == 0)
```

Как определять присутствие утечки?

- ⚙️ Пользуемся инструментами Xcode и верим, что они определяют
- 🔍 Определяем GOD-objects и следим за их количеством самостоятельно

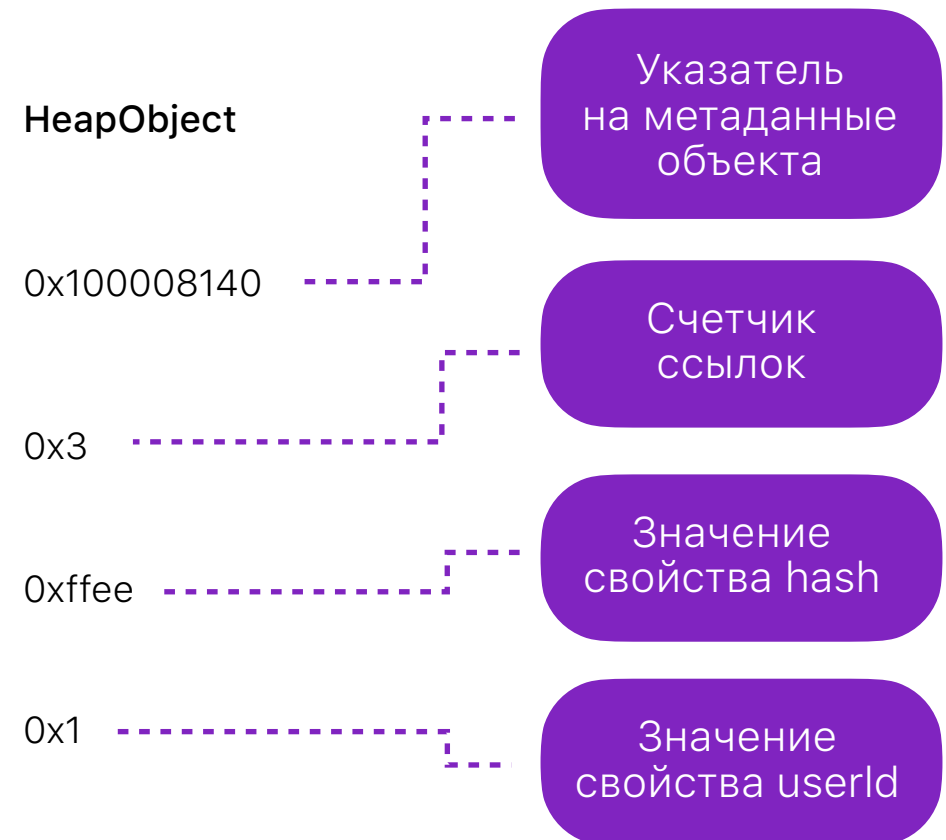
```
class GodObject {  
    static var objectsCount = 0  
  
    init() {  
        Self.objectsCount += 1  
    }  
  
    deinit {  
        Self.objectsCount -= 1  
    }  
}  
assert(GodObject.objectsCount == 0)
```

Модель памяти

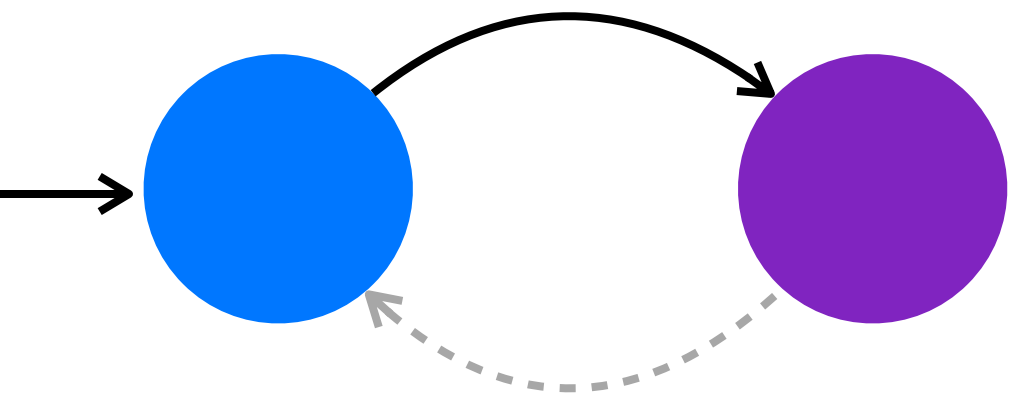


Хранение объекта

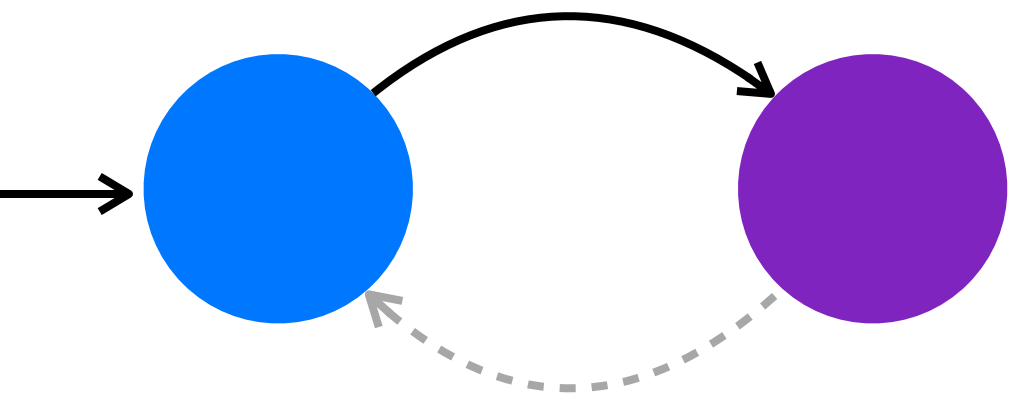
```
class CommitInfo {  
  let hash: Int  
  let userId: Int  
  
  init(hash: Int, userId: Int) {  
    self.hash = hash  
    self.userId = userId  
  }  
}  
let firstCommit = CommitInfo(hash: 0xffee, userId: 1)
```



Виды ссылок

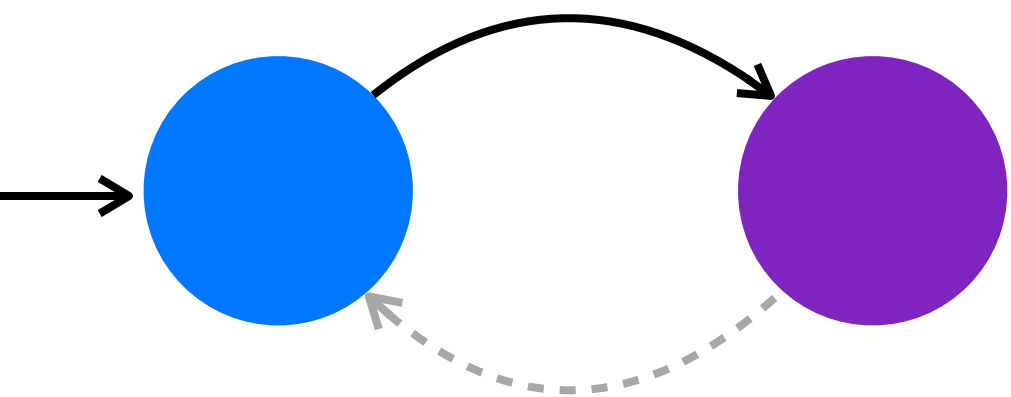


Виды ссылок



Strong

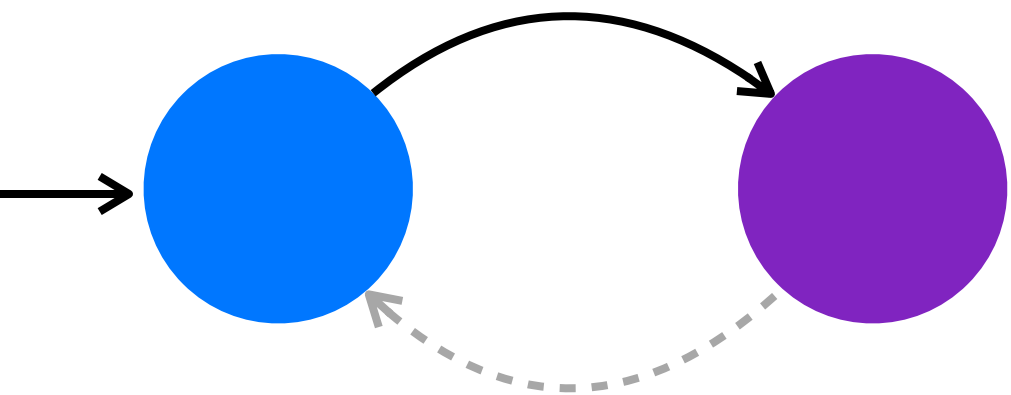
Виды ссылок



Strong

Weak

Виды ссылок

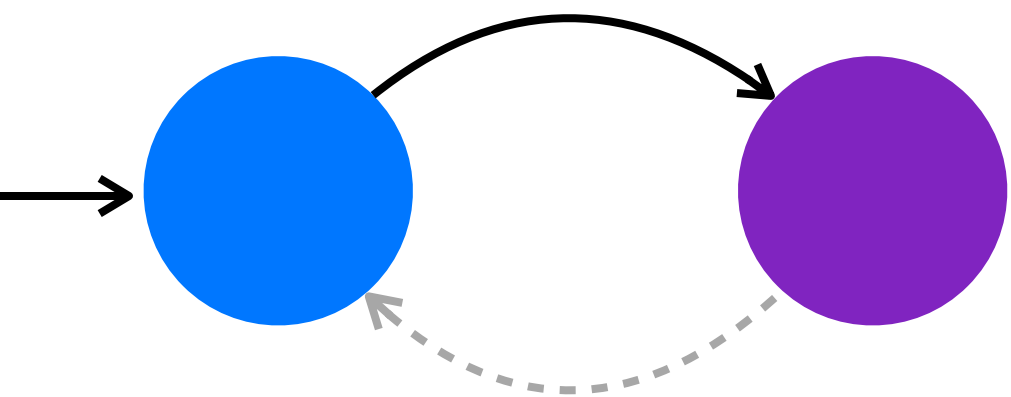


Strong

Weak

Unowned

Виды ссылок



Strong

Weak

Unowned

Unowned
(unsafe)

Счётчик ссылок (InlineRefCounts)

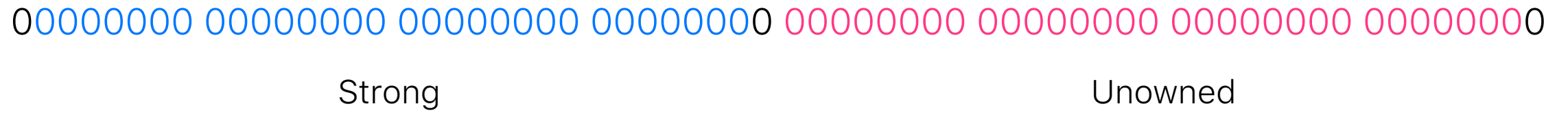
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Счётчик ссылок (InlineRefCounts)

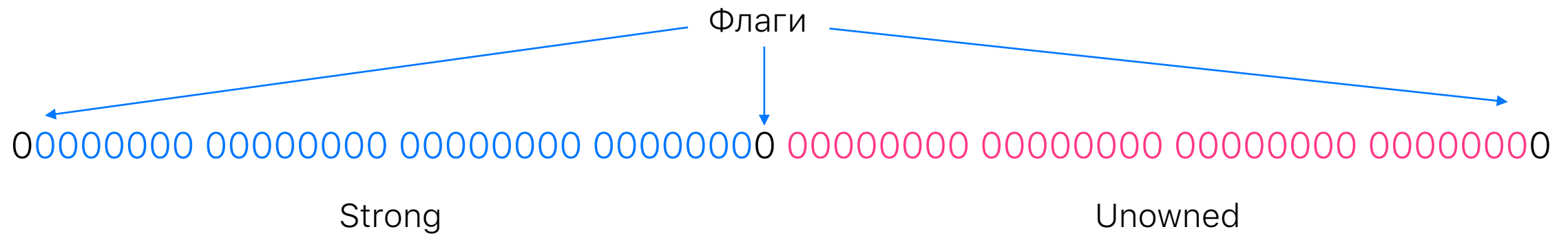
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Unowned

Счётчик ссылок (InlineRefCounts)



Счётчик ссылок (InlineRefCounts)



Счётчик ссылок (HeapObjectSideTableEntry)

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Счётчик ссылок (HeapObjectSideTableEntry)

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Ссылка на Side Table

Счётчик ссылок (HeapObjectSideTableEntry)

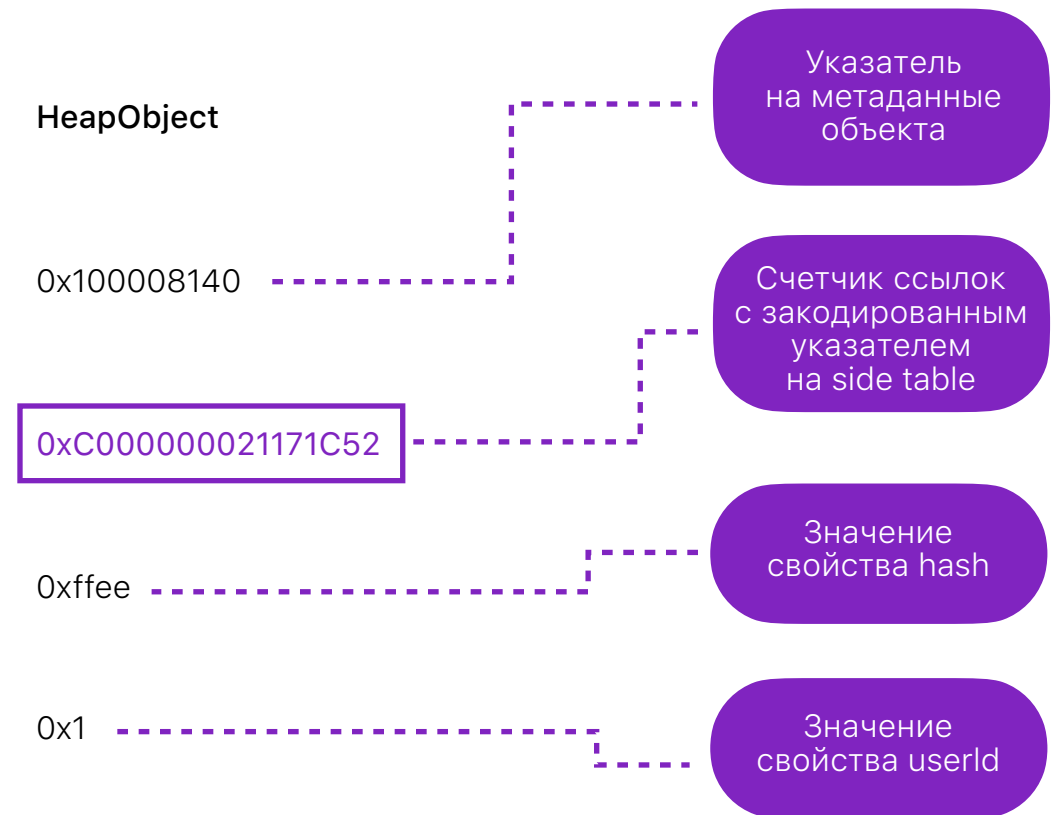
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Флаги

Ссылка на Side Table

Хранение объекта

```
class CommitInfo {  
  let hash: Int  
  let userId: Int  
  
  init(hash: Int, userId: Int) {  
    self.hash = hash  
    self.userId = userId  
  }  
}  
let firstCommit = CommitInfo(hash: 0xffee, userId: 1)  
weak var weakCommit = firstCommit
```



Side table

```
class HeapObjectSideTableEntry {
    std::atomic<HeapObject*> object;
    SideTableRefCounts refCounts;
}

class WeakReference {
    union {
        std::atomic<WeakReferenceBits> nativeValue;
#ifdef SWIFT_OBJC_INTEROP
        id nonnativeValue;
#endif
    };
}
```

```
apple/swift/blob/main/stdlib/public/runtime/WeakReference.h
apple/swift/blob/main/stdlib/public/SwiftShims/swift/shims/RefCount.h
```

Жизненный цикл объекта

- 1 **Live** – объект создан и делает какие-то полезные вещи.

- 2 **Deiniting**
- 3 **Deinited**
- 4 **Freed**
- 5 **Dead**

<https://github.com/apple/swift/blob/main/stdlib/public/SwiftShims/swift/shims/RefCount.h>
https://habr.com/ru/companies/vivid_money/articles/592599/

Почему надо бороться с утечками памяти?

Почему надо бороться с утечками памяти?



Цикл сильных ссылок

Почему надо бороться с утечками памяти?



Цикл сильных ссылок

- Захват `self` в `@escaping` closure

Почему надо бороться с утечками памяти?



Цикл сильных ссылок

- Захват `self` в `@escaping` closure



Удержание в глобальных переменных

Почему надо бороться с утечками памяти?



Цикл сильных ссылок

- Захват `self` в `@escaping` closure



Удержание в глобальных переменных

- Хранение в статическом поле

Как определять присутствие утечки?

- ⚙️ Пользуемся инструментами Xcode и верим, что они определяют
- 🔍 Определяем GOD-objects и следим за их количеством самостоятельно

```
class GodObject {  
    static var objectsCount = 0  
  
    init() {  
        Self.objectsCount += 1  
    }  
  
    deinit {  
        Self.objectsCount -= 1  
    }  
}  
assert(GodObject.objectsCount == 0)
```

Граф пам'яті



WWDC. iOS Memory Deep Dive

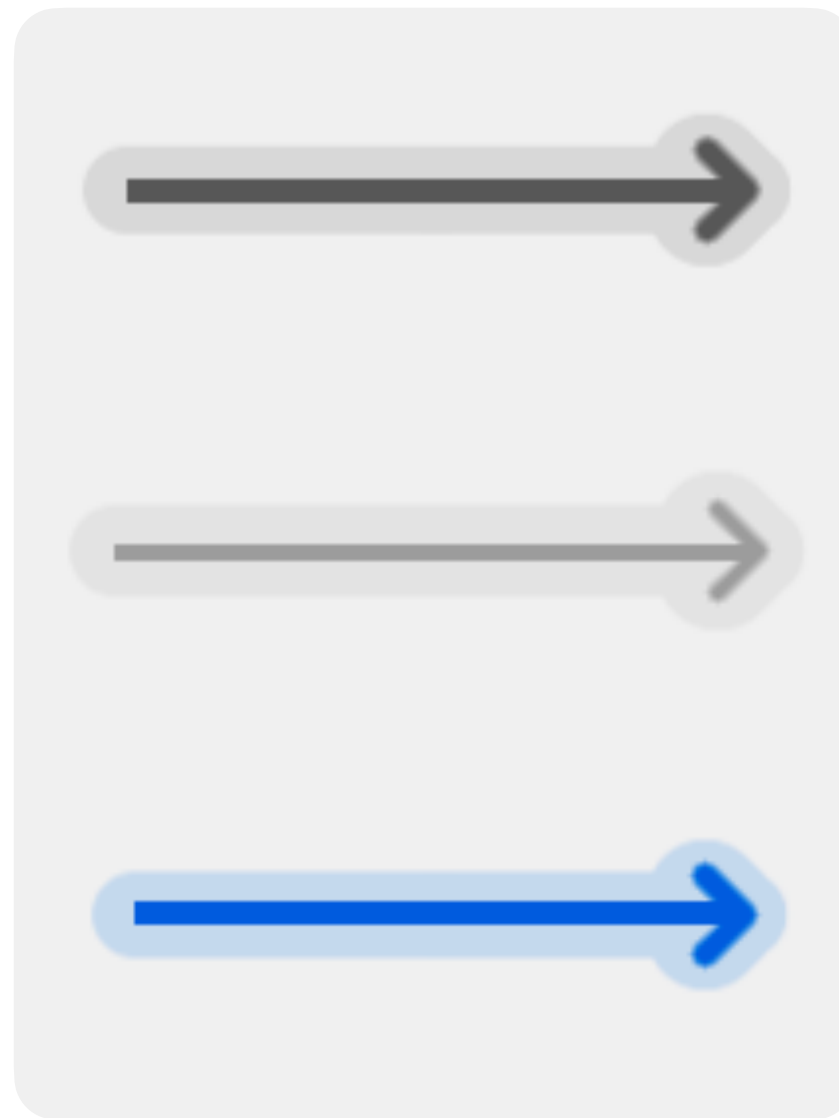


Легенда карты

Чёрная стрелка — в большинстве своём сильная ссылка.

Серая стрелка — в большинстве случаев слабая ссылка на объект.

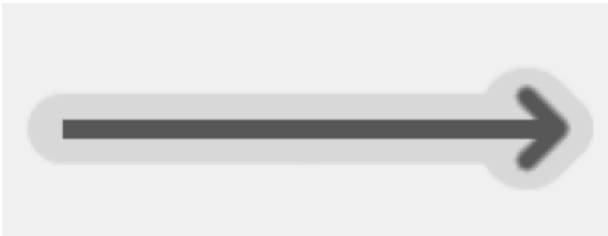
Синяя стрелка — связь, выбранная в дебагере.



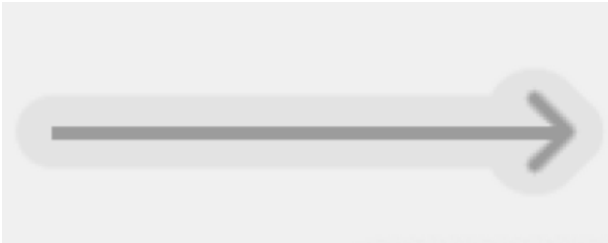
Легенда карты (Memory Graph)

Стрелка в графе

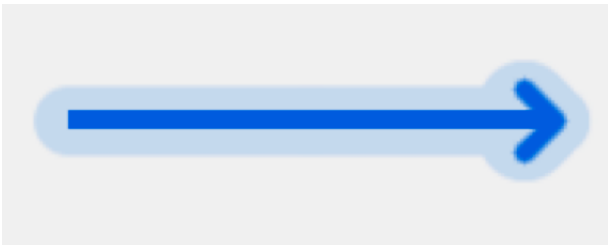
Связь в рантайме



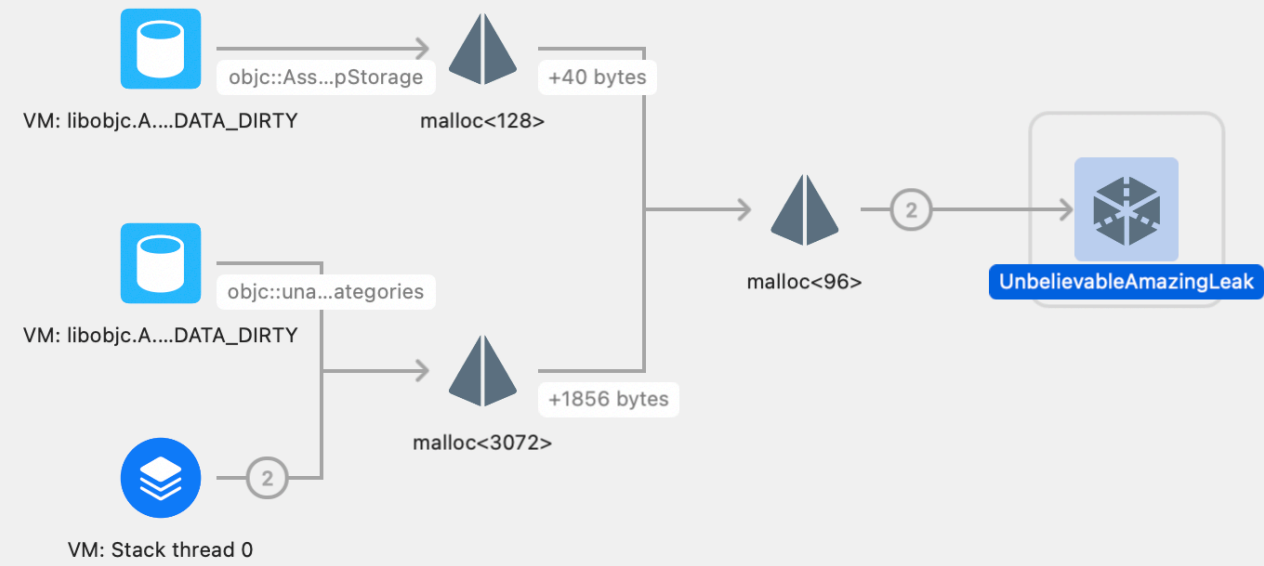
Strong

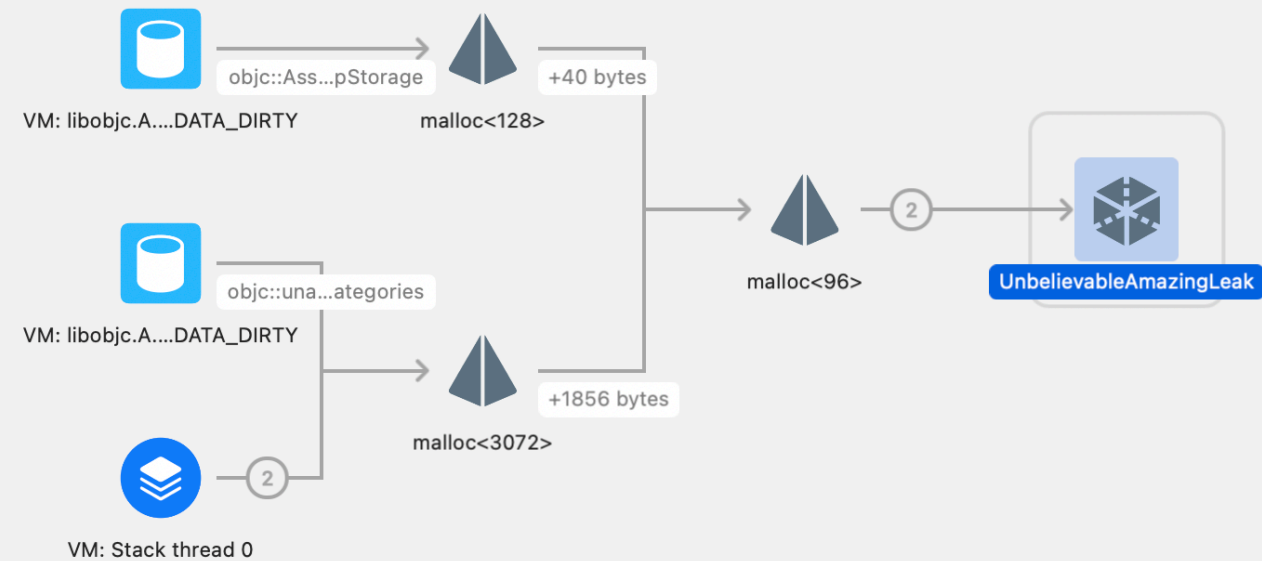


Weak



Выбранная связь в дебаггере





Info Arguments Options **Diagnostics**

- Runtime Sanitization** Address Sanitizer ⓘ
 Requires recompilation Detect use of stack after return
 Thread Sanitizer ⓘ
 Undefined Behavior Sanitizer ⓘ

- Runtime API Checking** Main Thread Checker
 Thread Performance Checker ⓘ

- Memory Management** Malloc Scribble
 Malloc Guard Edges
 Guard Malloc
 Zombie Objects
 Malloc Stack Logging
 ⓘ
 Memory Graph on Resource Exception ⓘ

- Metal** API Validation
 Shader Validation →

heme Manage Schemes... Shared

Определяем причины утечек

```
class A: GodObject {  
    var b: B?  
}
```

```
class B {  
    var a: A?  
}
```

```
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.a = a  
}
```

```
doSomething()
```

```
assert(A.objectsCount == 0)
```



```
Thread 1: Assertion failed
```

Собираем граф памяти

JustCode PID 42928

- CPU 0%
- Memory 4,5 MB
- Energy Impact Zero
- Disk Zero KB/s
- Network Zero KB/s

JustCode (2)

- A (1)
 - 0x134105c20
- B (1)
 - 0x134107050
- <unknown> (54)
 - non-object in zone D... (39)
 - 0x6000008a4040
 - 0x6000013a4000
 - 0x6000013a4030
 - 0x600001da0000
 - 0x600001da4000
 - 0x600001da4020
 - 0x600001da4040
 - 0x600001da4060

JustCode JustCode My Mac Paused JustCode

main Memory Graph for JustCode main

JustCode JustCode A 0x134105c20 → a

A diagram showing a memory graph. Two nodes, A and B, are connected by a blue double-headed arrow. A central node '2' is also connected to both A and B. A tooltip box is positioned below node '2', containing a right-pointing arrow followed by 'a' and a left-pointing arrow followed by 'b'.

Reference Details

Name a
Type strong
Source <B 0x134107050> [32] +16
Destination <A 0x134105c20> [32] +0

Собираем граф памяти

The screenshot displays the Xcode IDE with the JustCode memory graph tool active. The interface is divided into several sections:

- Left Sidebar:** Shows system and application resources. Under "JustCode (2)", the memory address `0x134105c20` is selected, indicated by a pink arrow pointing to the graph.
- Top Bar:** Displays "JustCode", "My Mac", and "Paused JustCode". The breadcrumb path is "main" > "Memory Graph for JustCode" > "main".
- Graph Area:** Shows a memory graph with two nodes, A and B, connected by a blue double-headed arrow. A central node labeled "2" is also connected to both A and B. A tooltip for node "2" shows a right-pointing arrow labeled "a" and a left-pointing arrow labeled "b".
- Reference Details Panel:** Located on the right, it provides information for the selected reference:
 - Name: a
 - Type: strong
 - Source: <B 0x134107050> [32] +16
 - Destination: <A 0x134105c20> [32] +0

Собираем граф памяти

The screenshot displays the JustCode IDE interface. On the left, a sidebar shows system and application resources. The main window displays a memory graph for the 'main' process. The graph shows two memory nodes, A and B, connected by a blue double-headed arrow. Node A is at address 0x134105c20 and node B is at address 0x134107050. A tooltip over the arrow shows a value of 2. A pink arrow points from the tooltip to the left sidebar, highlighting the memory address 0x134105c20. On the right, the 'Reference Details' panel shows the following information:

Reference Details	
Name	a
Type	strong
Source	<B 0x134107050> [32] +16
Destination	<A 0x134105c20> [32] +0

Собираем граф памяти

The screenshot displays the JustCode IDE interface. On the left, a sidebar shows system and application resources. The main window displays a memory graph for the 'main' process. The graph shows two memory nodes, A and B, connected by a blue double-headed arrow. Node A is at address 0x134105c20 and node B is at address 0x134107050. A callout box between the nodes shows a blue arrow pointing to 'a' and a red arrow pointing to 'b'. The 'Reference Details' panel on the right shows the details for the reference from B to A: Name 'a', Type 'strong', Source '<B 0x134107050> [32] +16', and Destination '<A 0x134105c20> [32] +0'. A pink arrow points from the 'Type strong' text to the 'Reference Details' panel.

JustCode JustCode > My Mac Paused JustCode

main Memory Graph for JustCode main

JustCode > JustCode > A > 0x134105c20 → a

Reference Details

Name a
Type strong
Source <B 0x134107050> [32] +16
Destination <A 0x134105c20> [32] +0

JustCode PID 42928
CPU 0%
Memory 4,5 MB
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s

JustCode (2)
A (1)
0x134105c20
B (1)
0x134107050
<unknown> (54)
non-object in zone D... (39)
0x6000008a4040
0x6000013a4000
0x6000013a4030
0x600001da0000
0x600001da4000
0x600001da4020
0x600001da4040
0x600001da4060

Собираем граф памяти

The screenshot displays the Xcode Instruments Memory Graph tool. The left sidebar shows system metrics for the JustCode process (PID 42928): CPU at 0%, Memory at 4.5 MB, Energy Impact at Zero, Disk at Zero KB/s, and Network at Zero KB/s. Below these are the process's memory spaces: JustCode (2), A (1), and an unknown space (54). The A space contains one object at address 0x134105c20, which is selected. The unknown space contains 39 non-objects in zone D...

The central graph area shows a memory graph with two objects, A and B, connected by a bidirectional arrow. Object A is at address 0x134105c20 and Object B is at address 0x134107050. A circle with the number 2 is positioned on the arrow between them.

The right sidebar provides details for the selected object:

- Object**
 - Class Name: A
 - Kind: Swift
 - Address: 0x134105c20
 - Size: 32 bytes
 - Malloc Zone: MallocStackLoggingLiteZone
- Hierarchy**
 - A
 - GodObject
 - _TtCs12_SwiftObject
- Backtrace**
 - 0 _malloc_zone_malloc_instrumented_or_legacy
 - 3 A.__allocating_init()
 - 4 doSomething()
 - 5 main
 - 6 start
 - 7 0xffffffffffffffff

Собираем граф памяти

The screenshot displays the Xcode Instruments Memory Graph tool. The left sidebar shows system metrics for the JustCode process (PID 42928): CPU at 0%, Memory at 4.5 MB, Energy Impact at Zero, Disk at Zero KB/s, and Network at Zero KB/s. Below these are the process's memory objects, with the object at address 0x134105c20 selected. The central area shows a memory graph with two objects, A and B, connected by a pointer. A red arrow points to object A. The right sidebar provides details for the selected object: Class Name A, Kind Swift, Address 0x134105c20, Size 32 bytes, and Malloc Zone MallocStackLoggingLiteZone. Below this is the Hierarchy (A, GodObject, _TtCs12_SwiftObject) and a Backtrace showing the call stack from start to the current location.

Object

- Class Name A
- Kind Swift
- Address 0x134105c20
- Size 32 bytes
- Malloc Zone MallocStackLoggingLiteZone

Hierarchy

- A
- GodObject
- _TtCs12_SwiftObject

Backtrace

- 0 _malloc_zone_malloc_instrumented_or_legacy
- 3 A.__allocating_init()
- 4 doSomething()
- 5 main
- 6 start
- 7 0xffffffffffffffff

Собираем граф памяти

The screenshot displays the Xcode Instruments interface with the Memory Graph tool active. The main window shows a memory graph with two objects, A and B, connected by a bidirectional arrow. Object A is highlighted with a blue square and a pink arrow pointing to it. Object B is also highlighted with a blue square. A pink arrow also points to the 'Class Name A' field in the right sidebar.

System Metrics:

- JustCode PID 42928
- CPU: 0%
- Memory: 4,5 MB
- Energy Impact: Zero
- Disk: Zero KB/s
- Network: Zero KB/s

Process Memory:

- JustCode (2)
- A (1)
- 0x134105c20 (highlighted)
- B (1)
- 0x134107050
- <unknown> (54)
- non-object in zone D... (39)
- 0x6000008a4040
- 0x6000013a4000
- 0x6000013a4030
- 0x600001da0000
- 0x600001da4000
- 0x600001da4020
- 0x600001da4040
- 0x600001da4060
- 0x600001da4080

Memory Graph:

- Object A (Address: 0x134105c20)
- Object B (Address: 0x134107050)
- Connection: 2

Object Details (Class Name A):

- Class Name: A
- Kind: Swift
- Address: 0x134105c20
- Size: 32 bytes
- Malloc Zone: MallocStackLoggingLiteZone

Hierarchy:

- A
- GodObject
- _TtCs12_SwiftObject

Backtrace:

- 0 _malloc_zone_malloc_instrumented_or_legacy
- 3 A.__allocating_init()
- 4 doSomething()
- 5 main
- 6 start
- 7 0xffffffffffffffff

Собираем граф памяти

The screenshot displays the Xcode Instruments Memory Graph tool. The left sidebar shows system metrics for the JustCode process (PID 42928): CPU (0%), Memory (4.5 MB), Energy Impact (Zero), Disk (Zero KB/s), and Network (Zero KB/s). Below these are the process's memory objects, with object A (0x134105c20) selected. The central area shows a memory graph with two objects, A and B, connected by a bidirectional arrow labeled '2'. A pink arrow points to object A. The right sidebar provides details for the selected object:

Object

- Class Name A
- Kind Swift
- Address 0x134105c20
- Size 32 bytes
- Malloc Zone MallocStackLoggingLiteZone

Hierarchy

- A
- GodObject
- _TtCs12_SwiftObject

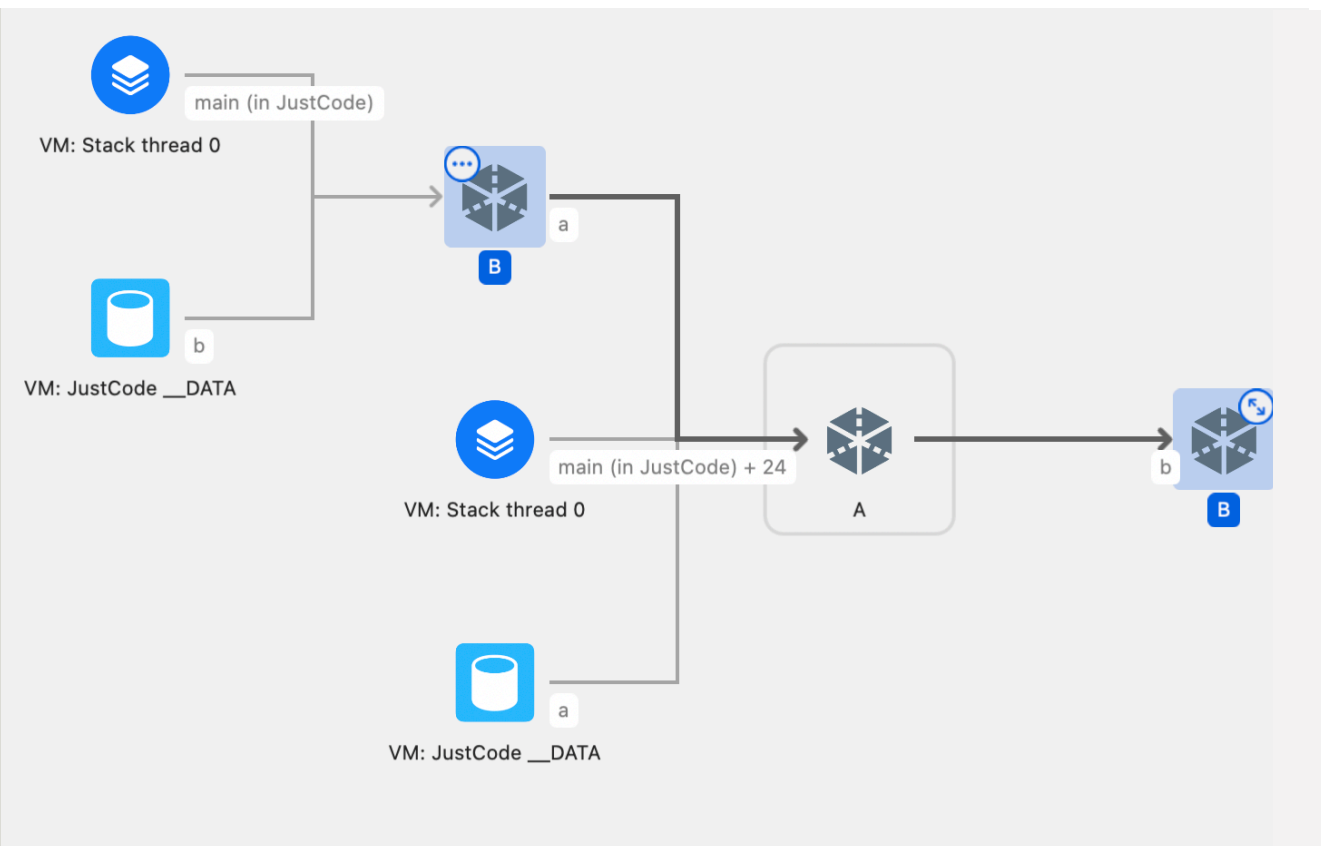
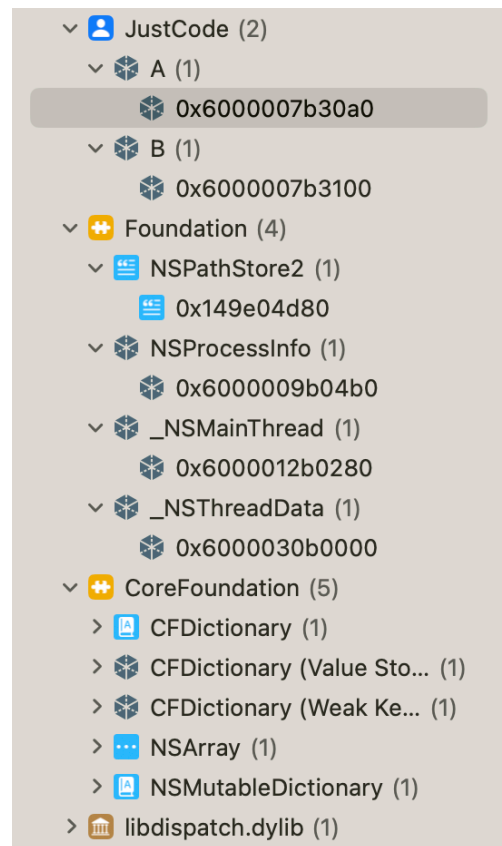
Backtrace

- 0 _malloc_zone_malloc_instrumented_or_legacy
- 3 A.__allocating_init()
- 4 doSomething()
- 5 main
- 6 start
- 7 0xffffffffffffffff

A pink arrow points to the 'doSomething()' entry in the backtrace.

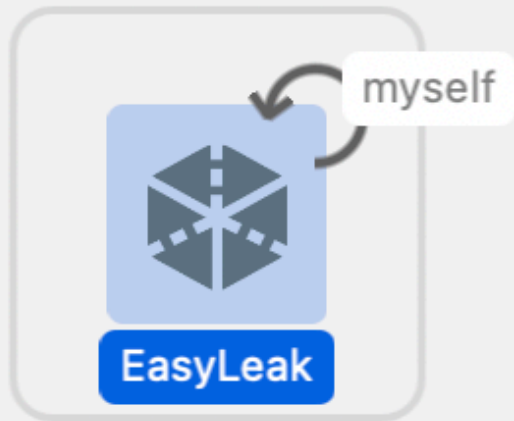
Собираем граф памяти, но...

```
class A: GodObject {  
    var b: B?  
}  
  
class B {  
    var a: A?  
}  
  
//func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.a = a  
//}  
  
//doSomething()  
RunLoop.current.run()
```




**Сколько минимум объектов
необходимо для утечки?**


1






Делаем плохо

```
class EasyLeak {  
    var myself: EasyLeak?  
}  
  
let easyLeak = EasyLeak()  
easyLeak.myself = easyLeak
```

▼  JustCode (1)

▼  EasyLeak (1)

 0x6000002033e0 

▼  Foundation (5)

Делаем хорошо

```
class EasyLeak {  
    var myself: EasyLeak?  
}
```

```
let easyLeak = EasyLeak()  
easyLeak.myself = easyLeak
```

```
class NoLeak {  
    weak var myself: NoLeak?  
}
```

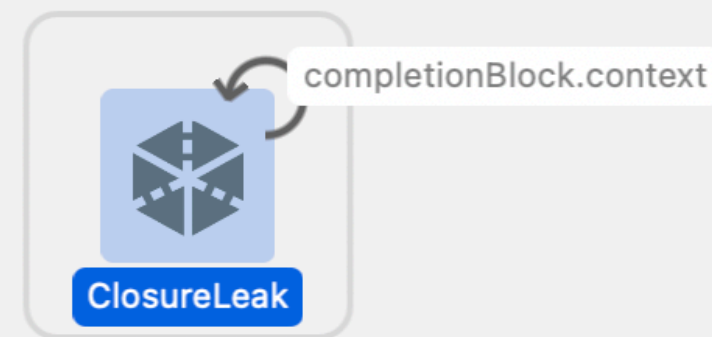
```
let noLeak = NoLeak()  
noLeak.myself = noLeak
```

Утечка через closure

```
class ClosureLeak {  
    var completionBlock: () -> Void = {}  
  
    init() {  
        self.completionBlock = {  
            print(self)  
        }  
    }  
}
```

```
func doClosureLeak() {  
    let closureLeak = ClosureLeak()  
}
```

```
doClosureLeak()
```

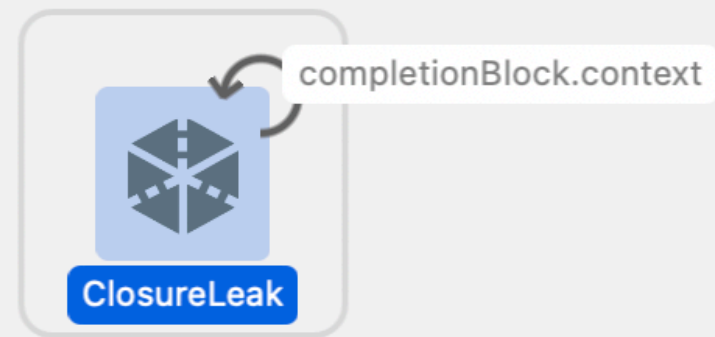


Утечка через closure

```
class ClosureLeak {  
    var completionBlock: () -> Void = {}  
  
    init() {  
        self.completionBlock = {  
            print(self)  
        }  
    }  
}
```

```
func doClosureLeak() {  
    let closureLeak = ClosureLeak()  
}
```

```
doClosureLeak()
```

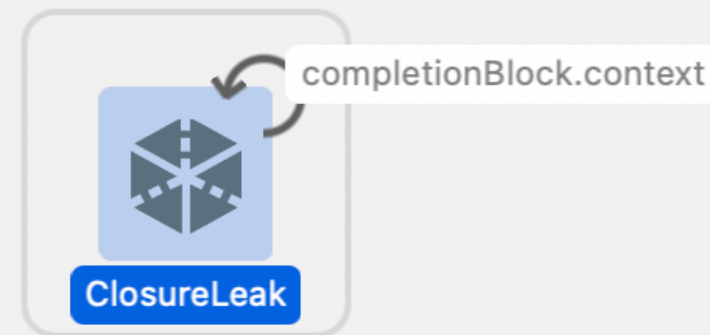


Утечка через closure

```
class ClosureLeak {  
    var completionBlock: () -> Void = {}  
  
    init() {  
        self.completionBlock = {  
            print(self)  
        }  
    }  
}
```

```
func doClosureLeak() {  
    let closureLeak = ClosureLeak()  
}
```

```
doClosureLeak()
```

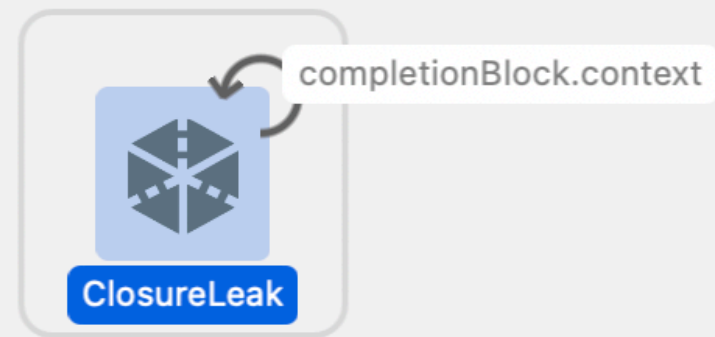


Утечка через closure

```
class ClosureLeak {  
    var completionBlock: () -> Void = {}  
  
    init() {  
        self.completionBlock = {  
            print(self)  
        }  
    }  
}
```

```
func doClosureLeak() {  
    let closureLeak = ClosureLeak()  
}
```

```
doClosureLeak()
```



Слабый захват в closure

```
init() {  
    self.completionBlock = {  
        print(self)  
    }  
}
```

Слабый захват в closure

```
init() {  
    self.completionBlock = { [weak self] in  
        guard let self else { return }  
        print(self)  
    }  
}
```


Хранение в статическом поле

```
class A: GodObject {  
    var b: B?  
}
```

```
class B {  
    static var a: A?  
}
```

```
func doSomething() {  
    let a = A()  
    B.a = a  
}
```

```
doSomething()
```

JustCode PID 75280

- CPU 0%
- Memory 4,2 MB
- Energy Impact Zero
- Disk Zero KB/s
- Network Zero KB/s

JustCode (1)

- A (1)
- 0x151f05590
- <unknown> (55)
- non-object in zone D... (40)
- 0x60000148c000
- 0x60000198c000
- 0x600002f8c040
- 0x60000348c000
- 0x60000348c030
- 0x600003888020
- 0x600003a8c000
- 0x600003a8c020
- 0x600003a8c040
- 0x600003a8c060
- Load More...
- non-object in zone Ma... (15)

JustCode > JustCode > A > 0x151f05590

VM: JustCode __DATA

static B.a

A

Object

- Class Name A
- Kind Swift
- Address 0x151f05590
- Size 32 bytes
- Malloc Zone MallocStackLoggingLit...

Hierarchy

- A
- GodObject
- _TtCs12_SwiftObject

Backtrace

- 0 _malloc_zone_malloc_instrum...
- 3 A.__allocating_init()
- 4 doSomething()
- 5 main
- 6 start
- 7 0xffffffffffffffff

Хранение в статическом поле



Не храните в глобальном поле объекты, чьё время жизни не равно времени жизни приложения



Что
разобрали?

Что
разобрали?



Цикл сильных ссылок

Что разобрали?



Цикл сильных ссылок

- Захват `self`
в `@escaping` closure

Что разобрали?



Цикл сильных ссылок

- Захват `self`
в `@escaping` closure



Удержание
в глобальных переменных

Что разобрали?



Цикл сильных ссылок

- Захват `self` в `@escaping` closure



Удержание в глобальных переменных

- Хранение в статическом поле

Алгоритм поиска (простая реализация)

Алгоритм поиска (простая реализация)

- 1 Собираем Граф Памяти

Алгоритм поиска (простая реализация)

- 1 Собираем Граф Памяти
- 2 Находим утёкший объект

Алгоритм поиска (простая реализация)

- 1 Собираем Граф Памяти
- 2 Находим утёкший объект
- 3 Идём по чёрным стрелкам

Алгоритм поиска (простая реализация)

- 1 Собираем Граф Памяти
- 2 Находим утёкший объект
- 3 Идём по чёрным стрелкам
- 4 Находим место, где создаётся цикл сильных ссылок

Алгоритм поиска (простая реализация)

- 1 Собираем Граф Памяти
- 2 Находим утёкший объект
- 3 Идём по чёрным стрелкам
- 4 Находим место, где создаётся цикл сильных ссылок
- 5 Разрываем цикл

Алгоритм поиска (простая реализация)

- 1 Собираем Граф Памяти
- 2 Находим утёкший объект
- 3 Идём по чёрным стрелкам
- 4 Находим место, где создаётся цикл сильных ссылок
- 5 Разрываем цикл
- 6 Шикуюем на пляже



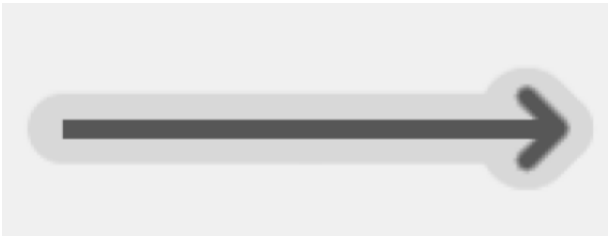
Кто теперь умеет бороться
с утечками?

Ловушки Xcode

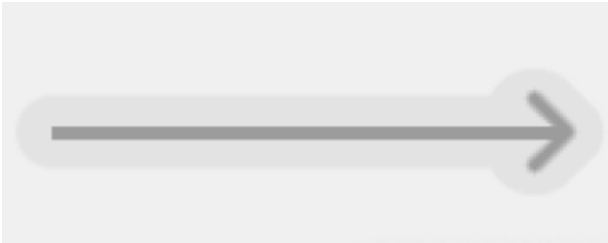
Легенда карты (Memory Graph)

Стрелка в графе

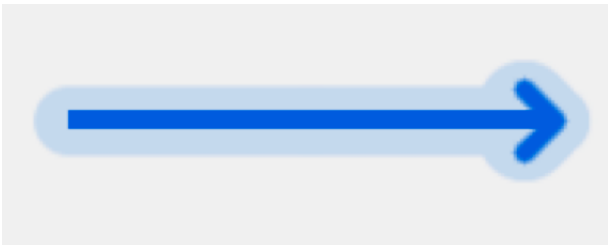
Связь в рантайме



Strong



Weak



Выбранная связь в дебаггере

Неверная стрелка: статическое поле

```
class A: GodObject {  
    var b: B?  
}
```

```
class B {  
    static var a: A?  
}
```

```
func doSomething() {  
    let a = A()  
    B.a = a  
}
```

```
doSomething()
```

JustCode PID 75280

- CPU 0%
- Memory 4,2 MB
- Energy Impact Zero
- Disk Zero KB/s
- Network Zero KB/s

JustCode (1)

- A (1)
- 0x151f05590
- <unknown> (55)
- non-object in zone D... (40)
- 0x60000148c000
- 0x60000198c000
- 0x600002f8c040
- 0x60000348c000
- 0x60000348c030
- 0x600003888020
- 0x600003a8c000
- 0x600003a8c020
- 0x600003a8c040
- 0x600003a8c060
- Load More...
- non-object in zone Ma... (15)

JustCode > JustCode > A > 0x151f05590

VM: JustCode __DATA

static B.a

A

Object

- Class Name A
- Kind Swift
- Address 0x151f05590
- Size 32 bytes
- Malloc Zone MallocStackLoggingLit...

Hierarchy

- A
- GodObject
- _TtCs12_SwiftObject

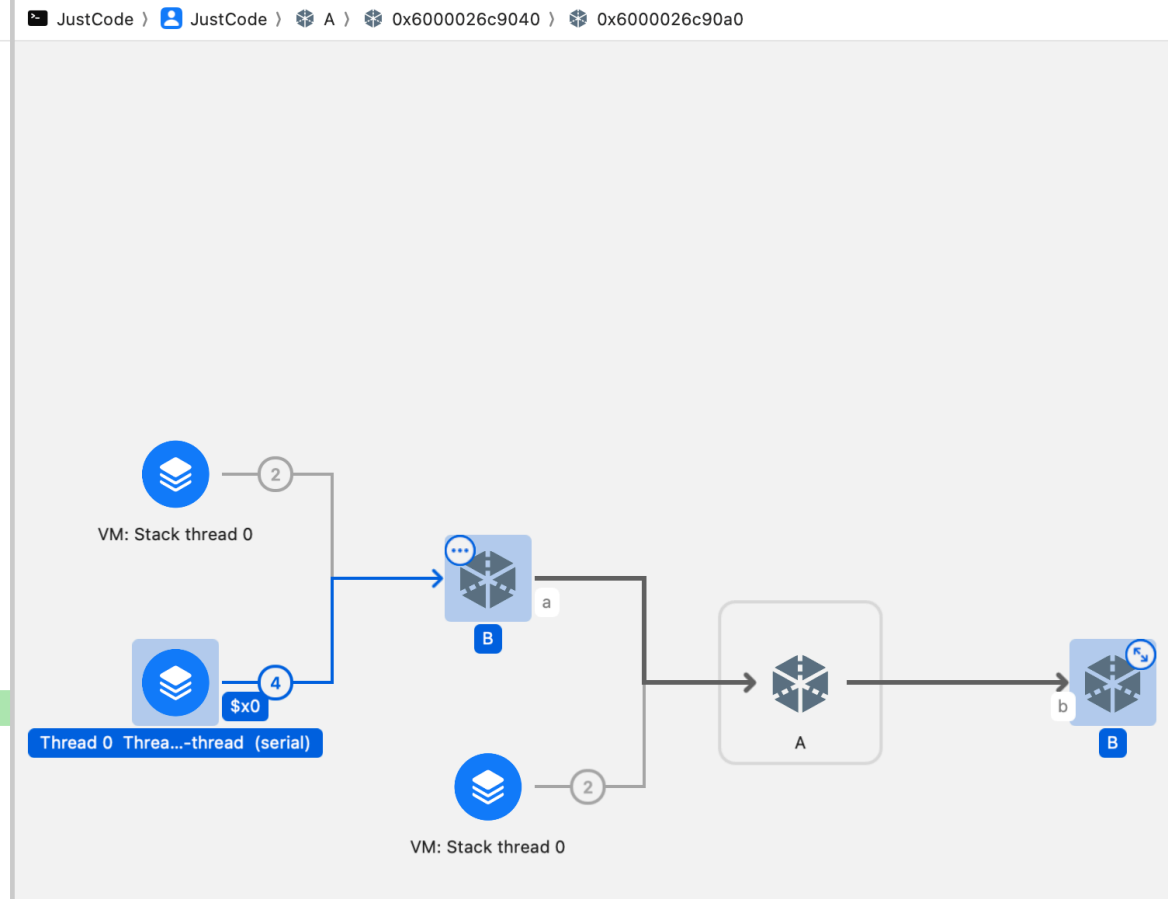
Backtrace

- 0 _malloc_zone_malloc_instrum...
- 3 A.__allocating_init()
- 4 doSomething()
- 5 main
- 6 start
- 7 0xffffffffffffffff

Неверная стрелка: Unowned

```
JustCode > JustCode > main > No Selection  
114  
115  
116  
117  
118  
119 class A: GodObject {  
120     unowned var b: B?  
121 }  
122  
123 class B {  
124     var a: A?  
125 }  
126  
127 func doSomething() {  
128     let a = A()  
129     let b = B()  
130     a.b = b  
131     b.a = a  
132 }  
133  
134 doSomething()  
135  
136 assert(A.objectsCount == 0)  
137
```

Thread 1: step over



Object

- Class Name B
- Kind Swift
- Address 0x6000026c90a0
- Size 32 bytes
- Malloc Zone DefaultMallocZone

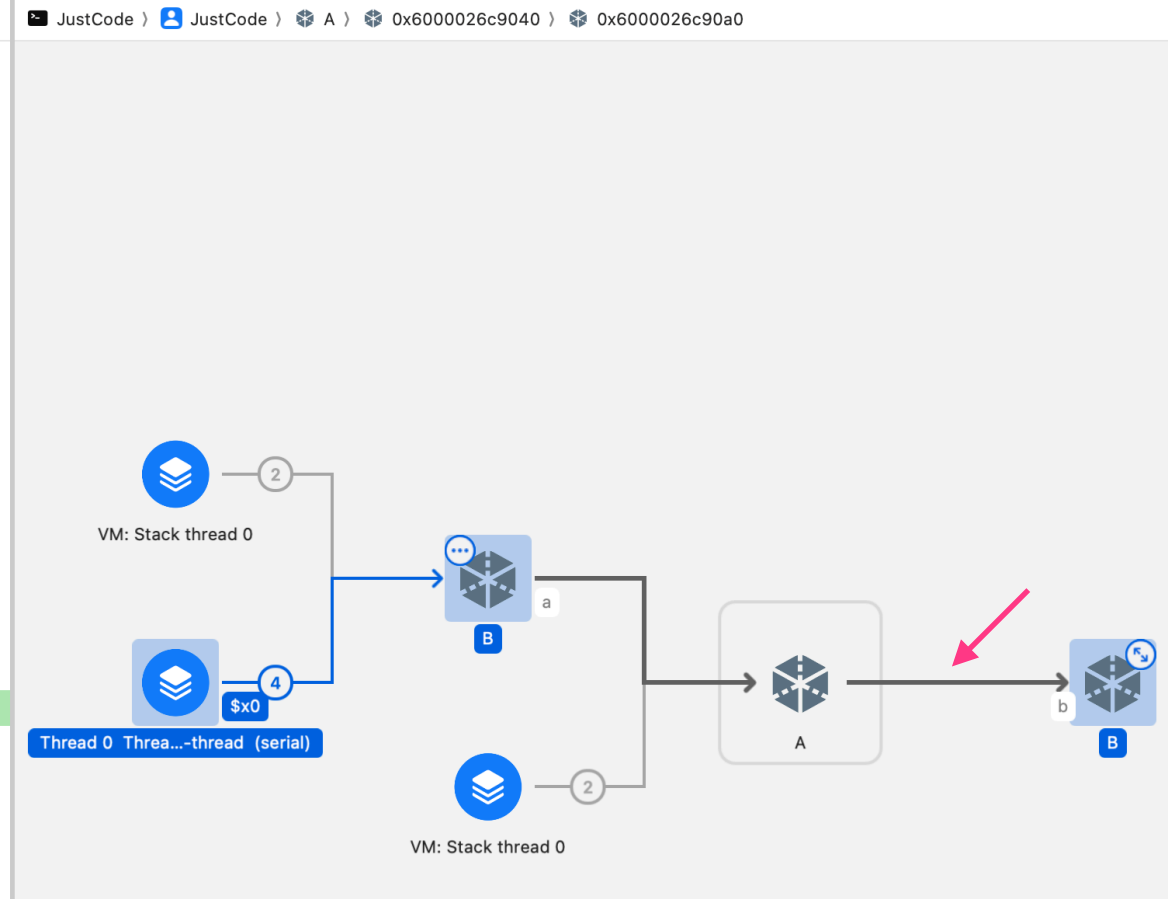
Hierarchy

- B
- _TtCs12_SwiftObject

Backtrace

Неверная стрелка: Unowned

```
JustCode > JustCode > main > No Selection  
114  
115  
116  
117  
118  
119 class A: GodObject {  
120     unowned var b: B?  
121 }  
122  
123 class B {  
124     var a: A?  
125 }  
126  
127 func doSomething() {  
128     let a = A()  
129     let b = B()  
130     a.b = b  
131     b.a = a  
132 }  
133  
134 doSomething()  
135  
136 assert(A.objectsCount == 0)  
137
```



Object

- Class Name B
- Kind Swift
- Address 0x6000026c90a0
- Size 32 bytes
- Malloc Zone DefaultMallocZone

Hierarchy

- B
- _TtCs12_SwiftObject

Backtrace

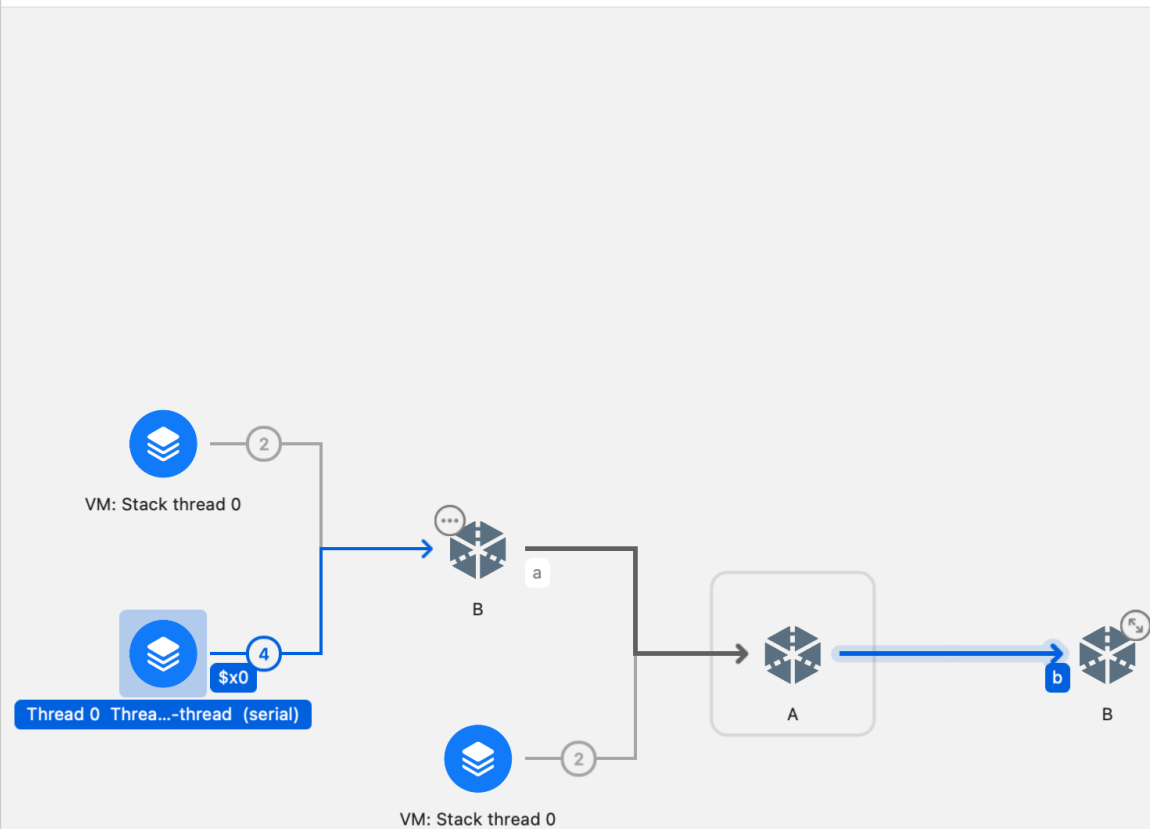
Неверная стрелка: Unowned

JustCode > JustCode > main > No Selection

```
114
115
116
117
118
119 class A: GodObject {
120     unowned var b: B?
121 }
122
123 class B {
124     var a: A?
125 }
126
127 func doSomething() {
128     let a = A()
129     let b = B()
130     a.b = b
131     b.a = a
132 }
133
134 doSomething()
135
136 assert(A.objectsCount == 0)
137
```

Thread 1: step over

JustCode > JustCode > A > 0x6000026c9040 > → b



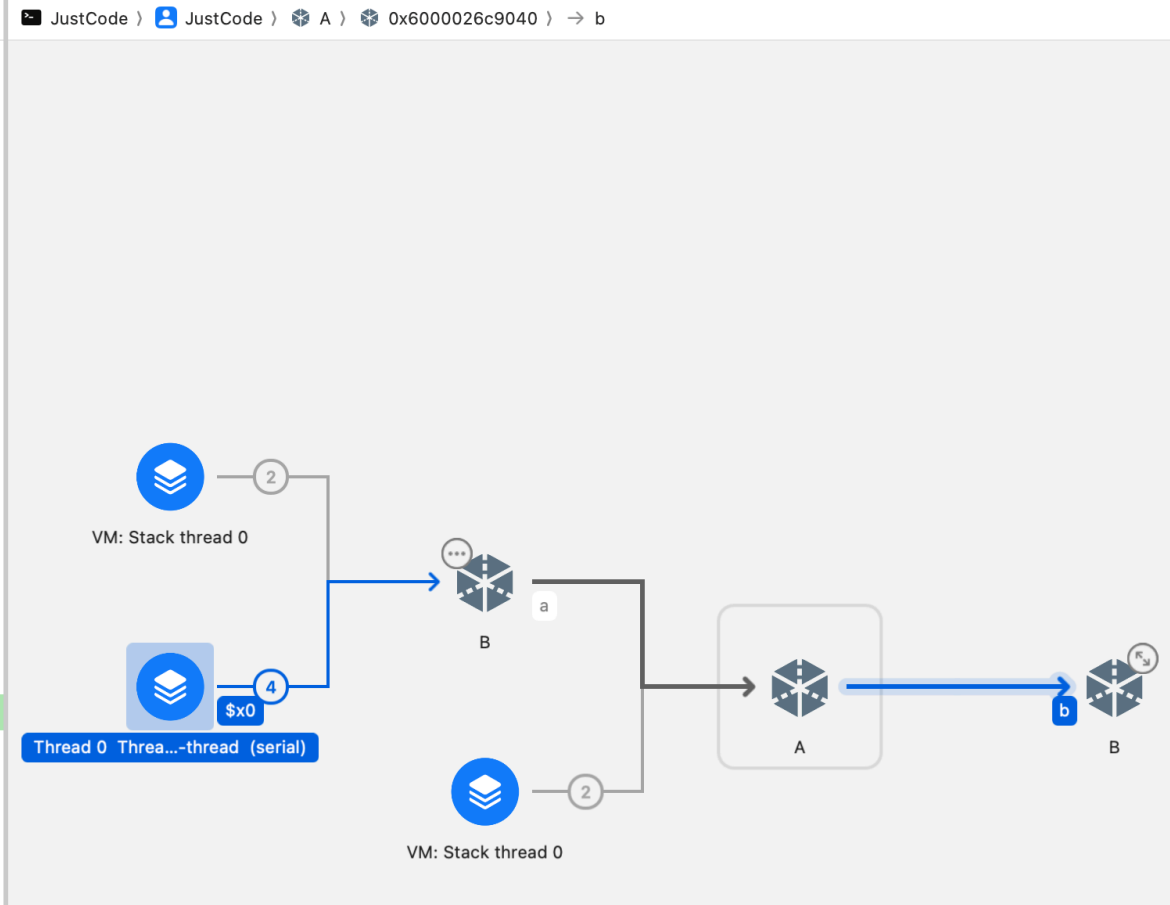
Reference Details

Name b
Type unowned
Source <A 0x6000026c9040> [32] +16
Destination <B 0x6000026c90a0> [32] +0

Неверная стрелка: Unowned

```
JustCode > JustCode > main > No Selection  
114  
115  
116  
117  
118  
119 class A: GodObject {  
120     unowned var b: B?  
121 }  
122  
123 class B {  
124     var a: A?  
125 }  
126  
127 func doSomething() {  
128     let a = A()  
129     let b = B()  
130     a.b = b  
131     b.a = a  
132 }  
133  
134 doSomething()  
135  
136 assert(A.objectsCount == 0)  
137
```

Thread 1: step over

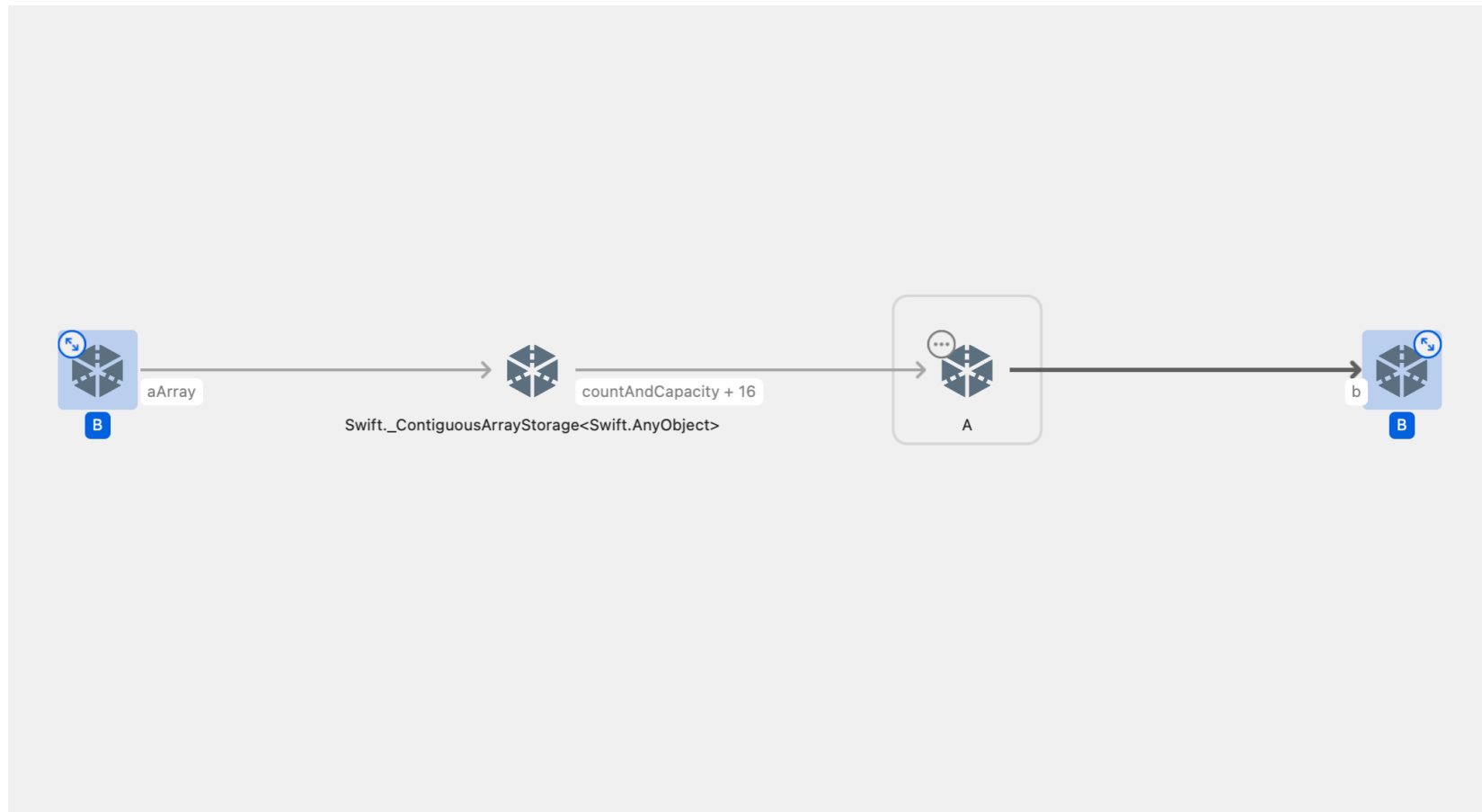


Reference Details

Name b
Type unowned ←
Source <A 0x6000026c9040> [32] +16
Destination <B 0x6000026c90a0> [32] +0

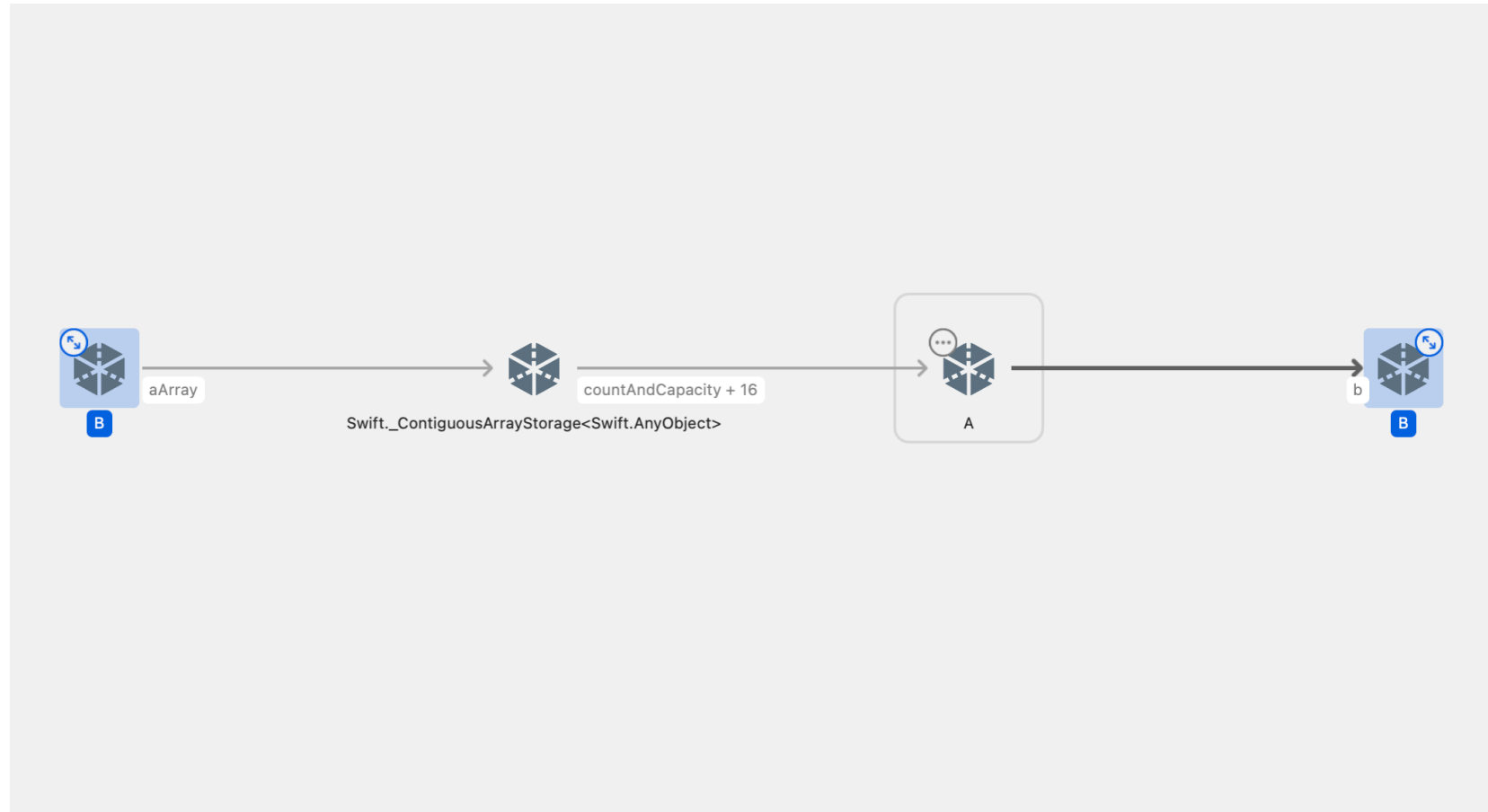
Неверная стрелка: реализация Swift коллекций

```
class A: GodObject {  
    var b: B?  
}  
  
class B {  
    var aArray: [A] = []  
}  
  
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.aArray = [a]  
}  
  
doSomething()
```



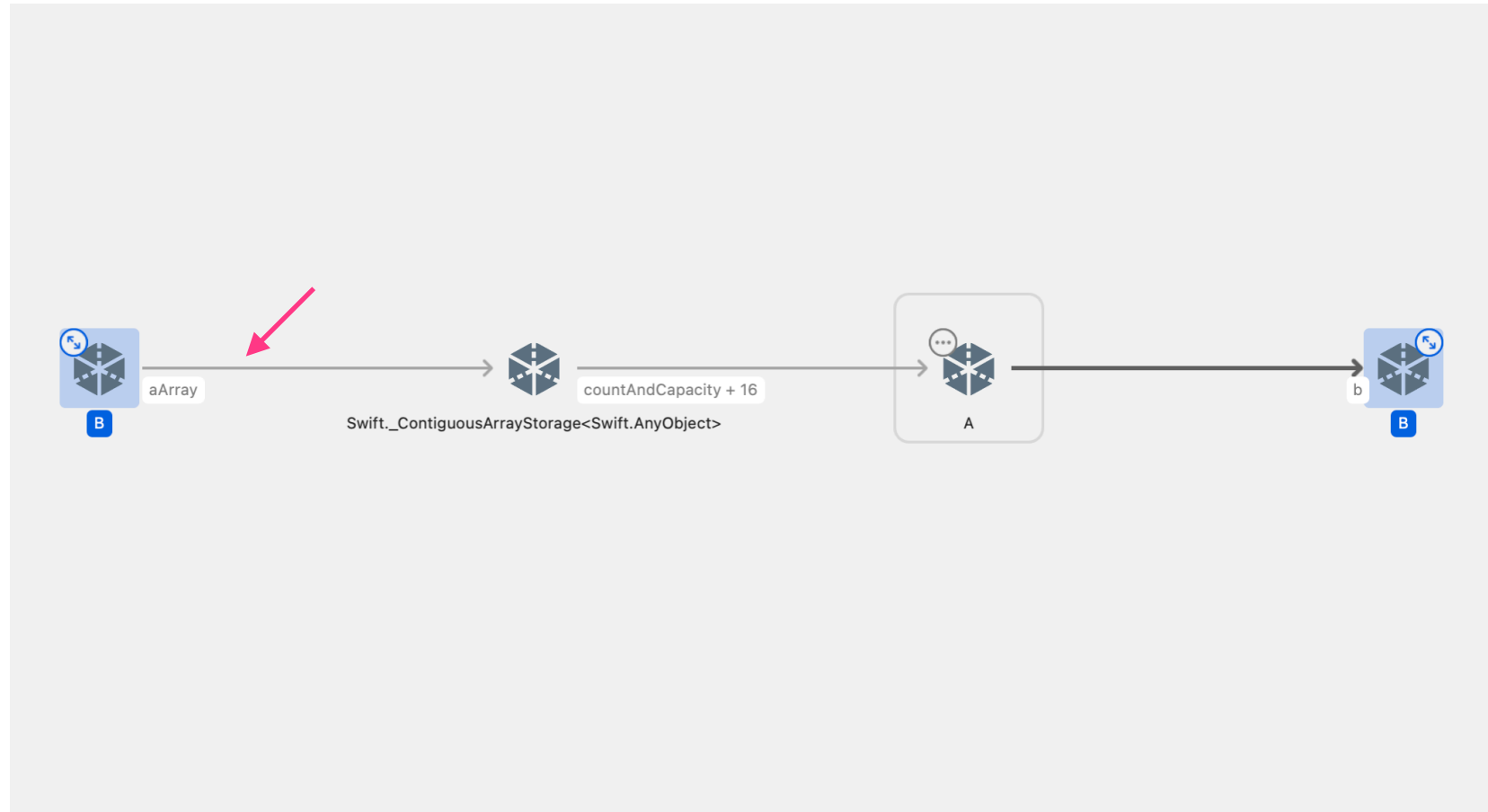
Неверная стрелка: реализация Swift коллекций

```
class A: GodObject {  
    var b: B?  
}  
  
class B {  
    var aArray: [A] = []  
}  
  
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.aArray = [a]  
}  
  
doSomething()
```



Неверная стрелка: реализация Swift коллекций

```
class A: GodObject {  
    var b: B?  
}  
  
class B {  
    var aArray: [A] = []  
}  
  
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.aArray = [a]  
}  
  
doSomething()
```



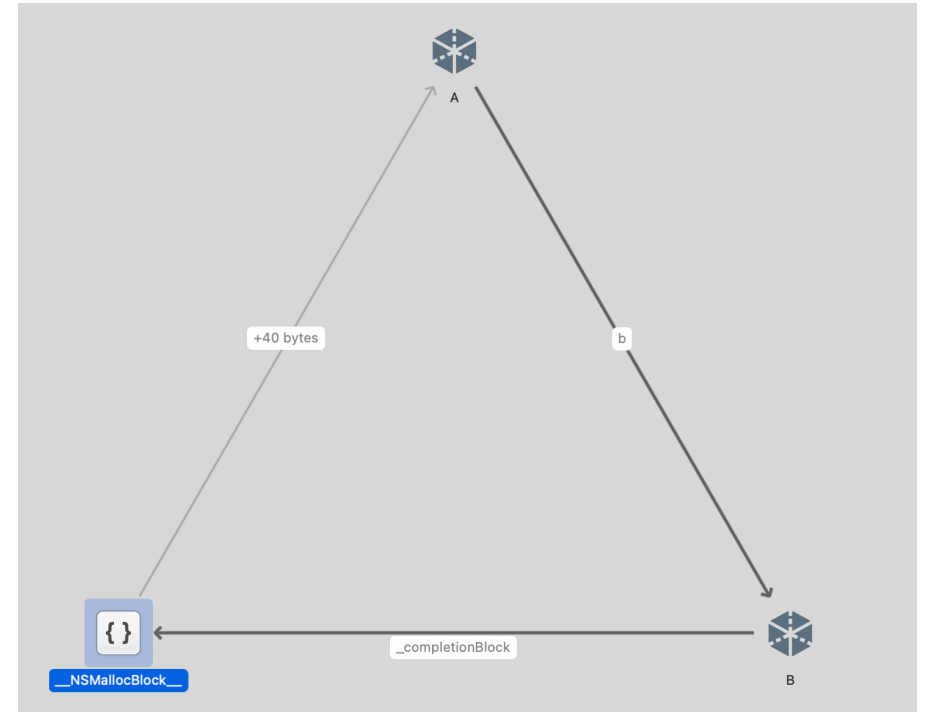
Неверная стрелка: Связь Objective C & Swift

```
@interface B : NSObject
@property (nonatomic) void (^completionBlock)(void);
- (instancetype)initWithBlock:(void (^)(void))block; ←
@end
```

```
class A: GodObject {
    var b: B?
}
```

```
func doLeak() {
    let a = A()
    let b = B { ←
        print(a)
    }
    a.b = b
}
```

```
doLeak()
```



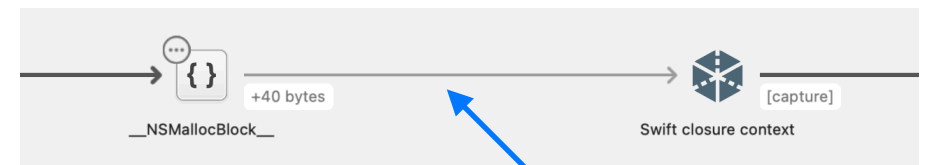
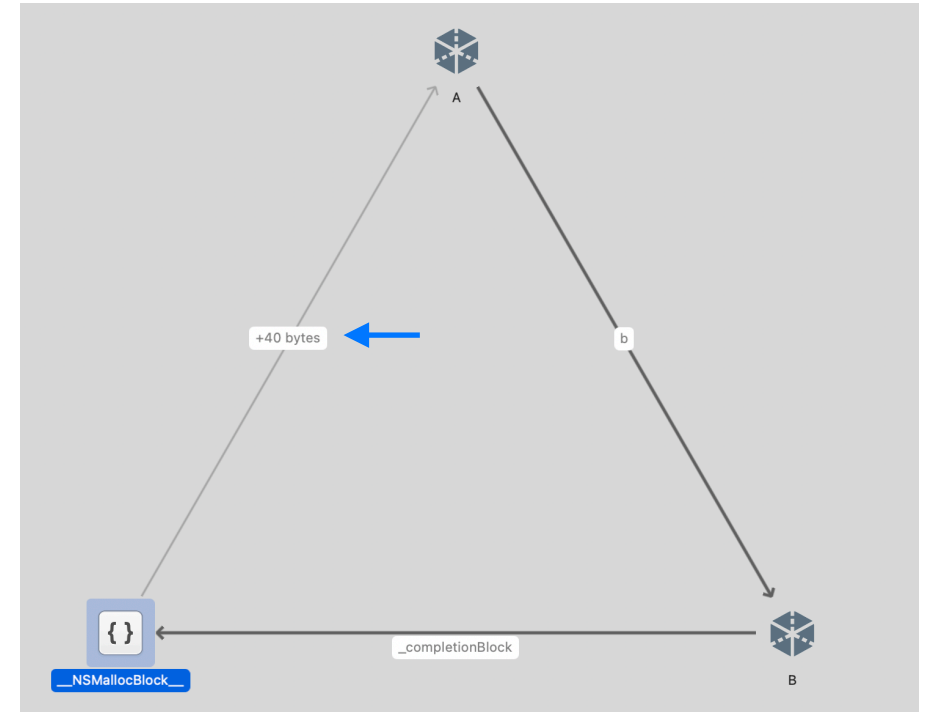
Неверная стрелка: Связь Objective C & Swift

```
@interface B : NSObject
@property (nonatomic) void (^completionBlock)(void);
- (instancetype)initWithBlock:(void (^)(void))block; ←
@end
```

```
class A: GodObject {
    var b: B?
}
```

```
func doLeak() {
    let a = A()
    let b = B { ←
        print(a)
    }
    a.b = b
}
```

```
doLeak()
```



Неверная стрелка: Протоколы

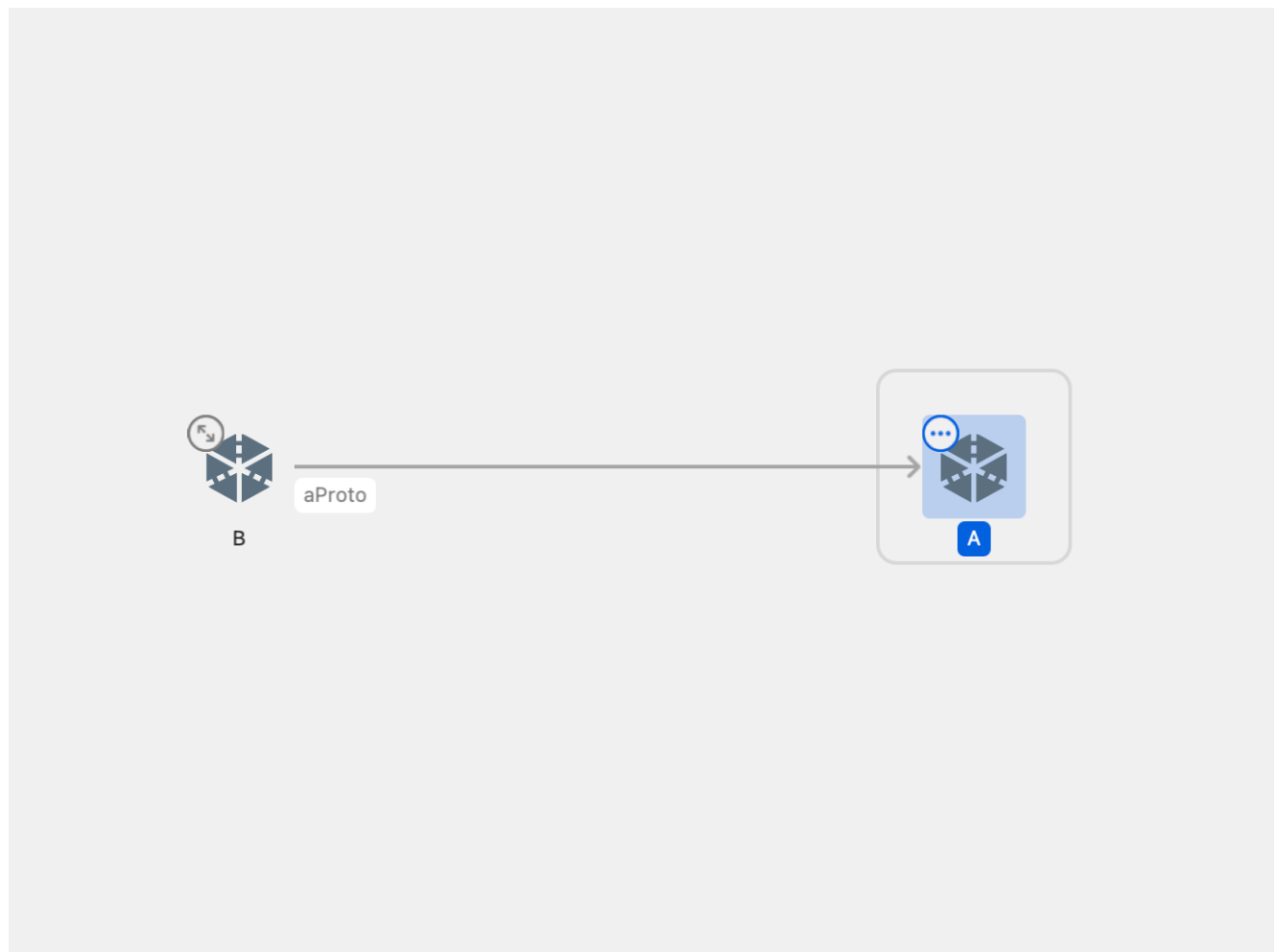
```
protocol ProtoA {}

class A: GodObject, ProtoA {
    var b: B?
}

class B {
    var aProto: ProtoA? = nil
}

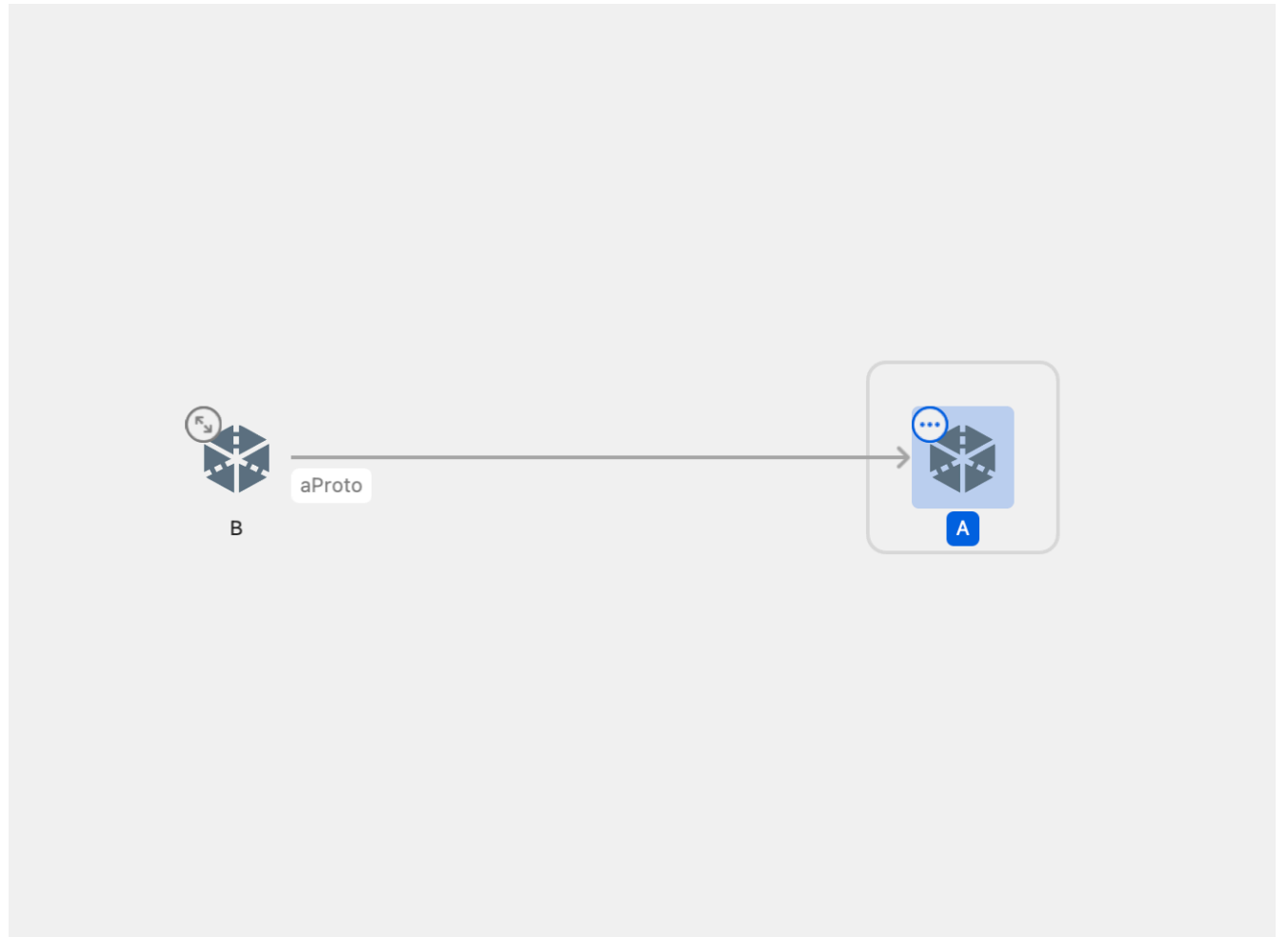
func doSomething() {
    let a = A()
    let b = B()
    a.b = b
    b.aProto = a
}

doSomething()
assert(A.objectsCount == 0)
```



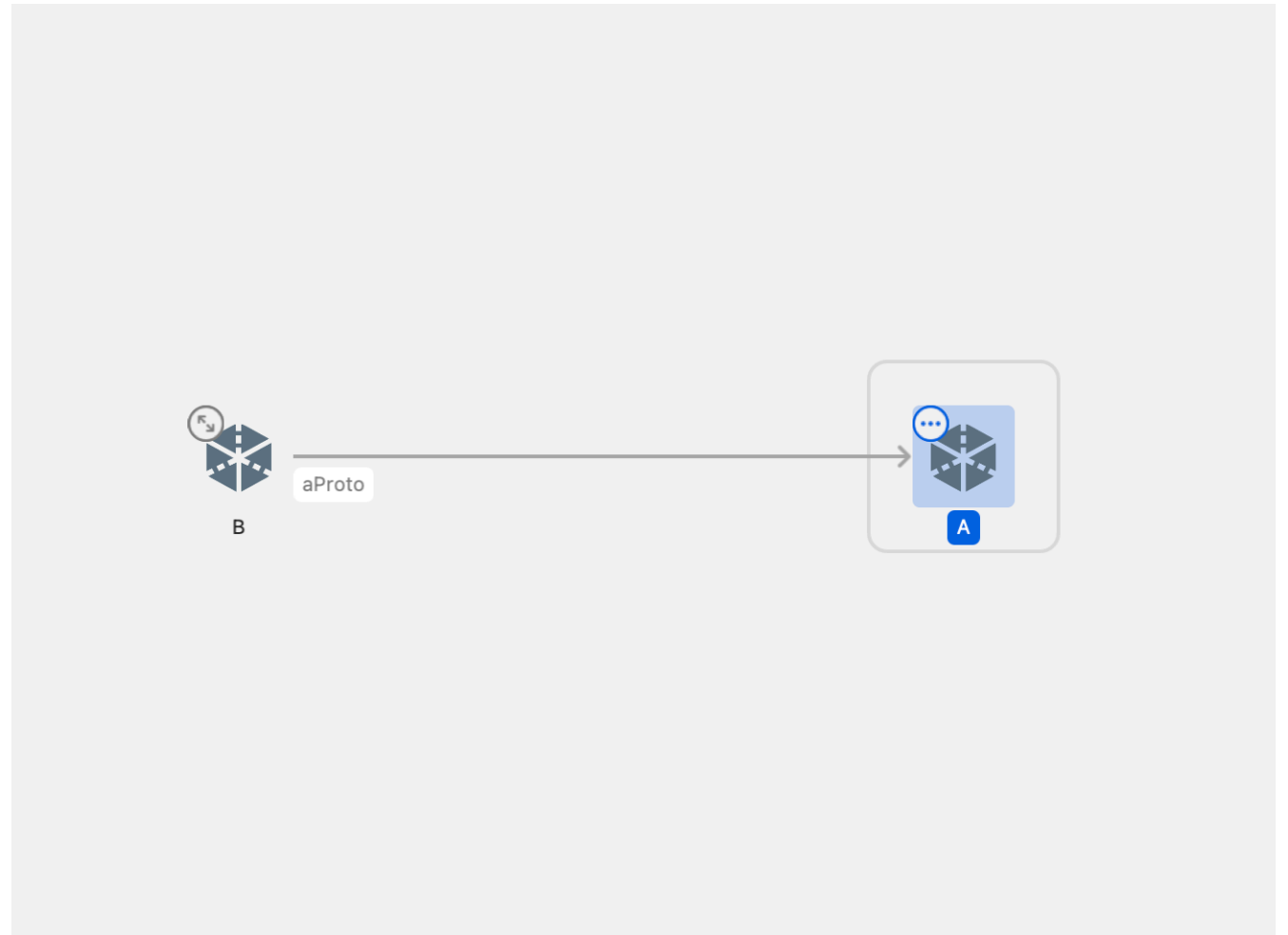
Неверная стрелка: Протоколы

```
protocol ProtoA {} ←  
  
class A: GodObject, ProtoA {  
    var b: B?  
}  
  
class B {  
    var aProto: ProtoA? = nil  
}  
  
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.aProto = a  
}  
  
doSomething()  
assert(A.objectsCount == 0)
```



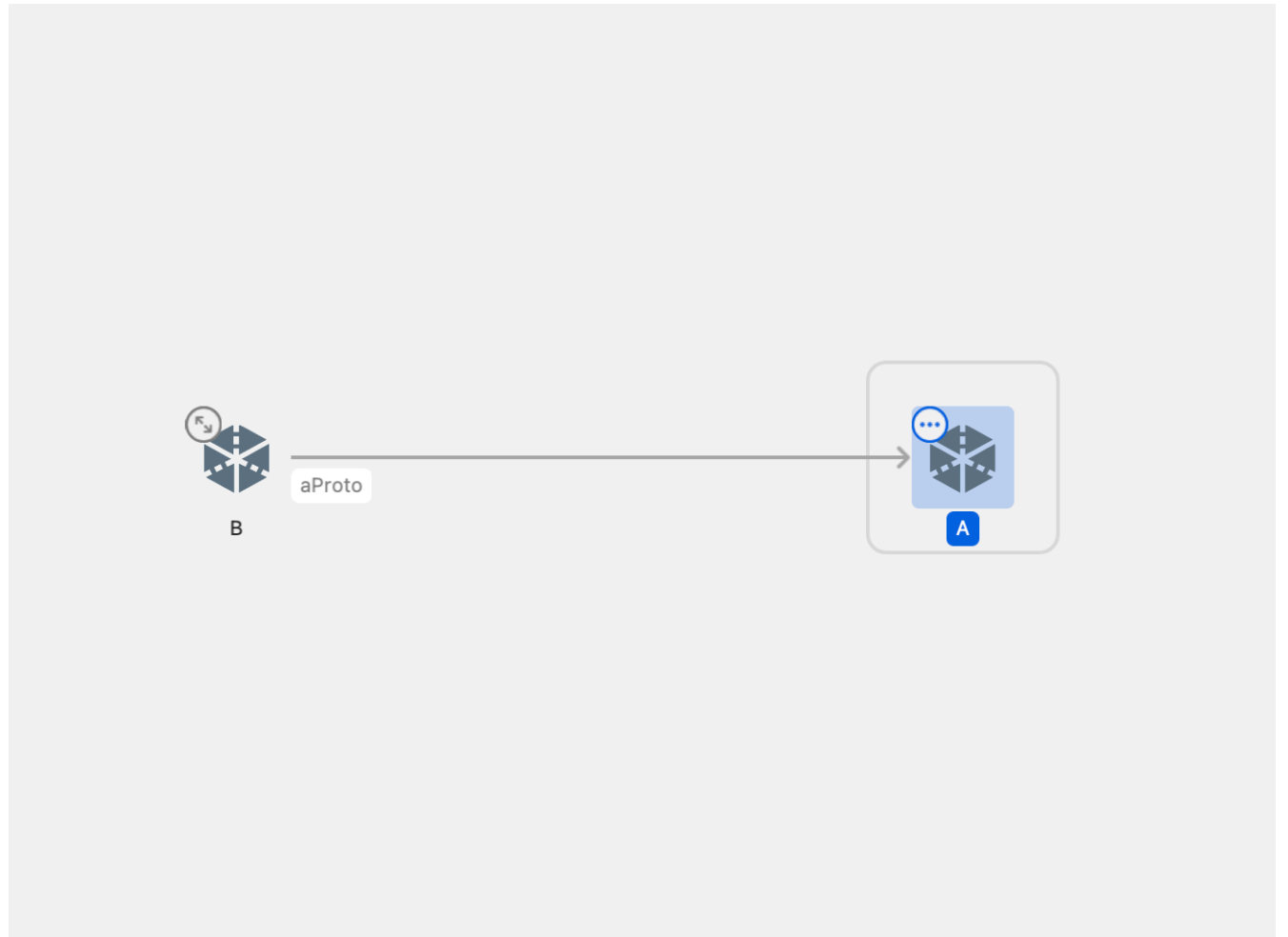
Неверная стрелка: Протоколы

```
protocol ProtoA {} ←  
  
class A: GodObject, ProtoA {  
    var b: B?  
}  
  
class B {  
    var aProto: ProtoA? = nil ←  
}  
  
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.aProto = a  
}  
  
doSomething()  
assert(A.objectsCount == 0)
```



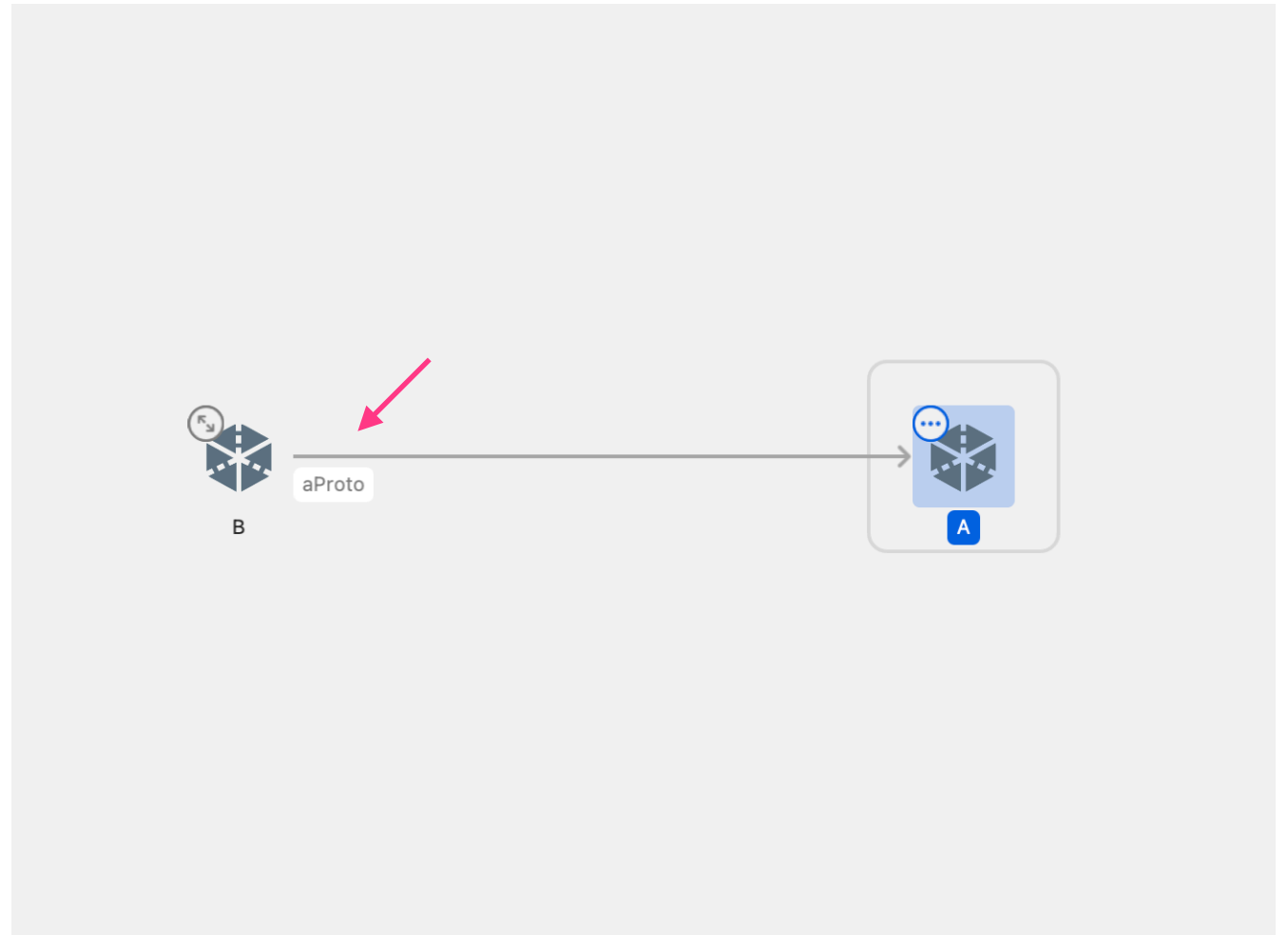
Неверная стрелка: Протоколы

```
protocol ProtoA {} ←  
  
class A: GodObject, ProtoA {  
    var b: B?  
}  
  
class B {  
    var aProto: ProtoA? = nil ←  
}  
  
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.aProto = a ←  
}  
  
doSomething()  
assert(A.objectsCount == 0)
```



Неверная стрелка: Протоколы

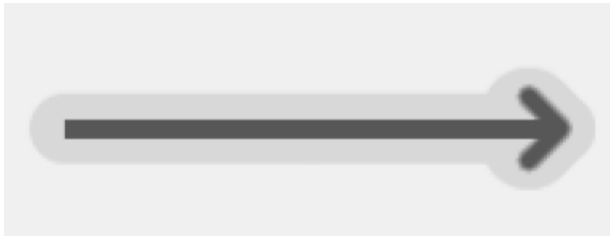
```
protocol ProtoA {} ←  
  
class A: GodObject, ProtoA {  
    var b: B?  
}  
  
class B {  
    var aProto: ProtoA? = nil ←  
}  
  
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.aProto = a ←  
}  
  
doSomething()  
assert(A.objectsCount == 0)
```



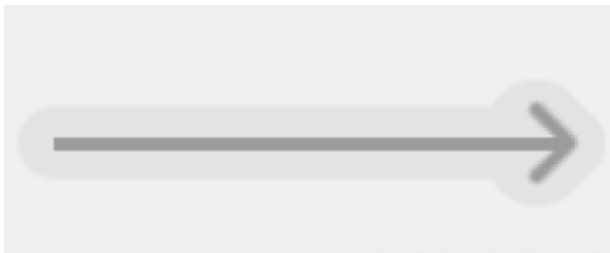
Легенда карты (Memory Graph)

Стрелка в графе

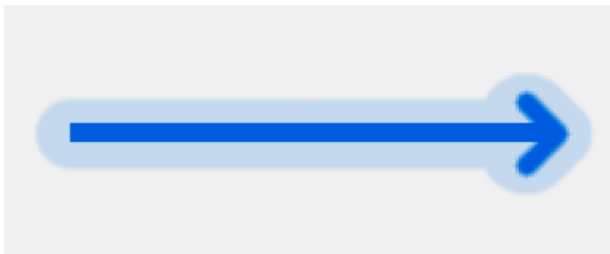
Связь в рантайме



Strong, Unowned



Weak,
Strong – статическое свойство,
Strong – Protocol,
Strong – реализация коллекций,
Strong – closure context,
Strong – интеропт Obj C & Swift



Выбранная связь в дебаггере

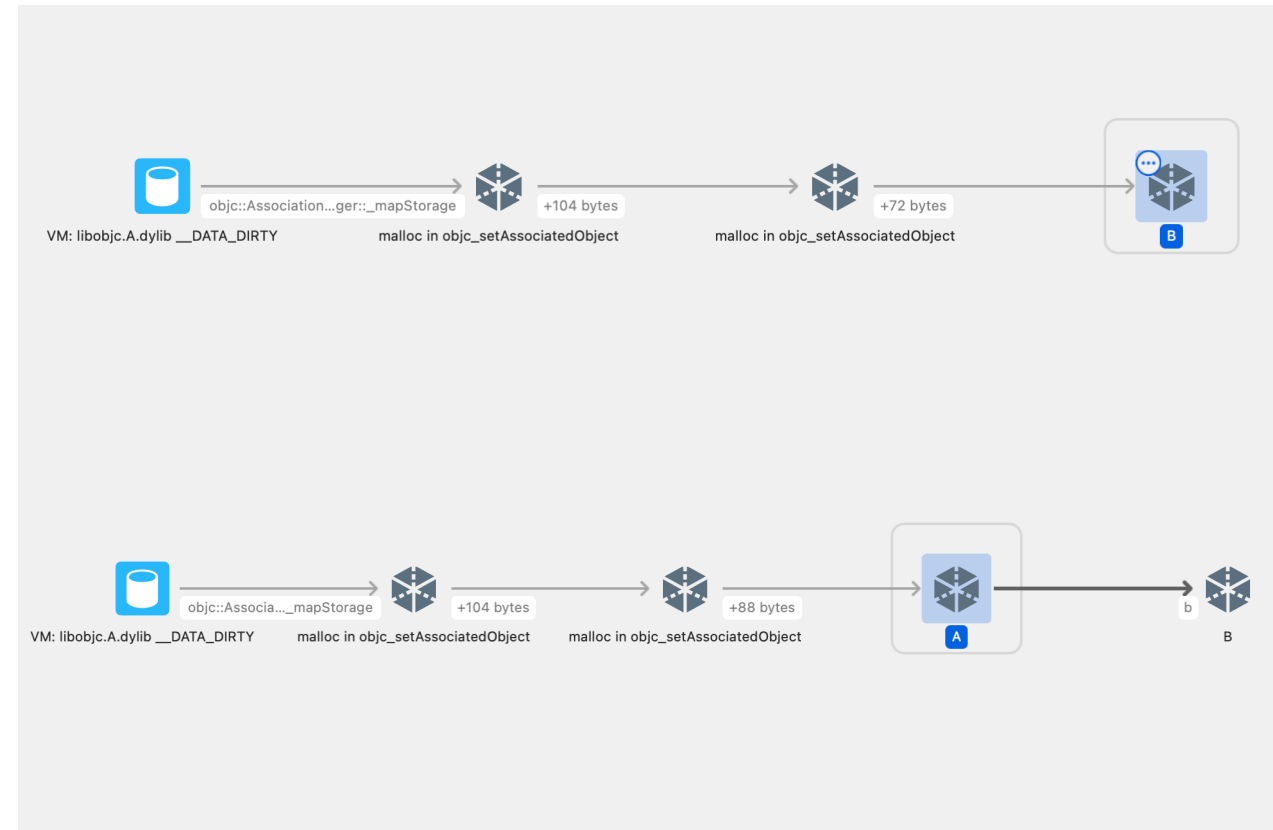
Самодельные хранимые переменные

```
class A: GodObject {  
    var b: B?  
}
```

```
class B {}
```

```
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.a = a  
}
```

```
doSomething()
```



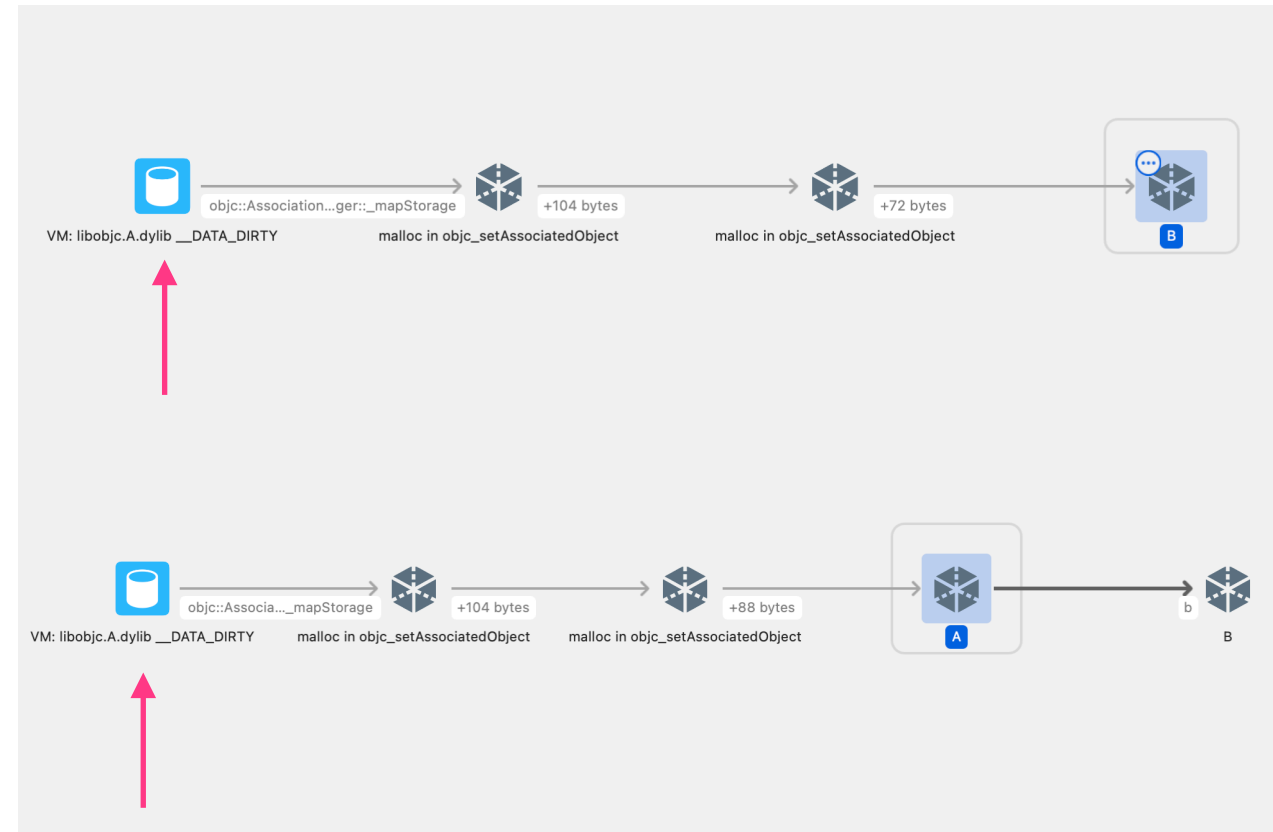
Самодельные хранимые переменные

```
class A: GodObject {  
    var b: B?  
}
```

```
class B {}
```

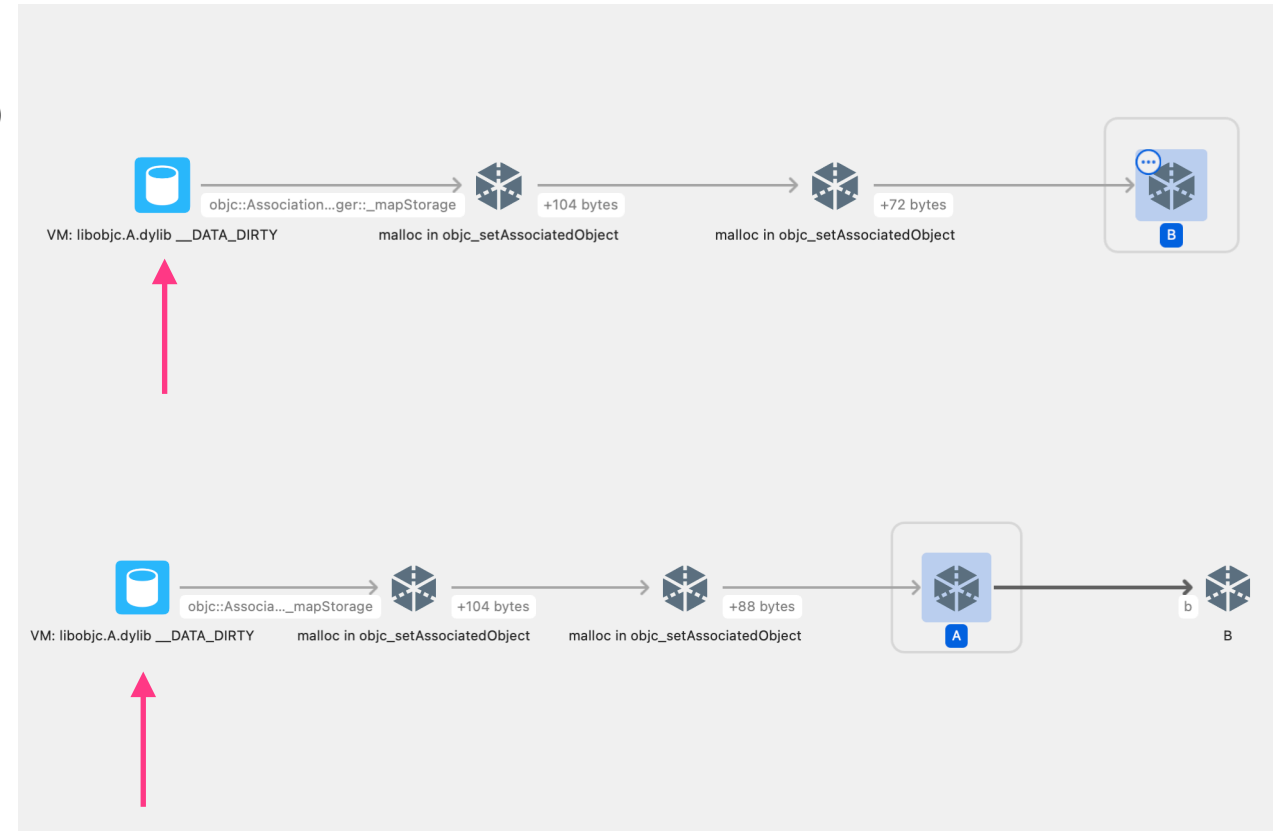
```
func doSomething() {  
    let a = A()  
    let b = B()  
    a.b = b  
    b.a = a  
}
```

```
doSomething()
```



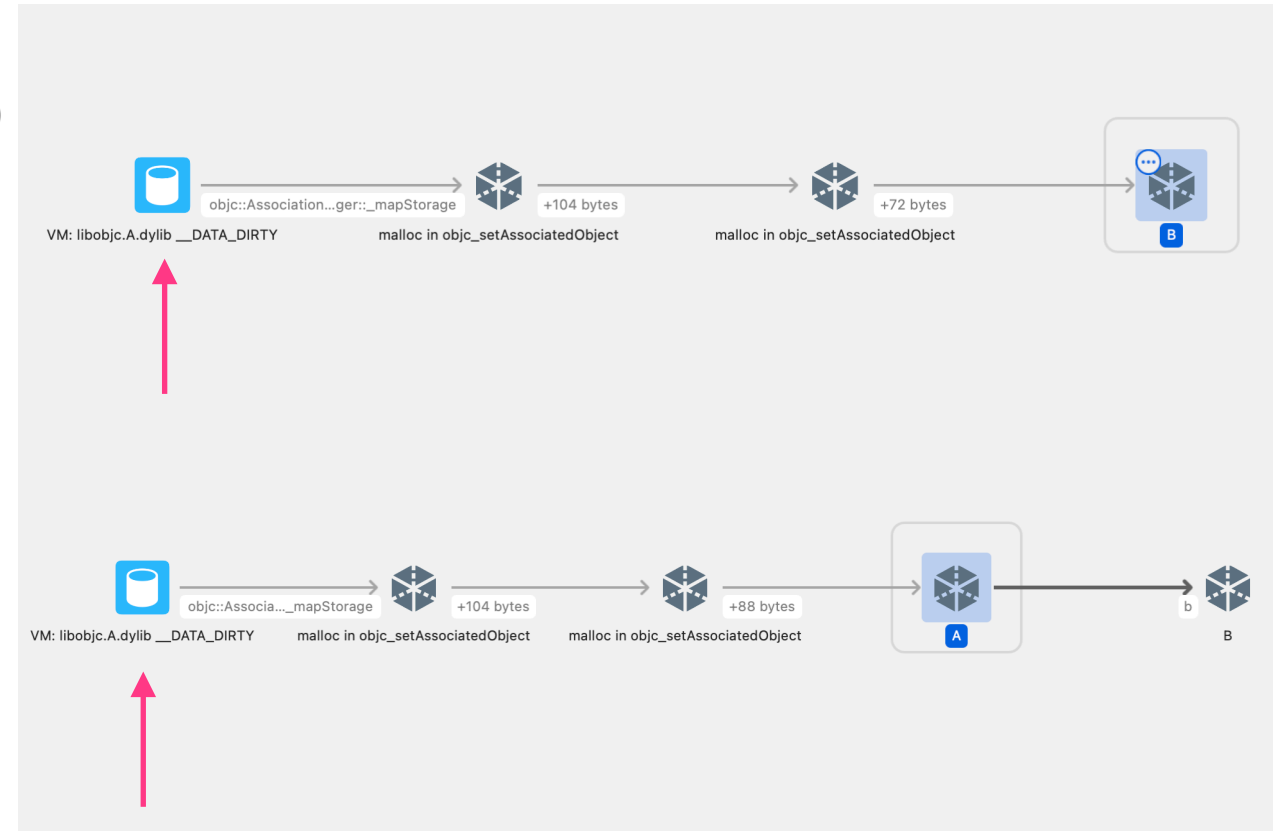
Самодельные хранимые переменные

```
extension B {  
  private static var sessionAssociatedObject: () = ()  
  var a: A? {  
    get {  
      objc_getAssociatedObject(  
        self,  
        &B.sessionAssociatedObject  
      ) as? A  
    }  
    set {  
      objc_setAssociatedObject(  
        self,  
        &B.sessionAssociatedObject,  
        newValue,  
        .OBJC_ASSOCIATION_RETAIN_NONATOMIC  
      )  
    }  
  }  
}
```



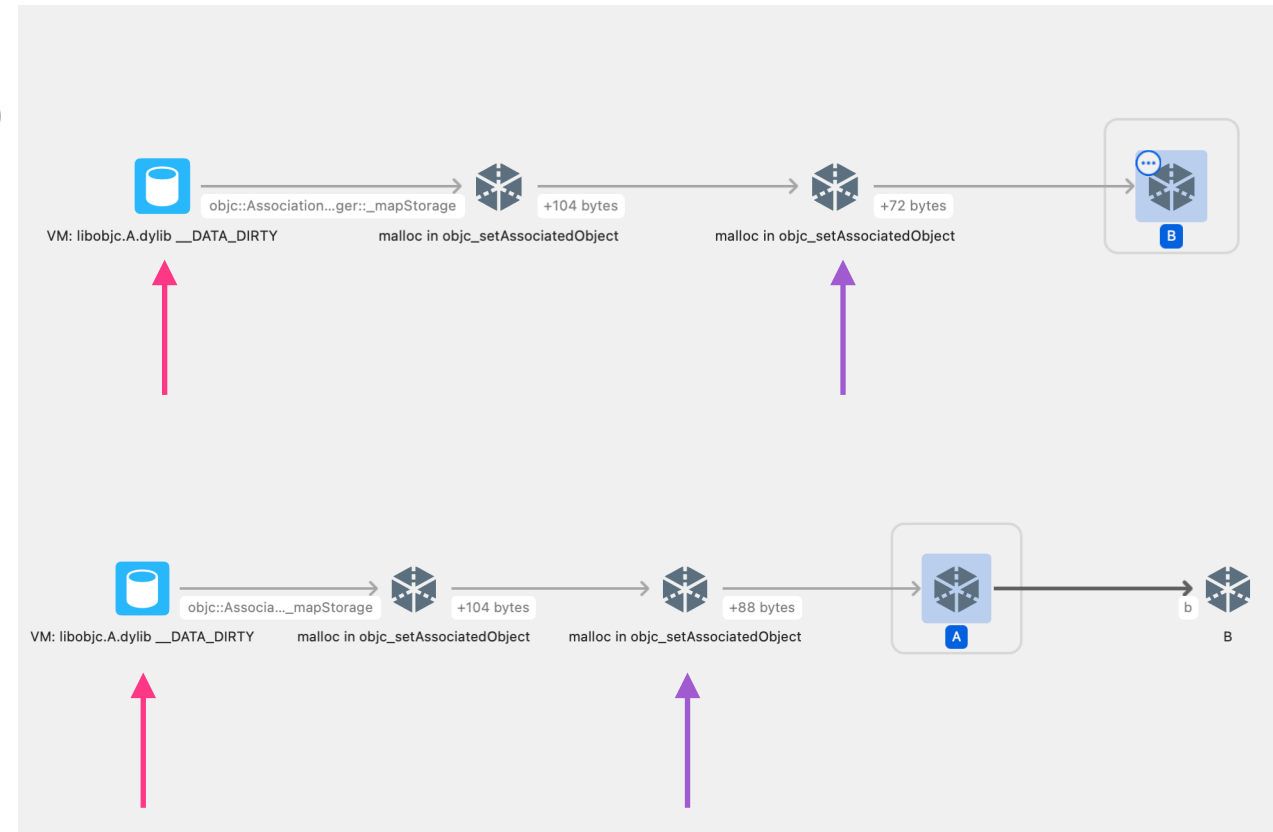
Самодельные хранимые переменные

```
extension B {  
  private static var sessionAssociatedObject: () = ()  
  var a: A? {  
    get {  
      objc_getAssociatedObject(  
        self,  
        &B.sessionAssociatedObject  
      ) as? A  
    }  
    set {  
      objc_setAssociatedObject(  
        self,  
        &B.sessionAssociatedObject,  
        newValue,  
        .OBJC_ASSOCIATION_RETAIN_NONATOMIC  
      )  
    }  
  }  
}
```



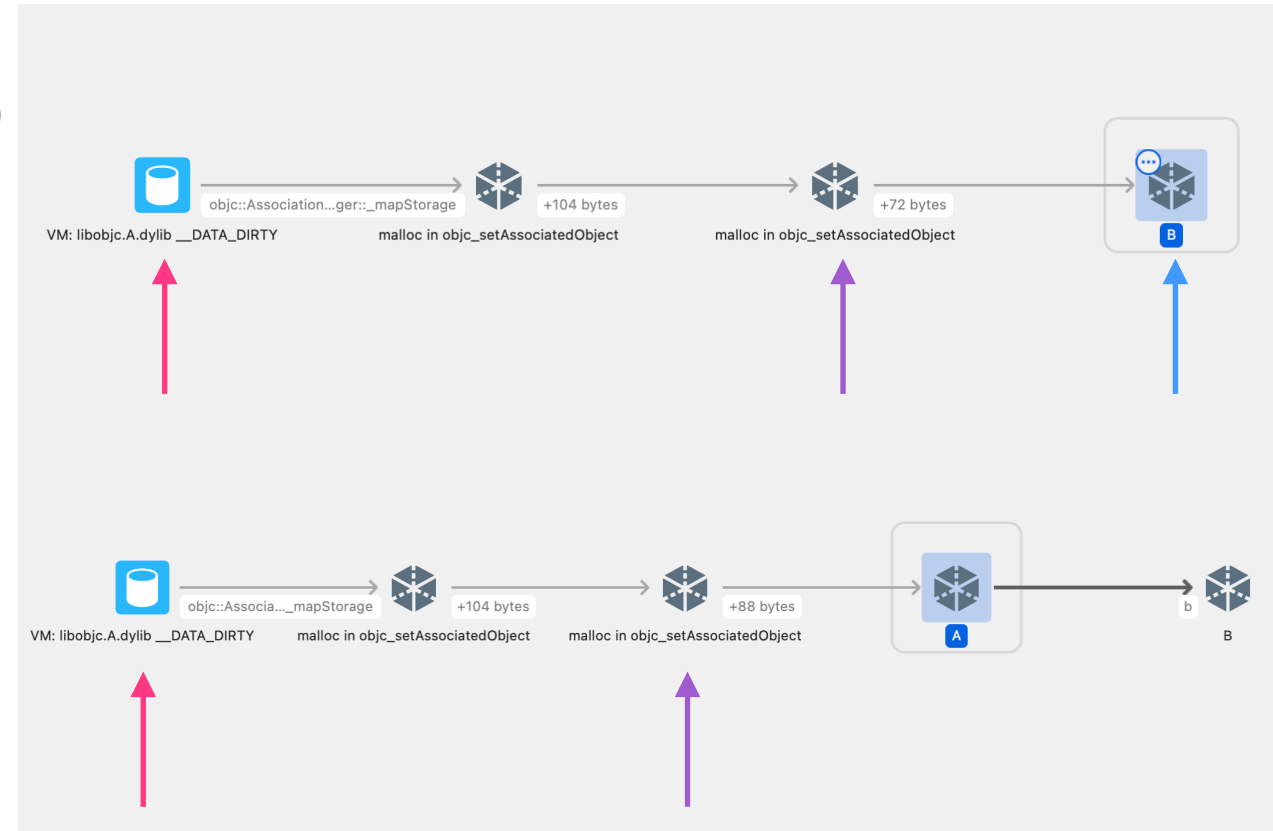
Самодельные хранимые переменные

```
extension B {  
  private static var sessionAssociatedObject: () = ()  
  var a: A? {  
    get {  
      objc_getAssociatedObject(  
        self,  
        &B.sessionAssociatedObject  
      ) as? A  
    }  
    set {  
      objc_setAssociatedObject( ←  
        self,  
        &B.sessionAssociatedObject,  
        newValue,  
        .OBJC_ASSOCIATION_RETAIN_NONATOMIC  
      )  
    }  
  }  
}
```



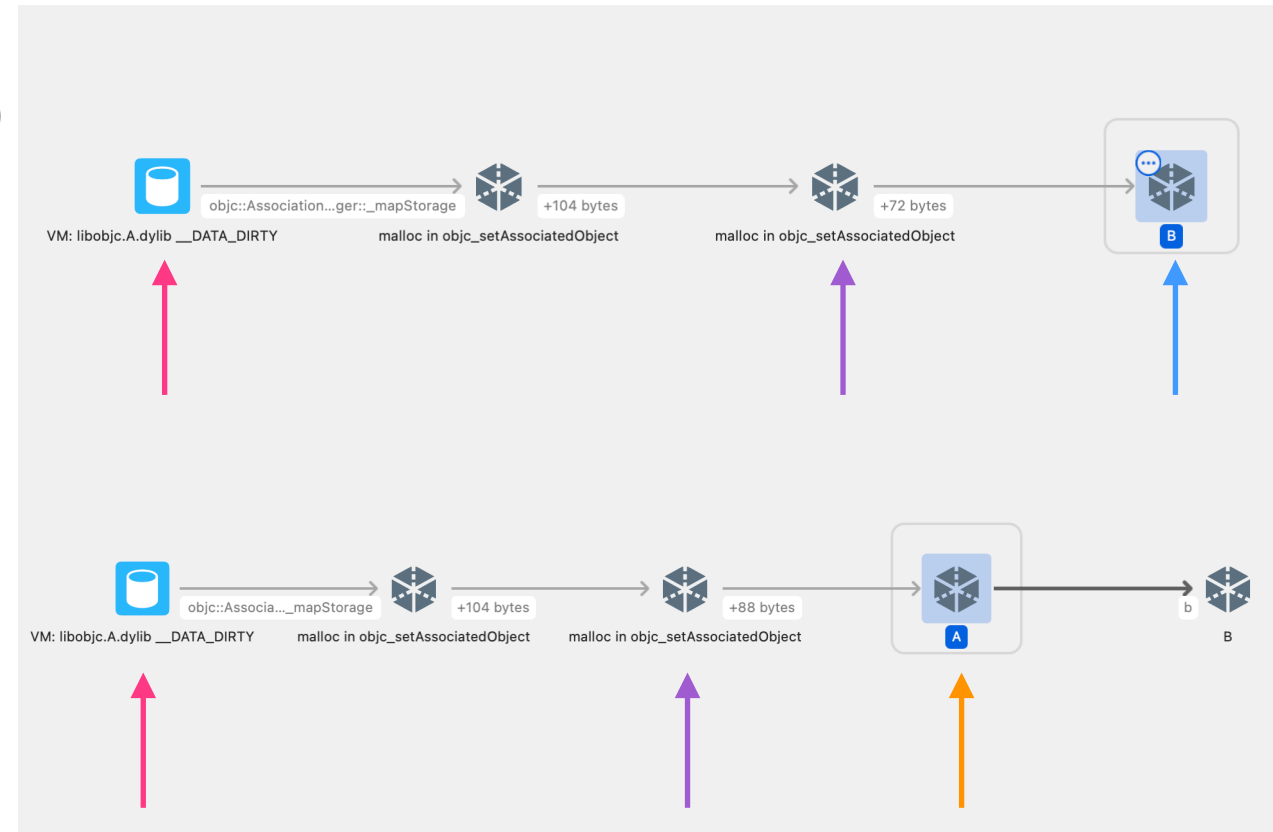
Самодельные хранимые переменные

```
extension B {  
  private static var sessionAssociatedObject: () = ()  
  var a: A? {  
    get {  
      objc_getAssociatedObject(  
        self,  
        &B.sessionAssociatedObject  
      ) as? A  
    }  
    set {  
      objc_setAssociatedObject(  
        self, ←  
        &B.sessionAssociatedObject,  
        newValue,  
        .OBJC_ASSOCIATION_RETAIN_NONATOMIC  
      )  
    }  
  }  
}
```



Самодельные хранимые переменные

```
extension B {  
  private static var sessionAssociatedObject: () = ()  
  var a: A? {  
    get {  
      objc_getAssociatedObject(  
        self,  
        &B.sessionAssociatedObject  
      ) as? A  
    }  
    set {  
      objc_setAssociatedObject(  
        self, ←  
        &B.sessionAssociatedObject,  
        newValue, ←  
        .OBJC_ASSOCIATION_RETAIN_NONATOMIC  
      )  
    }  
  }  
}
```



Автоматизация индикации утечек

У вас

У вас

1 Огромное количество UI тестов

У вас

- 1 Огромное количество UI тестов
- 2 Есть GOD-object, для которого вы знаете сценарий уничтожения

У вас

- 1 Огромное количество UI тестов
- 2 Есть GOD-object, для которого вы знаете сценарий уничтожения
- 3 Не охота править каждый тест

У вас

- 1 Огромное количество UI тестов
- 2 Есть GOD-object, для которого вы знаете сценарий уничтожения
- 3 Не охота править каждый тест

```
class GodObject {
    static var objectsCount = 0

    init() {
        Self.objectsCount += 1
    }

    deinit {
        Self.objectsCount -= 1
    }
}
assert(GodObject.objectsCount == 0)
```

**Пишем обёртку для всех тестов,
которая будет уничтожать GOD-object**

```
func testCaseWillStart(_ testCase: XCTestCase) {
    testCase.addTeardownBlock {
        step("Perform logout scenario") { _ in
            do {
                // смотрим, что приложение существует и не упало
            } catch {
                XCTFail("Application has been crashed")
            }
        }
    }
}
```


Новый тестплан утечек

Новый тестплан утечек

Все подходящие тесты

Новый тестплан утечек

Все подходящие тесты

Прогоны на TeamCity

Новый тестплан утечек

Все подходящие тесты

Прогоны на TeamCity

Контроль в течение дня*

Новый тестплан утечек

Все подходящие тесты

Прогоны на TeamCity

Контроль в течение дня*

* Не можем добавить в прогоны МРов,
так как очень долго

Тест с утечкой

All tests

▼ iOSVK / Merge Requests / vkclient / UI Tests Leaks iPhone vk-client-development / #101 2

▼ UITests: 2

▼ PhotoFlowTests 1

testDeletePhotoViaMultipick

3/3 runs ...

Stacktrace

 Copy to clipboard

===== Failed test run #1 =====

failed - Application has been crashed (/Users/macbuild/Applications/buildAgent/work/9fa45a61def58b3b/UITests/Sources/Infrastructure/XCTest/XCTestObservation/SessionLeakXCTestObserver.swift#CharacterRangeLen=0&EndingLineNumber=37&StartingLineNumber=37)

===== Failed test run #2 =====

failed - Application has been crashed (/Users/macbuild/Applications/buildAgent/work/9fa45a61def58b3b/UITests/Sources/Infrastructure/XCTest/XCTestObservation/SessionLeakXCTestObserver.swift#CharacterRangeLen=0&EndingLineNumber=37&StartingLineNumber=37)

===== Failed test run #3 =====

failed - Application has been crashed (/Users/macbuild/Applications/buildAgent/work/9fa45a61def58b3b/UITests/Sources/Infrastructure/XCTest/XCTestObservation/SessionLeakXCTestObserver.swift#CharacterRangeLen=0&EndingLineNumber=37&StartingLineNumber=37)

Open in build log

Успехи тестплана

■ UI Leaks: vk-client-development ☆









Прогон тест-плана UITestsLeaks из схемы vk-client-development

[<All branches>](#) [Actions](#) [Assign investigation...](#)

Overview [Change Log](#) [Pending Changes 100+](#) [Chains](#) [Issue Log](#) [Statistics](#) [Compatible Agents 72](#) [Settings](#) [WebHooks](#)

[Branches](#) [Builds](#)  31 pending in [Default](#)

[All](#)        

Build number	Branch	Status	Changes	Agent	Started	Duration	
#49	release-8.59	✓ Tests passed: 336, muted: 2; Buil...	 Andrey Zykov: 2		2 Dec 23 08:00	16m 55s	 ...
#48	dev	✓ Tests passed: 337, muted: 1; Buil...	 100+ changes		1 Dec 23 16:34	38m 59s	 ...
#47	dev	✓ Tests passed: 336, muted: 2; Buil...	 100+ changes		1 Dec 23 08:00	17m 33s	 ...
#44	release-8.58	! Tests failed: 2, passed: 324, igno...	 8 changes		29 Nov 23 08:00	20m 16s	 ...

Казнить
нельзя
помиловать

Жизненный цикл утечки

Жизненный цикл утечки

→ Утечки появляются постоянно

Жизненный цикл утечки

- Утечки появляются постоянно
- Необходимо оперативно реагировать

Жизненный цикл утечки

- Утечки появляются постоянно
- Необходимо оперативно реагировать
- Необходимо постоянно отвлекаться от своих задач

Жизненный цикл утечки

- Утечки появляются постоянно
- Необходимо оперативно реагировать
- Необходимо постоянно отвлекаться от своих задач
- Люди повторяют свои ошибки

Жизненный цикл утечки

- Утечки появляются постоянно
- Необходимо оперативно реагировать
- Необходимо постоянно отвлекаться от своих задач
- Люди повторяют свои ошибки
- Утечки появляются постоянно

**Создаём регламент по работе
с утечками**

Регламент



очередь

Регламент



очередь

Регламент



очередь

Регламент



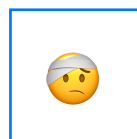
очередь

Регламент



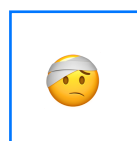
очередь

Регламент



очередь

Регламент



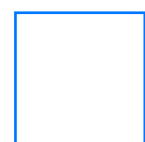
очередь

Регламент



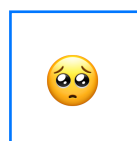
очередь

Регламент



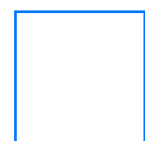
очередь

Регламент



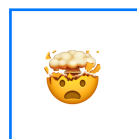
очередь

Регламент



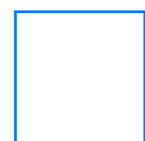
очередь

Регламент



очередь

Регламент



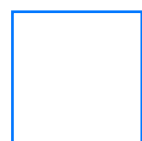
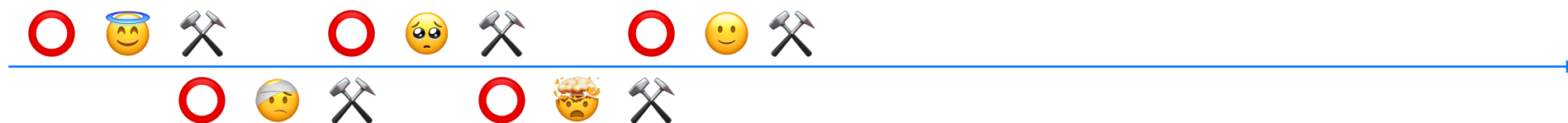
очередь

Регламент



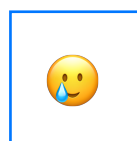
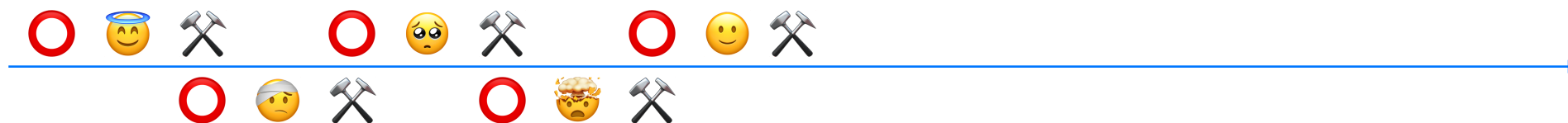
очередь

Регламент



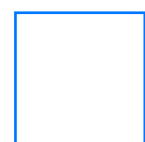
очередь

Регламент



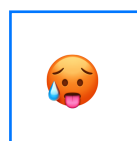
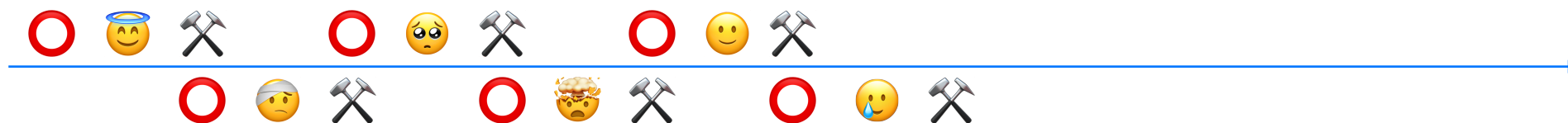
очередь

Регламент



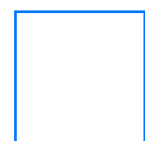
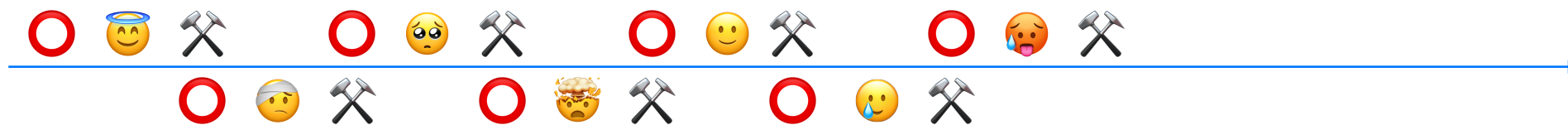
очередь

Регламент



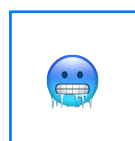
очередь

Регламент



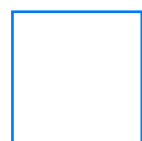
очередь

Регламент



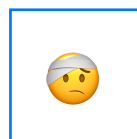
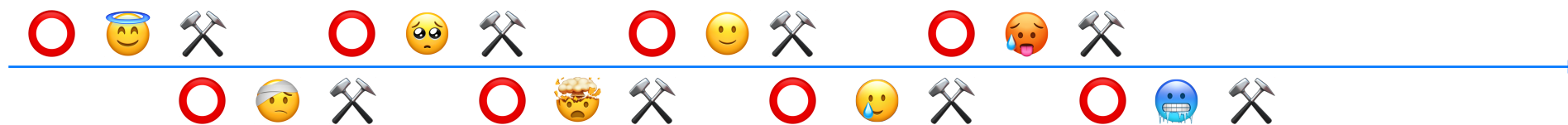
очередь

Регламент



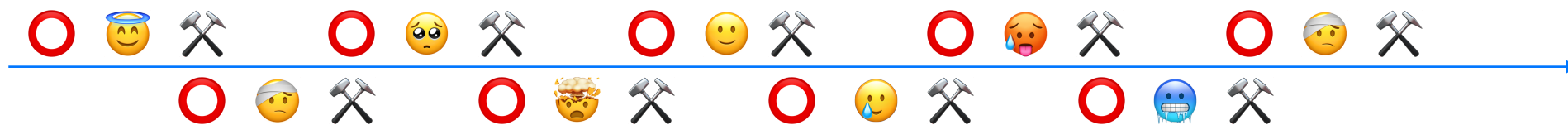
очередь

Регламент



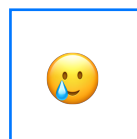
очередь

Регламент



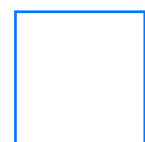
очередь

Регламент



очередь

Регламент



очередь

Плюсы командной работы



Плюсы командной работы

Расширение
навыков



Плюсы командной работы

Расширение
навыков

Распределение
нагрузки



Плюсы командной работы

Расширение
навыков

Распределение
нагрузки

Скорость
реакции



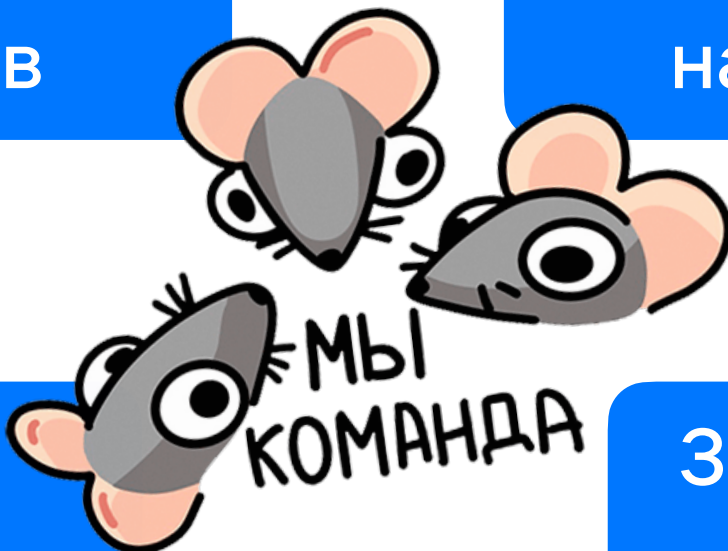
Плюсы командной работы

Расширение
навыков

Распределение
нагрузки

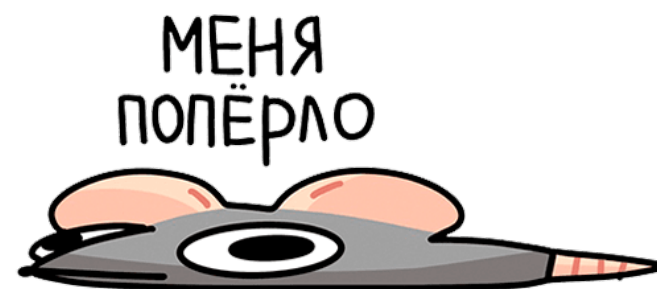
Скорость
реакции

Закрепление
ошибок



Моя
борьба

Что обсудили?



Что обсудили?

- 1 Что такое утечка



Что обсудили?

- 1 Что такое утечка
- 2 Графики с пользователями



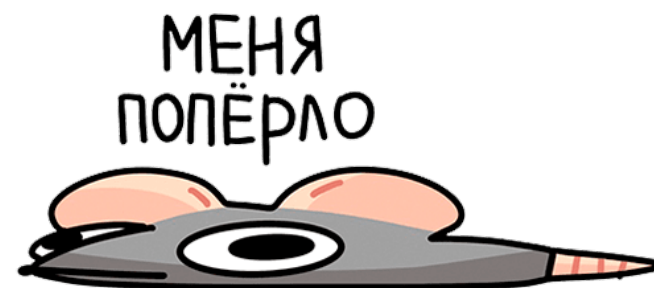
Что обсудили?

- 1 Что такое утечка
- 2 Графики с пользователей
- 3 Инструменты для индикации



Что обсудили?

- 1 Что такое утечка
- 2 Графики с пользователей
- 3 Инструменты для индикации
- 4 Инструменты для поиска



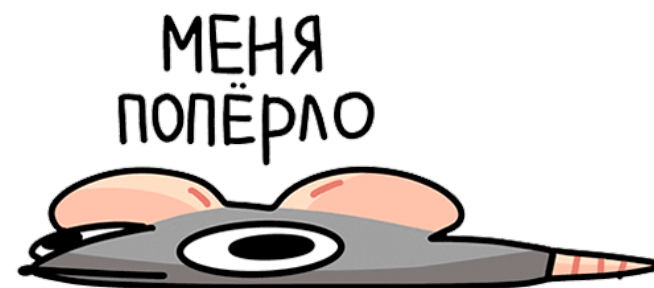
Что обсудили?

- 1 Что такое утечка
- 2 Графики с пользователей
- 3 Инструменты для индикации
- 4 Инструменты для поиска
- 5 Особые случаи утечек



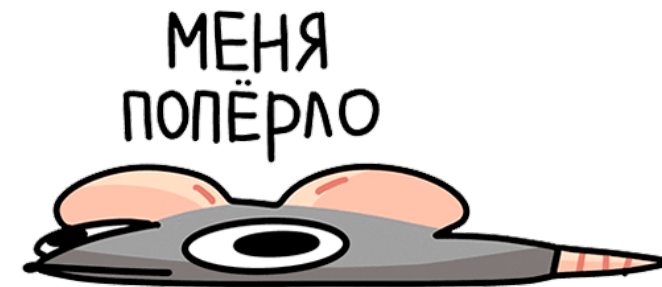
Что обсудили?

- 1 Что такое утечка
- 2 Графики с пользователей
- 3 Инструменты для индикации
- 4 Инструменты для поиска
- 5 Особые случаи утечек
- 6 Автоматизация индикации



Что обсудили?

- 1 Что такое утечка
- 2 Графики с пользователей
- 3 Инструменты для индикации
- 4 Инструменты для поиска
- 5 Особые случаи утечек
- 6 Автоматизация индикации
- 7 Приобщение коллег к общему благу





Спасибо за внимание!

Колосов Артём, iOS-разработчик в команде мобильной инфраструктуры ВКонтакте
vk.com/ADevil000

