Борьба с утечками памяти: от задачи до победы

Колосов Артём,
ВКонтакте

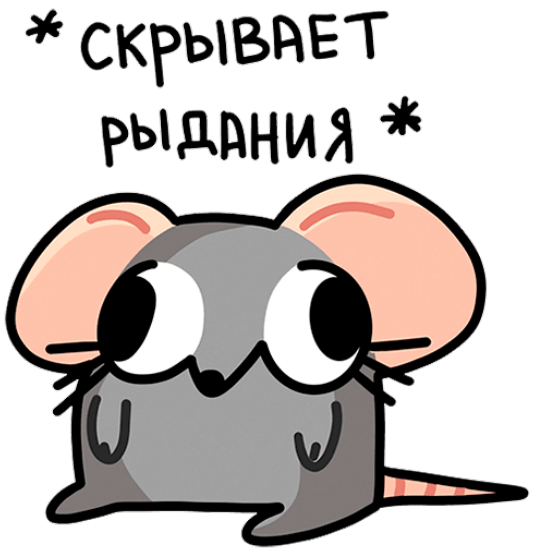
Колосов Артём

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



Что обсудим?

Что обсудим?

- Что такое утечка

Что обсудим?

- Что такое утечка
- Графики с пользователей

Что обсудим?

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

Что обсудим?

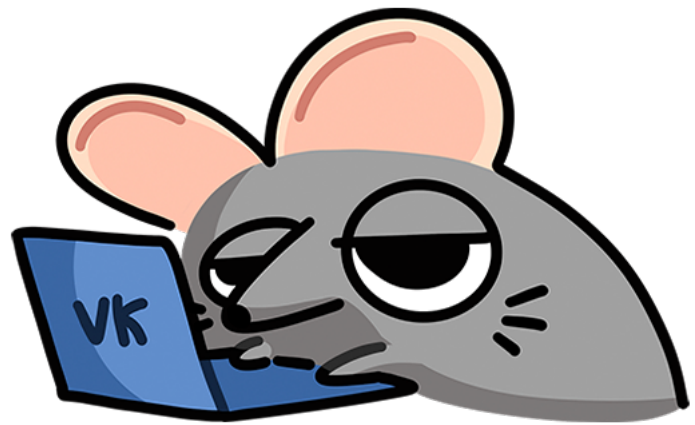
- Что такое утечка
- Графики с пользователей
- Инструменты для индикации
- Особые случаи утечек

Что обсудим?

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

Что обсудим?

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


VK

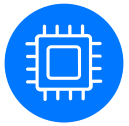

ЖЕСТЬ
СОГЛАСЕН


ВЫНЮХИВАЮ

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

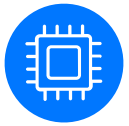
Почему надо бороться с утечками памяти?

Почему надо бороться с утечками памяти?



Сокращение потребляемой оперативной памяти

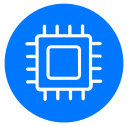
Почему надо бороться с утечками памяти?



Сокращение потребляемой оперативной памяти

Реже OOM

Почему надо бороться с утечками памяти?

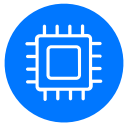


Сокращение потребляемой оперативной памяти

Реже OOM

Система реже выгружает приложение

Почему надо бороться с утечками памяти?

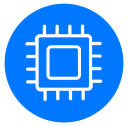


Сокращение потребляемой оперативной памяти

- Реже OOM
- Система реже выгружает приложение

Поддержание корректного состояния приложения

Почему надо бороться с утечками памяти?



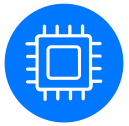
Сокращение потребляемой оперативной памяти

- 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.

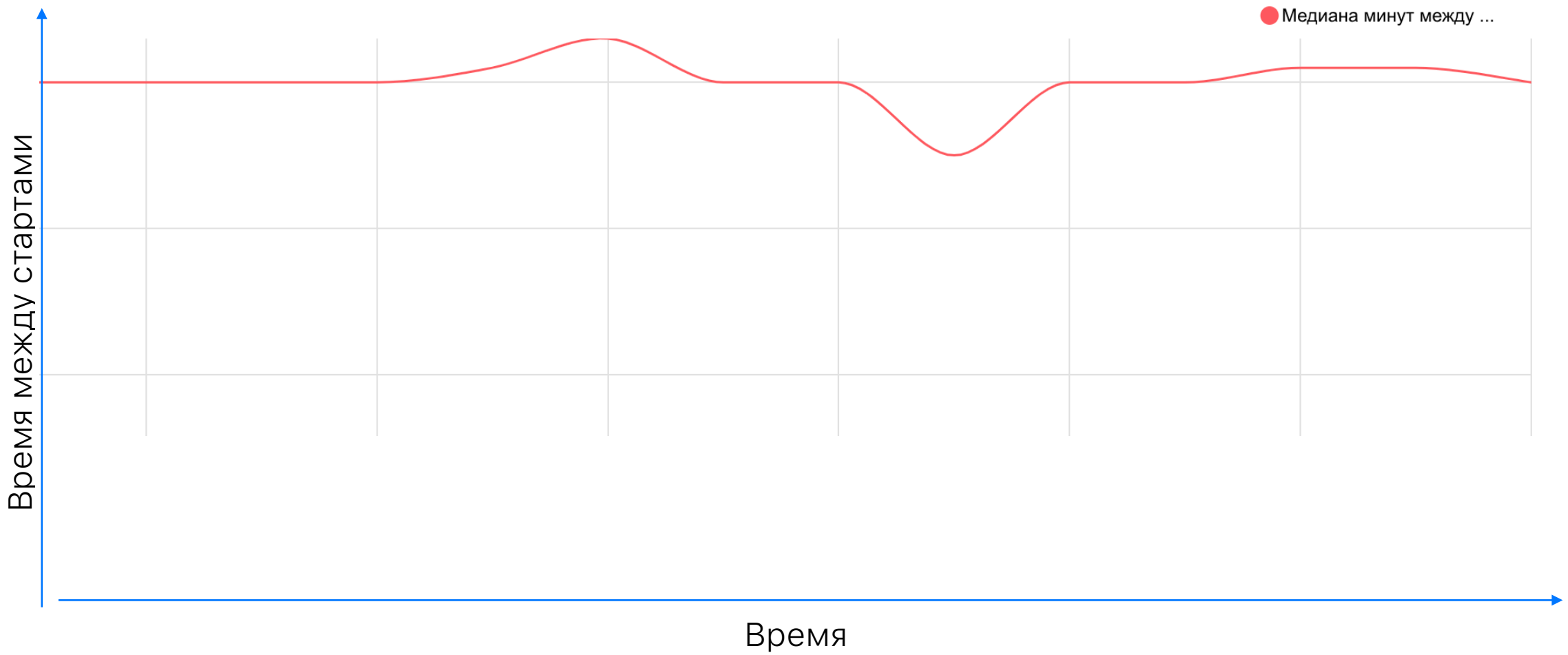
Documentation/MetricKit/MXAppExitMetric/

Закрытия по памяти

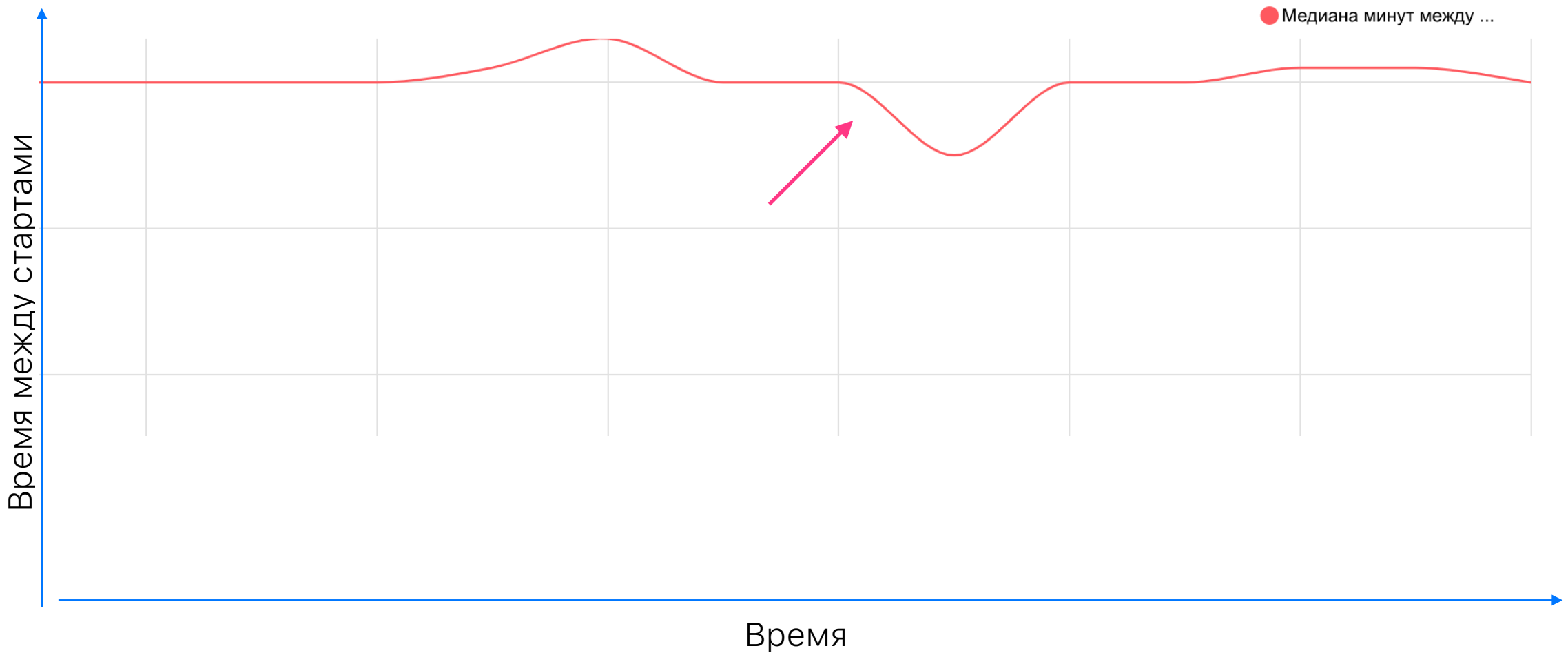
Закрытия по памяти

Закрытия по памяти

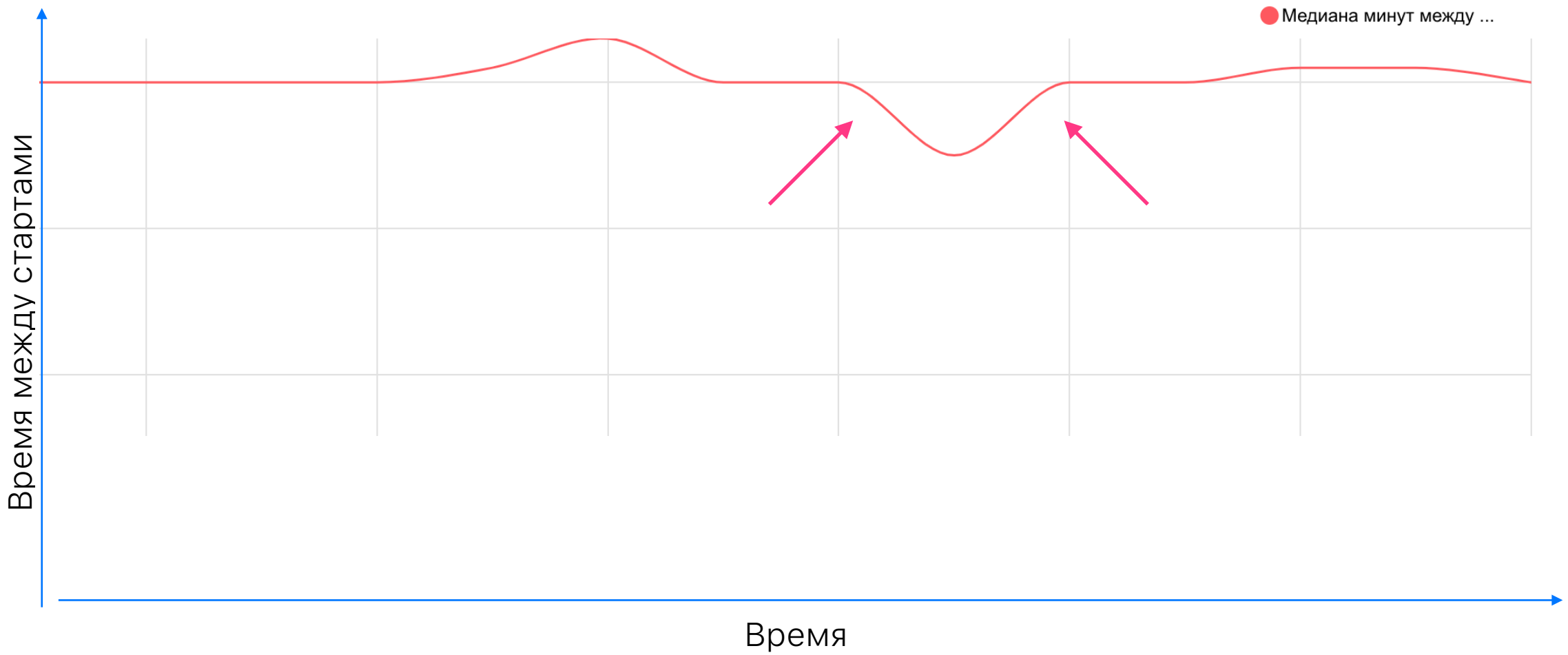
Медиана между холодными стартами



Медиана между холодными стартами



Медиана между холодными стартами



Как сделать,
чтобы не тормозило,
пожалуйста

Евгений Шаповалов

Определяем присутствие утечки

Как определять присутствие утечки?

Как определять присутствие утечки?

- Пользуемся инструментами Xcode и верим, что они определят

Как определять присутствие утечки?

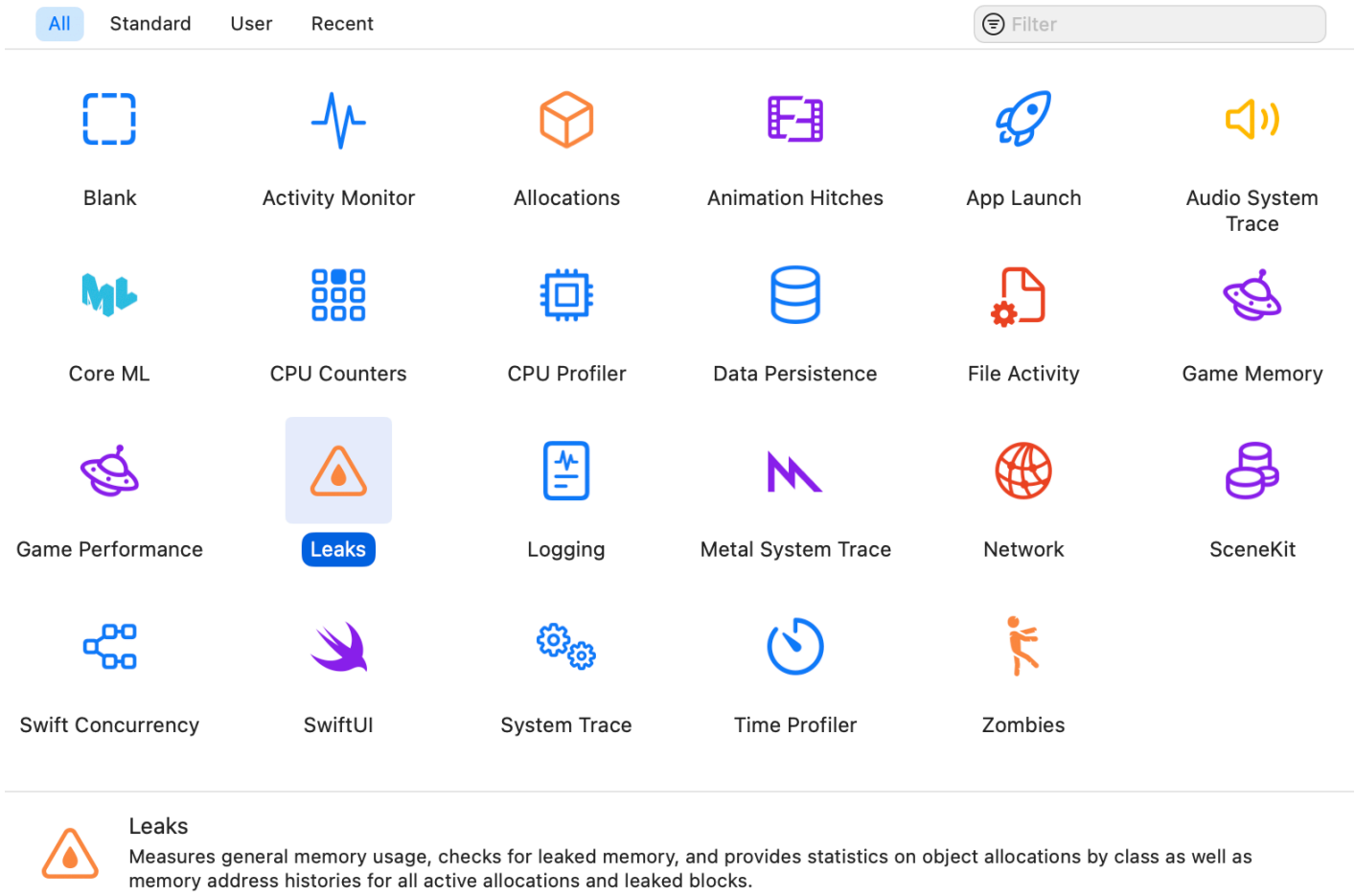
- Пользуемся инструментами Xcode и верим, что они определят
- Определяем GOD-objects и следим за их количеством самостоятельно

Как определять присутствие утечки?

- Пользуемся инструментами Xcode и верим, что они определят
- 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)
```

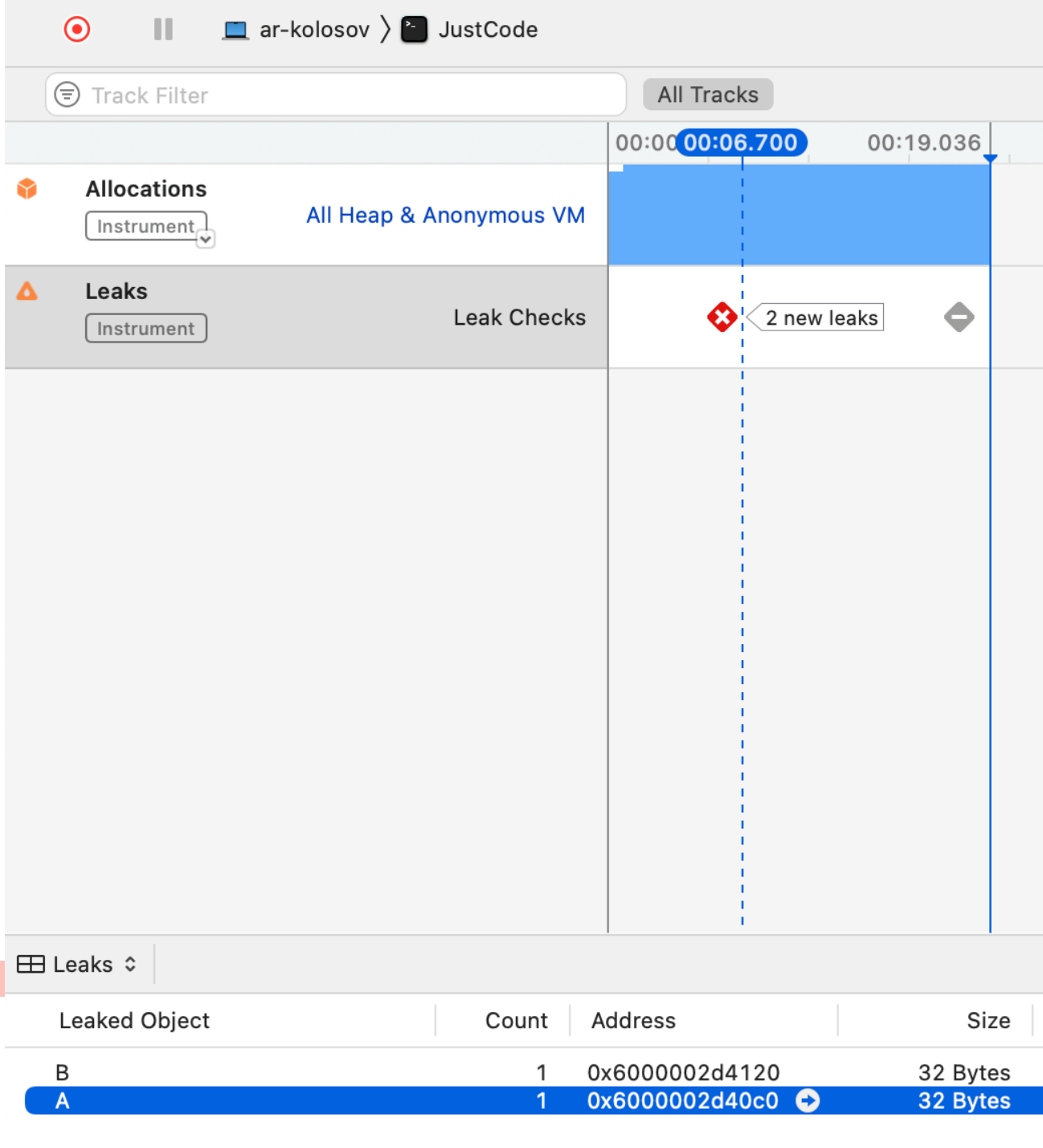
Выясняем наличие утечек



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



Instruments

Resolved Issues

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

Как определять присутствие утечки?

- Пользуемся инструментами Xcode и верим, что они определят
- 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?

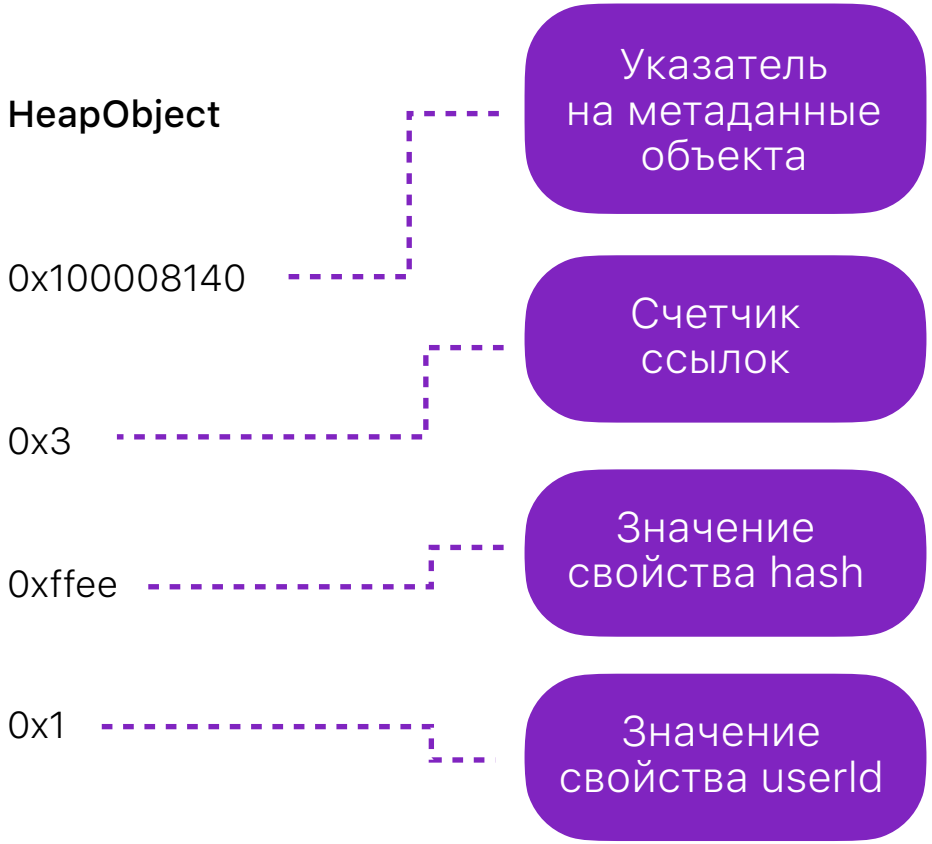
- Пользуемся инструментами Xcode и верим, что они определят
- Определяем GOD-objects и следим за их количеством самостоятельно

```
class GodObject {
    static var objectsCount = 0

    init() {
        Self.objectsCount += 1
    }

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

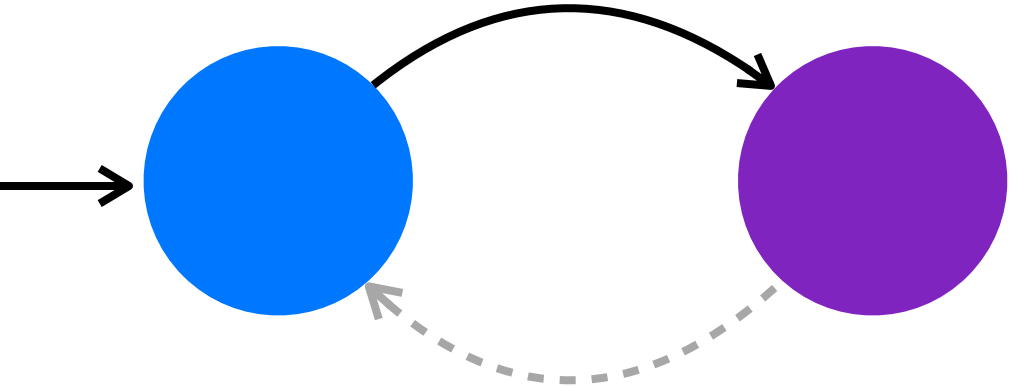
Модель памяти

Хранение объекта

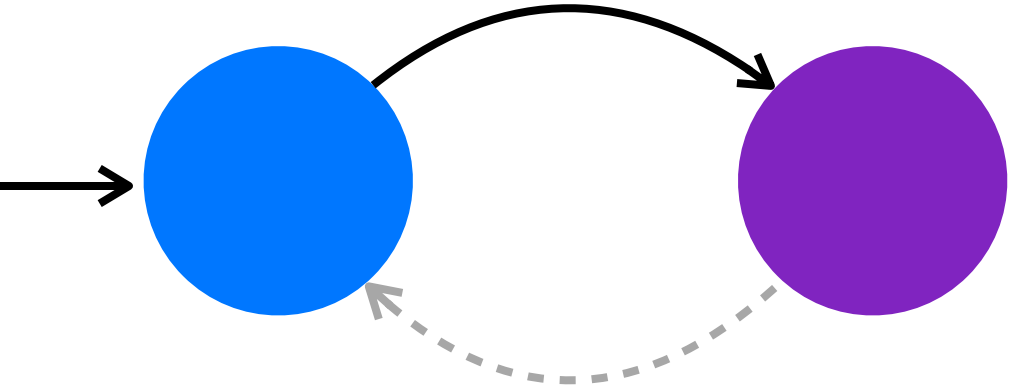
```
class CommitInfo {
    let hash: Int
    let userId: Int

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



Виды ссылок

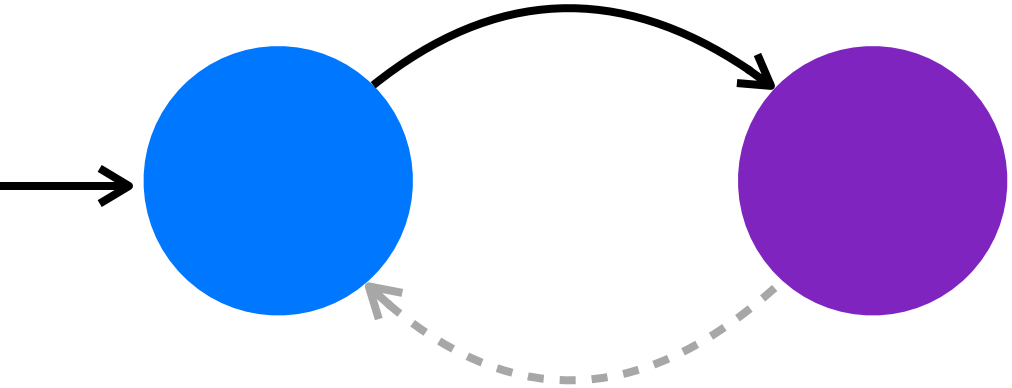


Виды ссылок



Strong

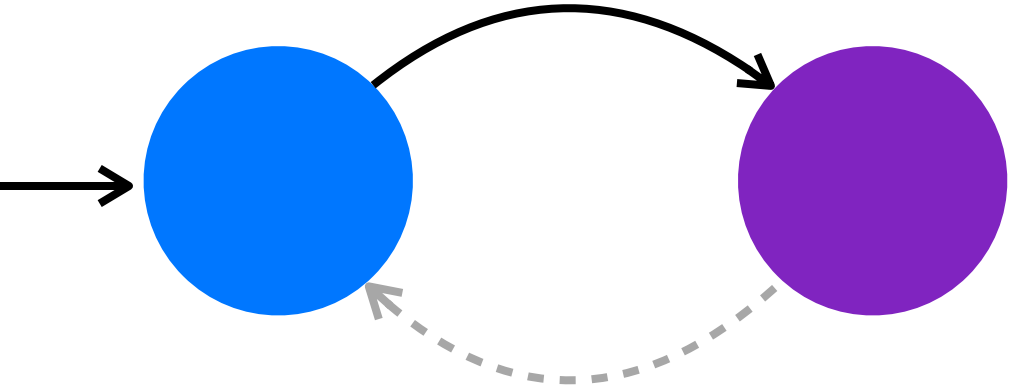
Виды ссылок



Strong

Weak

Виды ссылок

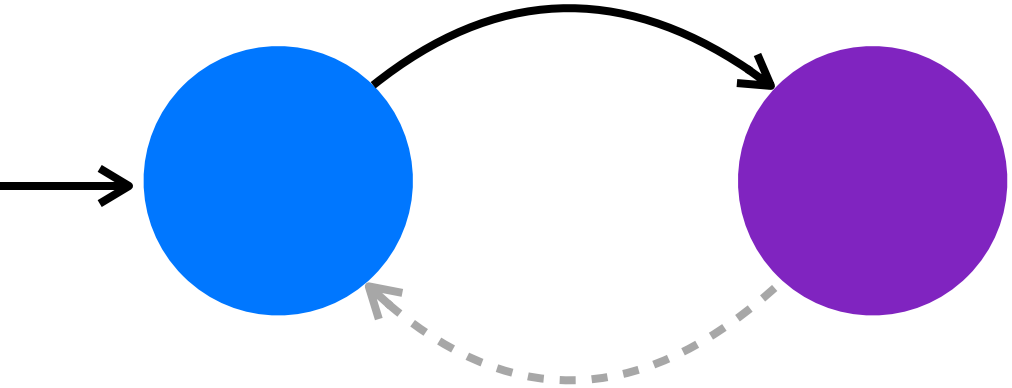


Strong

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

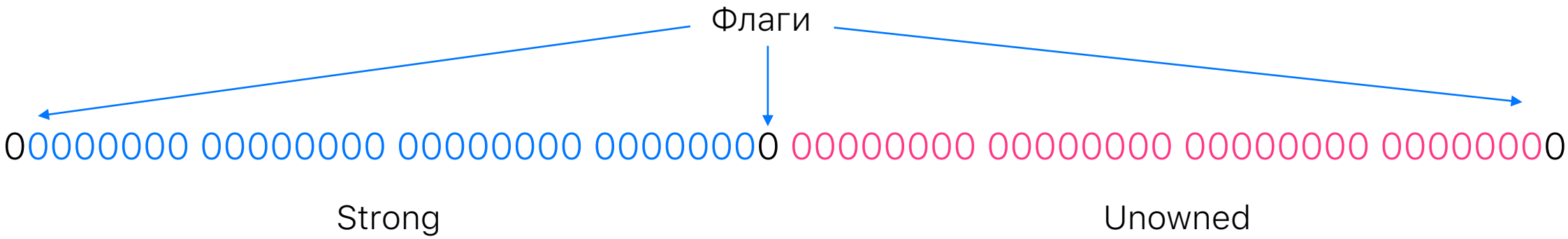
Счётчик ссылок (InlineRefCounts)

0OOOOOOO OOOOOOOO OOOOOOOO OOOOOOO0 OOOOOOOO OOOOOOOO OOOOOOOO OOOOOOO0

Strong

Unowned

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)

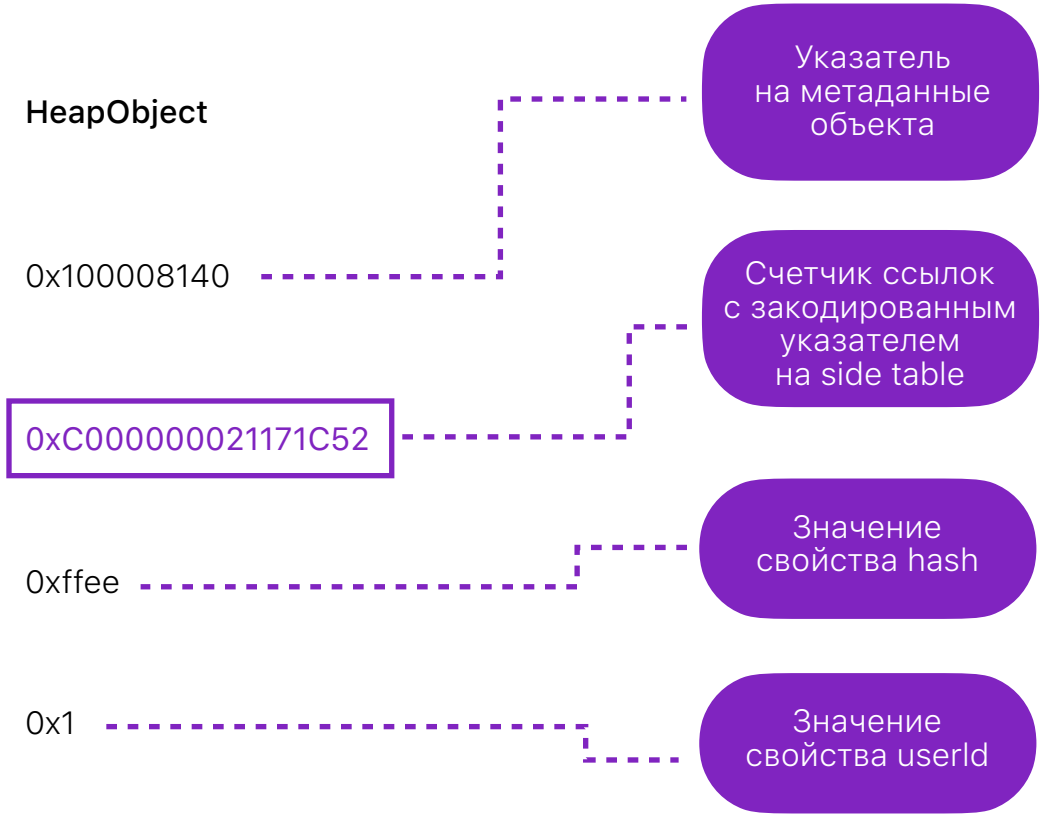
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

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;
#if 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 определят
- 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 памяти

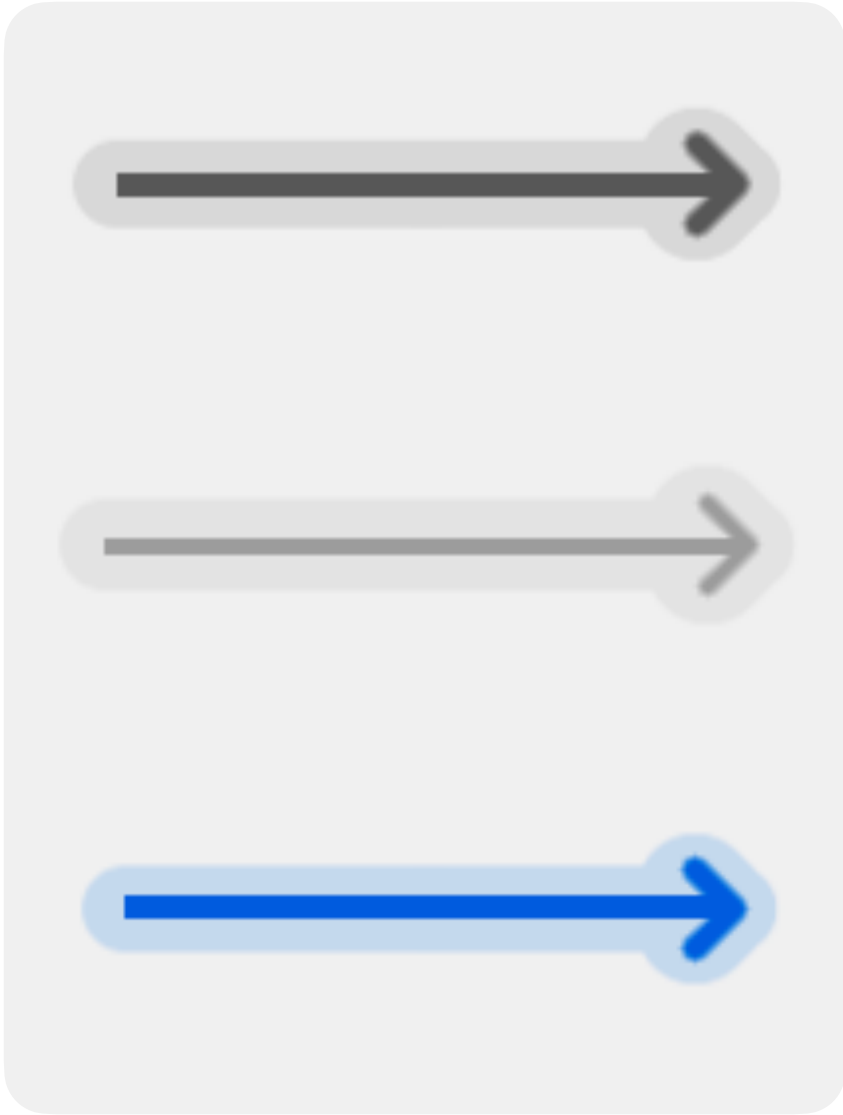
WWDC.
iOS Memory Deep Dive

Легенда карты

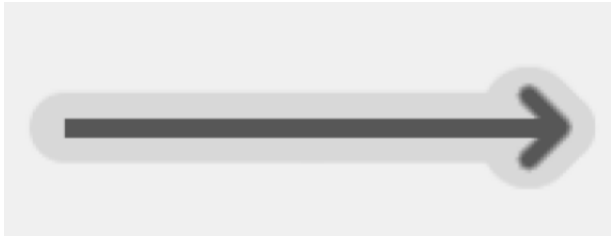
Чёрная стрелка — в большинстве своём сильная ссылка.

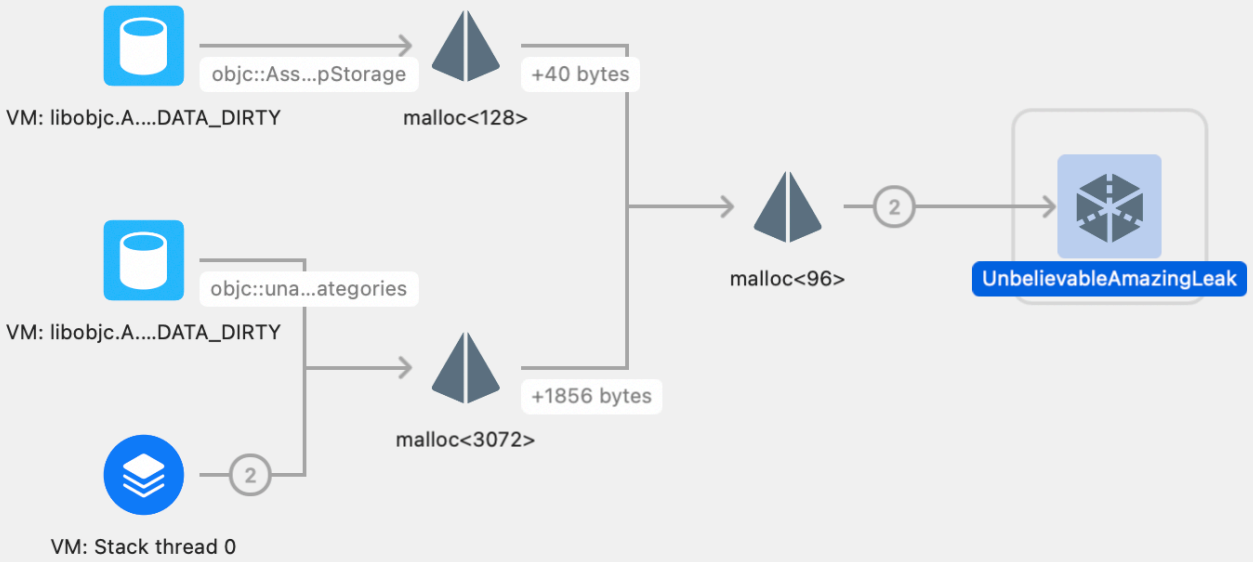
Серая стрелка — в большинстве случаев слабая ссылка на объект.

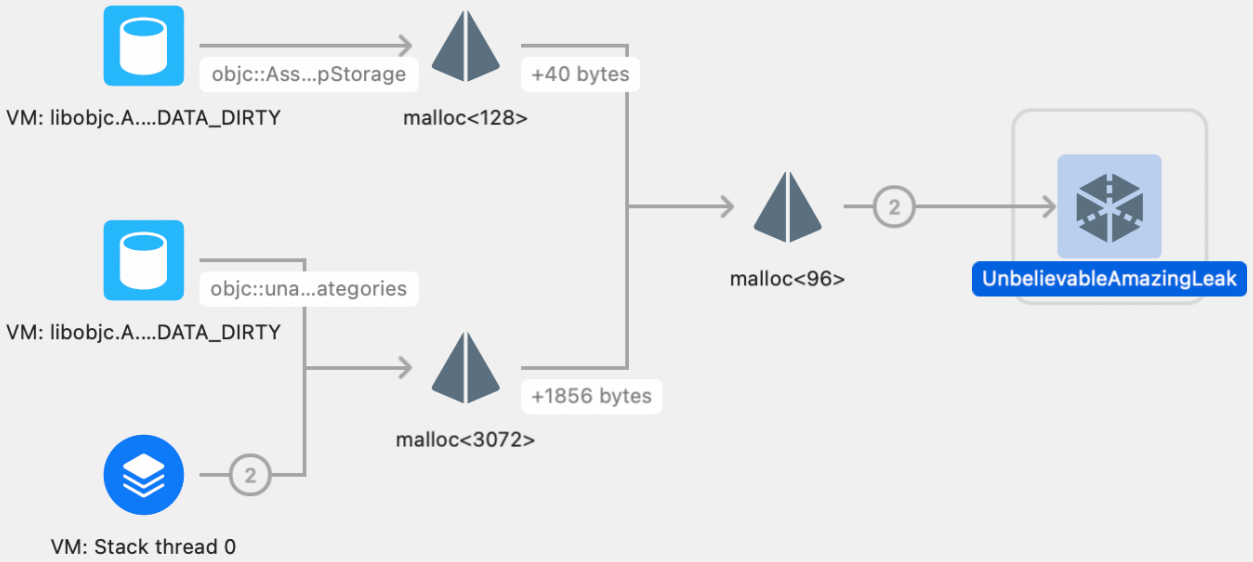
Синяя стрелка — связь, выбранная в дебагере.

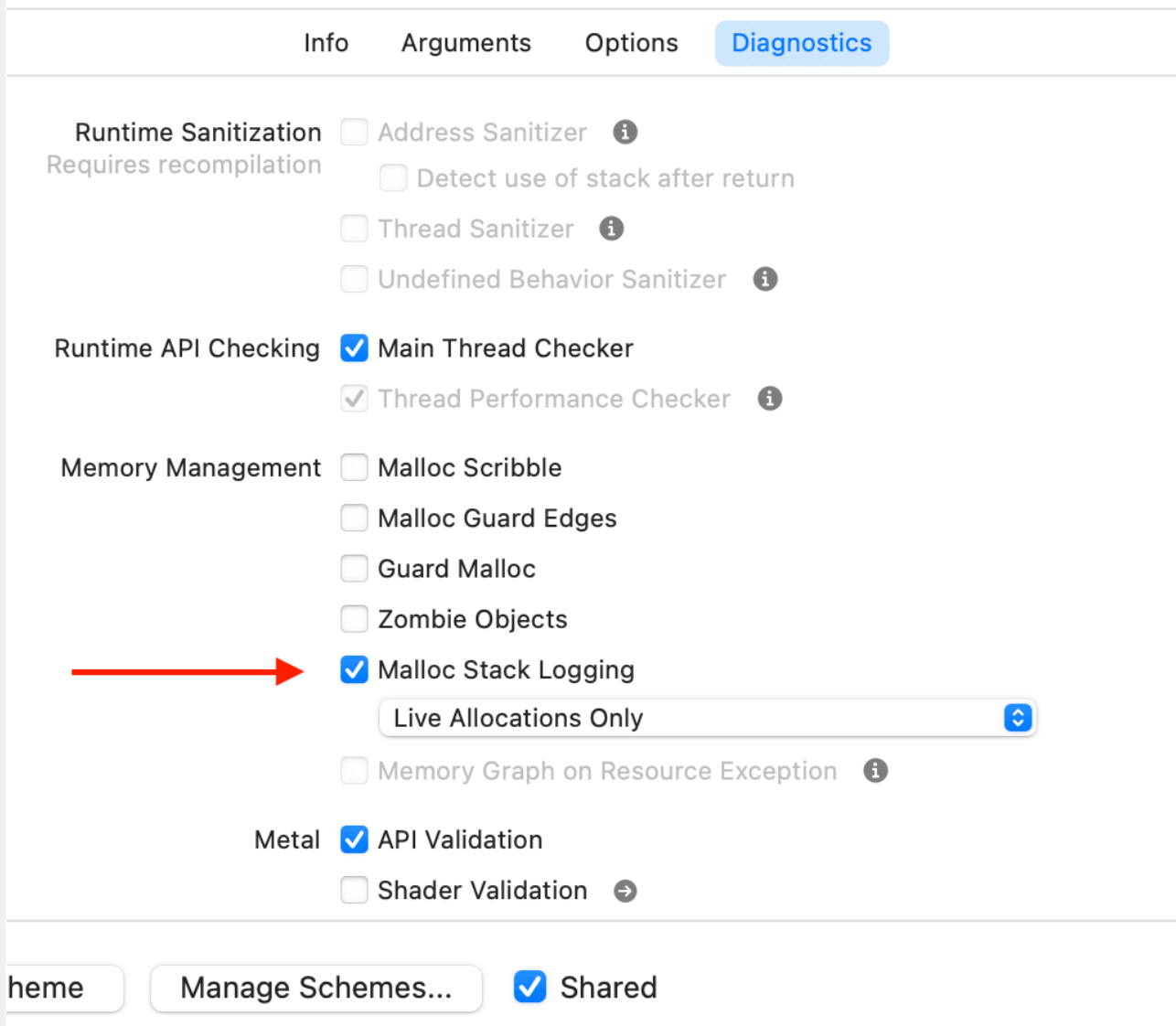


Легенда карты (Memory Graph)

Стрелка в графе	Связь в рантайме
	Strong
	Weak
	Выбранная связь в дебаггере


objc::Ass...pStorage
+40 bytes
VM: libobjc.A....DATA_DIRTY
malloc<128>
objc::una...ategories
VM: libobjc.A....DATA_DIRTY
malloc<96>
2
UnbelievableAmazingLeak
+1856 bytes
malloc<3072>
2
VM: Stack thread 0


objc::Ass...pStorage
+40 bytes
VM: libobjc.A....DATA_DIRTY
malloc<128>
objc::una...ategories
VM: libobjc.A....DATA_DIRTY
+1856 bytes
malloc<3072>
2
malloc<96>
2
UnbelievableAmazingLeak
VM: Stack thread 0

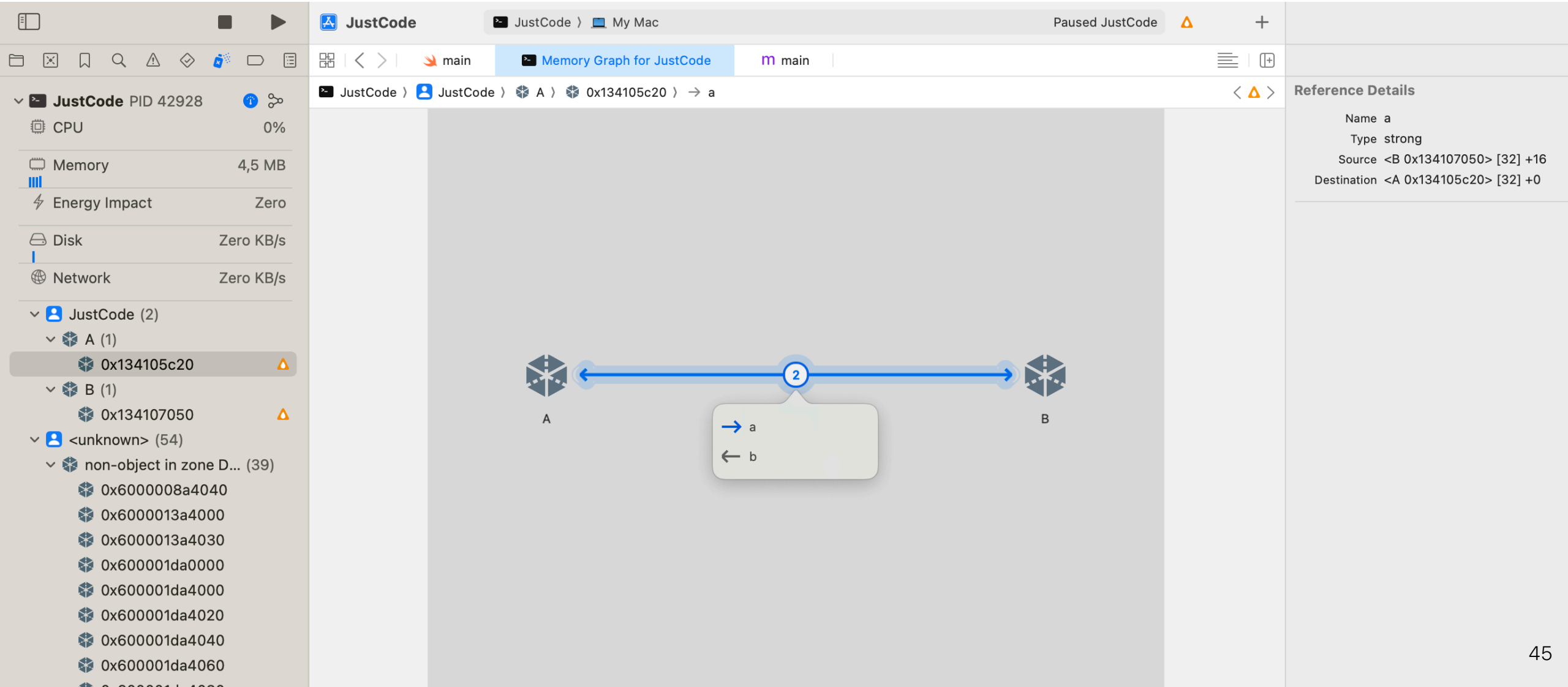

Info
Arguments
Options
Diagnostics
Runtime Sanitization
Requires recompilation
Address Sanitizer
Detect use of stack after return
Thread Sanitizer
Undefined Behavior Sanitizer
Runtime API Checking
Main Thread Checker
Thread Performance Checker
Memory Management
Malloc Scribble
Malloc Guard Edges
Guard Malloc
Zombie Objects
Malloc Stack Logging
Live Allocations Only
Memory Graph on Resource Exception
Metal
API Validation
Shader Validation
heme
Manage Schemes...
Shared

Определяем причины утечек

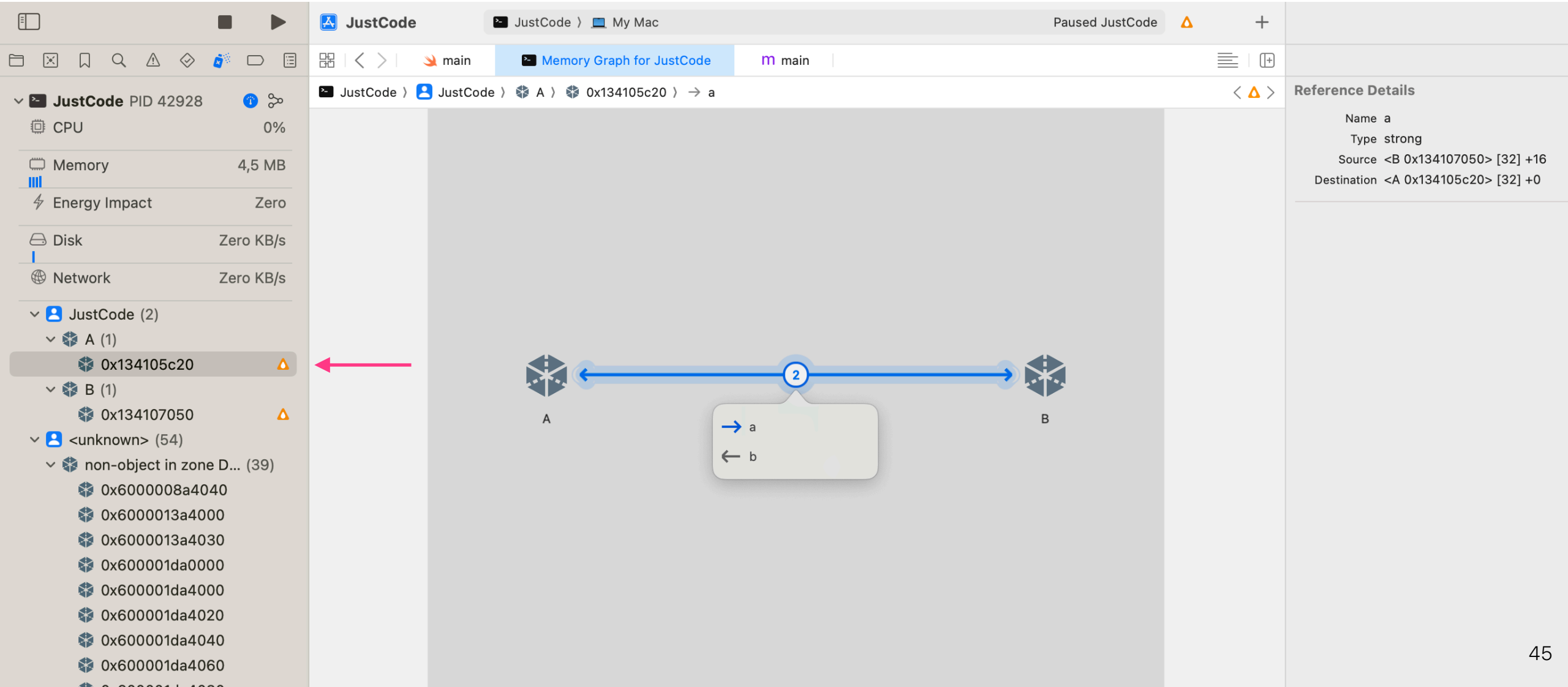
```swift
class A: GodObject {
    var b: B?
}

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

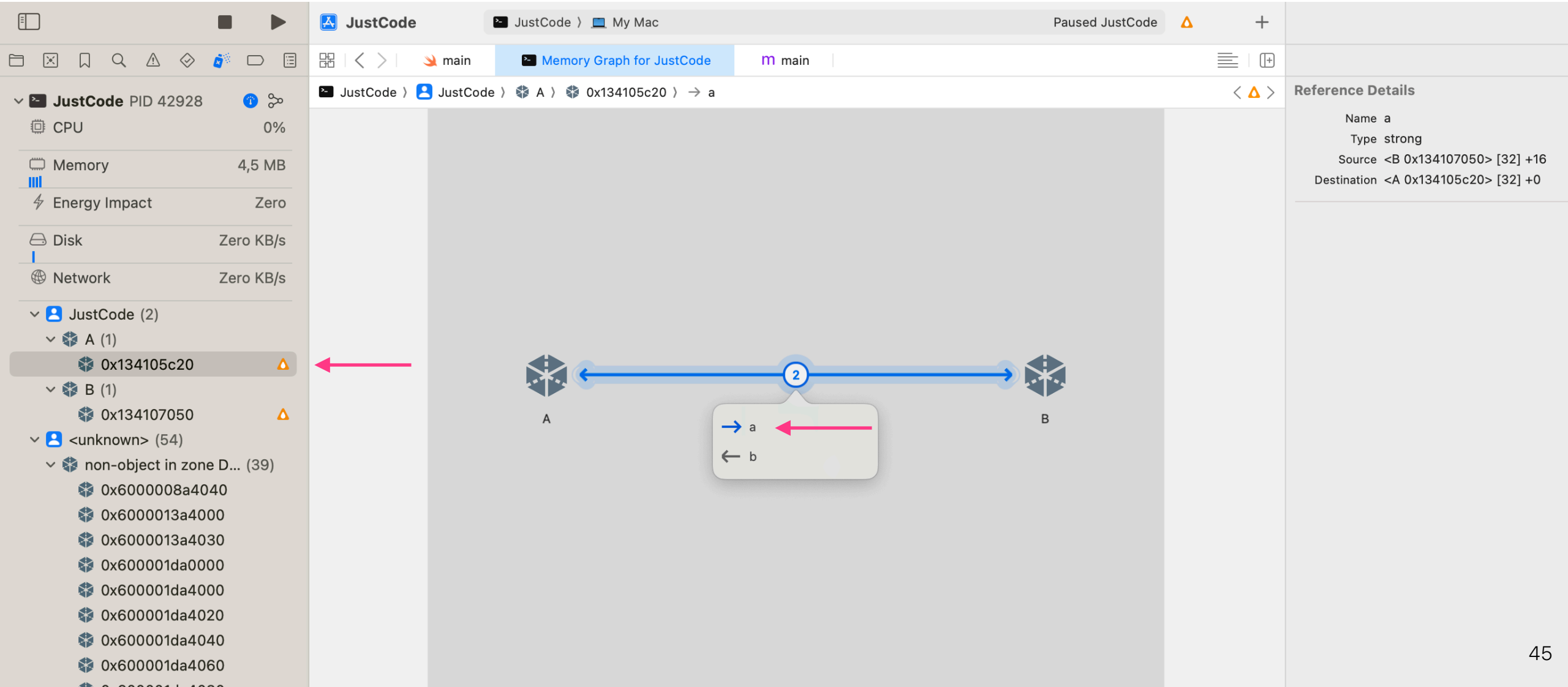
Собираем граф памяти



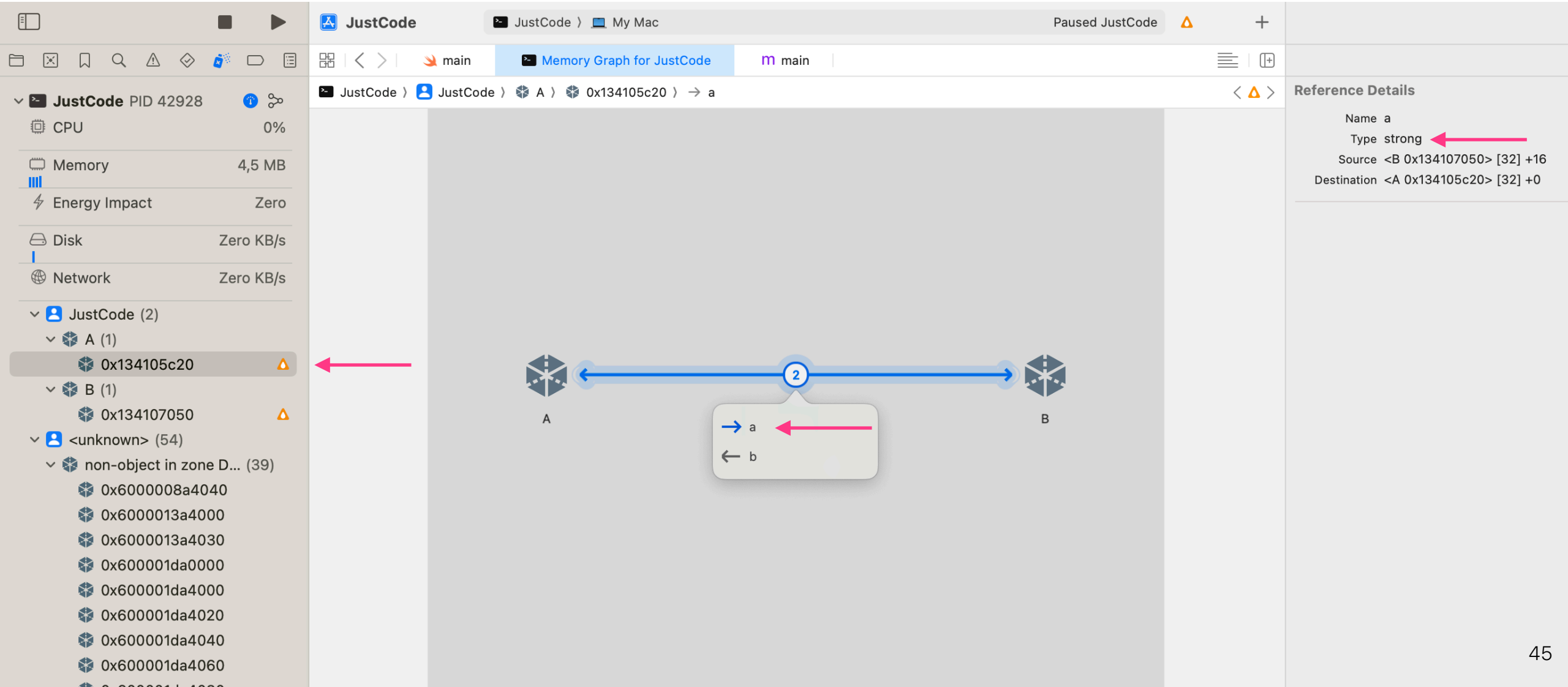
Собираем граф памяти



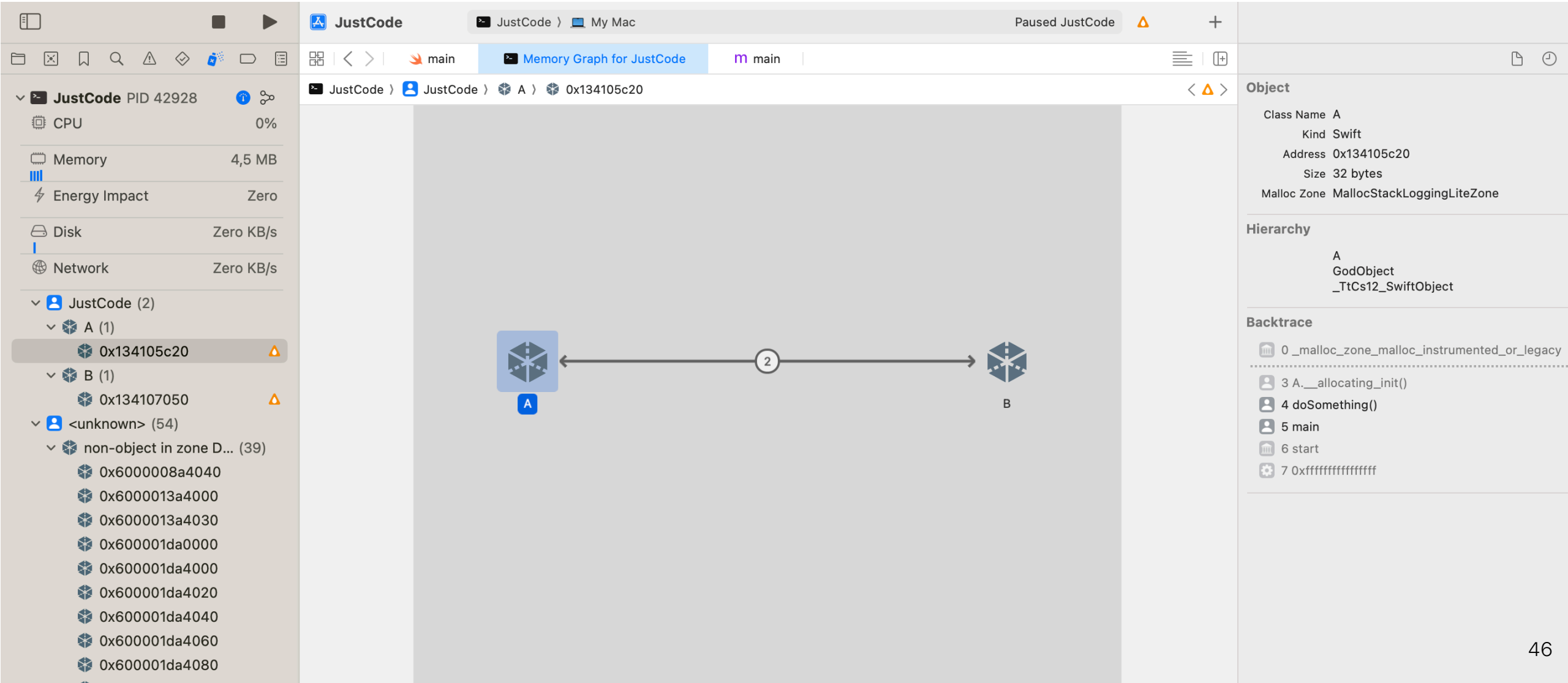
Собираем граф памяти



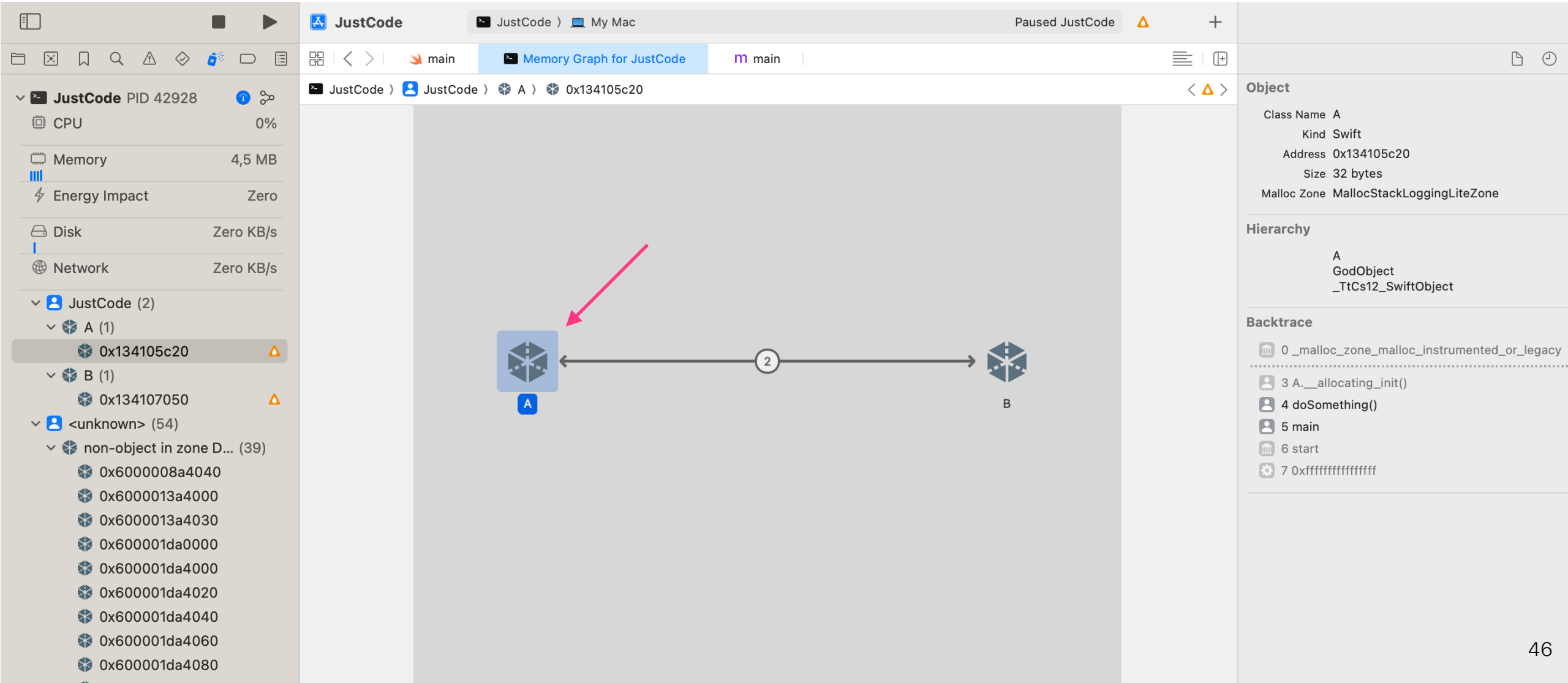
Собираем граф памяти



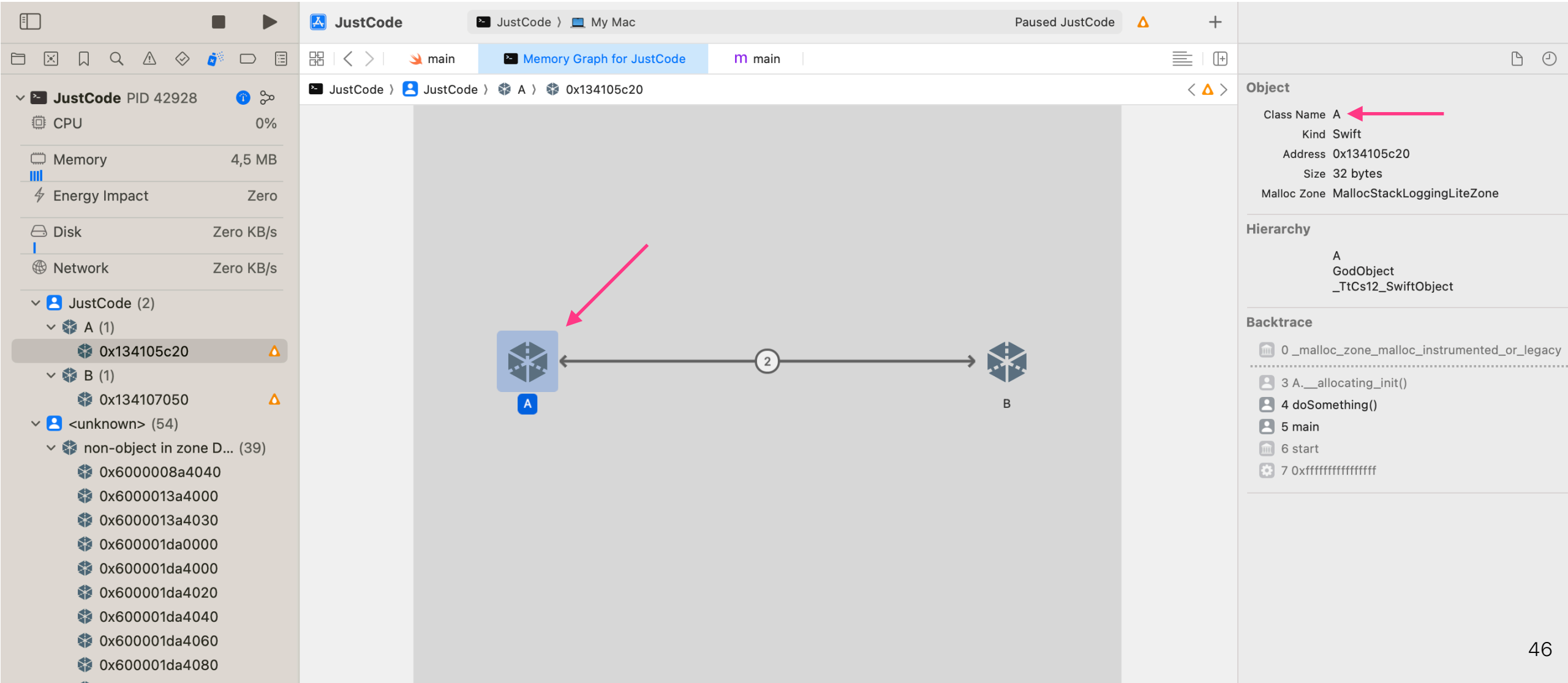
Собираем граф памяти



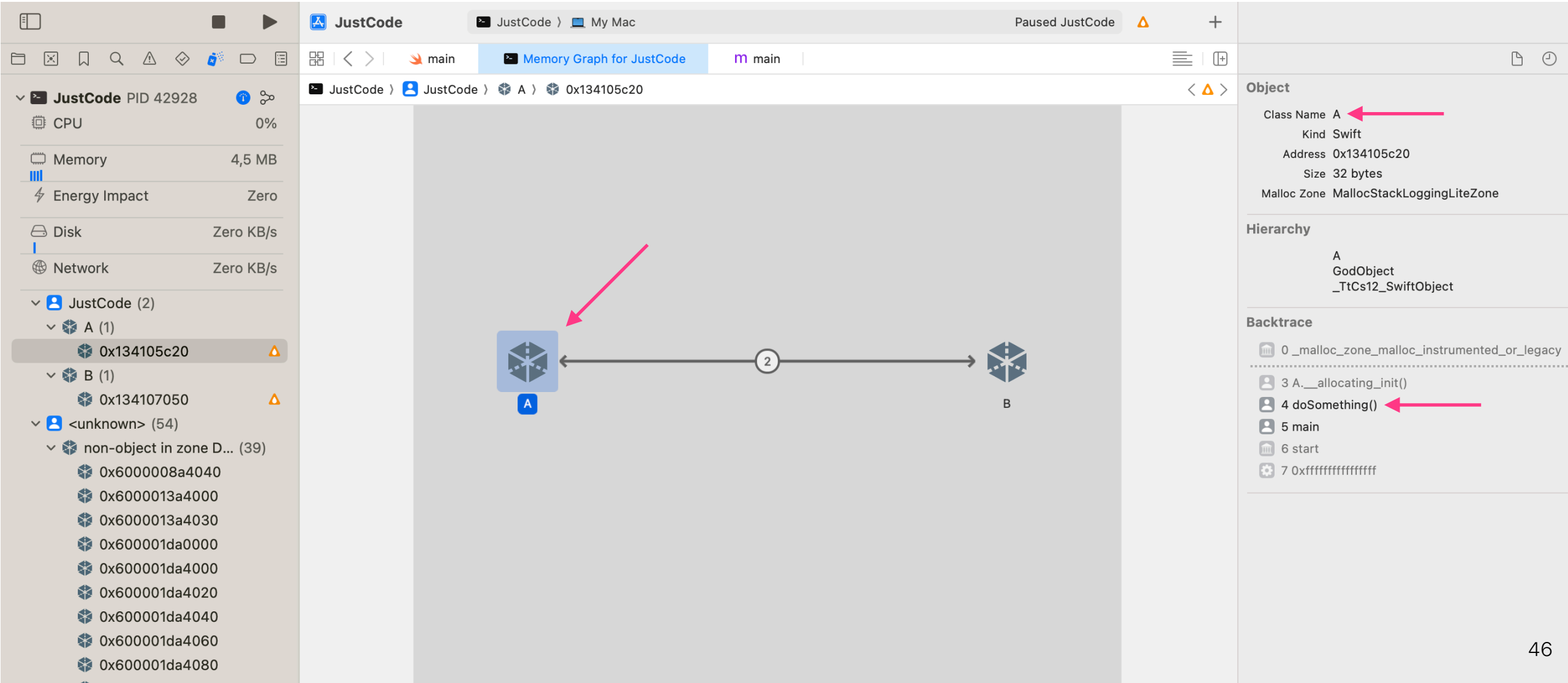
Собираем граф памяти



Собираем граф памяти

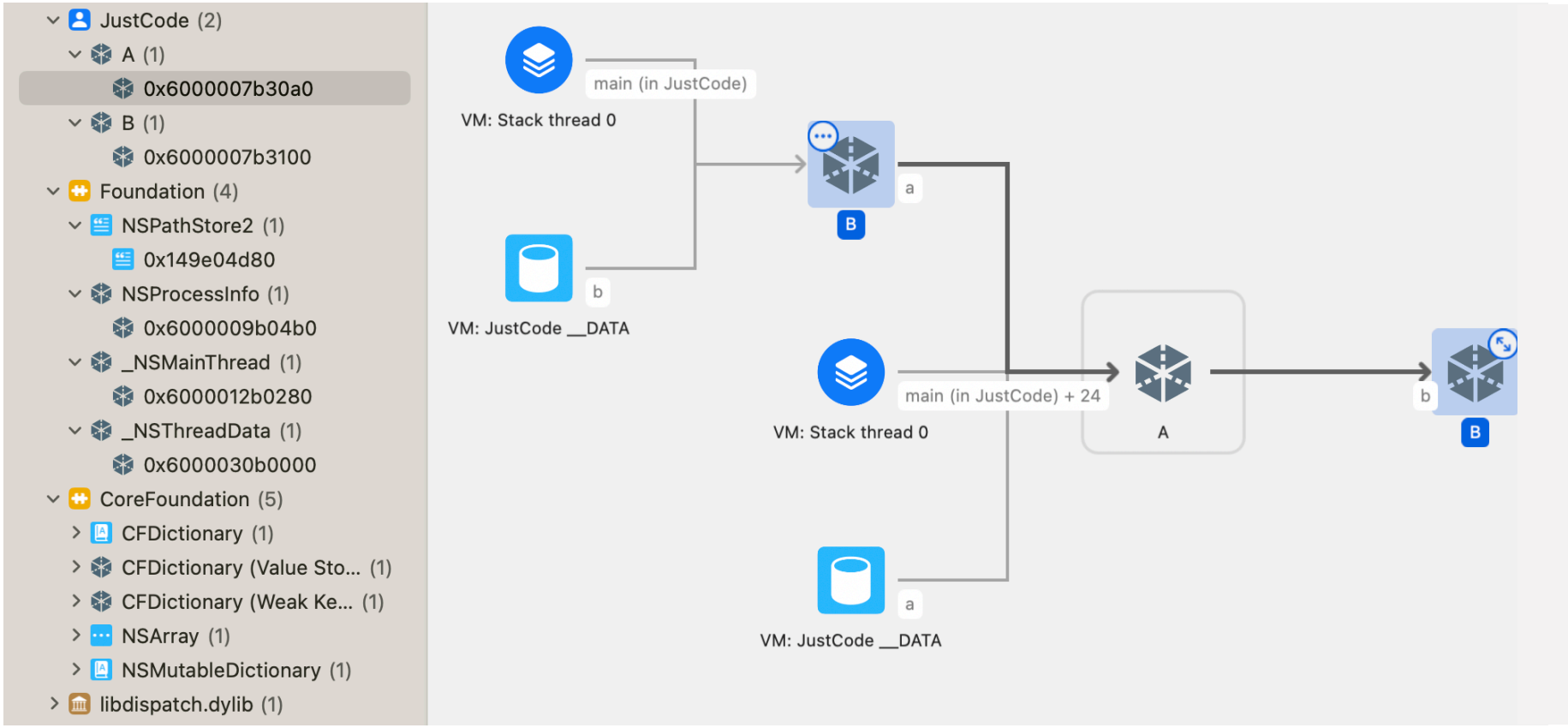


Собираем граф памяти



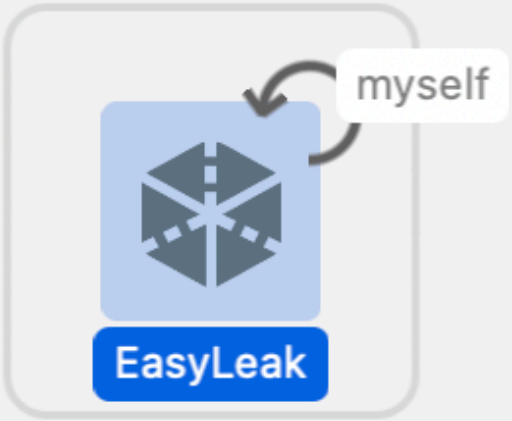
Собираем граф памяти, но...

```
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()
```



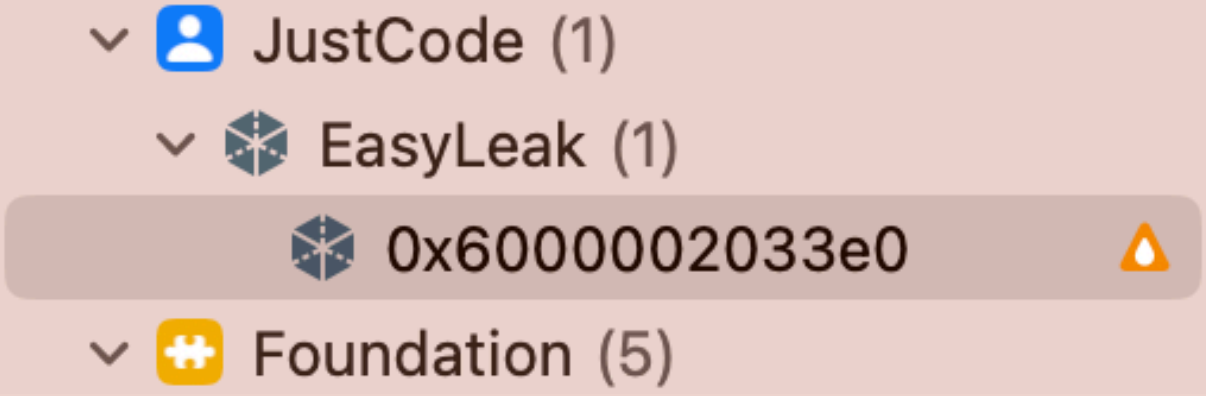
Сколько минимум объектов необходимо для утечки?

1



Делаем плохо

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

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

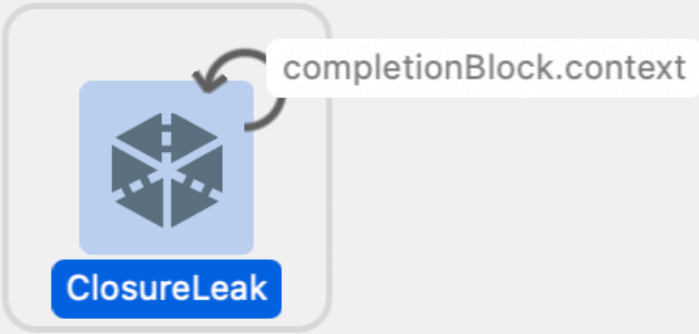


Делаем хорошо

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

let easyLeak = EasyLeak()
easyLeak.myself = easyLeak

class NoLeak {
    weak var myself: NoLeak?
}

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

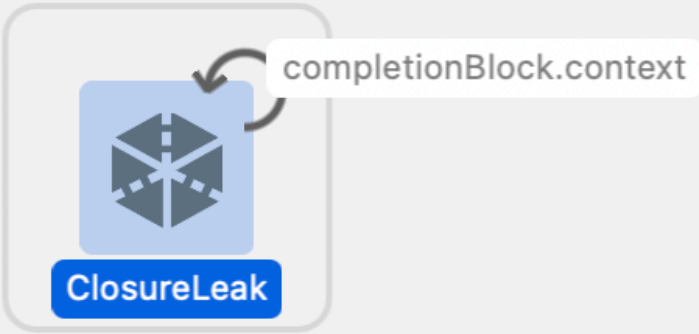
Утечка через closure

```
class ClosureLeak {
    var completionBlock: () -> Void = {}

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

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

doClosureLeak()
```



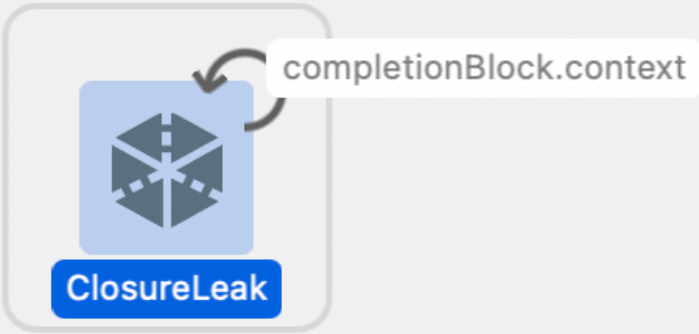
Утечка через closure

```
class ClosureLeak {
    var completionBlock: () -> Void = {}

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

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

doClosureLeak()
```



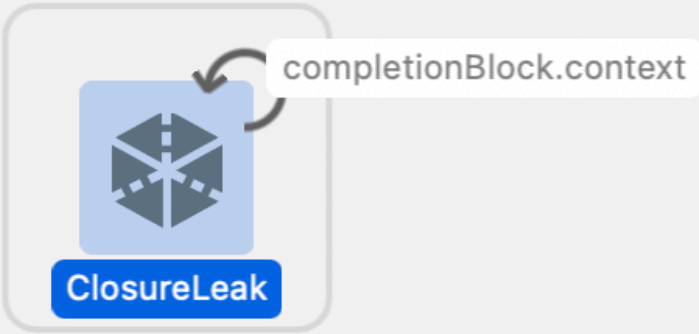
Утечка через closure

```
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

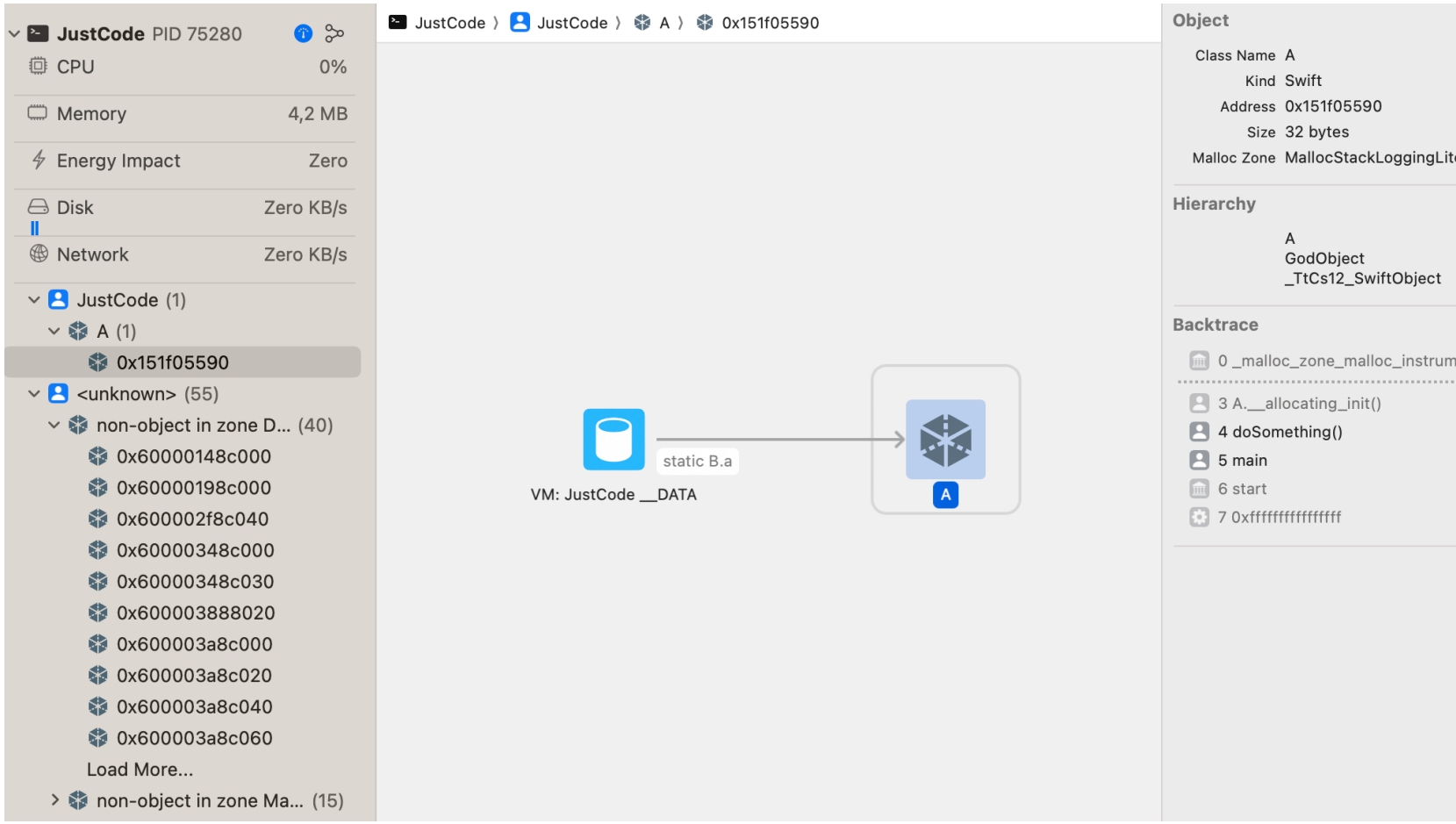
```
init() {
    self.completionBlock = {

        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()
```



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

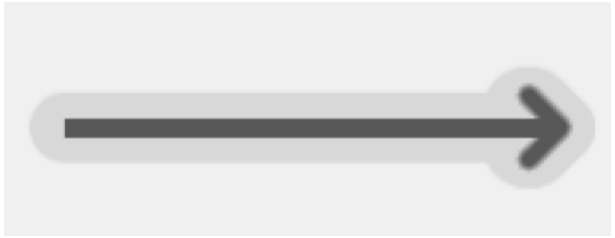
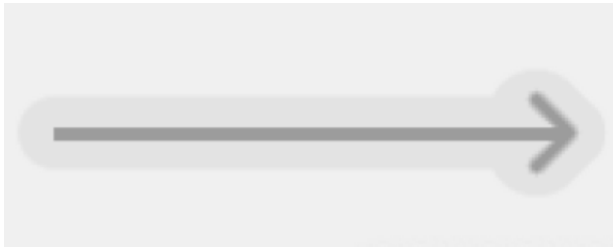
Алгоритм поиска (простая реализация)

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. Шикуем на пляже

Кто теперь умеет бороться с утечками?

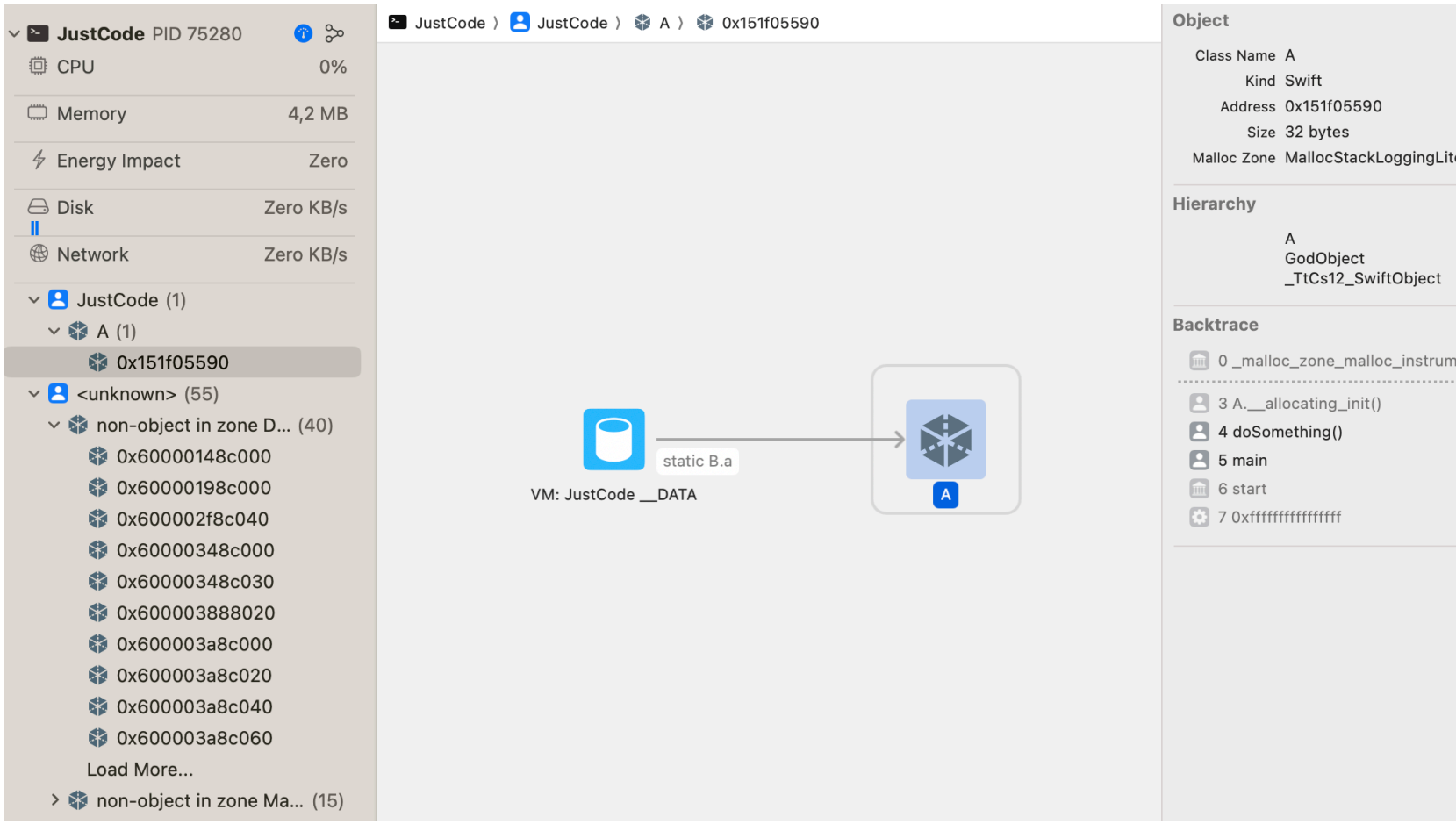
Ловушки Xcode

Легенда карты (Memory Graph)

Стрелка в графе	Связь в рантайме
	Strong
	Weak
	Выбранная связь в дебаггере

Неверная стрелка: статическое поле

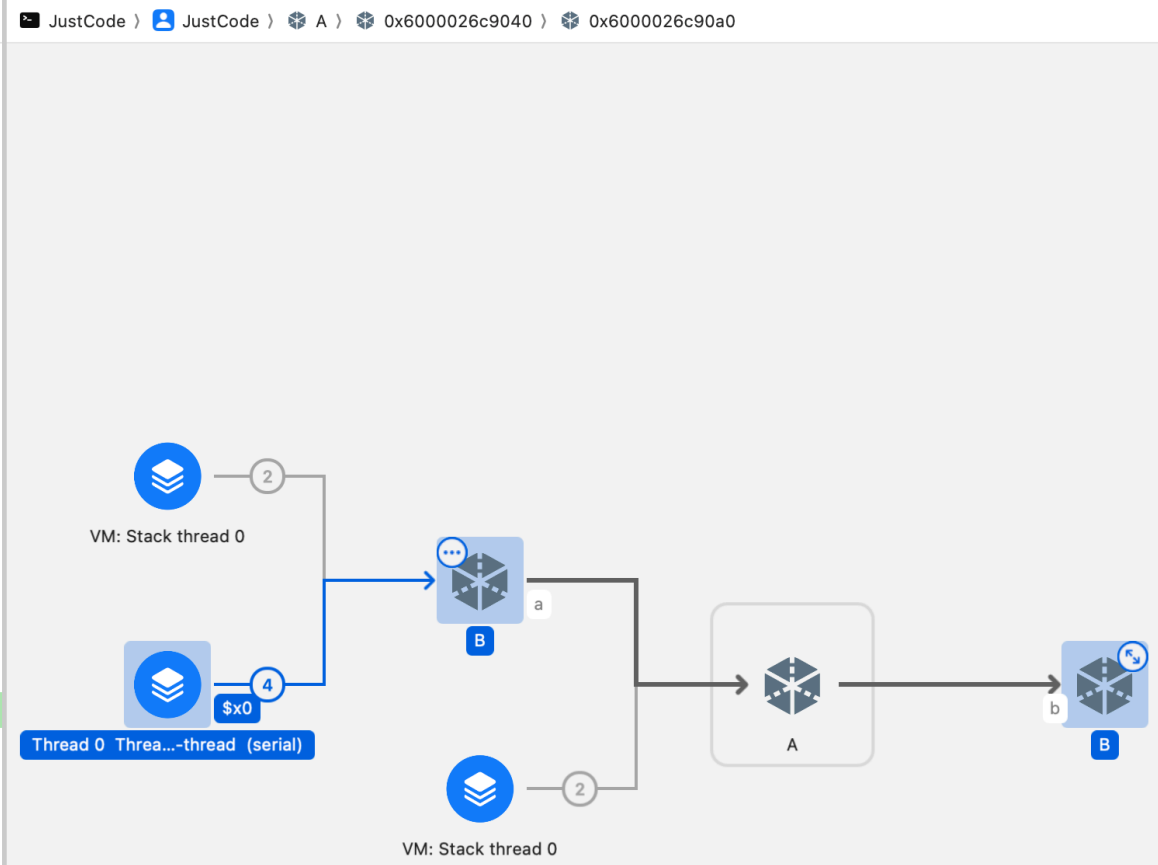
```
class A: GodObject {
    var b: B?
}

class B {
    static var a: A?
}

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

doSomething()
```



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 }                                   Thread 1: step over
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

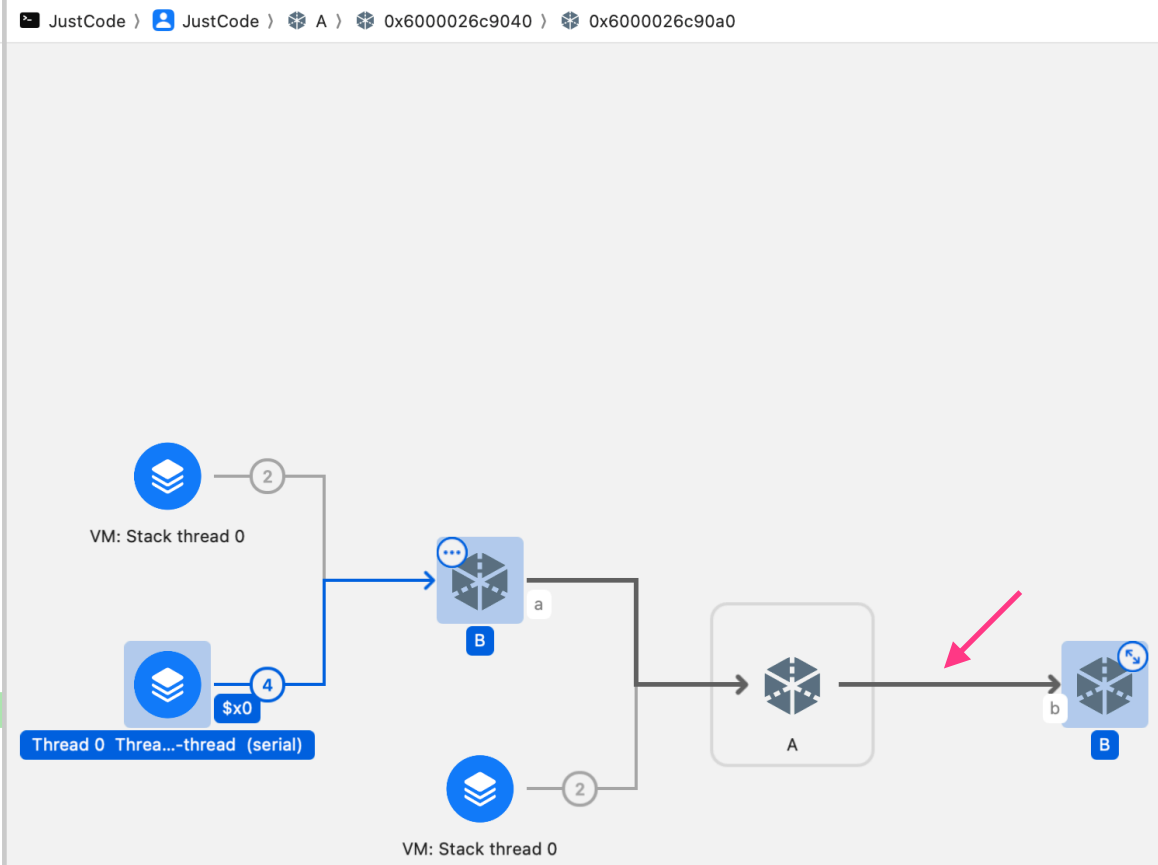
B
_TtCs12_SwiftObject

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 }                                   Thread 1: step over
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

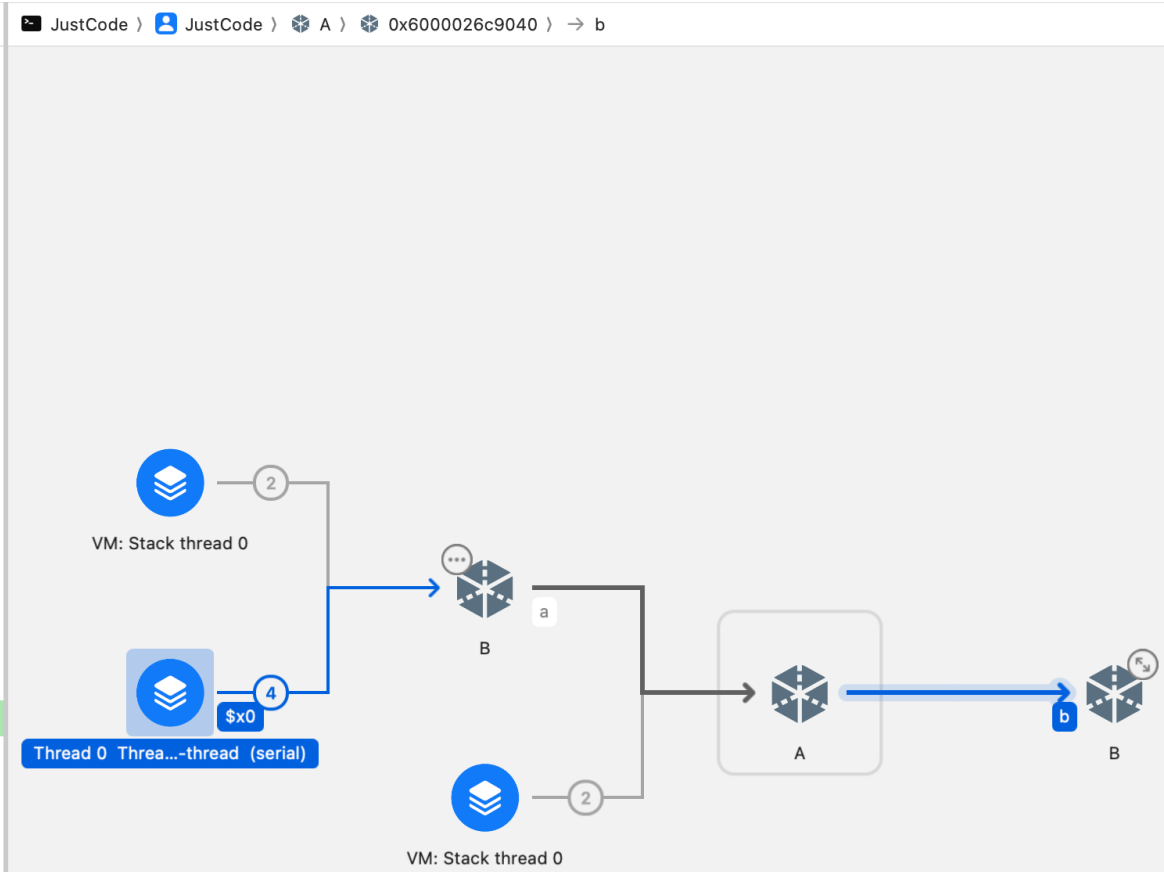
B
_TtCs12_SwiftObject

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 }                                   Thread 1: step over
133
134 doSomething()
135
136 assert(A.objectsCount == 0)
137
```



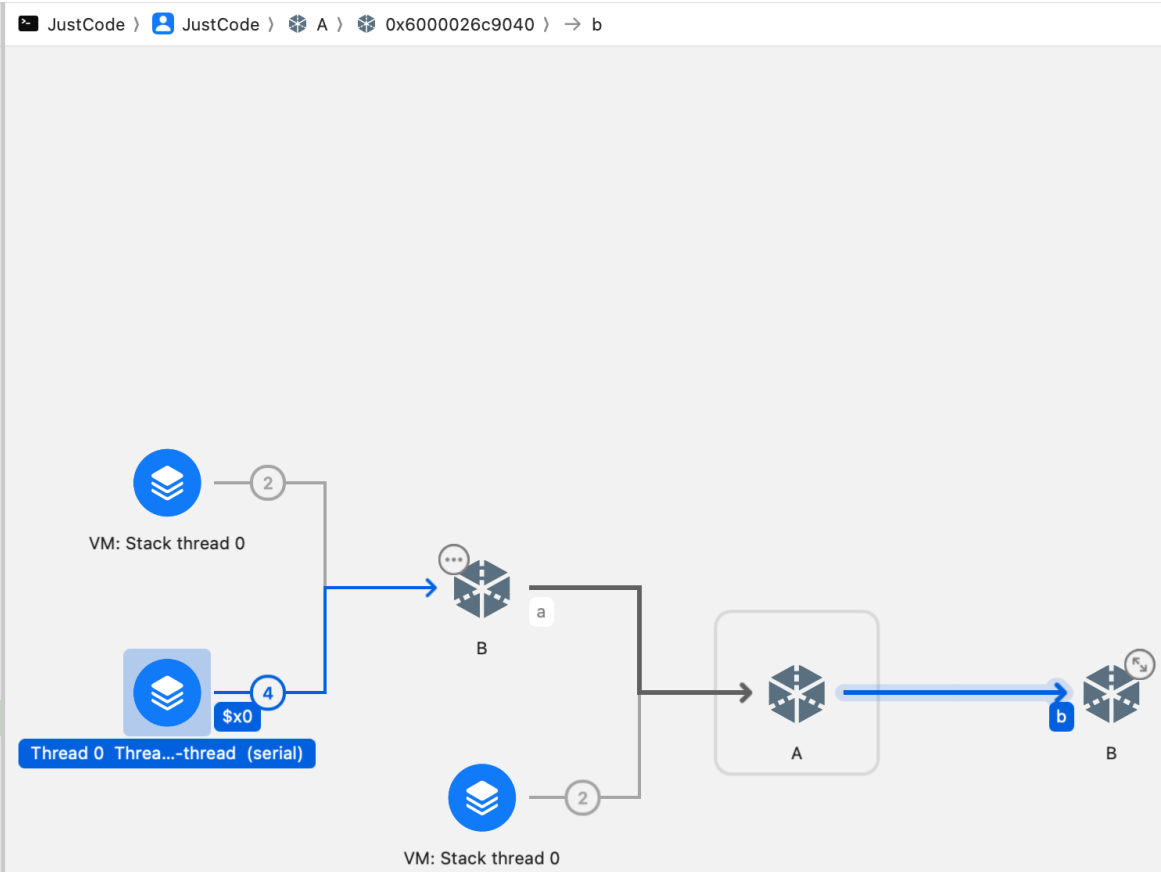
Reference Details

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 }                                   Thread 1: step over
133
134 doSomething()
135
136 assert(A.objectsCount == 0)
137
```

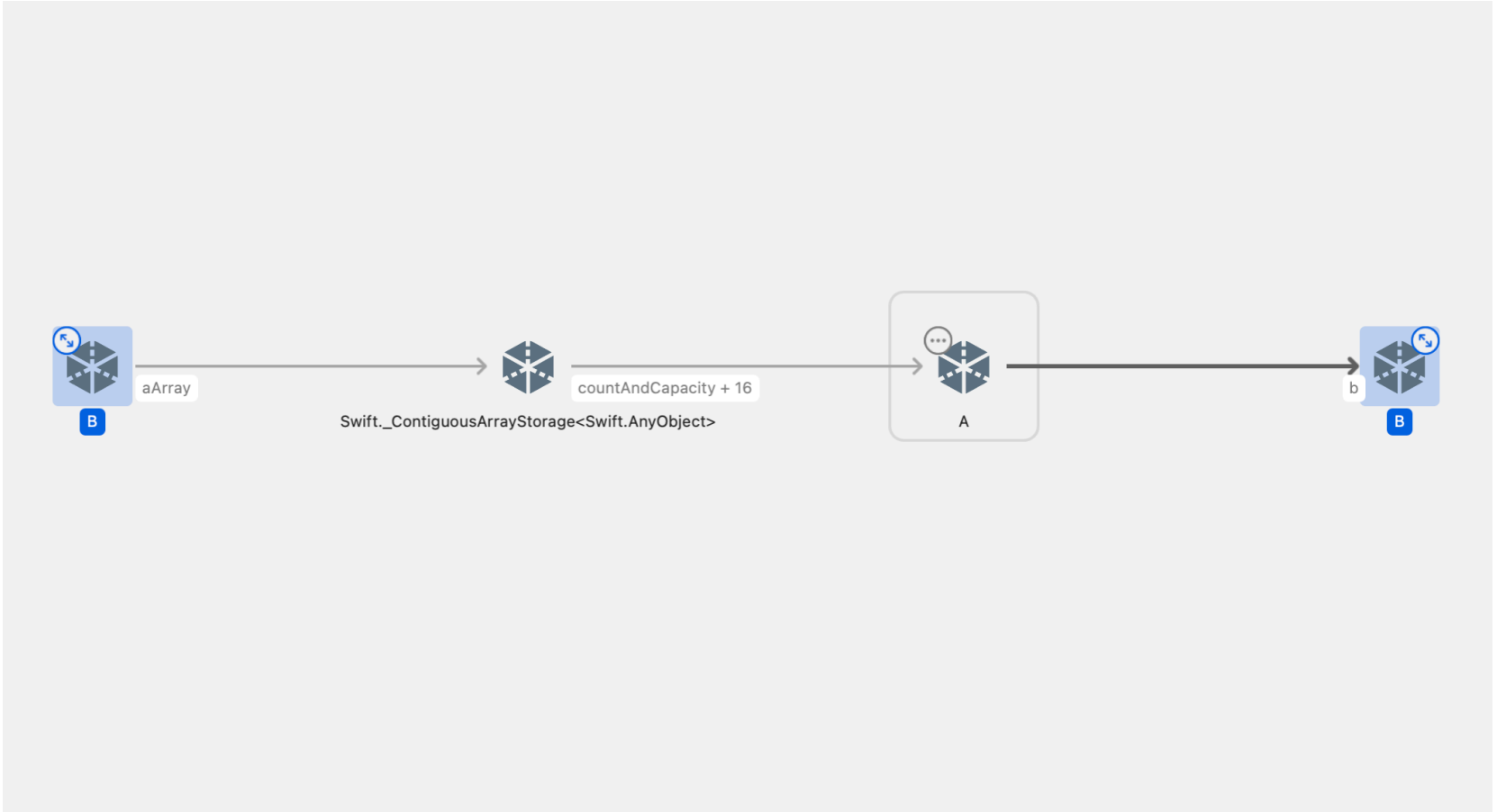


Reference Details

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

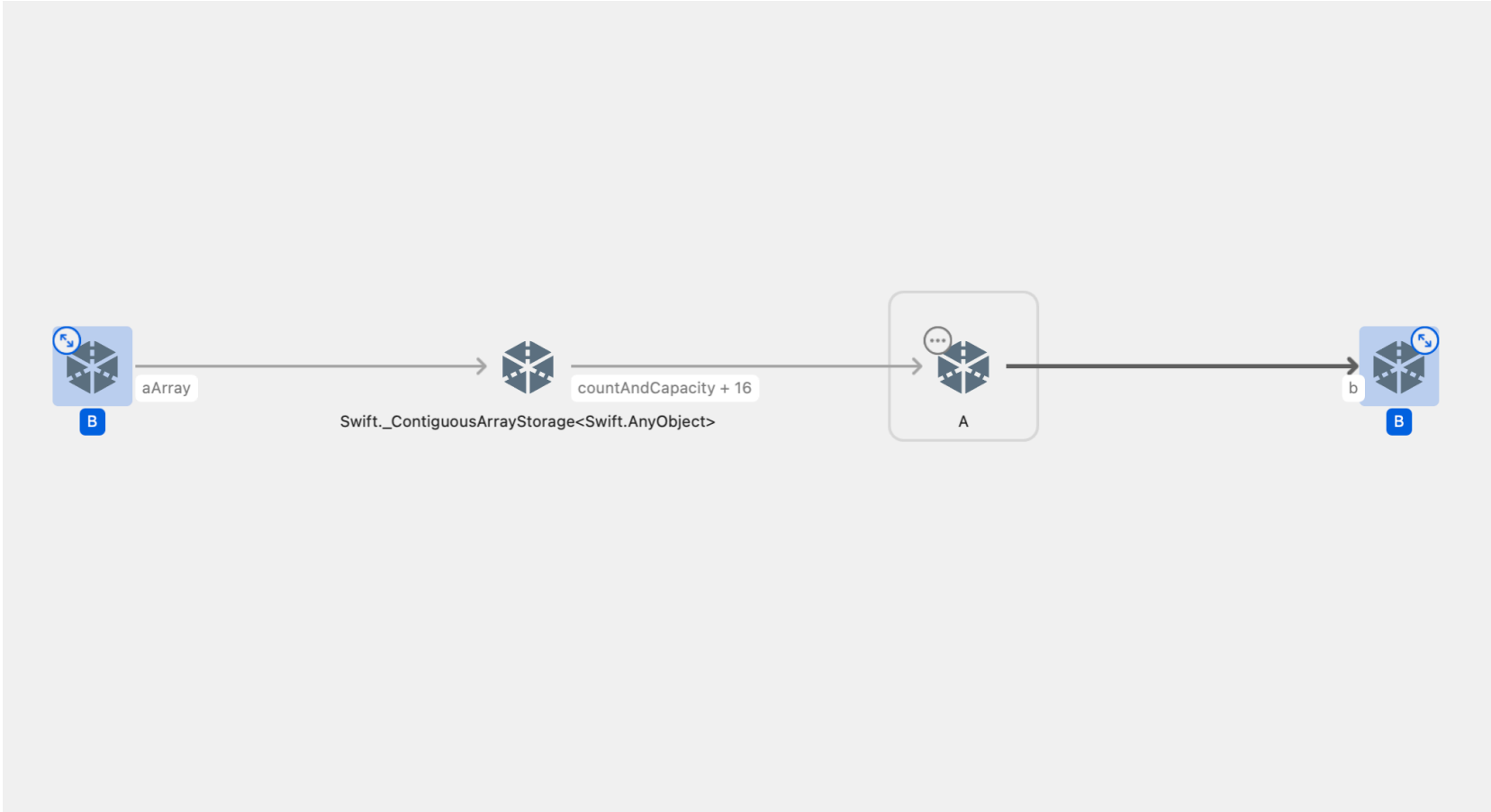
Неверная стрелка: реализация Swift коллекций

```
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()
```



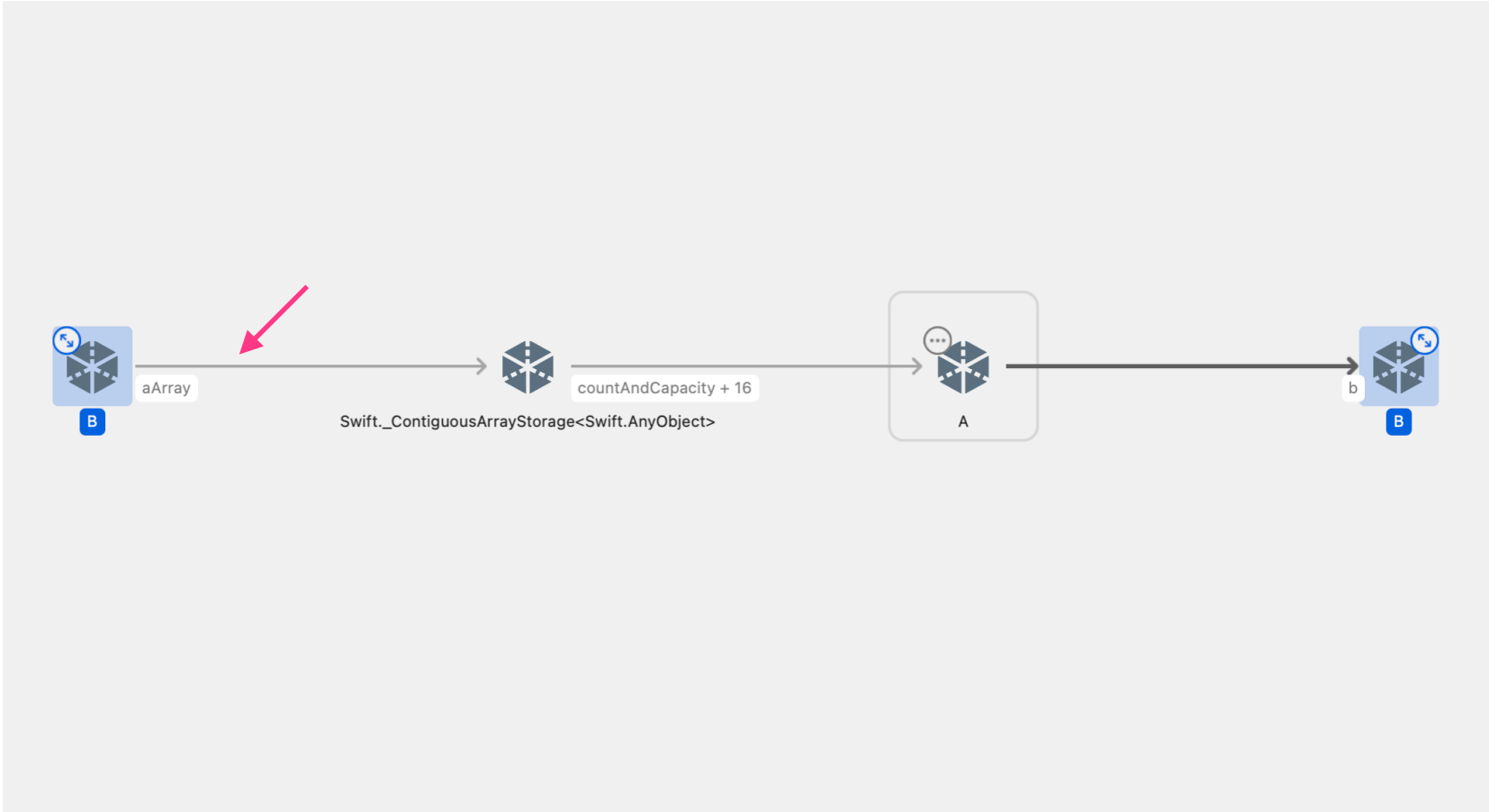
Неверная стрелка: реализация Swift коллекций

```
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

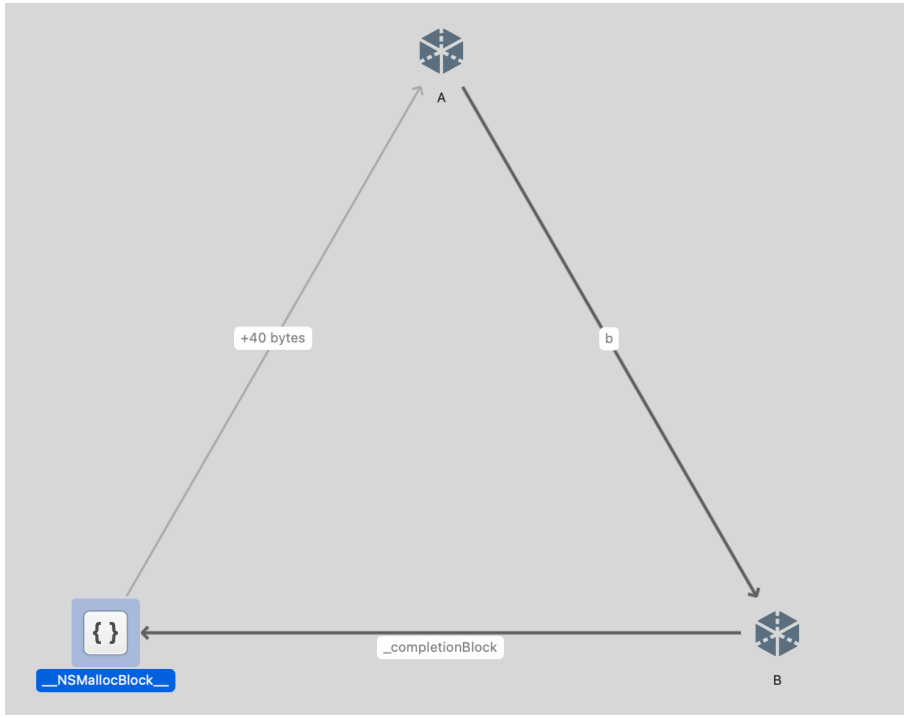
```
class A: GodObject {
    var b: B?
}

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

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

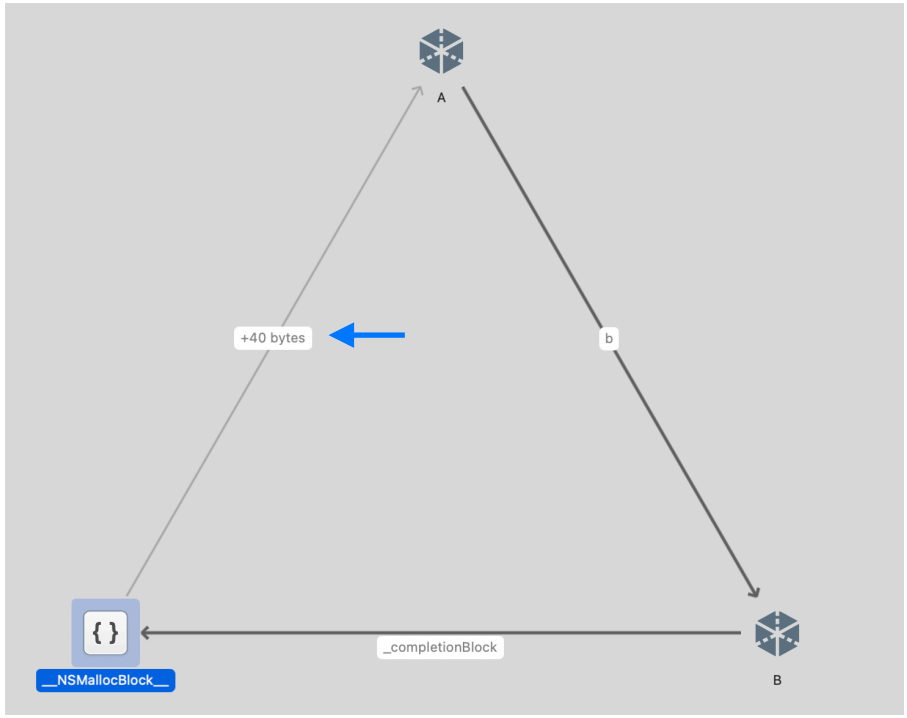
```swift
class A: GodObject {
    var b: B?
}

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

doLeak()
```



Неверная стрелка: Связь Objective C & Swift

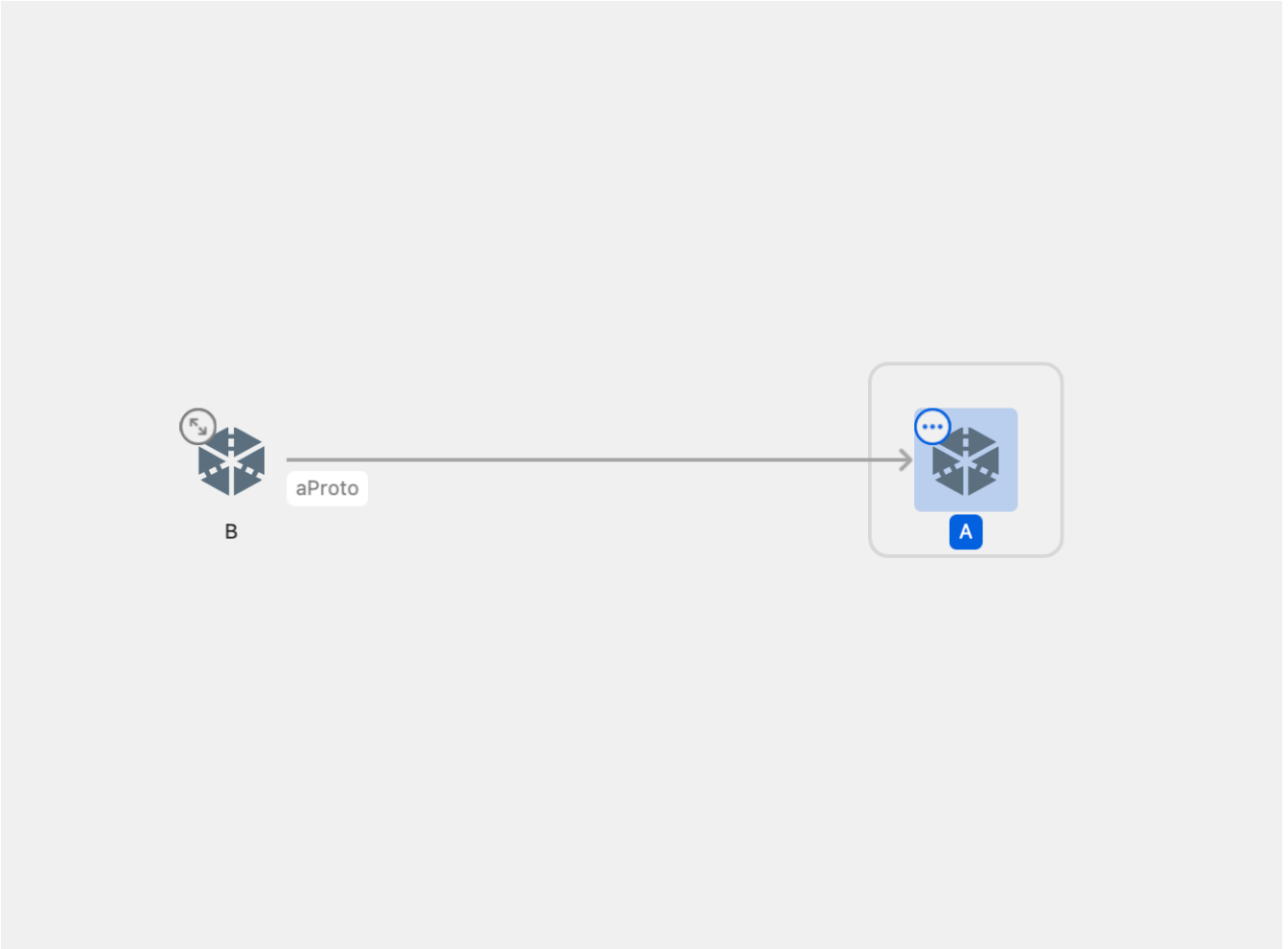
```objc
@interface B : NSObject
@property (nonatomic) void (^completionBlock)(void);
- (instancetype)initWithBlock:(void (^)(void))block;
@end
```

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

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

doLeak()
```



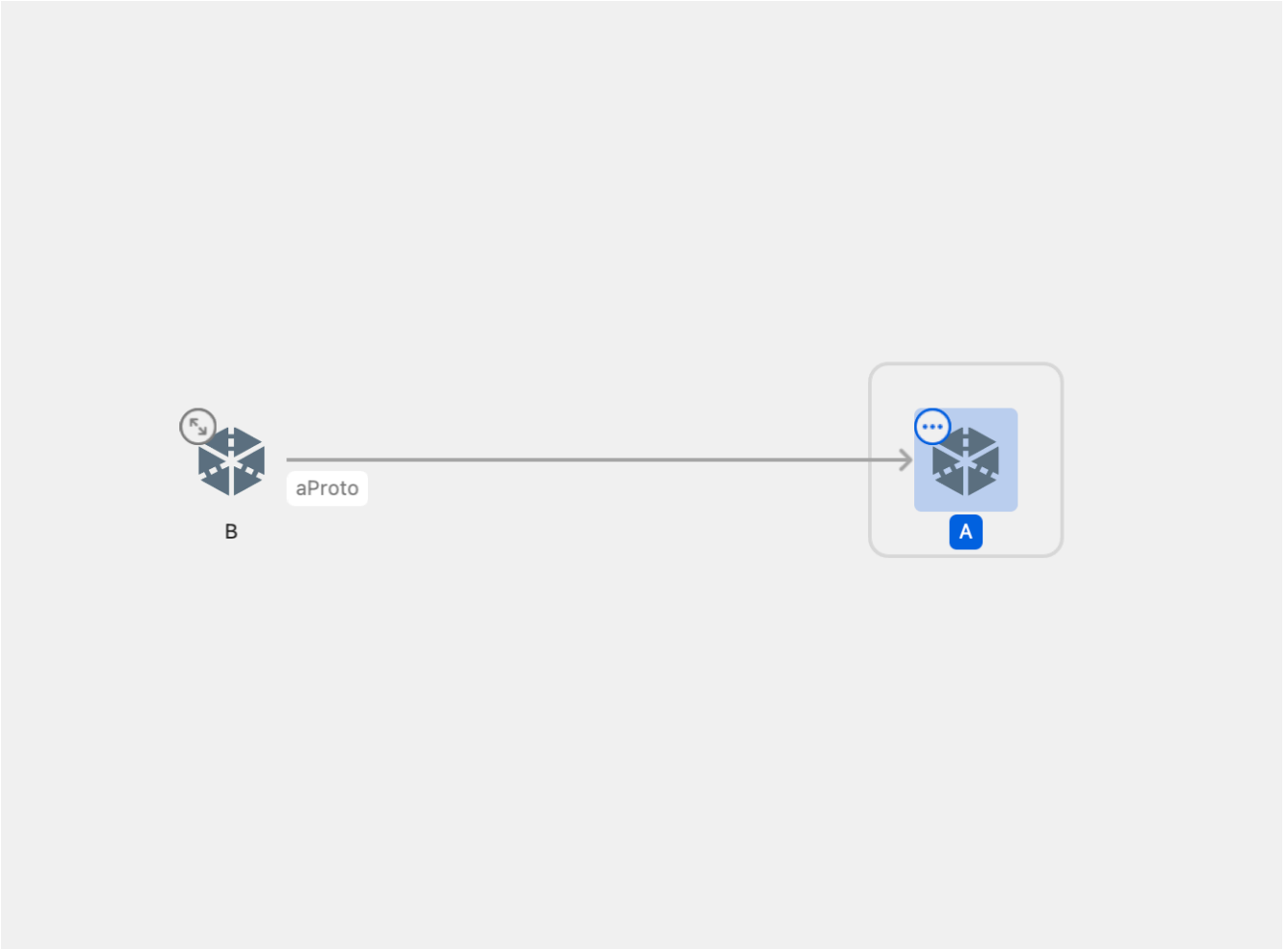
Неверная стрелка: Протоколы

```
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)
```



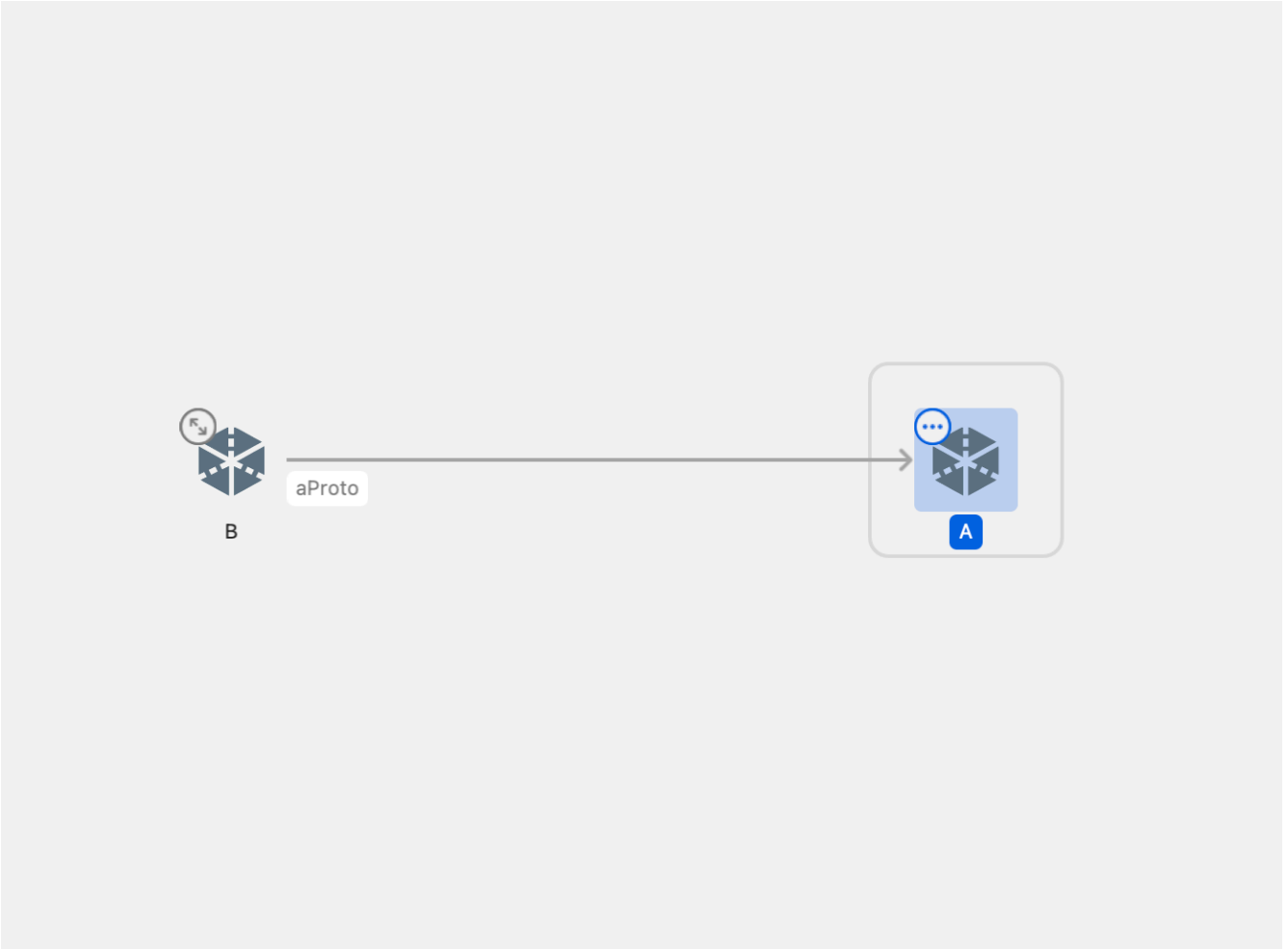
Неверная стрелка: Протоколы

```
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)
```



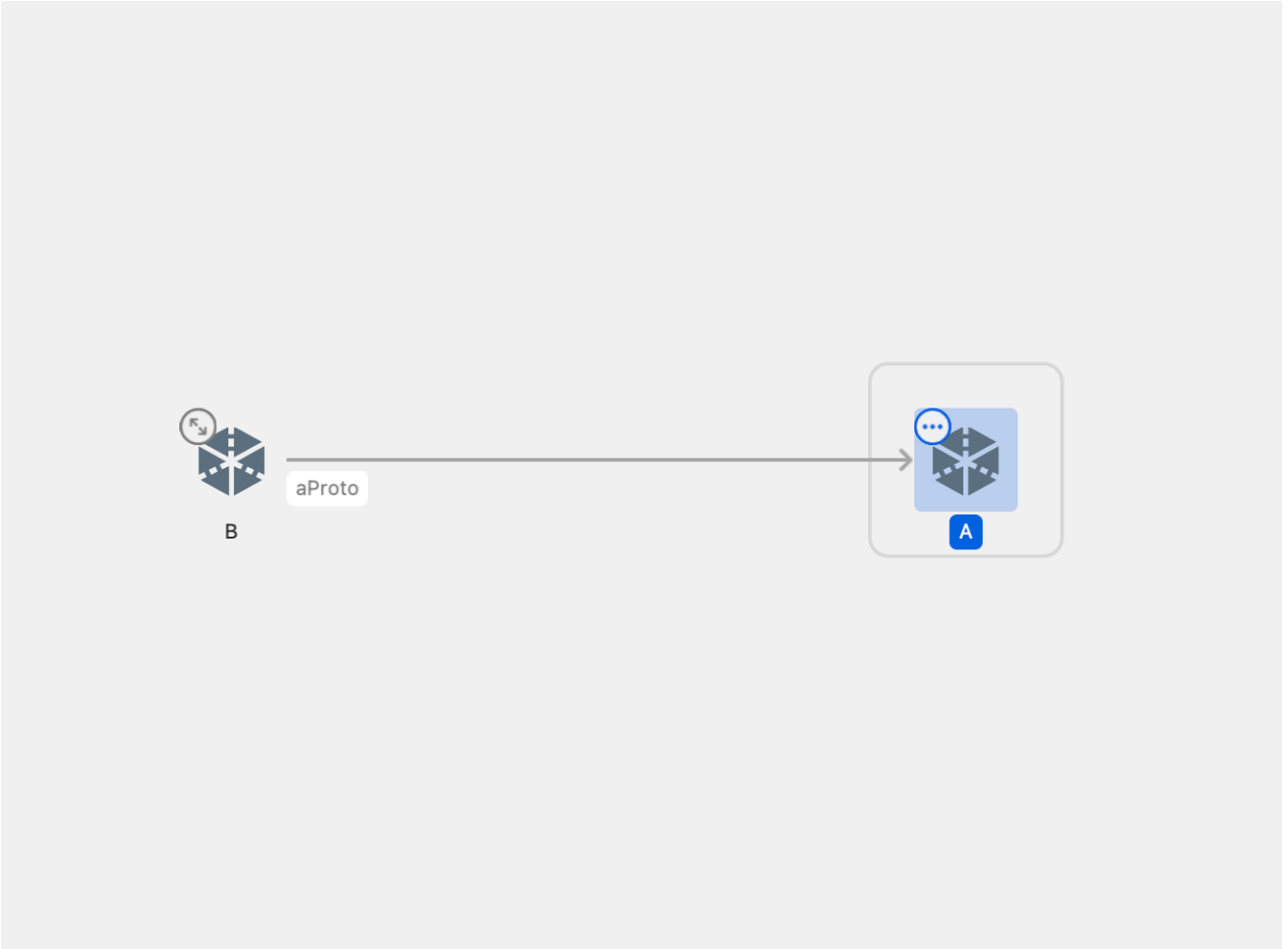
Неверная стрелка: Протоколы

```
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)
```



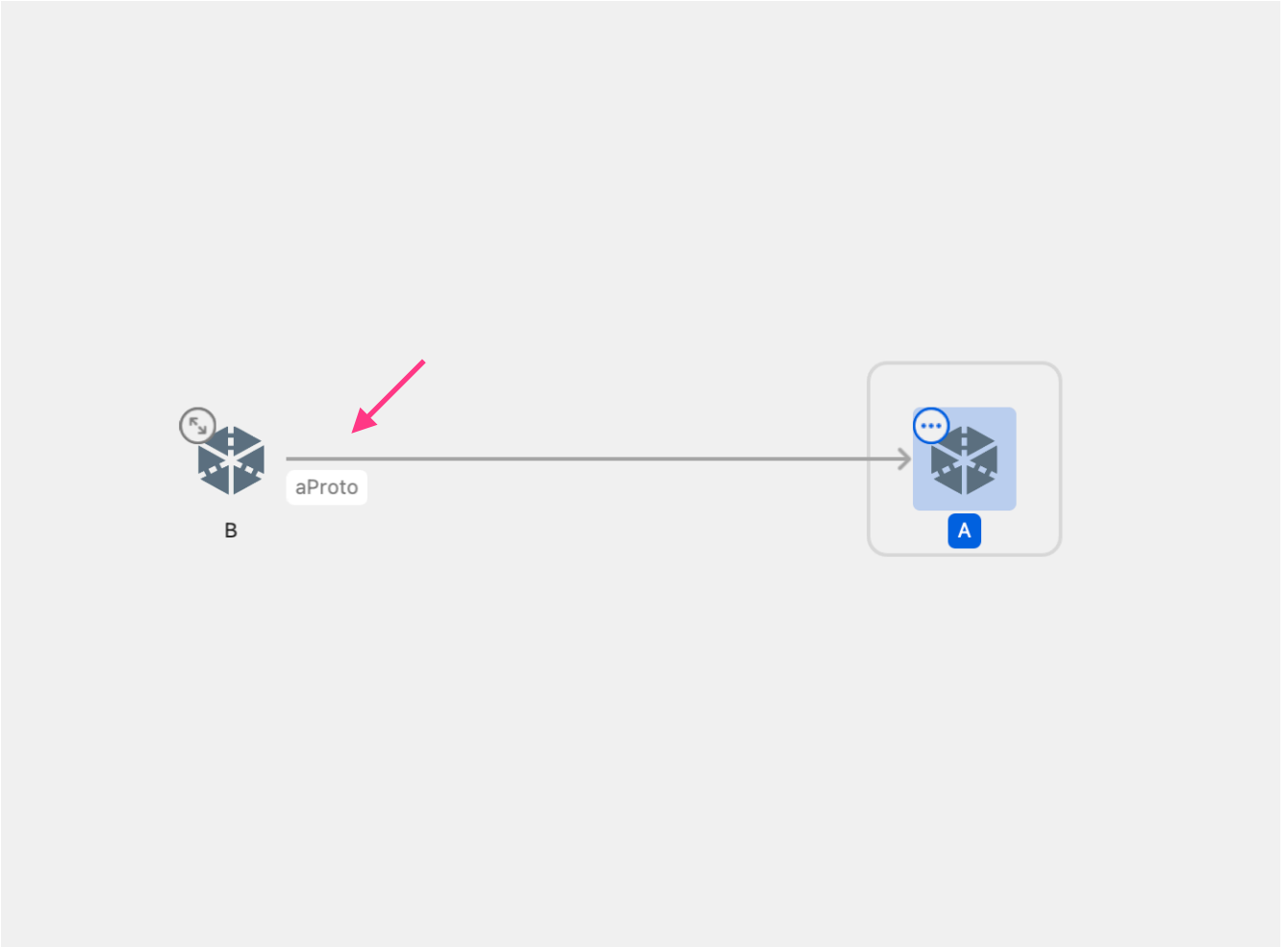
Неверная стрелка: Протоколы

```
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)
```

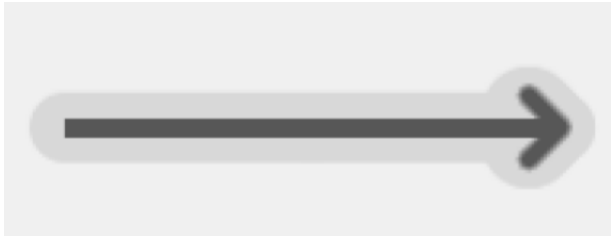


Неверная стрелка: Протоколы

```
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)
```

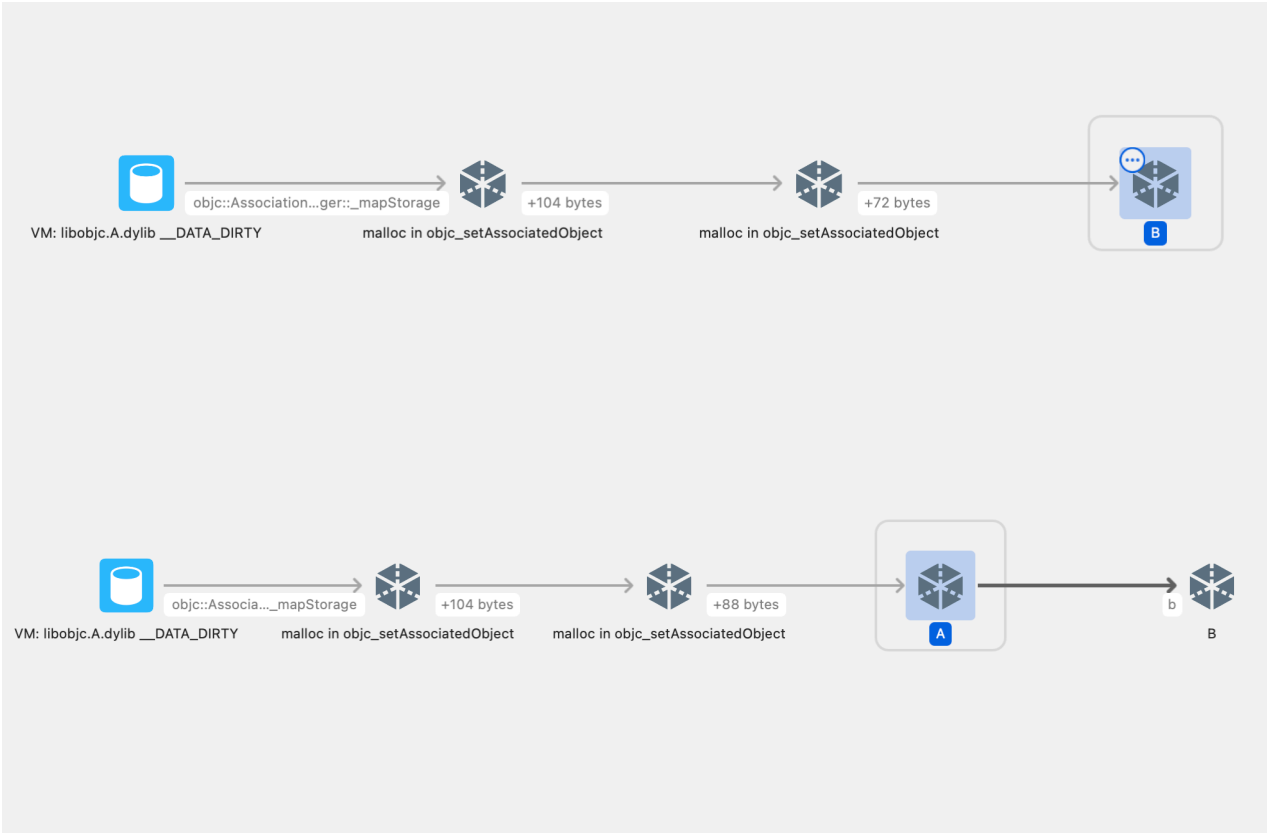


Легенда карты (Memory Graph)

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

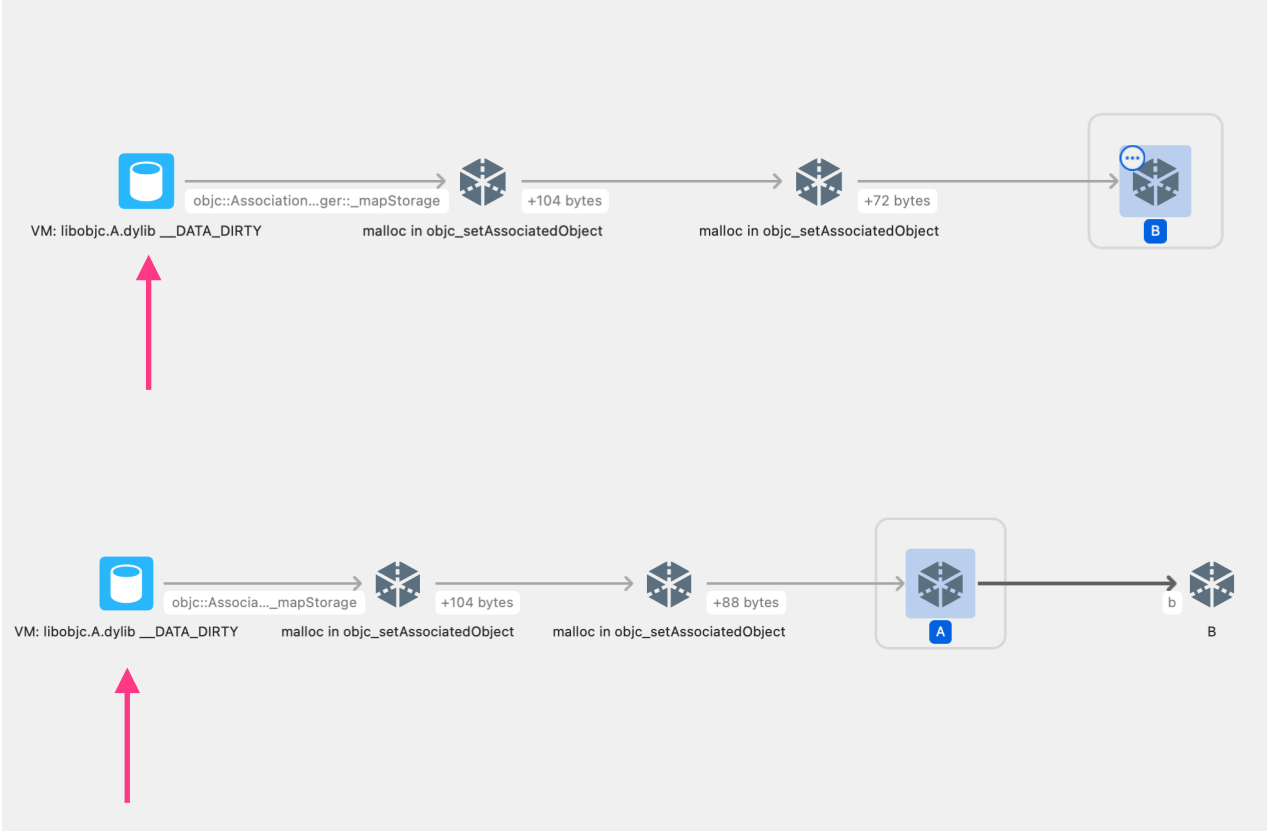
Самодельные хранимые переменные

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

class B {}

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

doSomething()
```



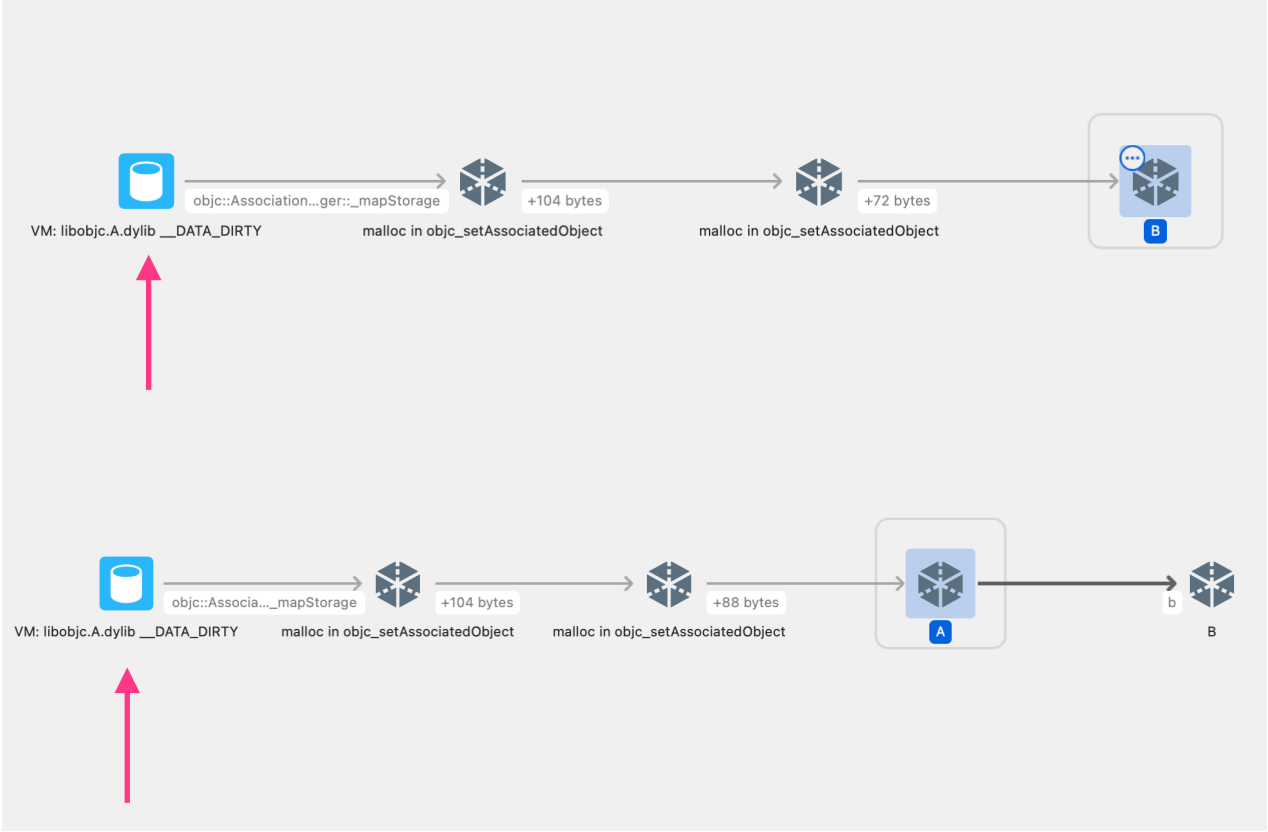
Самодельные хранимые переменные

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

class B {}

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

doSomething()
```



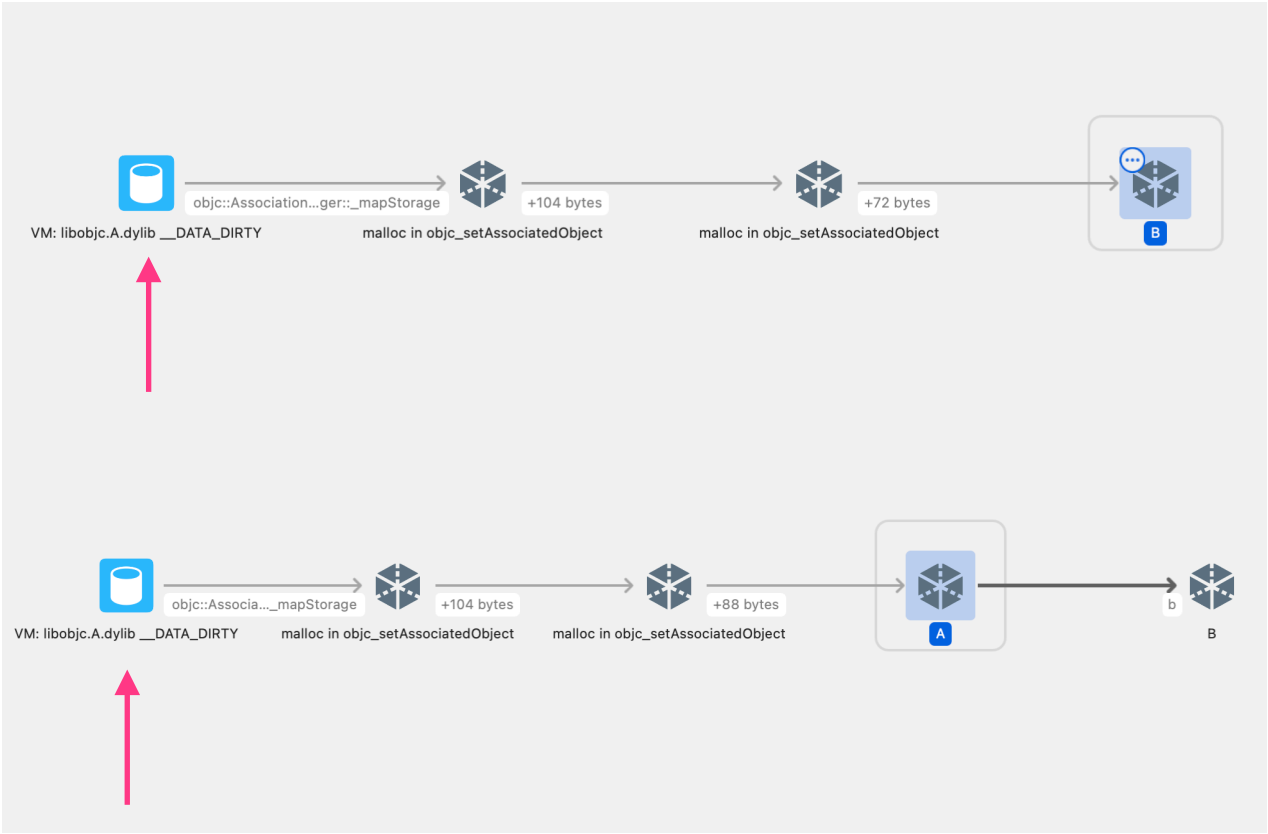
Самодельные хранимые переменные

```
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
            )
        }
    }
}
```



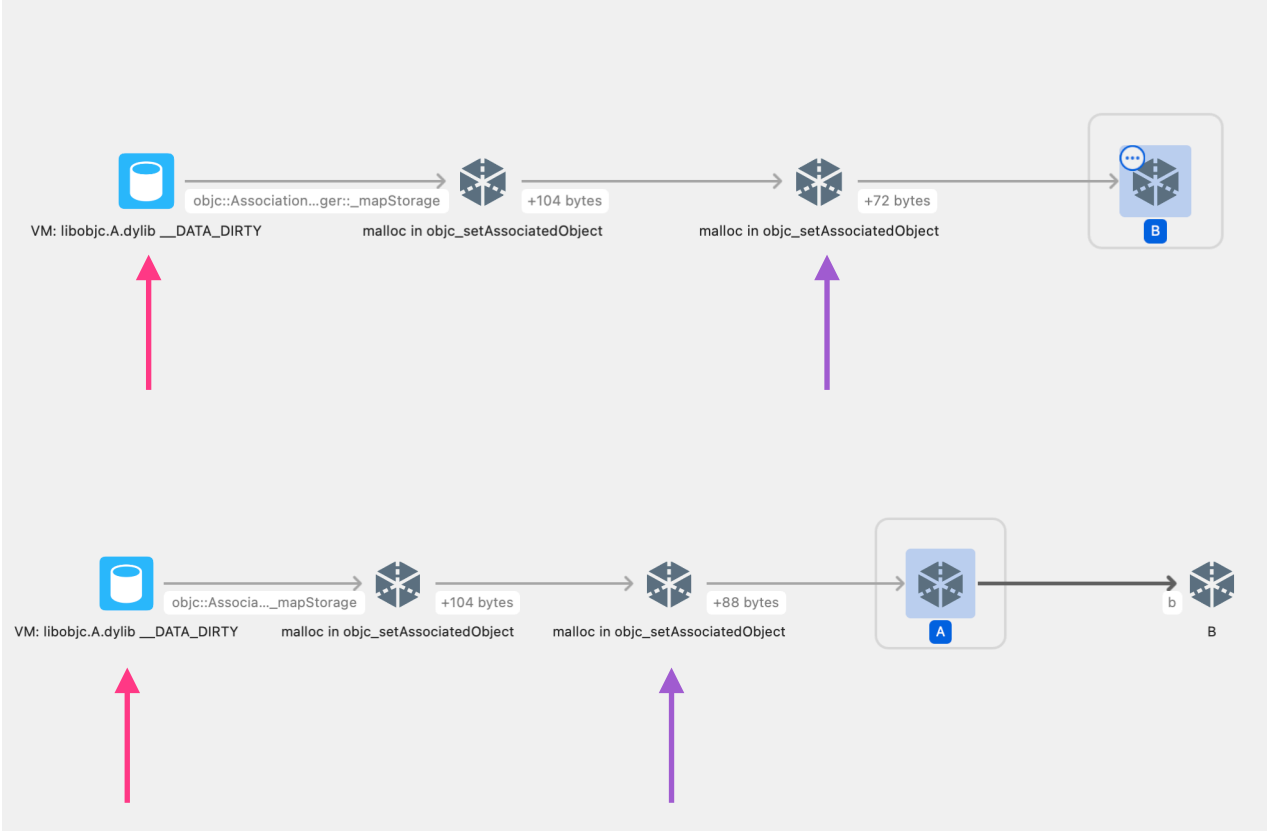
Самодельные хранимые переменные

```
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
            )
        }
    }
}
```



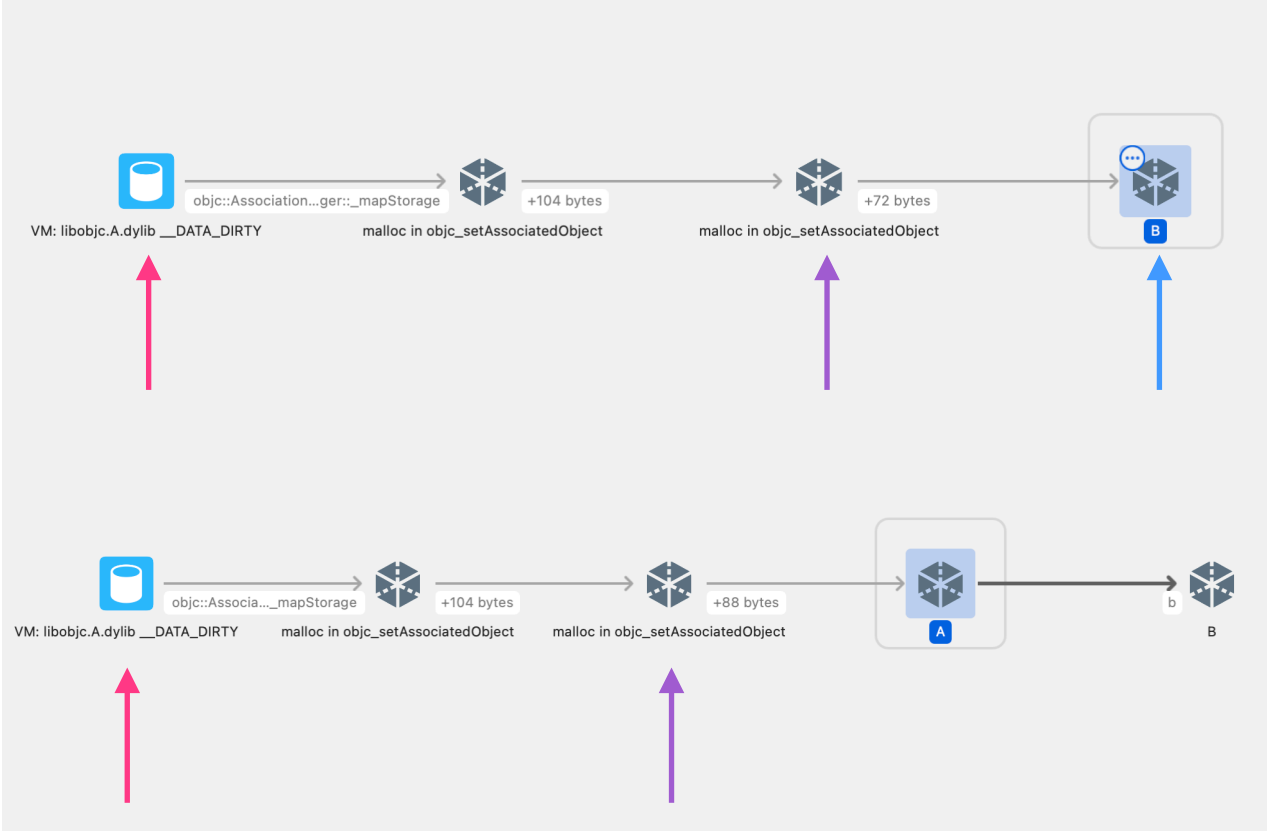
Самодельные хранимые переменные

```
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
            )
        }
    }
}
```



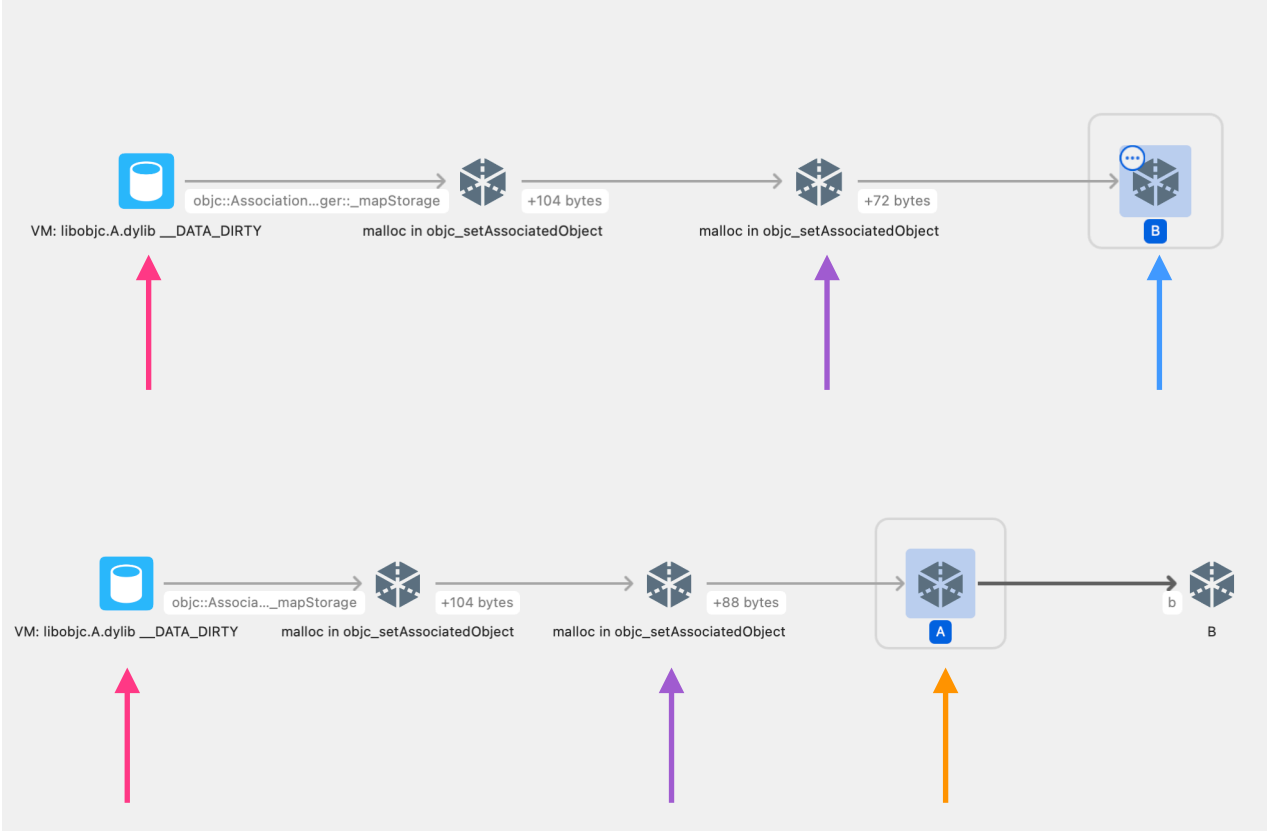
Самодельные хранимые переменные

```
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 MPов, 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 | Filter by agent name

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

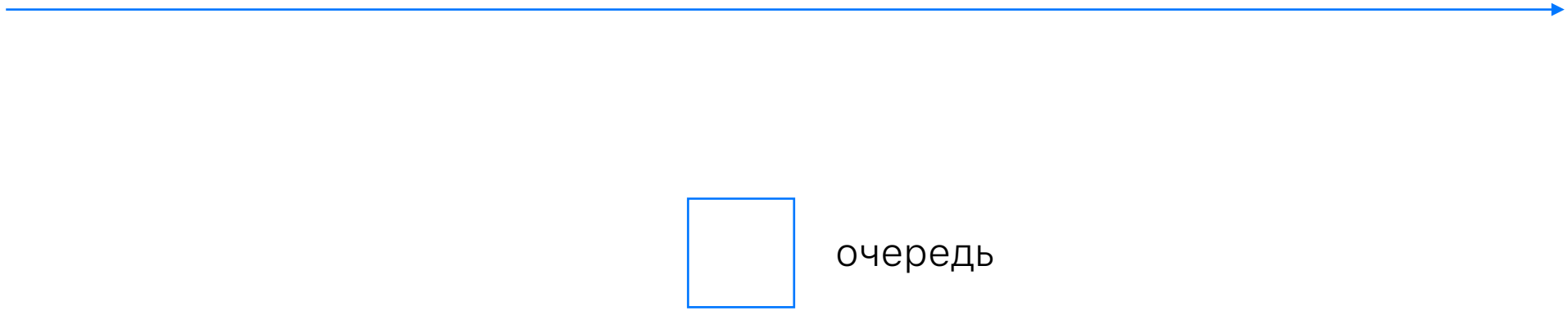
- Утечки появляются постоянно
- Необходимо оперативно реагировать
- Необходимо постоянно отвлекаться от своих задач
- Люди повторяют свои ошибки

Жизненный цикл утечки

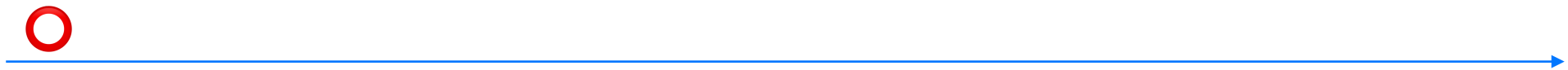
- Утечки появляются постоянно
- Необходимо оперативно реагировать
- Необходимо постоянно отвлекаться от своих задач
- Люди повторяют свои ошибки
- Утечки появляются постоянно

Создаём регламент по работе с утечками

Регламент

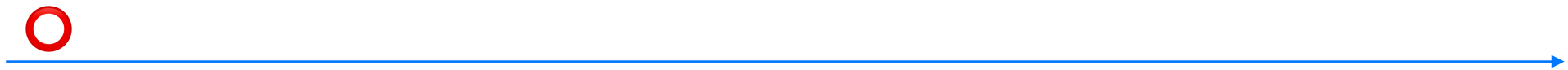


Регламент



очередь

Регламент



очередь

Регламент

очередь

Регламент

очередь

Регламент

очередь

Регламент

очередь

Регламент

очередь

Регламент

очередь

Регламент

очередь

Регламент

очередь

Регламент

очередь

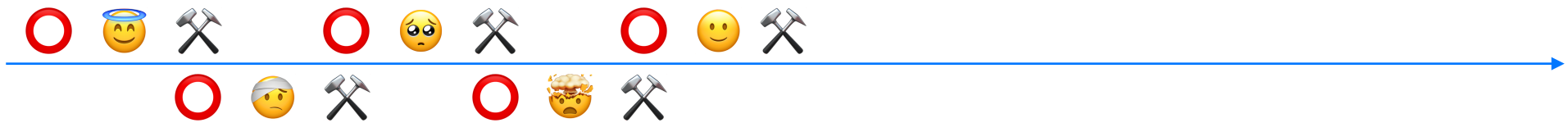
Регламент

очередь

Регламент

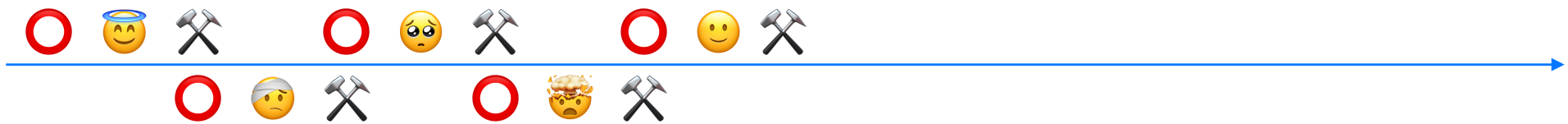
очередь

Регламент



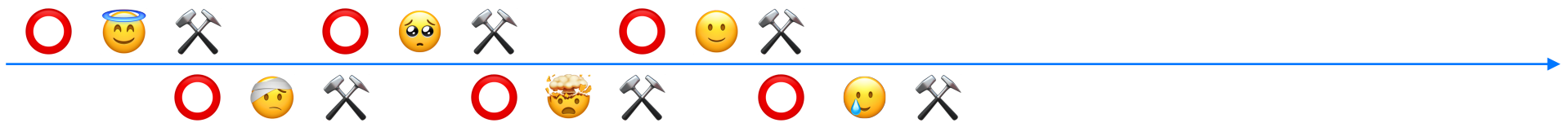
очередь

Регламент



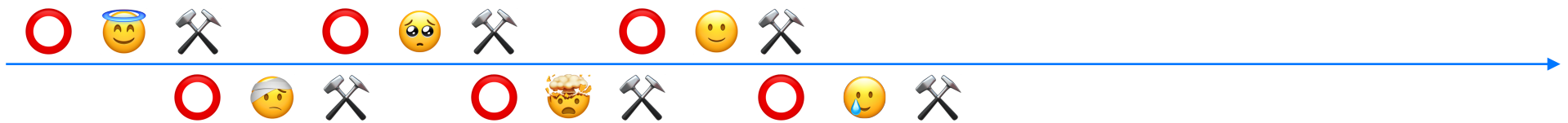
очередь

Регламент



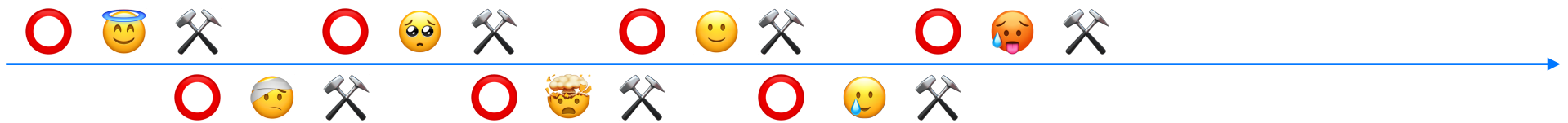
очередь

Регламент



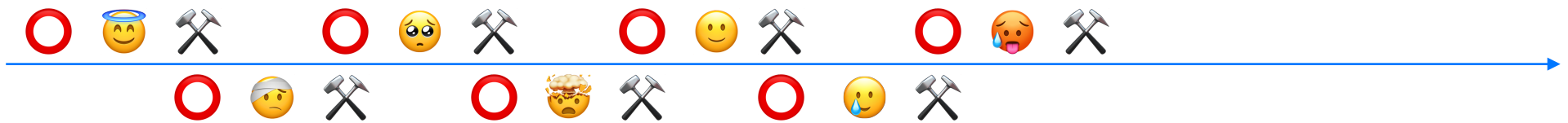
очередь

Регламент



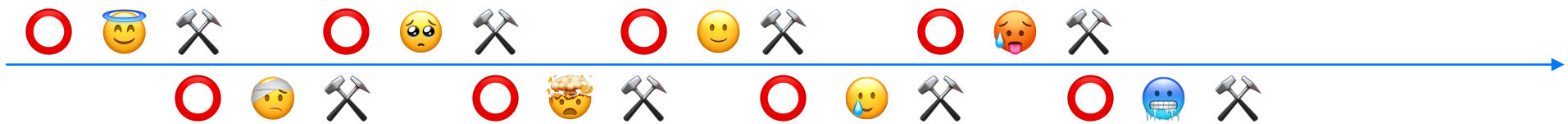
очередь

Регламент



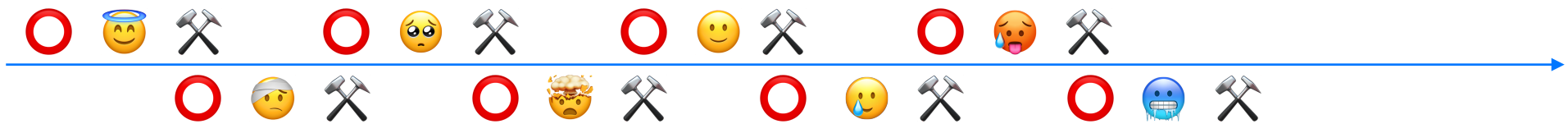
очередь

Регламент



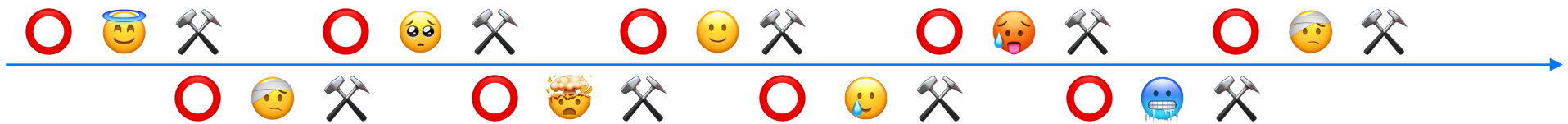
очередь

Регламент



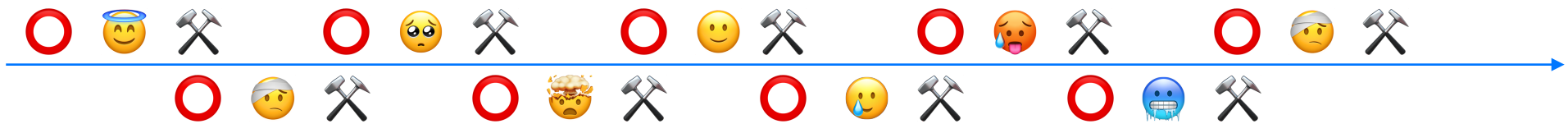
очередь

Регламент



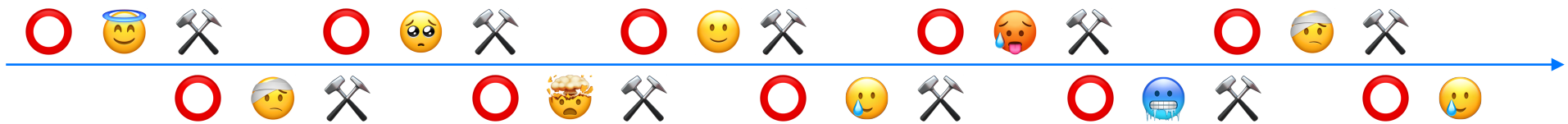
очередь

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

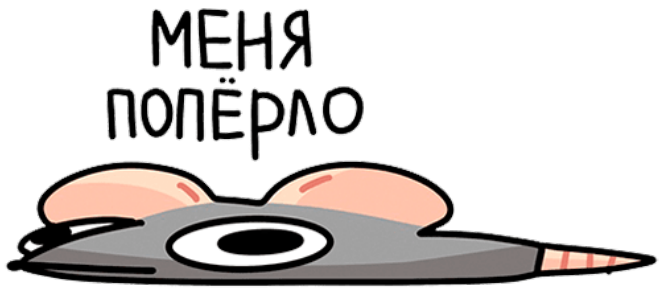
Распределение нагрузки

Скорость реакции

Закрепление ошибок

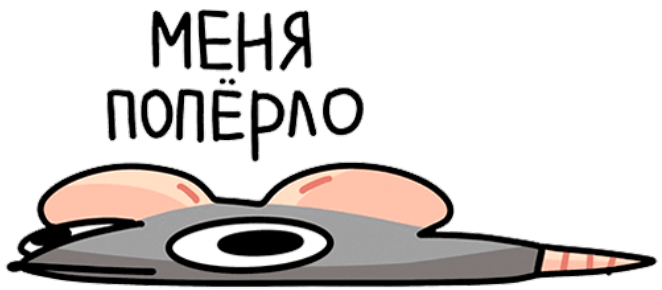
Моя борьба

Что обсудили?



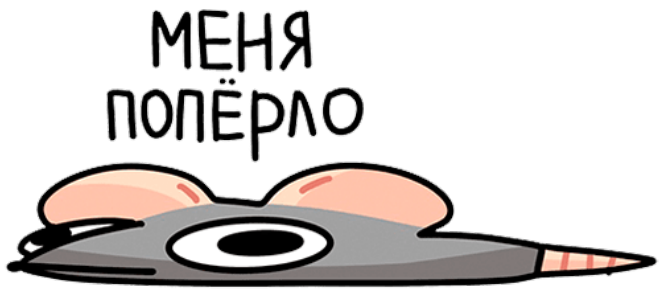
Что обсудили?

1. Что такое утечка



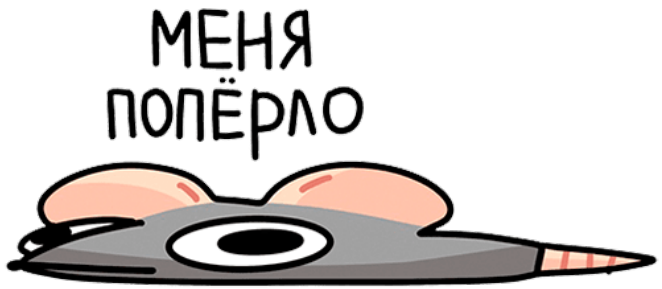
Что обсудили?

1. Что такое утечка
2. Графики с пользователей



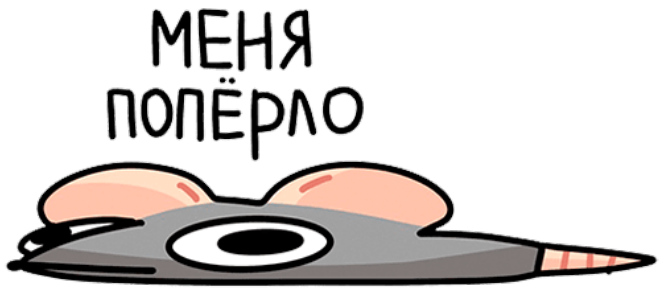
Что обсудили?

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



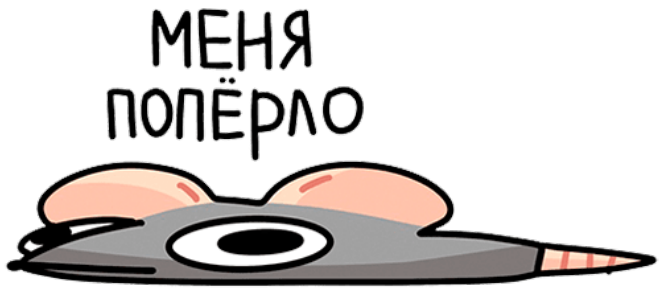
Что обсудили?

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



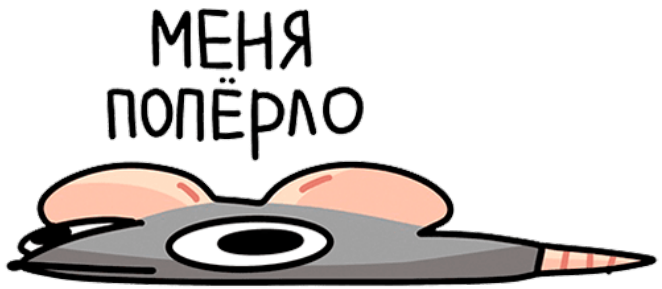
Что обсудили?

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



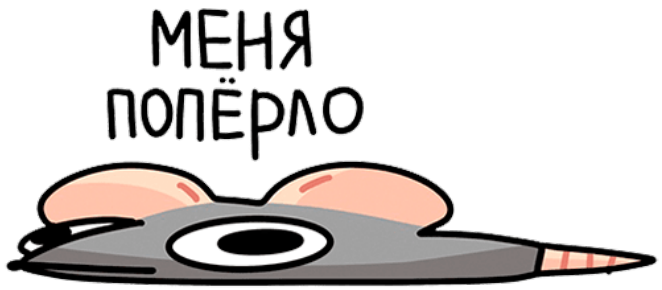
Что обсудили?

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