



Лекция 6.0

Concurrency

We can win the race



Very Low Level



pthreads

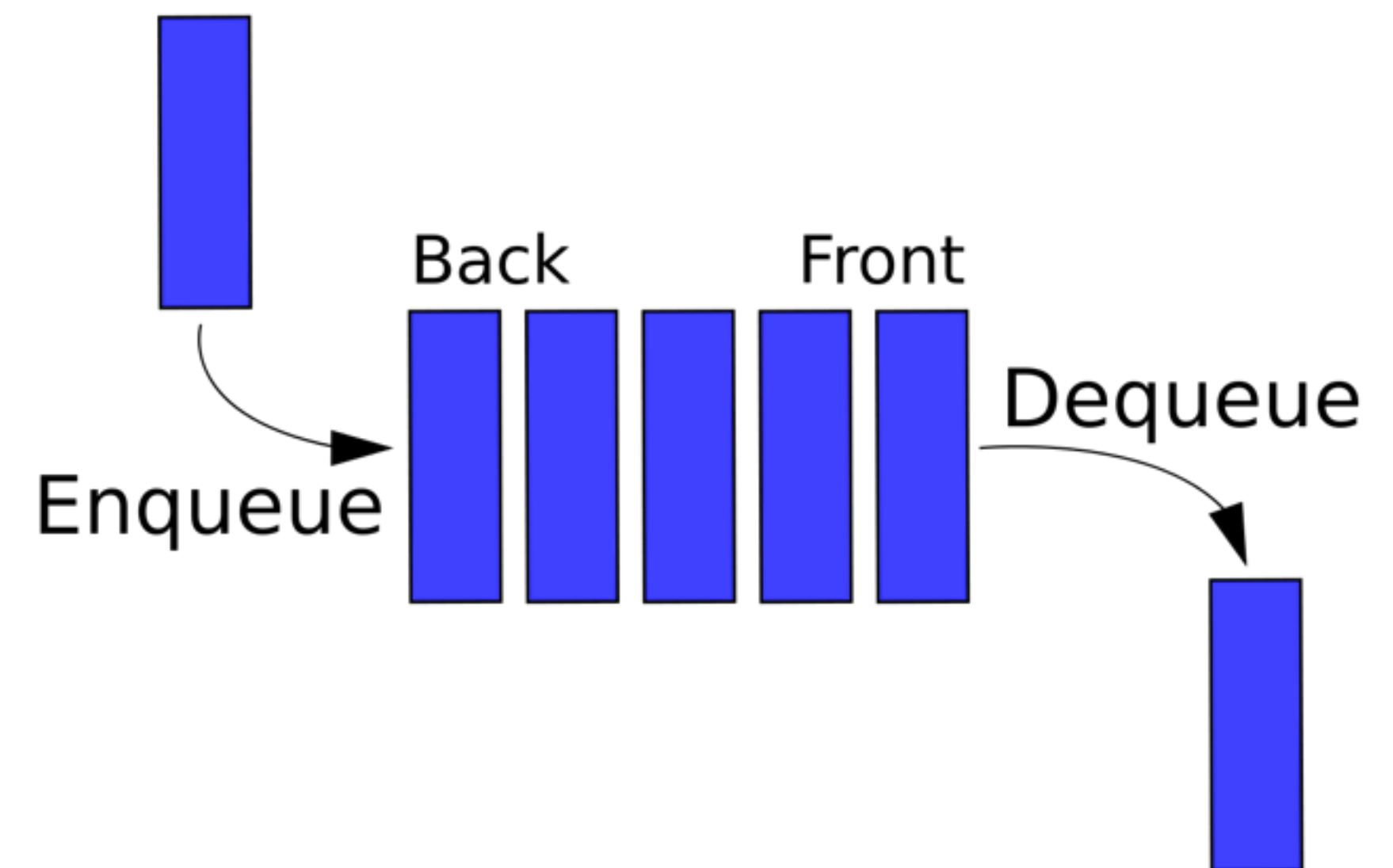


NSThread

Два варианта



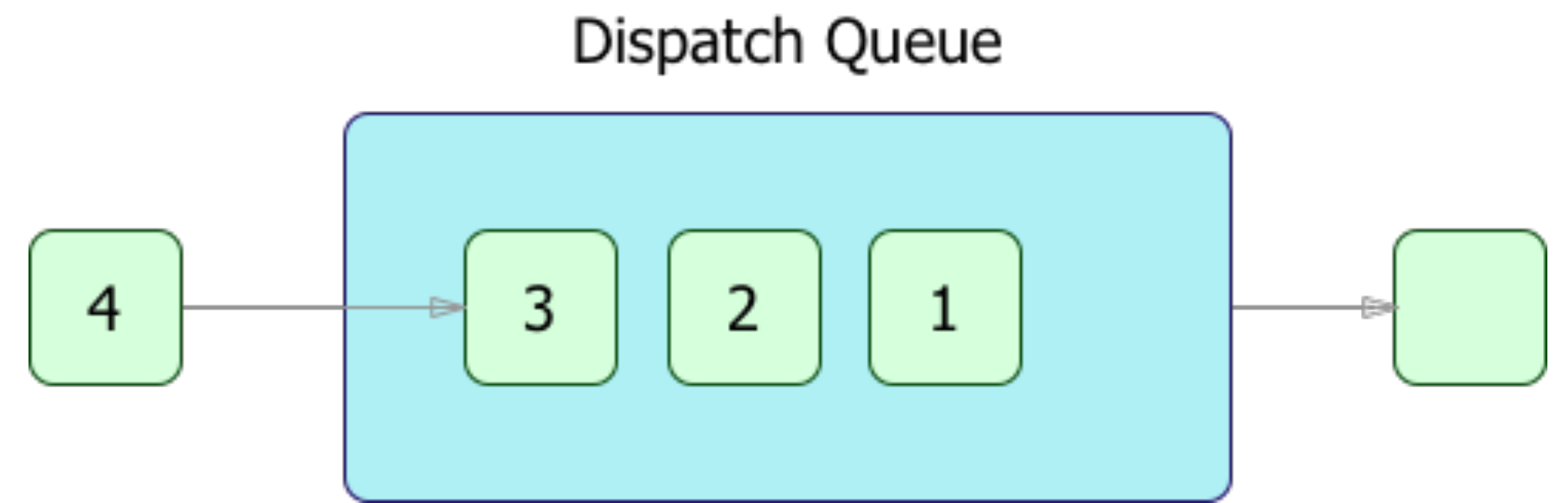
GCD (Low Level)



NSOperation

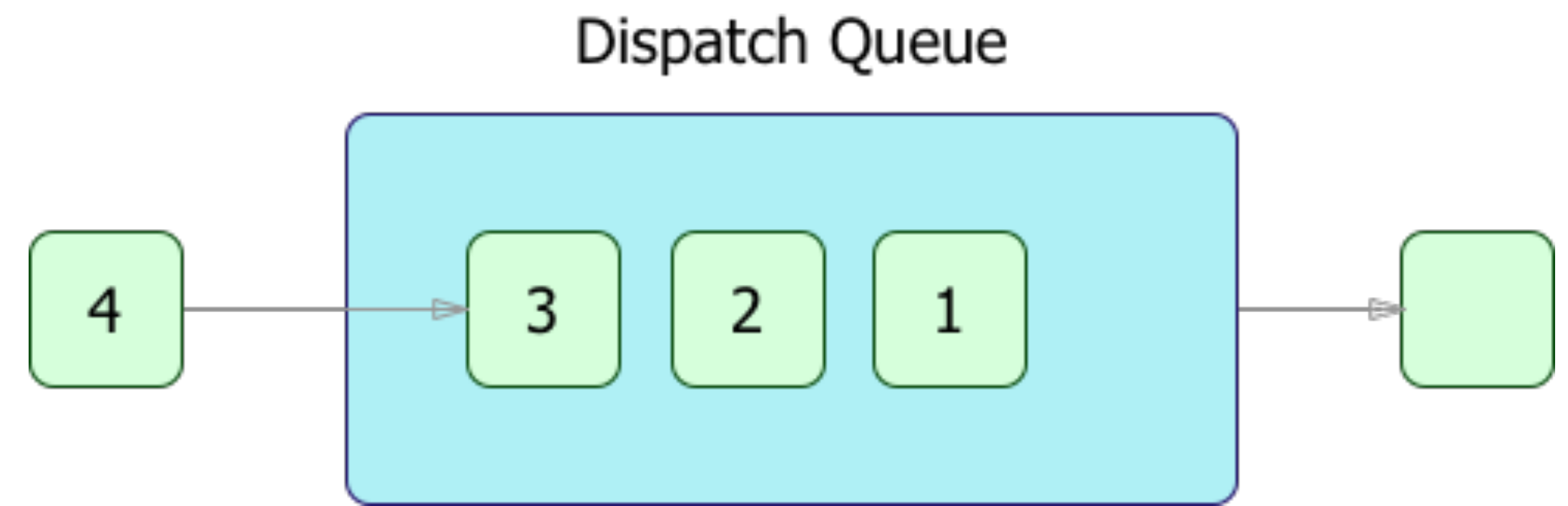
DispatchQueue

- › DispatchQueue
- › DispatchQueue.main
- › DispatchQueue.global(...)



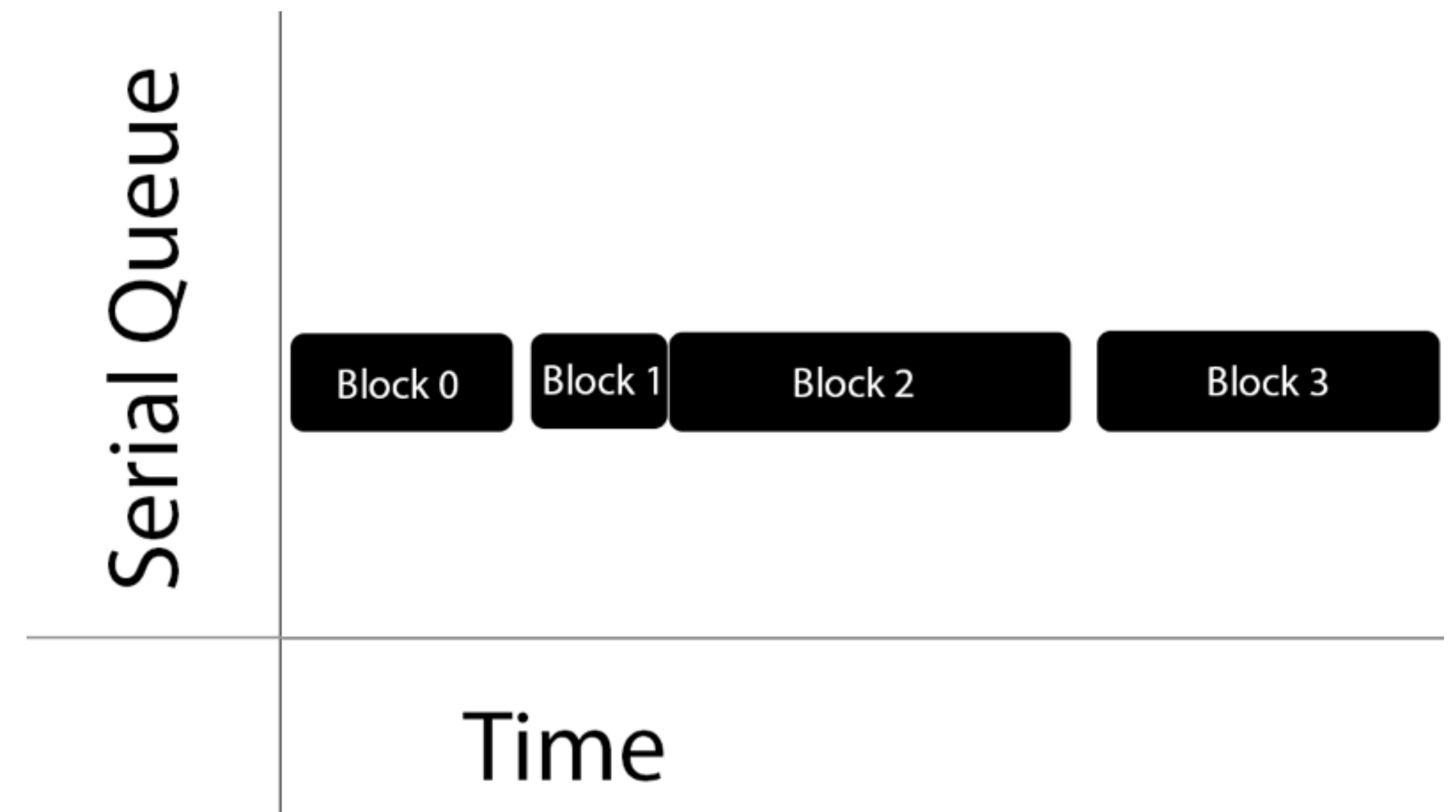
DispatchQueue SERIAL

› Задачи выполняются последовательно



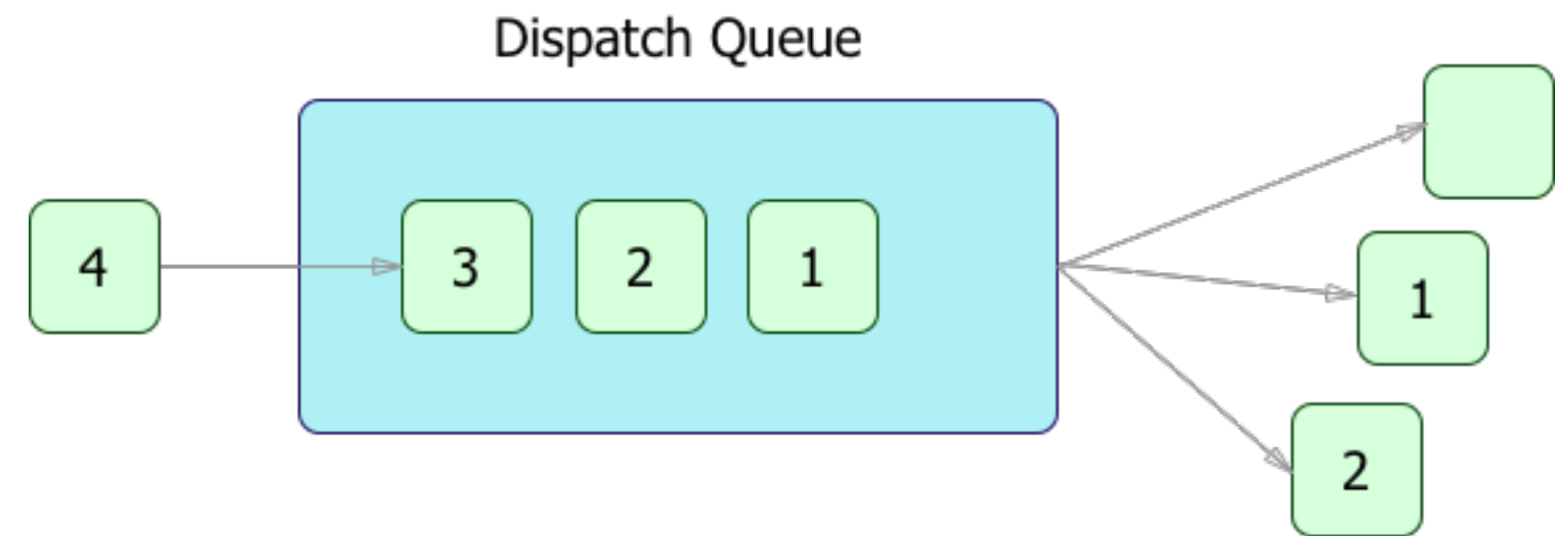
DispatchQueue SERIAL

› Задачи выполняются последовательно



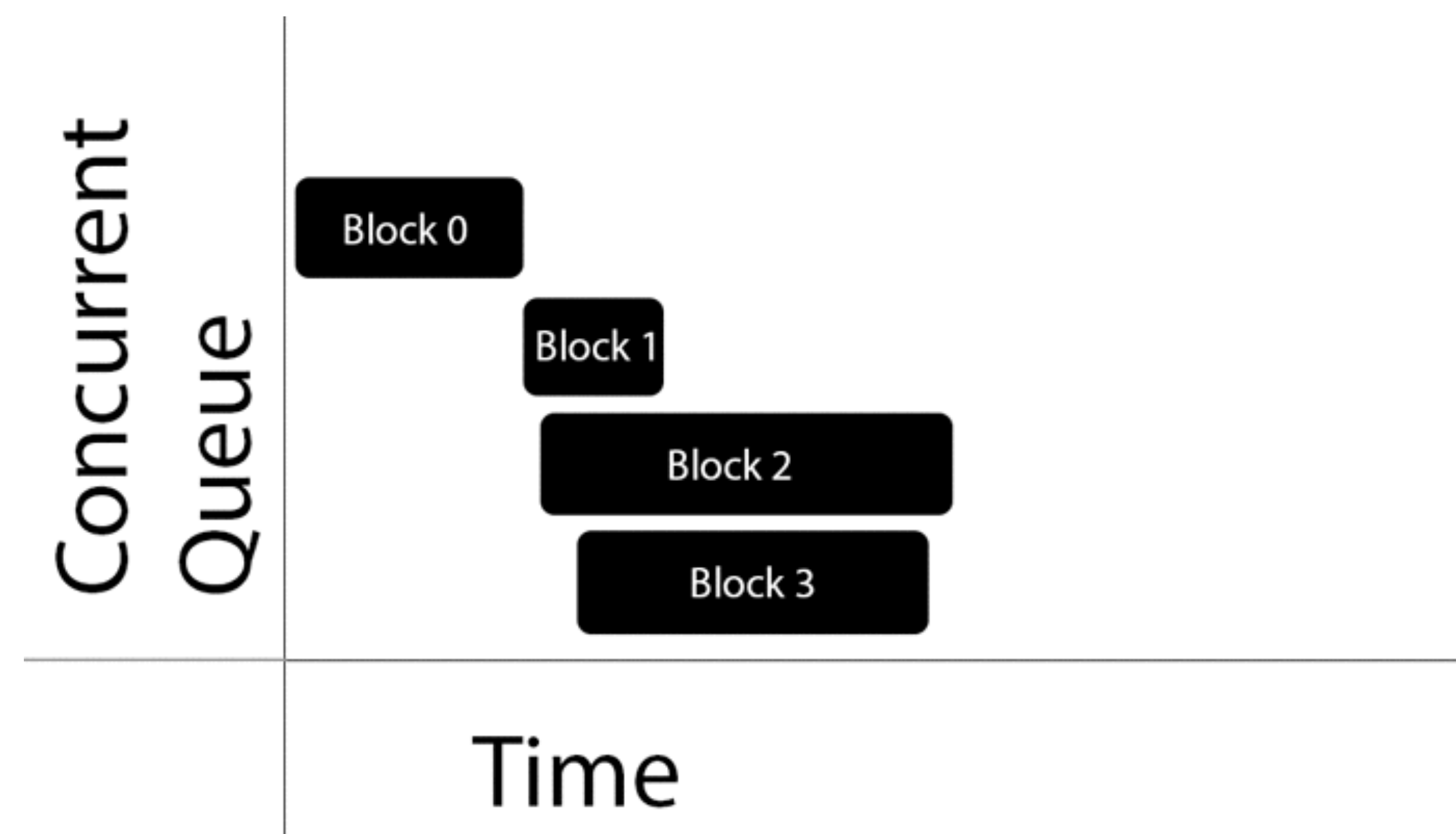
DispatchQueue Concurrent

- › Задачи выполняются параллельно
- › Порядок выхода из очереди - FIFO
- › Порядок выполнения - может варьироваться
- › Максимальный параллелизм - задается

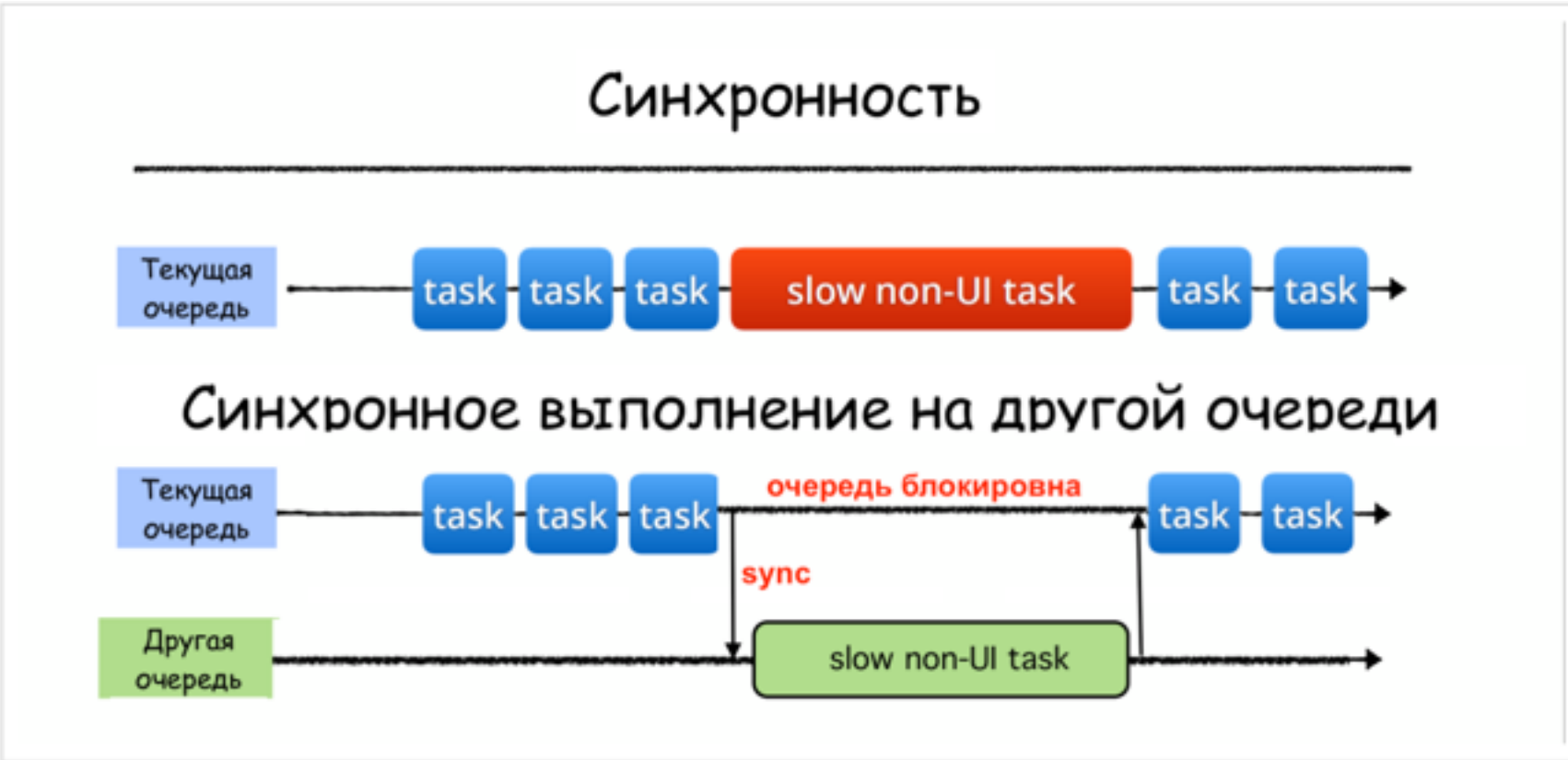


DispatchQueue Concurrent

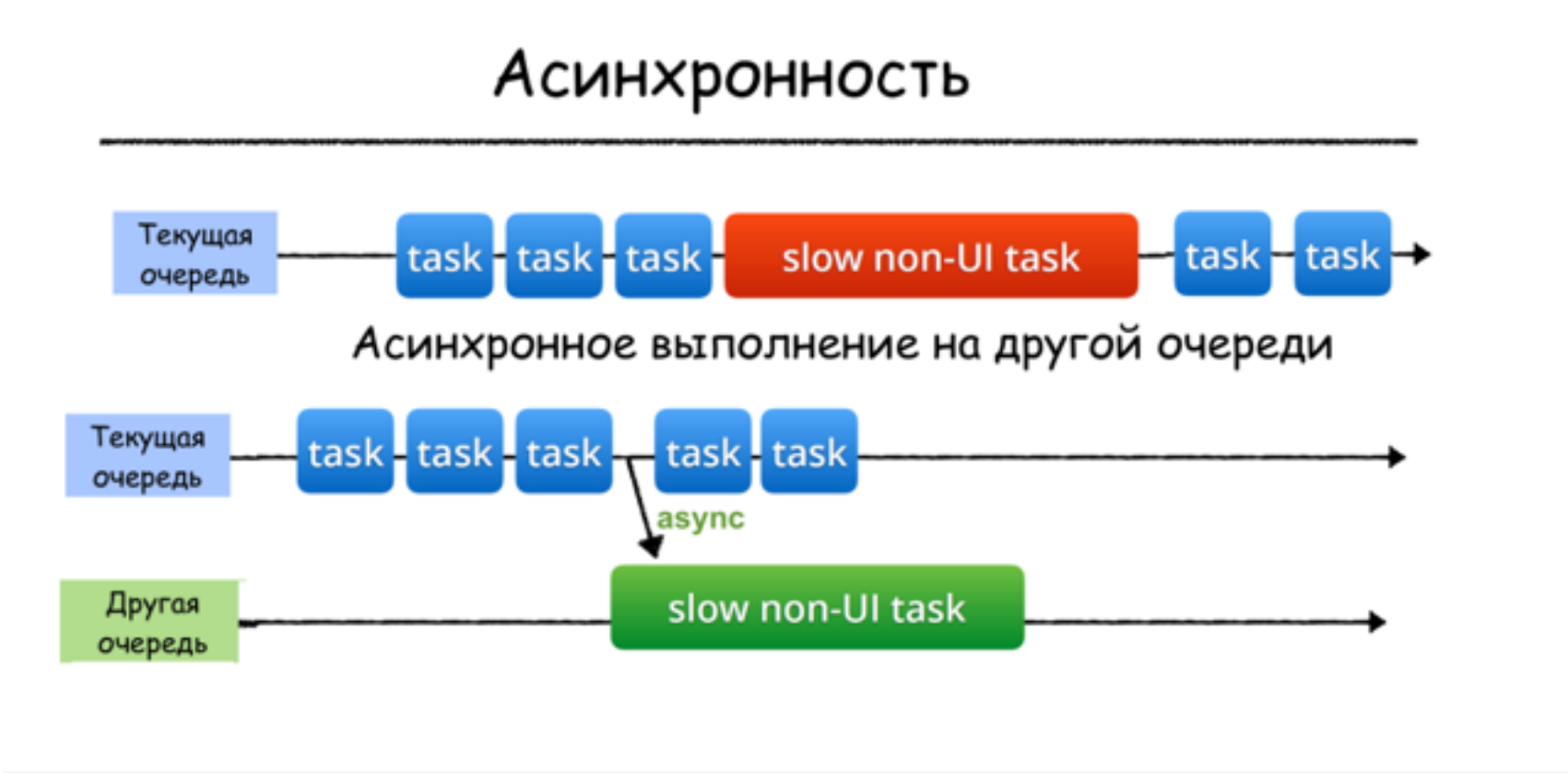
- › Задачи выполняются параллельно
- › Порядок выхода из очереди - FIFO
- › Порядок выполнения - может варьироваться
- › Максимальный параллелизм - задается



sync vs async

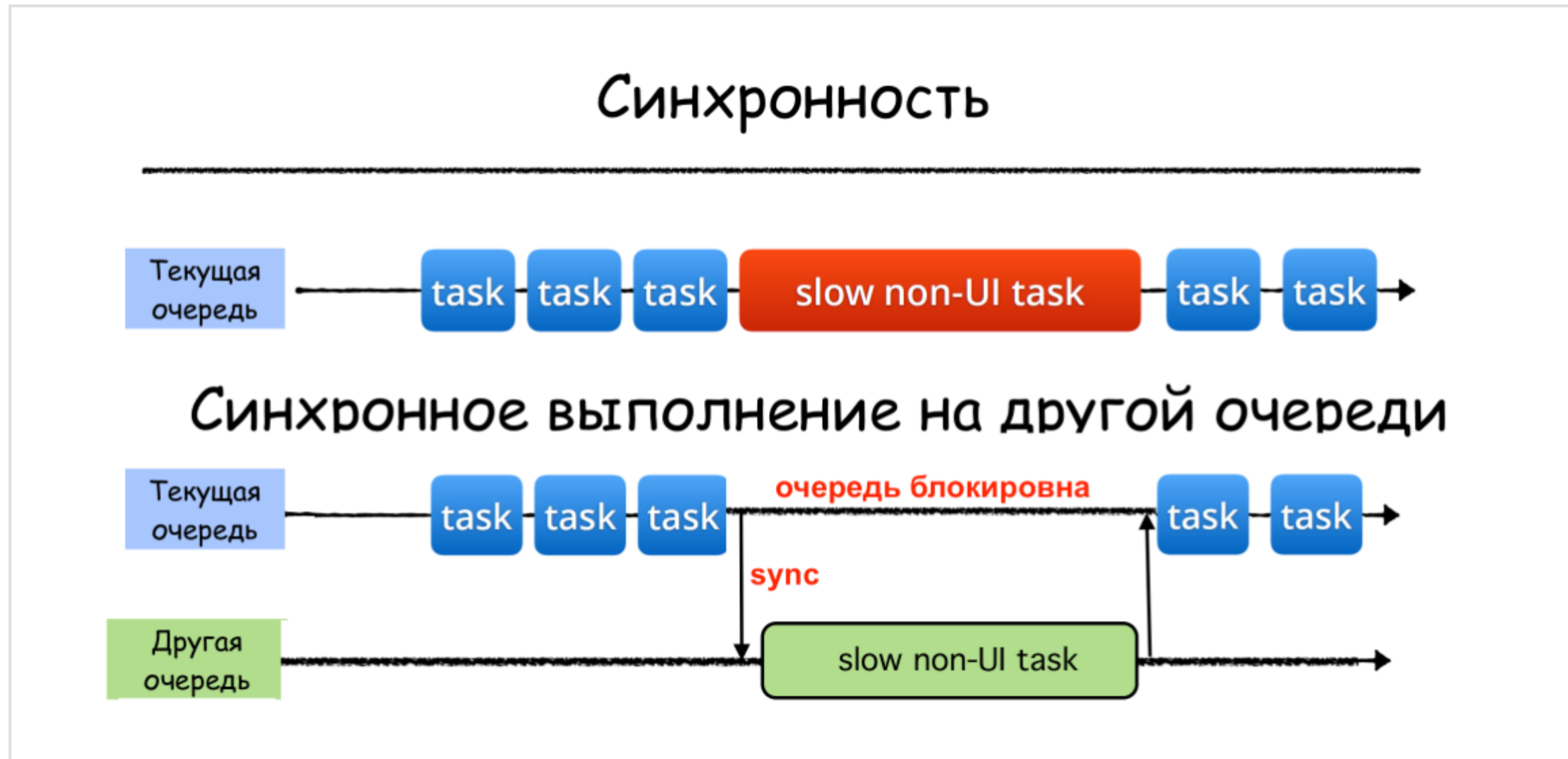


sync



async

sync

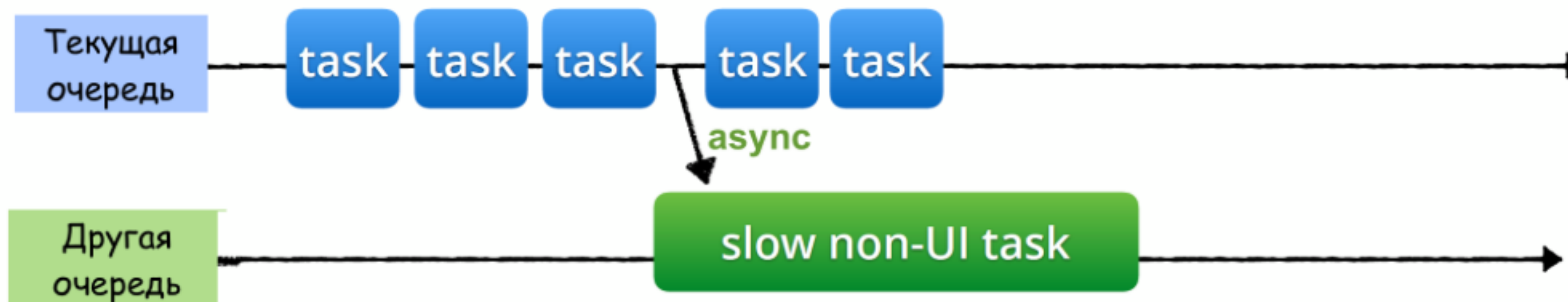


async

Асинхронность



Асинхронное выполнение на другой очереди



GCD: Делаем что-нибудь в фоне

```
DispatchQueue.global().async {  
    doBackgroundWork()  
}
```

GCD: Делаем что-нибудь в фоне и не только

```
DispatchQueue.global().async {  
    doBackgroundWork()  
    DispatchQueue.main.async {  
        doUIWork()  
    }  
}
```

GCD: no more dispatch_once

```
static id sharedInstance;  
dispatch_once(&onceToken, ^{  
    sharedInstance = [[self alloc] init];  
});
```

GCD: Семафорим

```
let semaphore = DispatchSemaphore(value: 0)
doSomeAsyncWork {
    semaphore.signal()
}

semaphore.wait(timeout: .now() + 5)
print("HI")
```

| Никогда не вызывайте wait
из главного потока

GCD: Ожидаем одновременного выполнения

```
let group = DispatchGroup()  
for item in items {  
    group.enter()  
    process(item: item) {  
        group.leave()  
    }  
}  
  
group.notify(queue: .main) {  
    ...  
}
```

GCD: Ожидаем одновременного выполнения

```
let group = DispatchGroup()  
for item in items {  
    group.enter()  
    process(item: item) {  
        group.leave()  
    }  
}  
  
group.wait()  
...
```

GCD: И еще много полезного



Поваренная книга (Swift 2)

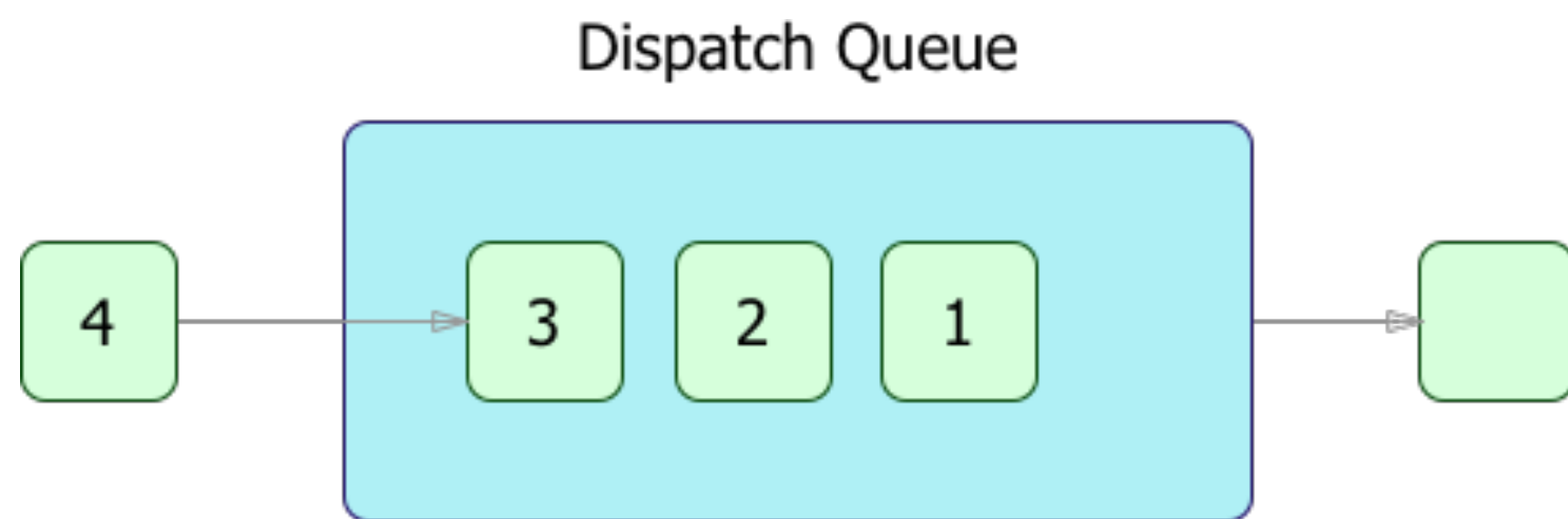


Apple Guide

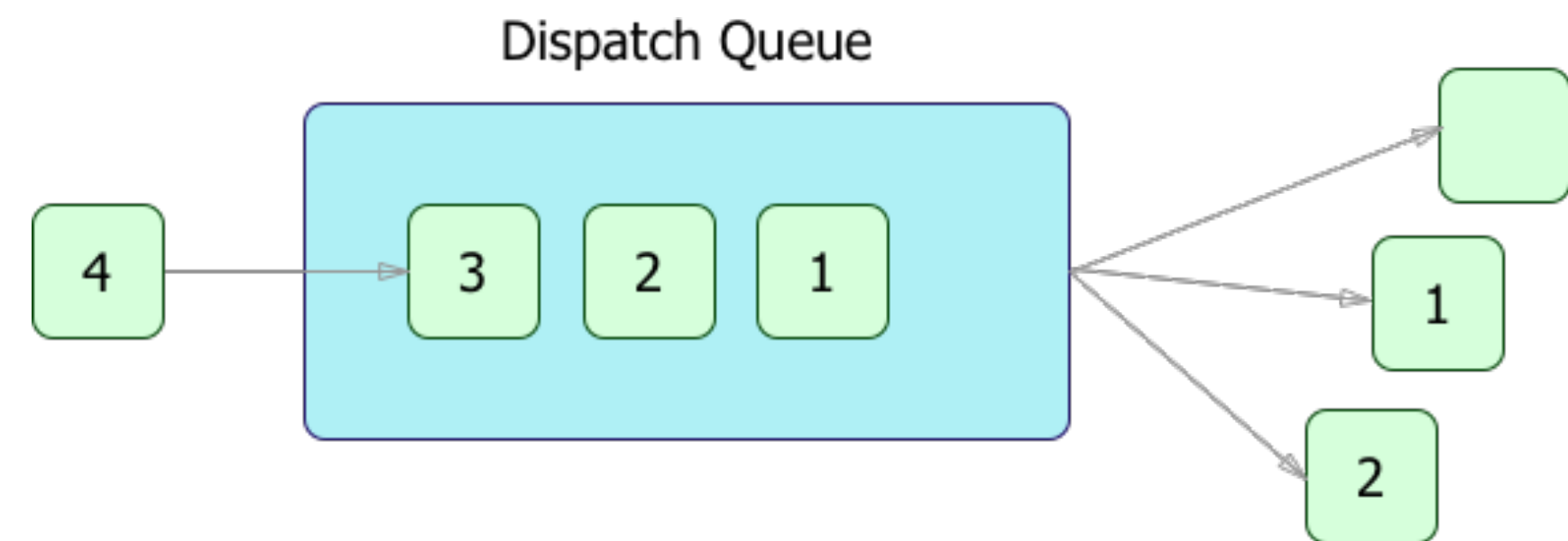
GCD Demo Time



SERIAL vs Concurrent

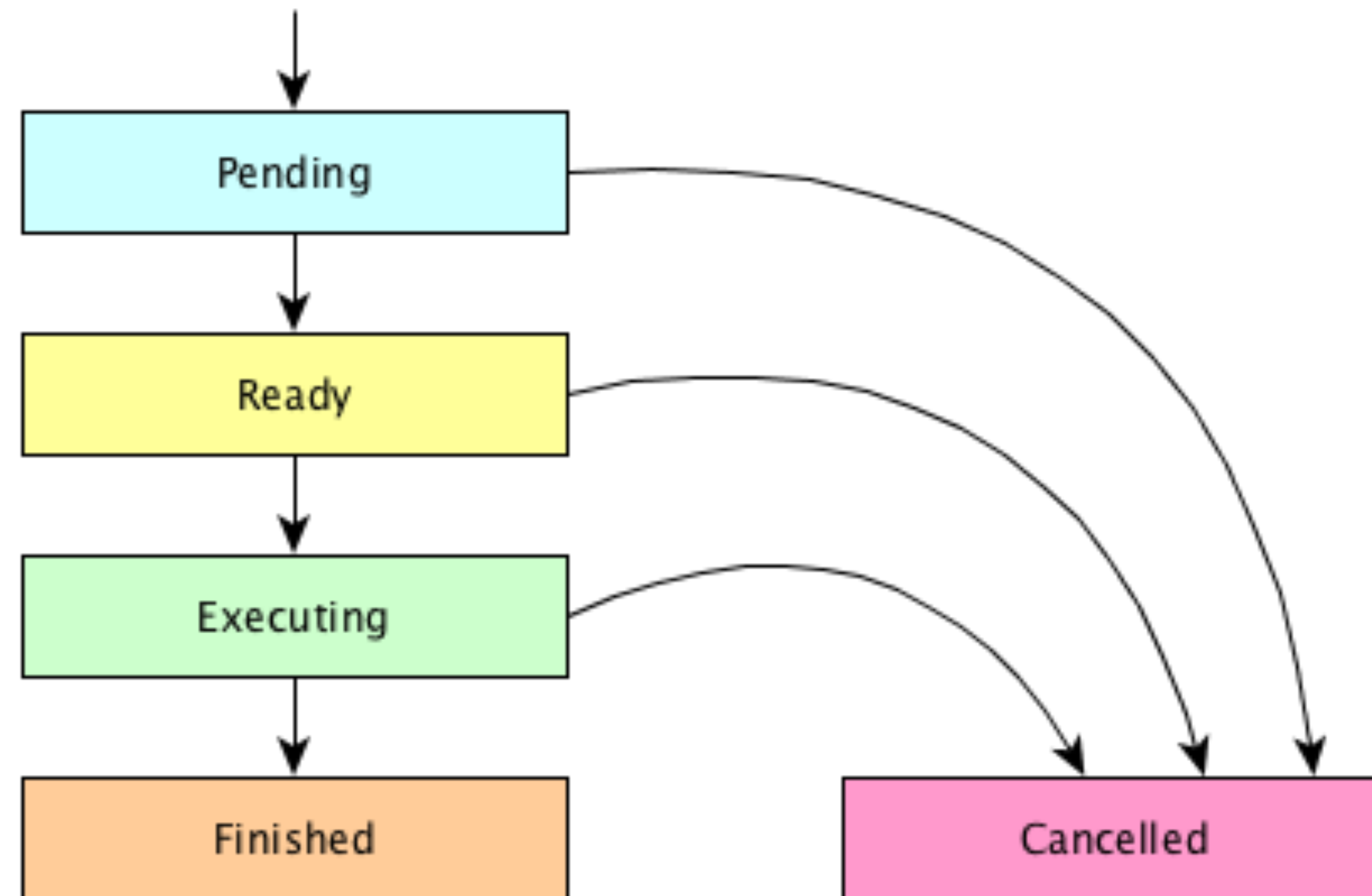


Serial

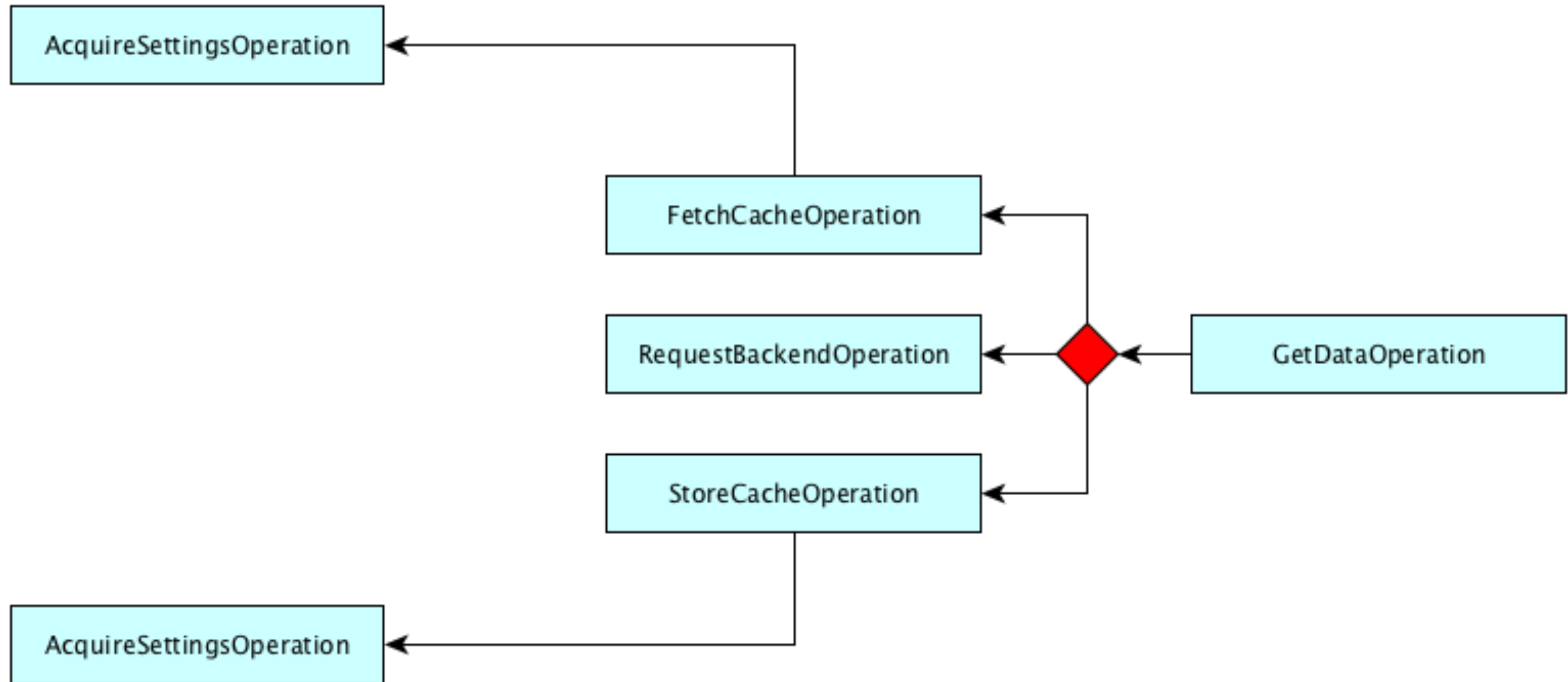


Concurrent

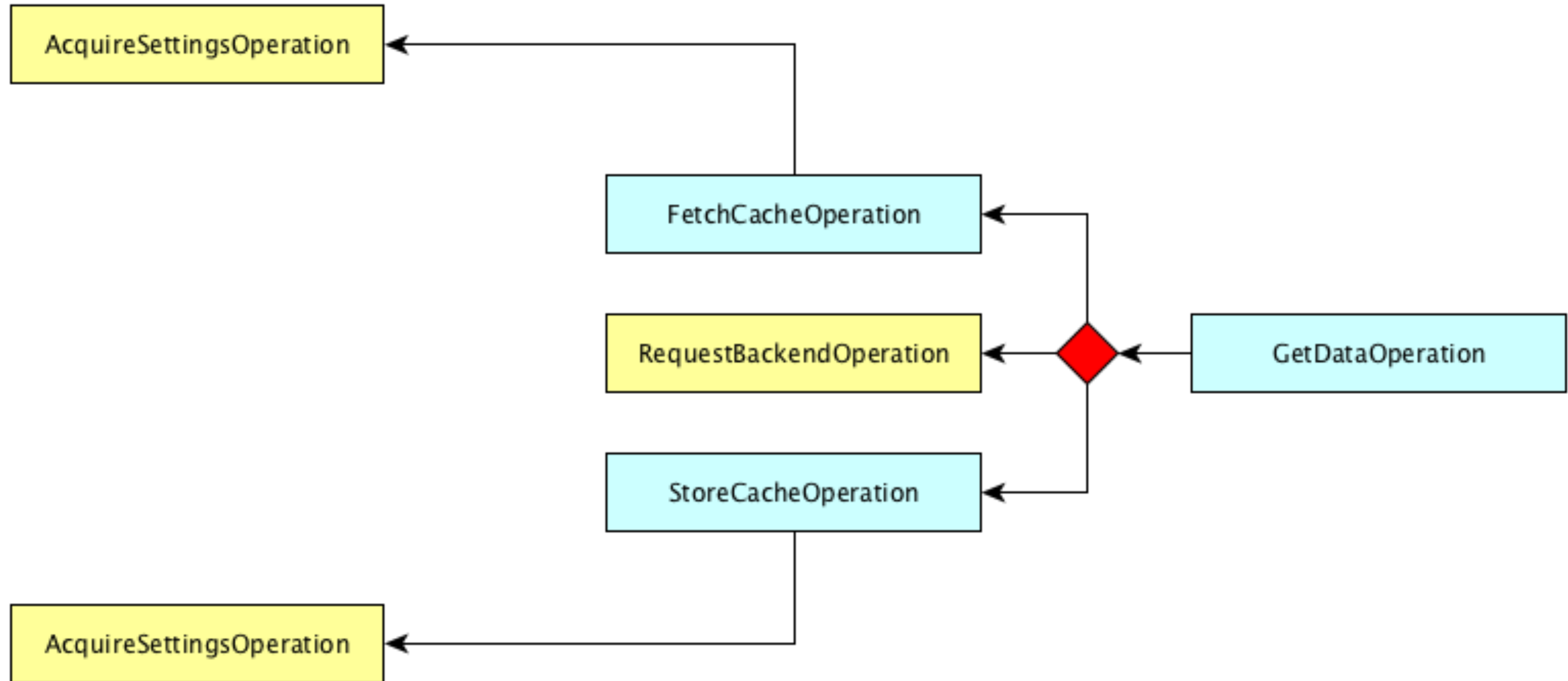
NSOperation



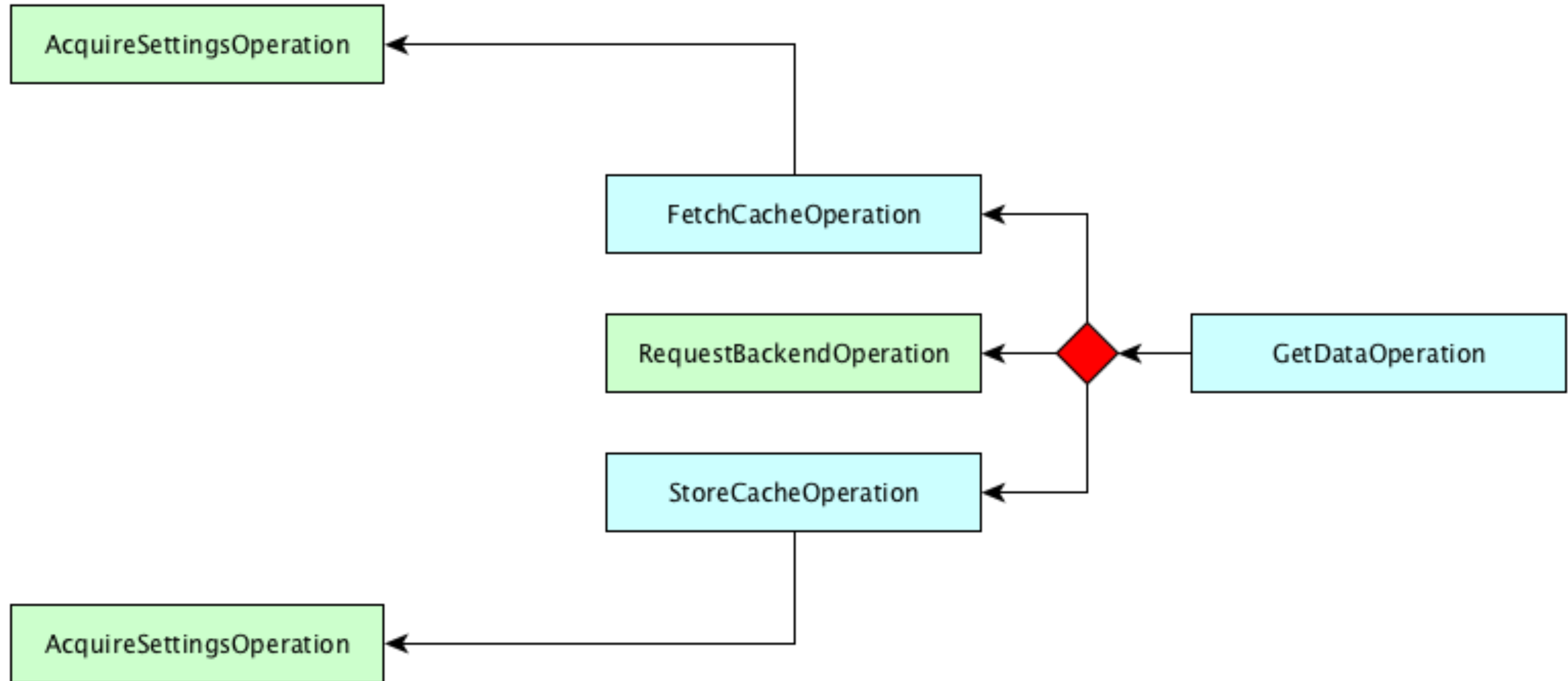
NSOperation: Dependencies



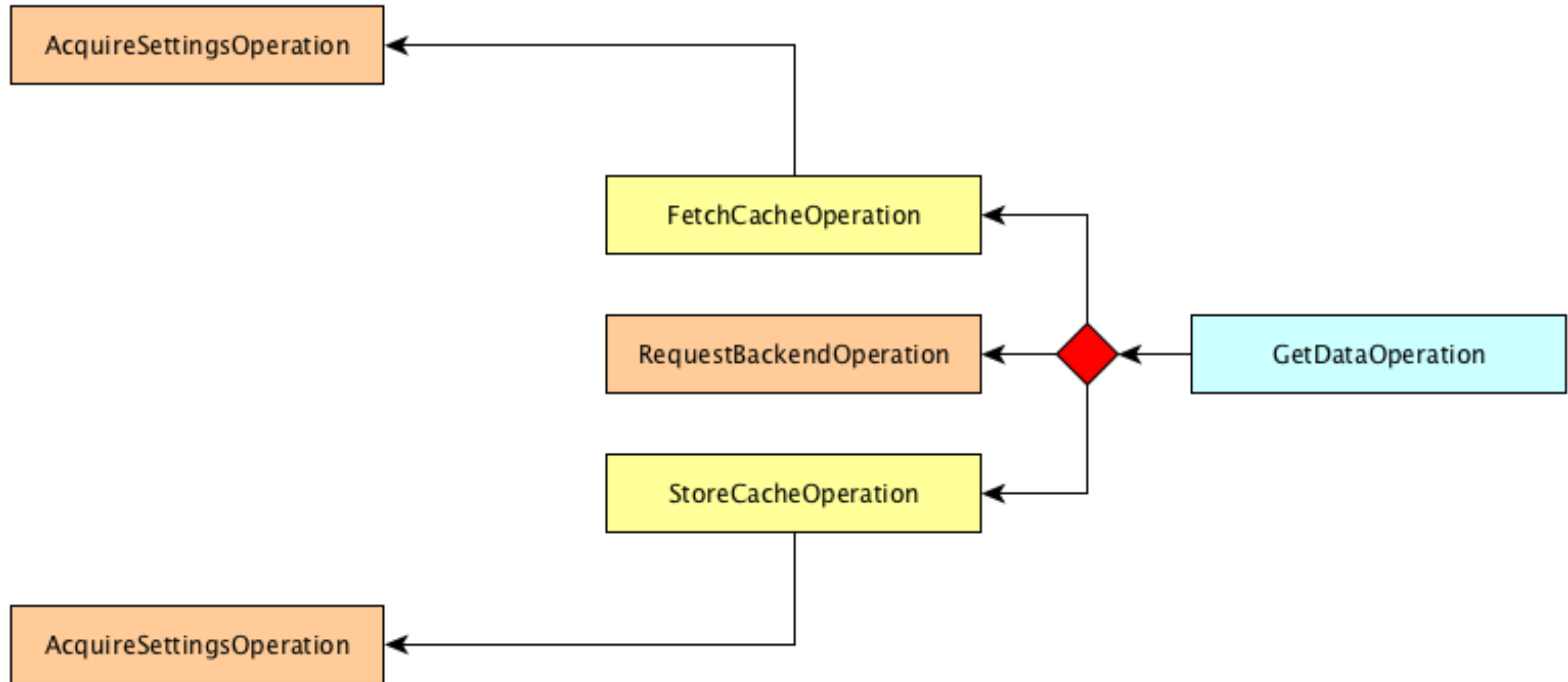
NSOperation: Dependencies



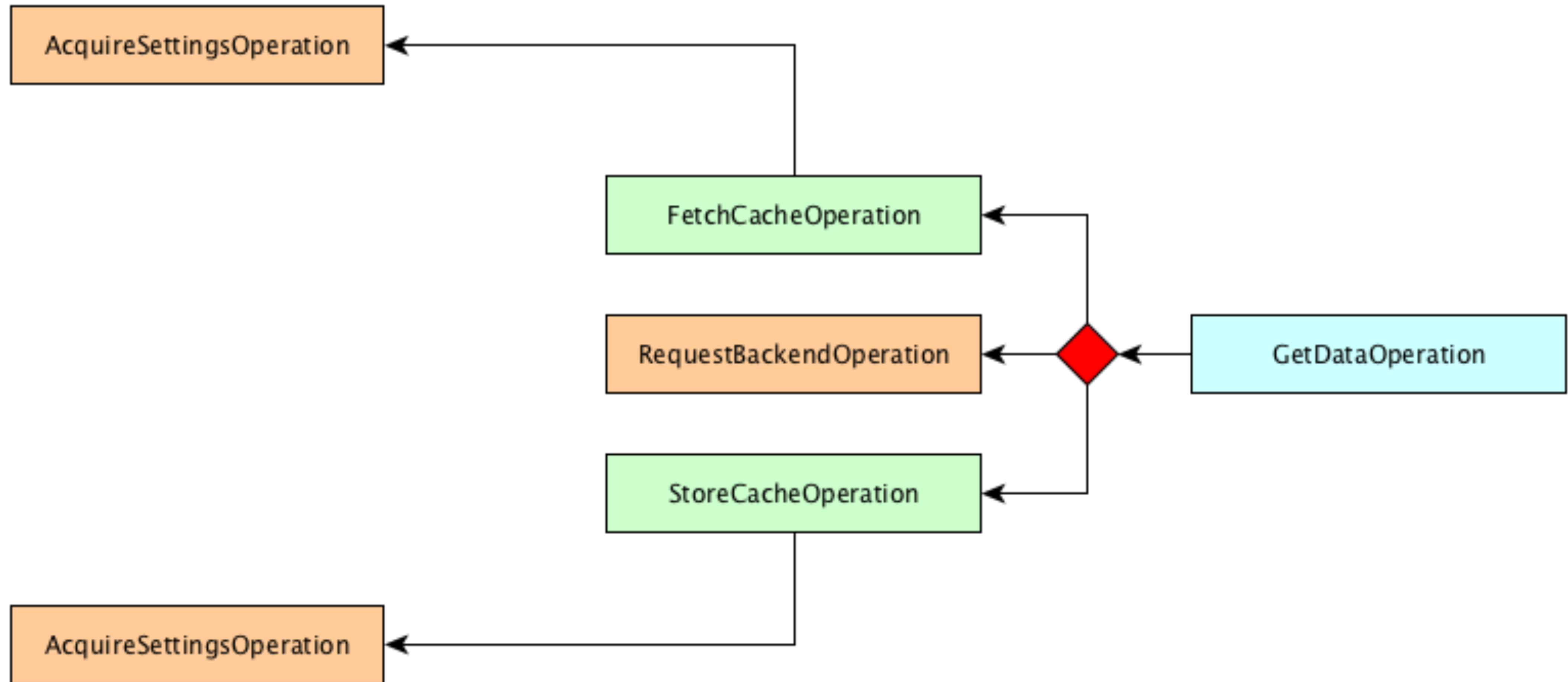
NSOperation: Dependencies



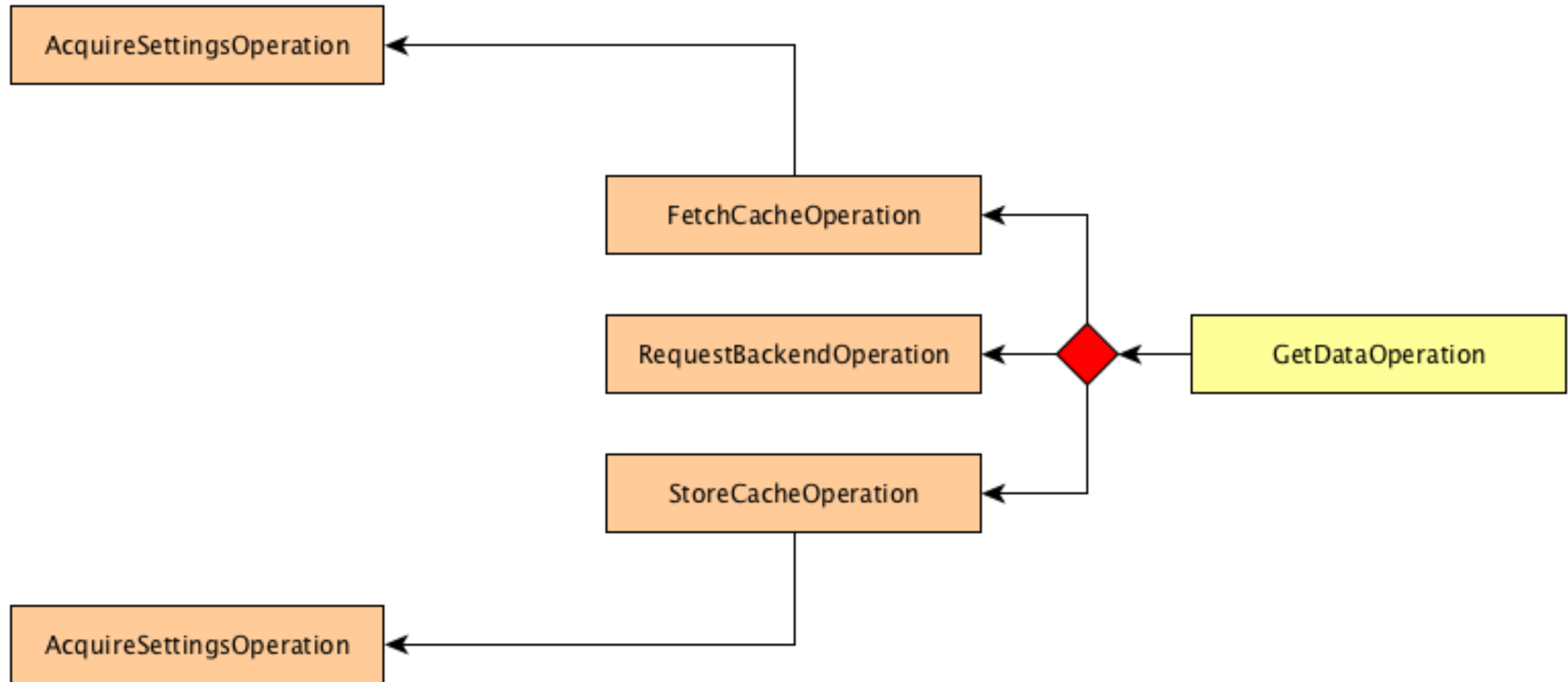
NSOperation: Dependencies



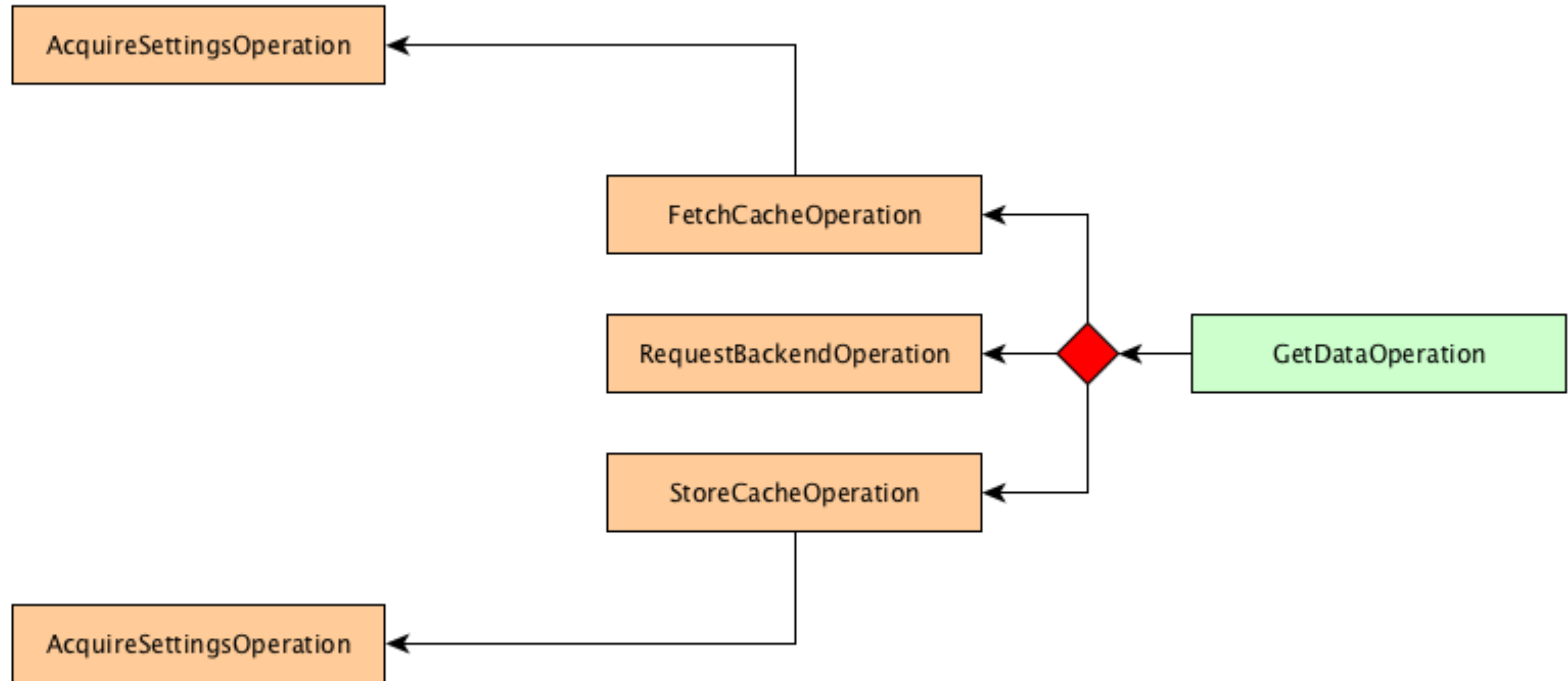
NSOperation: Dependencies



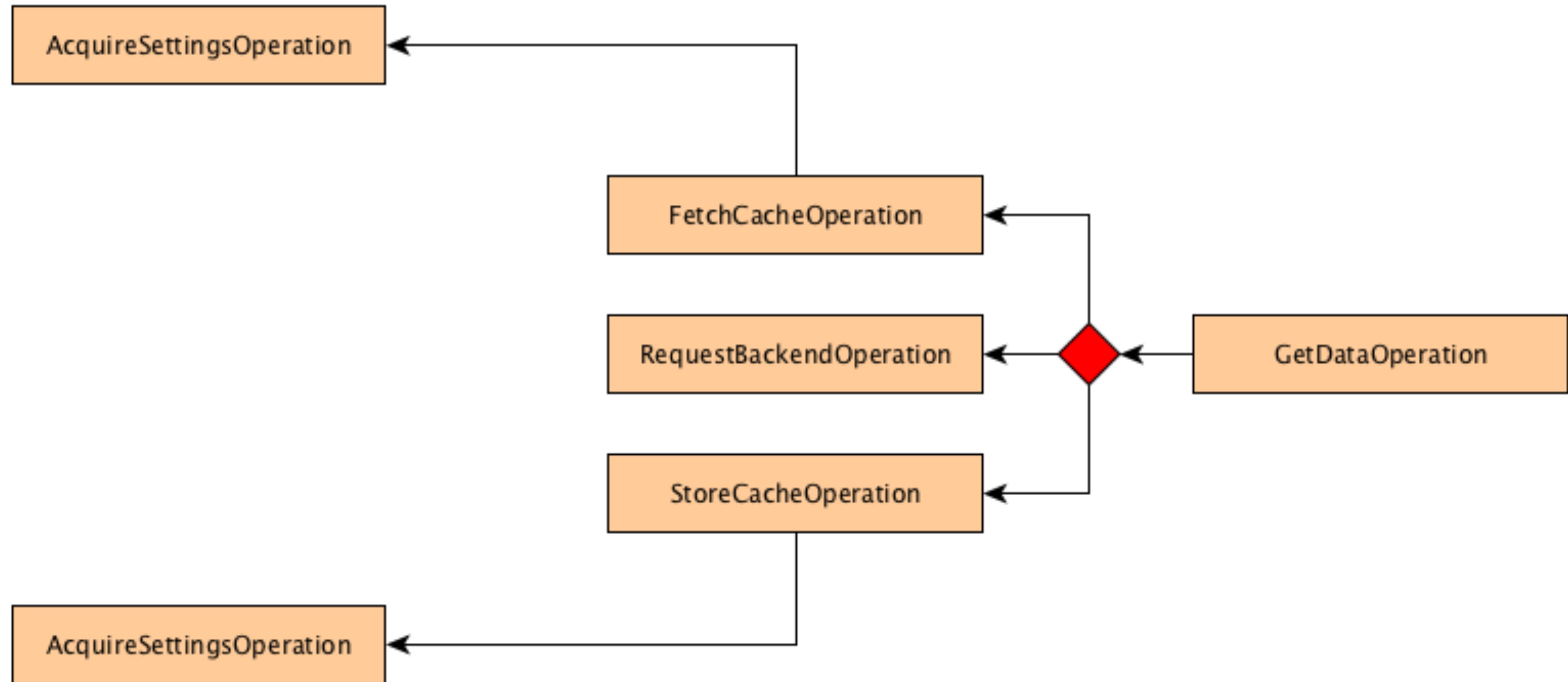
NSOperation: Dependencies



NSOperation: Dependencies



NSOperation: Dependencies



NSOperationQueue

```
let queue1 = OperationQueue.main

let queue2 = OperationQueue() // SERIAL
queue2.maxConcurrentOperationCount = 1

let queue3 = OperationQueue() // CONCURRENT
queue3.maxConcurrentOperationCount = 5
```

NSOperation

```
class MyOperation : Operation {  
    override func main() {  
        // do the work  
    }  
}  
  
let op1 = MyOperation()  
queue1.addOperation(op1)
```


NSOperation - BlockOperation

```
let op2 = BlockOperation {  
    // do the work  
}  
  
queue1.addOperation(op2)
```

NSOperation - AsyncOperation

```
class AsyncOperation : Operation {  
    var success: (() -> ())? = nil  
  
    fileprivate var _executing = false  
    fileprivate var _finished = false  
  
    ...  
}
```

NSOperation - AsyncOperation

```
class AsyncOperation : Operation {  
    ...  
    override func start() {  
        guard !isCancelled else {  
            finish()  
            return  
        }  
  
        willChangeValue(forKey: "isExecuting")  
        _executing = true  
        main()  
        didChangeValue(forKey: "isExecuting")  
    }  
    ...  
}
```

NSOperation - AsyncOperation

```
class AsyncOperation : Operation {  
    ...  
  
    override func main() {  
        // NOTE: should be overridden  
        finish()  
    }  
  
    ...  
}
```

NSOperation - AsyncOperation

```
class AsyncOperation : Operation {  
    ...  
  
    fileprivate func finish() {  
        success?()  
  
        willChangeValue(forKey: "isFinished")  
        _finished = true  
        didChangeValue(forKey: "isFinished")  
    }  
  
    ...  
}
```

NSOperation - AsyncOperation

```
class AsyncOperation : Operation {  
    ...  
  
    override var isAsynchronous: Bool {  
        return true  
    }  
  
    override var isExecuting: Bool {  
        return _executing  
    }  
  
    override var isFinished: Bool {  
        return _finished  
    }  
}
```

NSOperation - My AsyncOperation

```
class MyAsyncOperation: AsyncOperation {
    override func main() {
        DispatchQueue.main.asyncAfter(deadline: .now() + 3) { [weak self] in
            guard let sself = self else { return }
            // do the work
            sself.finish()
        }
    }
}

let op3 = MyAsyncOperation()
op3.success = {
    // do some post operation things
}
queue1.addOperation(op3)
```

NSOperation Demo



Вопросы?



Задача



Общие требования

- › Вам необходимо реализовать основные операции для вашего приложения (можно использовать стандартные, например, BlockOperation)
- › * Вам необходимо реализовать ваши операции как асинхронные операции (нельзя использовать стандартные)
- › ** Реализовать диспетчер для управления порядком выполнения операций (разные очереди)

Базовая задача

- › В вашей архитектуре вы продумывали базовые операции в вашем приложении, теперь их надо реализовать
- › Кроме этого, необходимо связать данные с интерфейсами, которые вы реализовывали в домашнем задании к 4 и 5 лекциям
- › Можно использовать стандартные операции

Усложнение 1 (*)

- › Необходимо реализовать ваши операции как асинхронные
- › Необходимо добиться их корректного выполнения

Усложнение 2 (**)

- › Необходимо реализовать объект - диспетчер (Dispatcher), который будет управлять несколькими очередями и планированием выполнения операций
- › Требуется поддержать несколько разных очередей для разных ресурсов (кэш, сеть, фоновое выполнение, задачи для интерфейса)

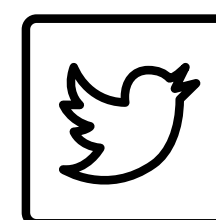
Контакты

Малых Денис

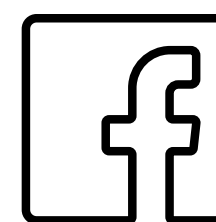
iOS Developer



mrdekk@yandex.ru



@mrdekk



mrdekk

Спасибо за внимание

