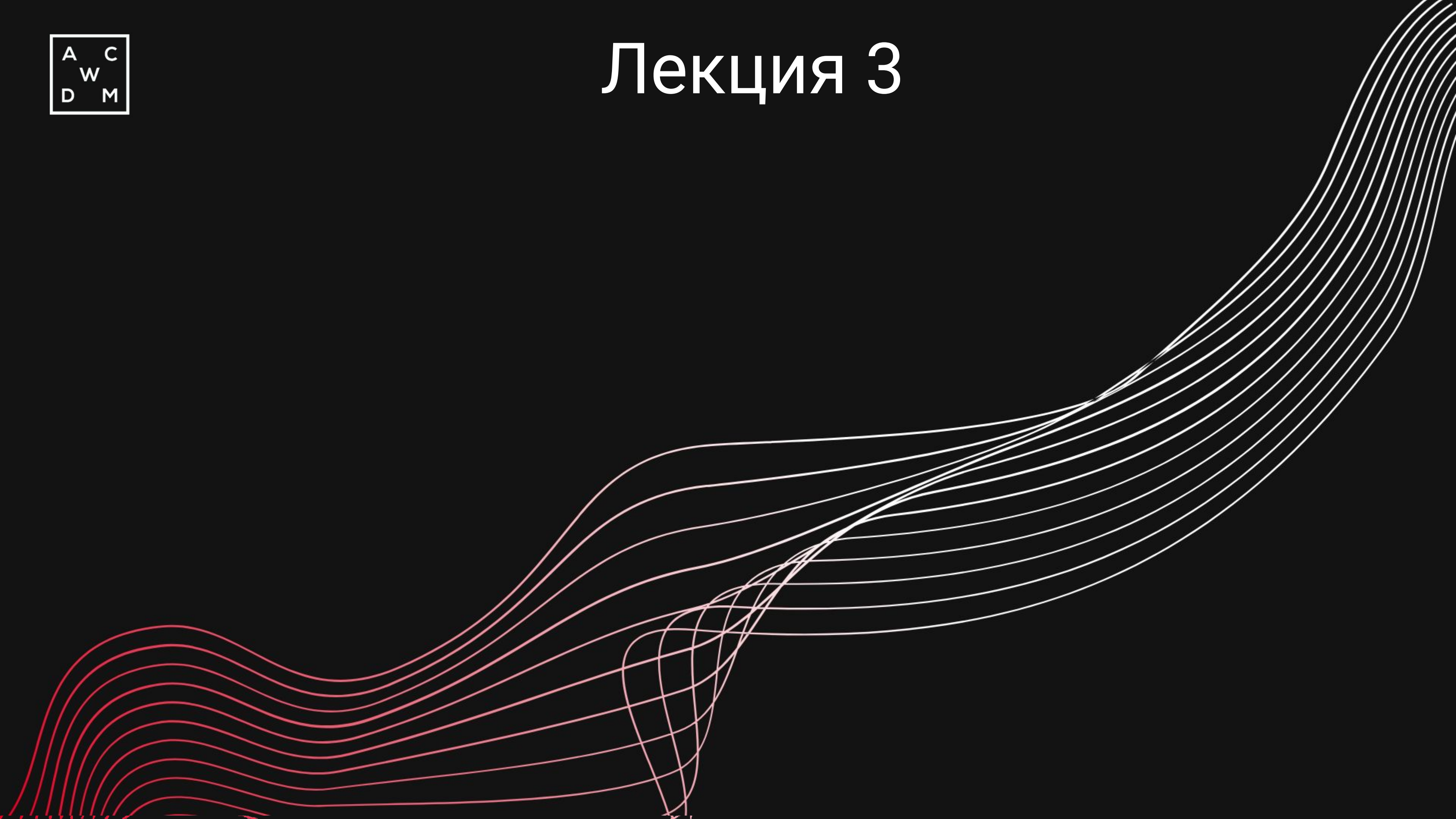


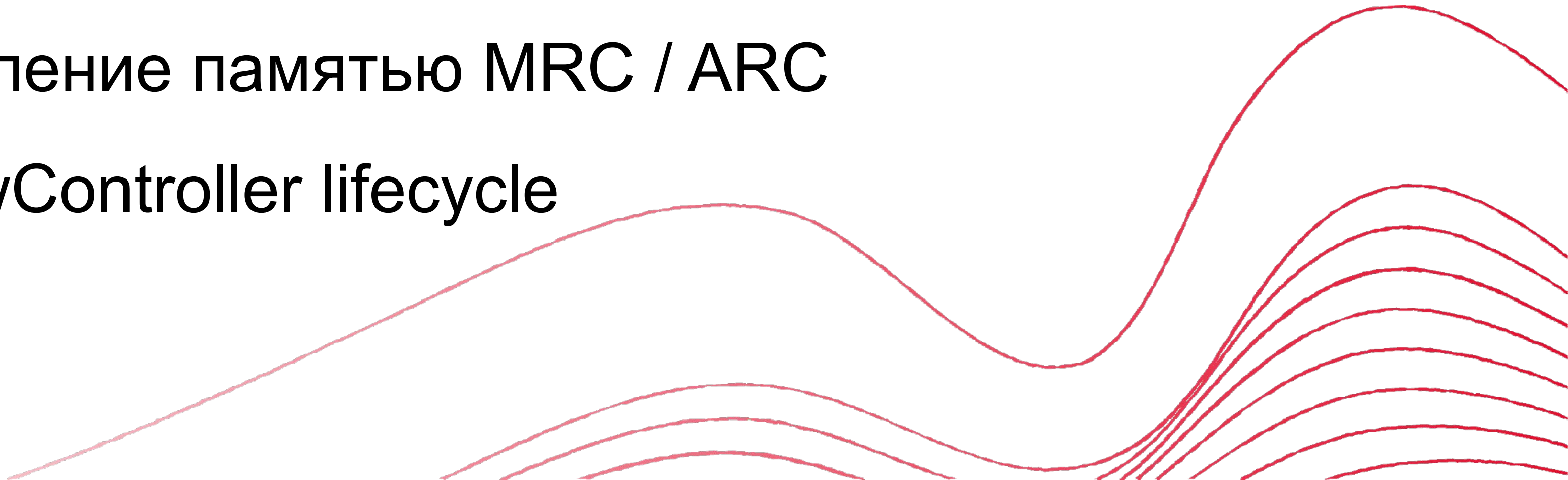


Лекция 3





Agenda

- Классы
 - Структуры, что такое Value type / Reference type
 - ООП
 - Управление памятью MRC / ARC
 - UIViewController lifecycle
- 



Классы (class ...)

- Разница между классом и экземпляром класса
- Свойства и методы
- Инициализация классов
- Операции над экземплярами класса и большие классы
- Наследование
- Реальные примеры классов

Подробная статья о структурах и классах:

<https://unetway.com/tutorial/swift-struktury-i-klassy>

Документация swift на русском, о классах и структурах:

<https://swiftbook.ru/content/languageguide/classes-and-structures/>



Value type / reference type

Структуры (struct ...)

- Как хранятся классы и структуры
- Оператор равенства для Reference type
- Инициализация структур
- let для классов / структур

Не лучшим образом оформленная статья про value / reference type на java, но именно прочитав ее я разобрался в свое время:

<https://www.tune-it.ru/web/bleizard/blog/-/blogs/2321688>

Объяснение работы value / reference type на swift:

<https://tproger.ru/translations/programming-concepts-stack-and-heap/>



ООП - объектно-ориентированное программирование

- **Инкапсуляция** — способ спрятать некоторые свойства и метода внутри класса, о которых не должно быть известно за его пределами.
- **Наследование** — способ расширить существующий класс, добавив в него новые метода или свойства.
- **Полиморфизм** — свойство объектов унаследованных от одного класса, позволяющие им создать свои реализации для функции базового класса.

Статья с отличным примером полиморфизма:

<https://metanit.com/swift/tutorial/3.13.php>

Простыми словами про ООП (хоть и на примере языка python):

<https://www.youtube.com/watch?v=XmCAGUo5k70>

Длинная и скучная, но подробная статья об ООП в примерах на swift:

<https://swiftbook.ru/post/tutorials/tutorial-obektno-orientirovannoe-programmirovanie-v-swift>



ARC и управление памятью

- Область видимости переменной
- Цикл ссылок на примере User и Phones
- Weak ссылка (всегда optional)
- Циклы ссылок в замыканиях

Подробная статья на тему ARC и управления памятью:

<https://habr.com/ru/post/451130/>

Про ARC на английском:

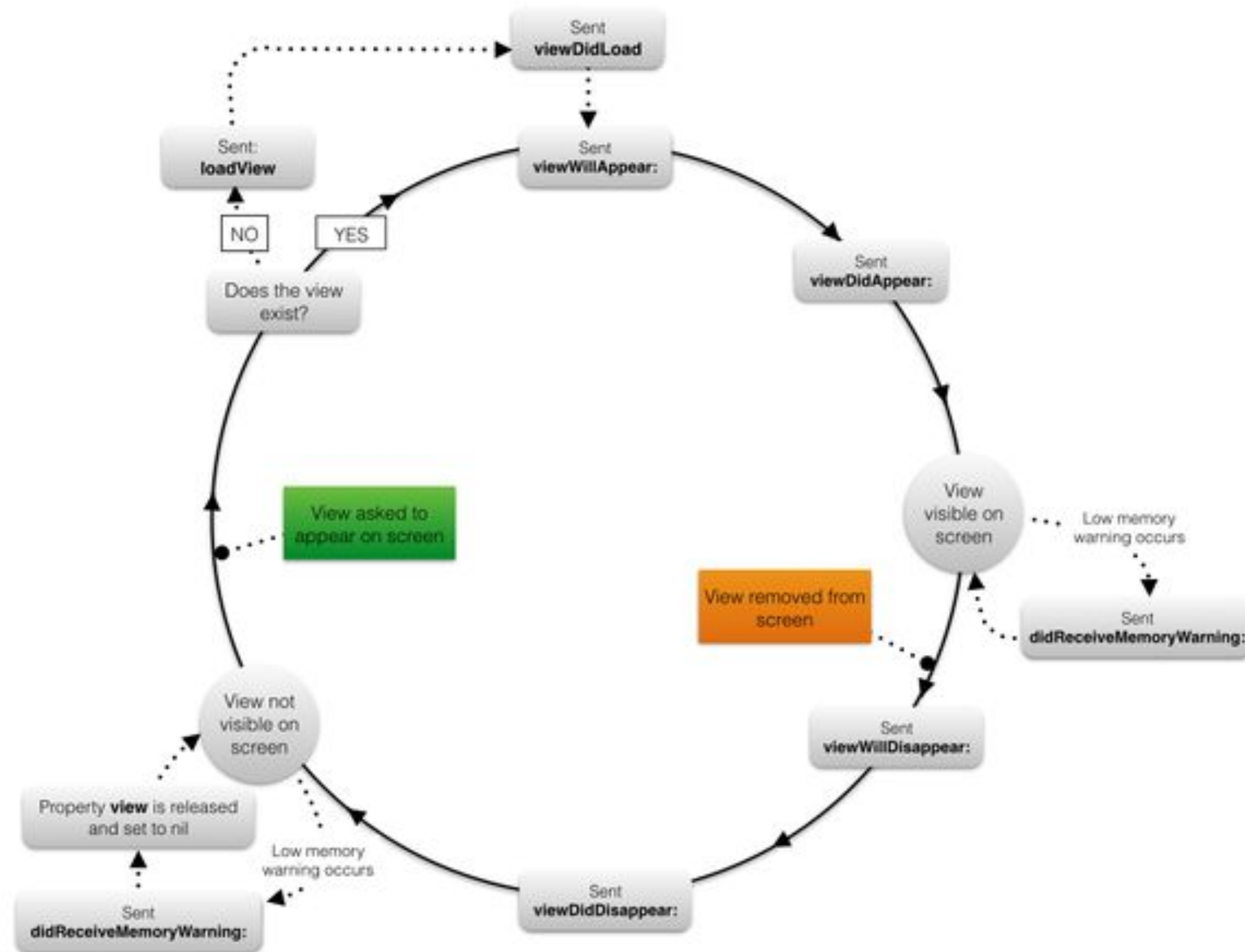
<https://docs.swift.org/swift-book/LanguageGuide/AutomaticReferenceCounting.html>

Про ARC на русском:

<https://swiftbook.ru/content/languageguide/automatic-reference-counting>



Жизненный цикл UIViewController





Жизненный цикл ViewController-a

- . **loadView()**
- . **viewDidLoad()**
- . **viewWillAppear(_ animated: Bool)**
- . **viewDidAppear(_ animated: Bool)**
- . **viewWillDisappear(_ animated: Bool)**
- . **viewDidDisappear(_ animated: Bool)**
- . **viewWillLayoutSubviews()**
- . **viewDidLayoutSubviews()**

Подробное объяснение жизненного цикла UIViewController()

<https://overcoder.net/q/9971/хотите-понять-жизненный-цикл-ios-uiviewcontroller>

Жизненный цикл вью контроллера на видео:

<https://www.youtube.com/watch?v=W9h6mpft04c>

Шпаргалка по методам (на английском):

<https://medium.com/@vipandey54/uiviewcontroller-lifecycle-7ca2d36f4f07>



Практика в Xcode

Класс Phone.

- а) Создайте класс Phone, который содержит переменные number, model и weight.
- б) Создайте три экземпляра этого класса.
- в) Выведите на консоль значения их свойств.
- г) Добавить в класс Phone методы: receiveCall, имеет один параметр – имя звонящего. Выводит на консоль сообщение “Звонит \(\name)”. getNumber – возвращает номер телефона. Вызвать эти методы для каждого из объектов.
- д) Добавить инициализатор в класс Phone, который принимает на вход три параметра для инициализации переменных класса - number, model и weight.
- е) Добавить инициализатор, который принимает на вход два параметра для инициализации переменных класса - number, model.
- ж) Добавить инициализатор без параметров.
- з) Вызвать из инициализатора с двумя параметрами конструктор с тремя.
- и) Добавьте метод receiveCall, который принимает два параметра - имя звонящего и номер телефона звонящего. Вызвать этот метод.
- к) Создать метод sendMessage. Данный метод принимает на вход номера телефонов, которым будет отправлено сообщение. Метод выводит на консоль номера этих телефонов.



Практика в Xcode

Класс Calculator

Создайте класс Calculator, который не содержит свойств, но содержит методы:

- Прибавить - принимает два числа и возвращает их сумму
- Вычесть - принимает два числа и возвращает их разницу
- Умножить - принимает два числа и возвращает результат их умножения
- Разделить - принимает два числа и возвращает результат их деления

Класс SuperCalculator (продвинутый калькулятор)

Пусть он будет наследоваться от Calculator и содержать методы:

- возвести в степень - принимает два параметра - число и степень в которую его нужно возвести. Возвращает результат возведения в степень
- получить процент - принимает два параметра - число и процент который мы хотим получить. Например число 50 и число 10, функция возвращает 5, так как 5 это 10% от 50-ти.

Создайте экземпляр класса SuperCalculator и протестируйте все методы со своими значениями, выведя результаты в консоль через команду: `print(SuperCalculator.myFunc())`



Практика в Xcode

Класс Animal

Создать класс Animal и наследующие его классы Dog, Cat, Horse.

Класс Animal содержит свойства food, location и методы makeNoise, eat, sleep.

Метод makeNoise, например, может выводить на консоль "*Такое-то животное спит*". Dog, Cat, Horse переопределяют методы makeNoise, eat. Добавьте переменные в классы Dog, Cat, Horse, характеризующие ТОЛЬКО ЭТИХ ЖИВОТНЫХ.

Создайте класс Ветеринар, в котором определите метод func treatAnimal(animal: Animal). Пусть этот метод распечатывает food и location пришедшего на прием животного.

В методе main создайте массив типа Animal, в который запишите животных всех имеющихся у вас типов. В цикле отправляйте их на прием к ветеринару.



Термины:

Свойство, проперти класса/структуры, переменная - контейнер в котором храниться какое-то значение (типа Bool, Int, String...)

Функция/метод - именованный блок кода, который можно выполнить повторно (вызвав его). Если придирааться, можно нагуглить разницу по первой ссылке. Но на практике эти слова используют как синонимы.

Класс - строительный блок программы. Может содержать проперти и функции. Является reference type (ссылочным типом).

Структура - то же, что и класс, но не умеет наследоваться и является value type (типом значения).

ARC - автоматический подсчет ссылок (automatic reference counting).

MRC - мануальный подсчет ссылок (manual reference counting).



Термины:

Инкапсуляция — способ спрятать некоторые свойства и метода внутри класса, о которых не должно быть известно за его пределами.

Наследование — способ расширить существующий класс, добавив в него новые метода или свойства.

Полиморфизм — свойство объектов унаследованных от одного класса, позволяющие им создать свои реализации для функции базового класса.

Жизненный цикл UIViewController - набор методов, которые срабатывают автоматически в определенные моменты во время выполнения программы.

viewDidLoad() - метод выполняющийся 1 раз при загрузке ViewController-a.

viewWillAppear() - метод, выполняющийся каждый раз, когда контроллер появляется на экране.



Все ссылки:

Подробная статья о структурах и классах:

<https://unetway.com/tutorial/swift-struktury-i-klassy>

Документация swift на русском, о классах и структурах:

<https://swiftbook.ru/content/languageguide/classes-and-structures/>

Не лучшим образом оформленная статья про value / reference type на java, но именно прочитав ее я разобрался в свое время:

<https://www.tune-it.ru/web/bleizard/blog/-/blogs/2321688>

Объяснение работы value / reference type на swift:

<https://tproger.ru/translations/programming-concepts-stack-and-heap/>

Статья с отличным примером полиморфизма:

<https://metanit.com/swift/tutorial/3.13.php>

Простыми словами про ООП (хоть и на примере языка python):

<https://www.youtube.com/watch?v=XmCAGUo5k70>

Длинная и скучная, но подробная статья об ООП в примерах на swift:

<https://swiftbook.ru/post/tutorials/tutorial-obektno-orientirovannoe-programmirovanie-v-swift>

Подробная статья на тему ARC и управления памятью:

<https://habr.com/ru/post/451130/>

Про ARC на английском:

<https://docs.swift.org/swift-book/LanguageGuide/AutomaticReferenceCounting.html>

Про ARC на русском:

<https://swiftbook.ru/content/languageguide/automatic-reference-counting>

Подробное объяснение жизненного цикла UIViewController()

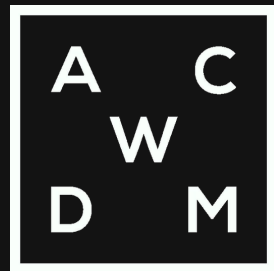
<https://overcoder.net/q/9971/хотите-понять-жизненный-цикл-ios-uiviewcontroller>

Жизненный цикл вью контроллера на видео:

<https://www.youtube.com/watch?v=W9h6mpftO4c>

Шпаргалка по методам (на английском):

<https://medium.com/@vipandey54/uiviewcontroller-lifecycle-7ca2d36f4f07>



Web
Academy

CREATE YOUR IT FUTURE.

WITH WEB ACADEMY

