

ЛЕКЦІЯ №5

Підготовчі завдання до лекції на тему “Управління пам'яттю в Swift”

Дата проведення: __.__.20__

Лектор: Далекорей Анатолій

СПИСОК ЛІТЕРАТУРИ	2
САМОКОНТРОЛЬ	3
ПРАКТИЧНЕ ЗАВДАННЯ	4
ТЕХНІЧНІ ВИМОГИ	8
ЗВОРОТНІЙ ЗВ'ЯЗОК	9

СПИСОК ЛІТЕРАТУРИ

Ознайомтеся зі списком літератури наведеним нижче. Описані джерела надать необхідну базову інформацію для засвоєння матеріалу лекції та виконання практичного завдання.

1. Objective-C Memory Management Essentials By Gibson Tang, Maxim Vasilkov (978-1-84969-712-5)..
2. The Swift Programming Language, Apple Inc,
 - chapter Automatic Reference Counting.
3. Weak vs Unowned (krakendev.io/blog/weak-and-unowned-references-in-swift).
4. Weak vs unowned (stackoverflow.com/questions/24320347/shall-we-always-use-unowned-self-inside-closure-in-swift).
5. Мэтт Гэлловей - Сила Objective-C 2.0 - 2014. (Chapter 5)..
6. А. Махер - "Программирование для iPhone. Высший уровень" (розділ 1.3).
7. Memory management - (https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/MemoryMgmt/Articles/MemoryMgmt.html#//apple_ref/doc/uid/10000011-SW1).

САМОКОНТРОЛЬ

Ознайомтеся зі списком ключових слів, що характеризують матеріал лекції. Володіння усіма описаними термінами є розумінням матеріалу лекції.

1. MMR.
2. ARC.
3. retain.
4. release.
5. retain cycle.
6. unowned.
7. capture list.
8. allocation/deallocation.
9. weak/strong.
10. zombie object.
11. autorelease pool
12. retain counter
13. memory leak
14. NSAutoreleasePool
15. dealloc/deinit
16. copy object and related processes
17. garbage collector

ПРАКТИЧНЕ ЗАВДАННЯ

Виконайте завдання наведені нижче. Кожне завдання складено у межах матеріалу лекції та не потребує додаткових знань. Дивіться технічні вимоги до виконання практичного завдання у відповідному розділі.

ЗАВДАННЯ №1

ОПИС:

Робота з пам'яттю.

ДЕТАЛІ:

Реалізуйте пункти описані нижче.

ПУНКТИ ВИКОНАННЯ:

1. Створити клас *Person* і описати в ньому властивість *name*.
2. В ініціалізаторі присвоїти ім'я і вивести в лог
3. Описати *deinit* і вивести в лог інформацію
4. Створити об'єкт типу *Person*
5. Запусти *playground* - і звернути увагу на те, що лог з *deinit* ніколи не викликається
6. Створити *scope* (наприклад *do {}*) для створеного об'єкту і перезапустити *playground*. Звернути увагу на логи
7. Створити об'єкт *Son* (злогами в *init/dealloc* як і в *Person*) з властивістю *parent* і додати властивість *child* в *Person*
8. Зберегти посилання *Person* в *Son*, *Son* в *Person*
9. Запустити код - звернути увагу на логи (*deinit* не викликається)
10. Модифікувати код так, щоб *deinit* з кожного об'єкту викликався

ЗАВДАННЯ №2

ОПИС:

Практика з *reference cycles in closures*.

ДЕТАЛІ:

Реалізуйте пункти описані нижче.

ПУНКТИ ВИКОНАННЯ:

Частина 1

1. Створити об'єкт *Operand* з властивістю *number*
2. Додати в об'єкт властивість типу *operand: Operand?*

3. Створити closure для обчислення суми 2-х чисел з кількох *Operand* об'єктів
4. Написати код який створить retainCycle
5. Написати код який вирішує проблему з попереднього пункту

Частина 2

1. Додати в клас *Operand* closure як властивість, яка робить якусь операцію (без параметрів). Виконати цей блок і *init*.

Приклад

```
var aBlock: () -> ()? = nil
init() {
    print("init")
    aBlock = { in
        print(self?.aConstant)
    }
}

deinit {
    print("deinit")
}
```

2. Створити об'єкт типу *Operand* і відразу занілити його
3. Перевірити логи (deinit не викликається)
4. Модифікувати код з використанням weak / unowned в capture list для вирішення даної проблеми

ЗАВДАННЯ №3 (Optional)*

ОПИС:

Робота з MMR. Objective-C

ДЕТАЛІ:

Реалізуйте пункти нижче.

ПУНКТИ ВИКОНАННЯ:

1. Створіть проект Objc-C для консольного додатку macOSx
2. При створенні класів використати -fno-objc-arc флаг для файлів (<https://stackoverflow.com/questions/6646052/how-can-i-disable-arc-for-a-single-file-in-a-project>)
3. Створити клас *MyNumber* з властивістю *value* використавши принципи MMR
4. В main функції створити об'єкти типу *MyNumber*
5. Присвоїти значення *value*
6. Вивести значення в консоль
7. Забезпечити звільнення пам'яті від об'єктів

8. Додати альтернативний спосіб з використанням autorelease pool, autorelease

ЗАВДАННЯ №4

ОПИС:

Практика з Debug Memory Graph.

ДЕТАЛІ:

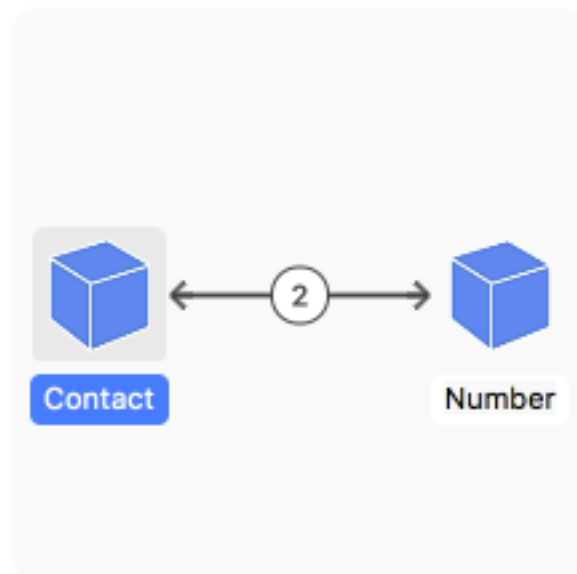
Реалізуйте пункти описані нижче.

ПУНКТИ ВИКОНАННЯ:

1. Завантажте код з посилання ([Download the starter project](https://www.raywenderlich.com/)) (Використано <https://www.raywenderlich.com/>)
2. Запустіть проект і видаліть кілька рядків (використайте swipe)
3. Натисніть Debug Memory Graph



4. Знайдіть в Debug Memory Graph знак оклику (фіолетовий)
5. Виберіть підсвічений пункт



6. Згідно (<https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/MemoryMgmt/Articles/mmPractical.html>) - parent object should have a strong hold on a child object by convention — not the other way around.
7. Знайдіть і замініть код, згідно опису нижчу

```
class Number {  
    unowned var contact: Contact  
    // Other code...  
}  
  
class Contact {  
    var number: Number?  
    // Other code...  
}
```

8. Запустіть проект і дослідіть поведінку лічильників посилань. Проблема має зникнути.

ТЕХНІЧНІ ВИМОГИ

Додаткові вимоги для успішного виконання тестового завдання

Операційна система: OS X Sierra or higher

Середовище розробки: Xcode 9.X or higher

Платформа: Playground for Swift, Command line template for Objective-C

Мова програмування: Swift / Objective-C

Місце здачі: нова гілка репозиторію

ЗВОРОТНІЙ ЗВ'ЯЗОК

У разі виникнення будь-яких питань стосовно матеріалу лекції можна звернутися до

Лектор: Далекорей Анатолій

E-mail: toljadal@gmail.com

Skype: відповідний канал