



Piscine iOS Swift - Day 00

Calculette

Summary: This document contains the subject for the D00 for the iOS Swift piscine of
42

Contents

I	Consignes	2
II	Introduction	3
III	Exercise 00: Hello World	4
IV	Exercise 01: Supersize me	5
V	Exercise 02: Moar buttons!	6
VI	Exercise 03: Make some code!	7
VII	Exercise 04: Overflows	8

Chapter I

Consignes

Sauf contradiction explicite, les consignes suivantes seront valables pour tous les jours de cette Piscine.

- Seul ce sujet sert de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices. L'url de votre dépôt **GIT** pour cette journée est disponible sur votre intranet.
- Vos exercices seront évalués par vos camarades de Piscine.
- En plus de vos camarades, vous pouvez être évalués par un programme appelé la Moulinette. La Moulinette est très stricte dans sa notation car elle est totalement automatisée. Il est donc impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les mauvaises surprises.
- Les exercices shell doivent s'exécuter avec `/bin/sh`.
- Vous ne devez laisser aucun autre fichier que ceux explicitement spécifiés par les énoncés des exercices dans votre dépôt de rendu.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Toutes les réponses à vos questions techniques se trouvent dans les **man** ou sur Internet.
- Pensez à discuter sur le forum Piscine de votre Intra et sur Slack !
- Lisez attentivement les exemples car ils peuvent vous permettre d'identifier un travail à réaliser qui n'est pas précisé dans le sujet à première vue.
- Réfléchissez. Par pitié, par Thor, par Odin !

Chapter II

Introduction

To start the development of an iOS application in Swift, you have to understand how Xcode works.

Xcode is an [IDE](#) developed by Apple that allows to design applications for Mac OS X, iOS, watchOS and tvOS.


Swift is an open source multi-paradigm programming language developed by Apple. It is very young since its first version was released on June 2nd 2014.

On this first day of piscine, you will learn to use Xcode and discover Swift developing a little calculator application.

This application will help you make your first steps in the realm of mobile development discovering how to create links between the view and the code. This application will only take wholes and basic operations into account.

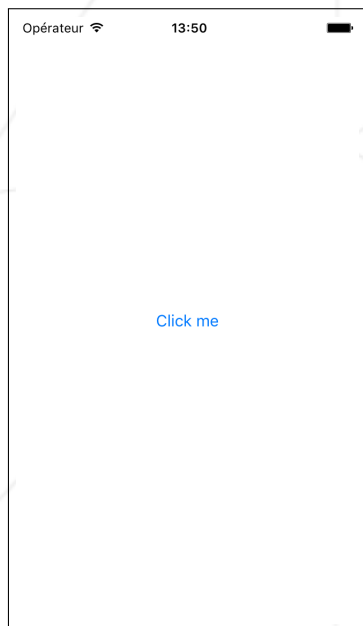
Chapter III

Exercise 00: Hello World

	Exercice : 00
Hello World	
Files to turn in : Swift Standard Library, UIKit	
Authorised functions : n/a	
Notes : n/a	


For this first exercise, you must create your first Xcode project for iOS in Swift language... Obviously. As far as I know, this is not an Objective-C piscine. Fortunately.

Create a main view, a **UIButton** that, when clicked, displays **ANY KIND** of message in the deXcode debug console.



Chapter IV

Exercise 01: Supersize me

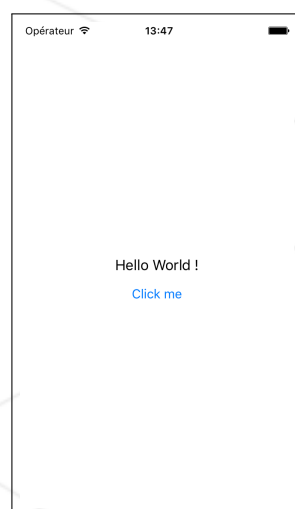
	Exercice : 01
Supersize me	
Files to turn in : Swift Standard Library, UIKit	
Authorised functions : n/a	
Notes : n/a	

Add a **UILabel** to the project in your main view that, when the **UIButton** is clicked, changes the label's text. You must also deal with the **AutoLayout**.

The UIButton and the UILabel must be horizontally centered on all the devices, even in landscape mode.




Using a StackView can be useful for the autolayout.



Chapter V

Exercise 02: Moar buttons!

	Exercise : 02
Moar buttons!	
Files to turn in : Swift Standard Library, UIKit	
Authorised functions : n/a	
Notes : n/a	

It lacks keys, don't you think?

You will now add all the keys of a simple calculator. Of course, it will be **UIButton**:

- Numbers from 0 to 9
- 'AC' will reinitialize the calculator
- '=' will execute the operation with both operands
- Operators '+', '-', '/', '*'
- 'NEG' will take the opposite of the current number


Once all the **UIButton**s have been properly set, make sure the **AutoLayout** is still manages (on all devices and in all modes).

Keys assigned to numbers must be able to modify the **UILabel** changing the number displayed above, but you won't have to program other keys for the moment.

Take the opportunity to add a little debug. For each push on a **UIButton**, the action must appear in the debug console (the format is free).

Chapter VI

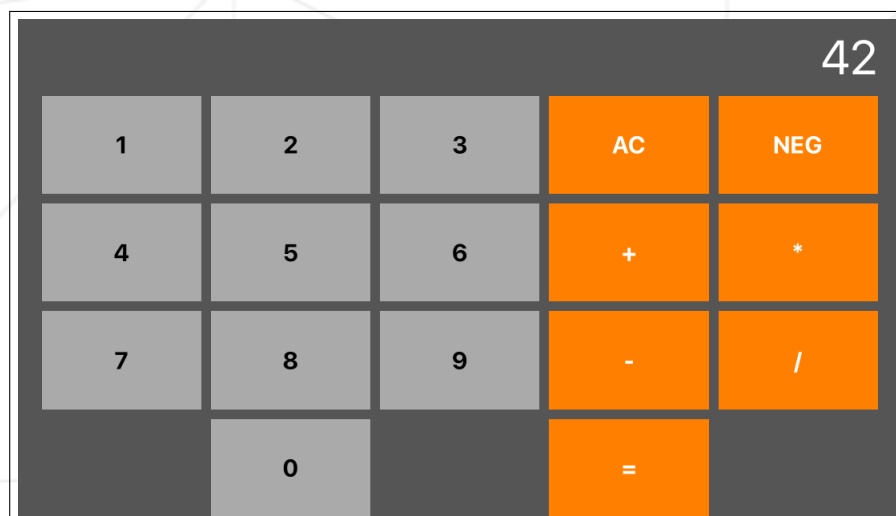
Exercice 03: Make some code!

	Exercice : 03
Make some code!	
Files to turn in : Swift Standard Library, UIKit	
Authorised functions : n/a	
Notes : n/a	

Now, you will code actions to execute when an operation is requested. The **UILabel** must be able to display the operation result and you must be able to chain the operations. Likewise, the **AutoLayout** must always be managed.




Beware the division by 0!



Chapter VII

Exercise 04: Overflows

	Exercise : 04
Overflows	
Files to turn in : Swift Standard Library, UIKit	
Authorised functions : n/a	
Notes : n/a	

If you've properly ran your tests in the previous exercises, you may have noticed your application crashes when the numbers become to big (positive as well as negative). This is what they call an **overflow**.

Neglecting the **overflows** can cause [heavy damage](#)!

In this exercise, you will fix the problem implementing **overflows** management.



Your application must NEVER crash!