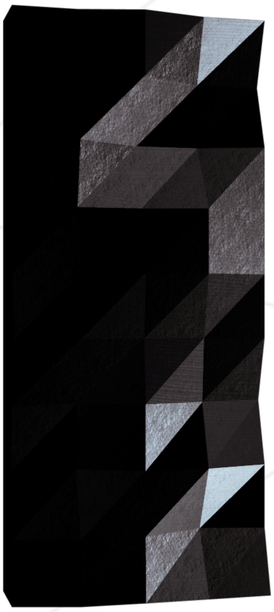


Piscine iOS Swift - Day 00

Calculette

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

Contents

Chapter I

Instructions

- Only this page will serve as reference. Do not trust rumors.
- Read attentively the whole document before beginning.
- Your exercises will be corrected by your piscine colleagues.
- The document can be relied upon, do not blindly trust the demos which can contain not required additions.
- You will have to deliver an app every day (except for Day 01) on your git repository, where you deliver the file of the Xcode project.
- Here it is the official manual of Swift and of Swift Standard Library
- It is forbidden to use other libraries, packages, pods...before Day 07
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Think about discussing on the forum Piscine of your Intra!
- Use your brain!!!

⚠ The videos on Intra were produced before Swift 3. Remove the prefix "NS" which you see in front of the class/struct/function in the code in the videos to use them in Swift 3.

⚠ Intra indicates the date and the hour of closing for your repositories. This date and hour also corresponds to the beginning of the peer-evaluation period for the corresponding piscine day. This peer-evaluation period lasts exactly 24h. After 24h passed, your missing peer grades will be completed with a 0.

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.

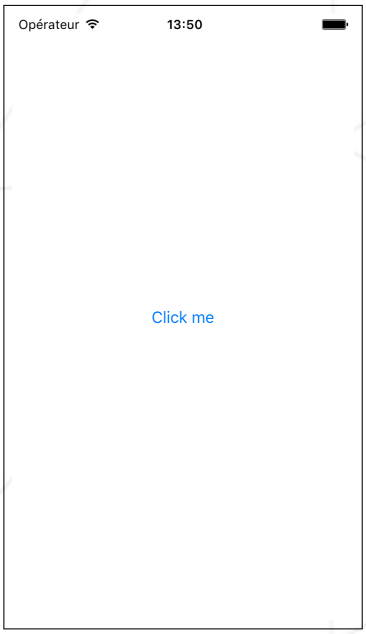
Chapter III

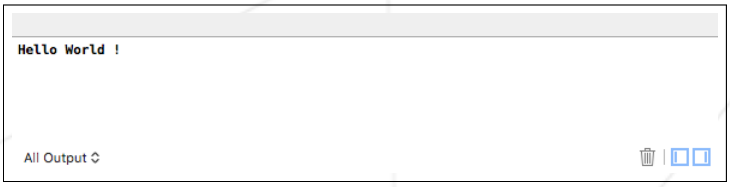
Exercise 00: Hello World

	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.





```
Hello World !
```

Chapter IV

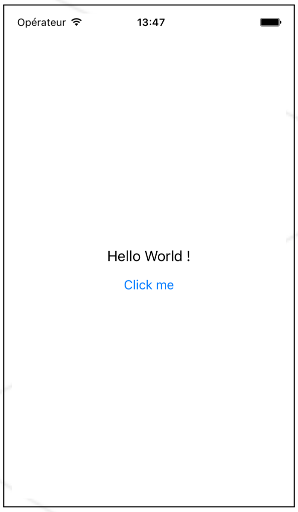
Exercise 01: Supersize me

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

Exercice 02: Moar buttons!

	Exercice : 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 **UIButtons** 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).

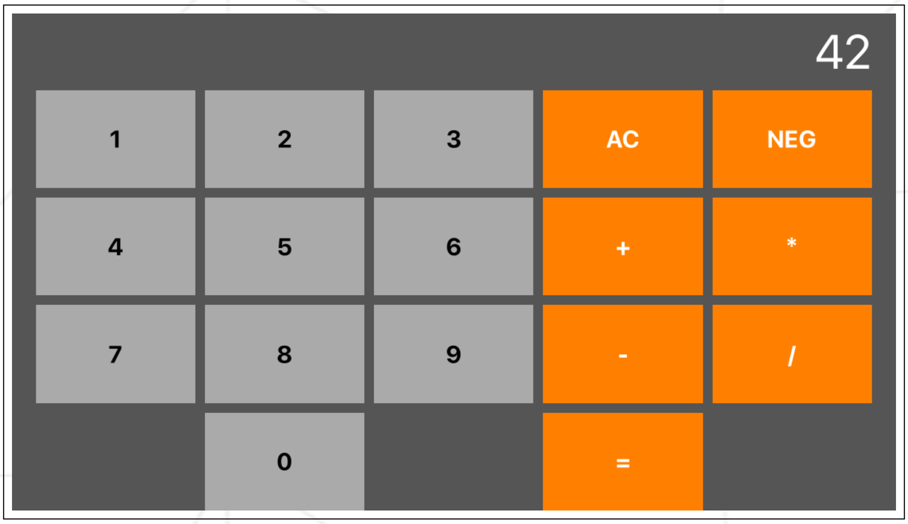
Chapter VI

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

	Exercice : 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!`