



Piscine iOS Swift - Day 00

Calculator

Maxime LEMORT mlemort@student.42.fr
42 Staff pedago@42.fr

*Summary: This document contains the subject for Day 00 for the „Piscine iOS Swift”
from [42](#)*

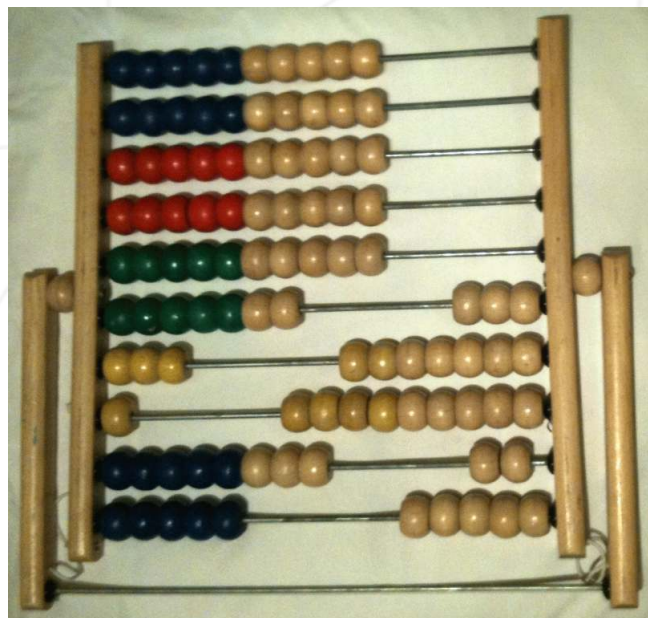
Contents

I	Foreword	2
I.1	Etymology	3
I.2	School abacus	3
I.3	Uses by the blind	4
I.4	Binary abacus	4
II	General Instructions	5
III	Introduction	7
IV	Exercise 00 : Hello World	8
V	Exercise 01 : Supersize me	9
VI	Exercise 02 : Moar buttons !	10
VII	Exercise 03 : Make some code!	11
VIII	Exercise 04 : Overflows	12

Chapter I

Foreword

Here is what wikipedia has to say about the abacus:



The abacus (plural abaci or abacuses), also called a counting frame, is a calculating tool that was in use in Europe, China and Russia, centuries before the adoption of the written Hindu–Arabic numeral system. The exact origin of the abacus is still unknown. Today, abaci are often constructed as a bamboo frame with beads sliding on wires, but originally they were beans or stones moved in grooves in sand or on tablets of wood, stone, or metal.

Abaci come in different designs. Some designs, like the bead frame consisting of beads divided into tens, are used mainly to teach arithmetic, although they remain popular in the post-Soviet states as a tool. Other designs, such as the Japanese soroban, have been used for practical calculations even involving several digits. For any particular abacus design, there usually are numerous different methods to perform a certain type of calculation, which may include basic operations like addition and multiplication, or even more complex ones, such as calculating square roots. Some of these methods may work with non-natural numbers (numbers such as 1.5 and $3/4$).

Although today many use calculators and computers instead of abaci to calculate, abaci still remain in common use in some countries. Merchants, traders and clerks in some parts of Eastern Europe, Russia, China and Africa use abaci, and they are still used to teach arithmetic to children. Some people who are unable to use a calculator because of visual impairment may use an abacus.

I.1 Etymology

The use of the word abacus dates before 1387 AD, when a Middle English work borrowed the word from Latin to describe a sandboard abacus. The Latin word came from Greek abax which means something without base, and improperly, any piece of rectangular board or plank. Alternatively, without reference to ancient texts on etymology, it has been suggested that it means “a square tablet strewn with dust”, or “drawing-board covered with dust (for the use of mathematics)” (the exact shape of the Latin perhaps reflects the genitive form of the Greek word, abakos). Whereas the table strewn with dust definition is popular, there are those that do not place credence in this at all and in fact state that it is not proven. Greek abax itself is probably a borrowing of a Northwest Semitic, perhaps Phoenician, word akin to Hebrew, “dust” (or in post-Biblical sense meaning “sand used as a writing surface”).

The preferred plural of abacus is a subject of disagreement, with both abacuses and abaci in use. The user of an abacus is called an abacist.

I.2 School abacus

Around the world, abaci have been used in pre-schools and elementary schools as an aid in teaching the numeral system and arithmetic.

In Western countries, a bead frame similar to the Russian abacus but with straight wires and a vertical frame has been common. It is still often seen as a plastic or wooden toy.

The wire frame may be used either with positional notation like other abaci (thus the 10-wire version may represent numbers up to 9,999,999,999), or each bead may represent one unit (so that e.g. 74 can be represented by shifting all beads on 7 wires and 4 beads on the 8th wire, so numbers up to 100 may be represented). In the bead frame shown, the gap between the 5th and 6th wire, corresponding to the color change between the 5th and the 6th bead on each wire, suggests the latter use.

The red-and-white abacus is used in contemporary primary schools for a wide range of number-related lessons. The twenty bead version, referred to by its Dutch name rekenrek (“calculating frame”), is often used, sometimes on a string of beads, sometimes on a rigid framework.

I.3 Uses by the blind

An adapted abacus, invented by Tim Cranmer, called a Cranmer abacus is still commonly used by individuals who are blind. A piece of soft fabric or rubber is placed behind the beads so that they do not move inadvertently. This keeps the beads in place while the users feel or manipulate them. They use an abacus to perform the mathematical functions multiplication, division, addition, subtraction, square root and cube root.

Although blind students have benefited from talking calculators, the abacus is still very often taught to these students in early grades, both in public schools and state schools for the blind. The abacus teaches mathematical skills that can never be replaced with talking calculators and is an important learning tool for blind students. Blind students also complete mathematical assignments using a braille-writer and Nemeth code (a type of braille code for mathematics) but large multiplication and long division problems can be long and difficult. The abacus gives blind and visually impaired students a tool to compute mathematical problems that equals the speed and mathematical knowledge required by their sighted peers using pencil and paper. Many blind people find this number machine a very useful tool throughout life.

I.4 Binary abacus

The binary abacus is used to explain how computers manipulate numbers. The abacus shows how numbers, letters, and signs can be stored in a binary system on a computer, or via ASCII. The device consists of a series of beads on parallel wires arranged in three separate rows. The beads represent a switch on the computer in either an “on” or “off” position.

Chapter II

General Instructions

- Only this document will serve as reference. Do not trust rumors.
- Read carefully the whole subject before beginning.
- Watch out! This document could potentially change up to an hour before submission.
- This project will be corrected by humans only.
- The document can be relied upon, do not blindly trust the demos which can contain unrequired additions.
- You will have to submit one app every day (except for Day 01) on your git repository, submit the folder 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, etc. before Day 07
- Got a question ? Ask your peer on the right. Otherwise, try your peer on the left.
- You can discuss on the Piscine forum of your Intra !
- By Odin, by Thor ! 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 in order 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 0.

Chapter III

Introduction

To begin the developpement of an iOS application in Swift properly, it's important to understand how Xcode works.

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


Swift is a multi-paradigm open source programmation langage developped by Apple as well. It is very recent since its first version was released on the 2nd of June 2014.

In this first day of the bootcamp you'll learn how to use Xcode and discover Swift by creating a small calculator application.

This application will allow you to toddle in the world of mobile development by making you discover how to create links between views and the code. This application will only support integers and basic operations.

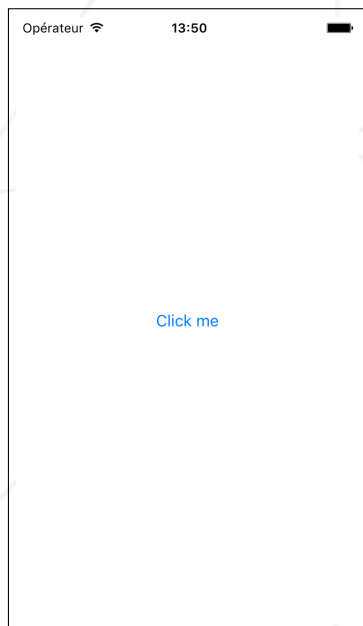
Chapter IV

Exercise 00 : Hello World

	Exercice : 00
Hello World	
Files to turn in : .xcodeproj and all the necessary files	
Allowed functions : Swift Standard Library, UIKit	
Notes : n/a	


For this first exercise you'll have to create your first Xcode project, for iOS in Swift... Yes, until proven otherwise, fortunately this isn't an Objective-C bootcamp.

Create on the main view, a **UIButton** which, when clicked on, will display an **ORDINARY** message in Xcode's debug console.



Chapter V

Exercise 01 : Supersize me

	Exercice : 01
Supersize me	
Files to turn in : .xcodeproj and all the necessary files	
Allowed functions : Swift Standard Library, UIKit	
Notes : n/a	

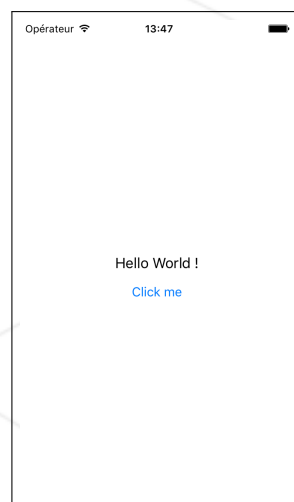
Add to this project an **UILabel** in your main view which will, when clicking on the **UIButton**, change the text of the Label.

You also have to manage the **AutoLayout**.

The **UIButton** as well as the **UILabel** must be horizontally centered on every device, even in landscape mode.




The use of **StackView** could prove helpful for the autolayout.



Chapter VI

Exercise 02 : Moar buttons !

	Exercice : 02
Moar buttons !	
Files to turn in : .xcodeproj and all the necessary files	
Allowed functions : Swift Standard Library, UIKit	
Notes : n/a	

Don't you think that there is some missing buttons?
You will now add every keys of a simple calculator, obviously you will use **UIButton**:

- Numbers from 0 to 9
- 'AC' to reset the calculator
- '=' which will compute the operation with the 2 operands
- The basic operators '+', '-', '/', '*'
- 'NEG' which will change the sign of the current number


When you have every **UIButton**s properly placed, make sure that the **AutoLayout** is still managed (ie on every device on every orientation).

The buttons assigned to numbers must be able to update the **UILabel** when changing the number displayed on it, but the other ones don't have to be programmed for now.

You'll also have to add a little bit of debug. Each time a **UIButton** is pressed, its action must be described in the debug console (the formatting is free).

Chapter VII

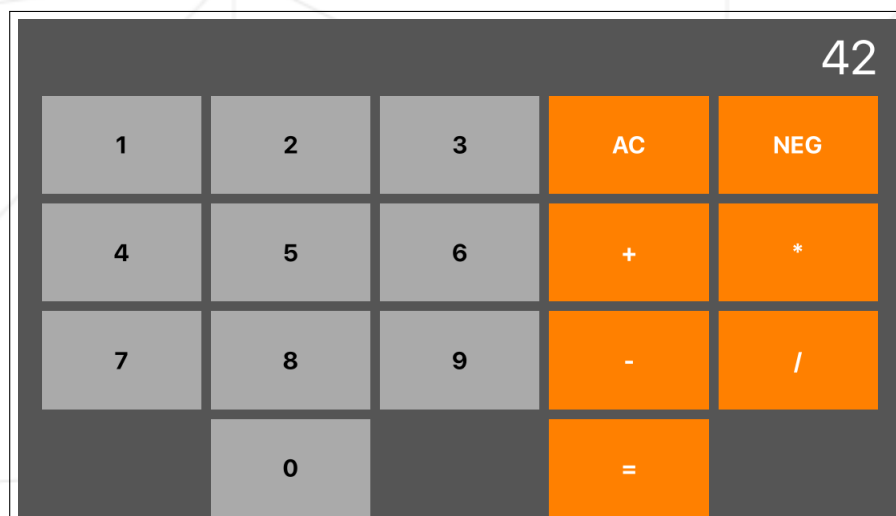
Exercise 03 : Make some code!

	Exercice : 0 03
Make some code!	
Files to turn in : .xcodeproj and all the necessary files	
Allowed functions : Swift Standard Library, UIKit	
Notes : n/a	

Now, you'll have to code the actions to run when an operation is clicked.
The **UILabel** must be able to display the result of the operation and it must be possible to chain operations. As usual, the **AutoLayout** must still be properly managed.




Be careful about the 0 division !



Chapter VIII

Exercise 04 : Overflows

	Exercise : 04
Overflows	
Files to turn in : .xcodeproj and all the necessary files	
Allowed functions : Swift Standard Library, UIKit	
Notes : n/a	

If you did accurate tests during the last exercise, you probably noticed that your app crashes when numbers are too big (positive as well as negative). You're facing a basic **overflow**.

Remember how much **damages** can an **overflows** omission cause!

In this exercise, you have to address this problem by incorporation **overflows** management.



Your app cannot crash, under any circumstances!