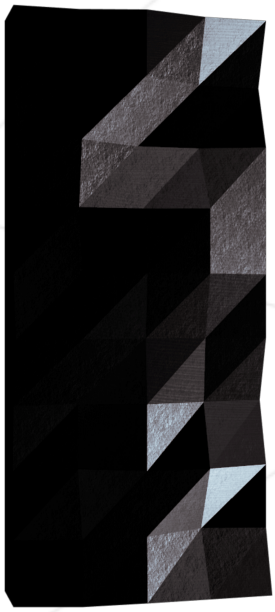


Piscine iOS Swift - Day 00

Calculator

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

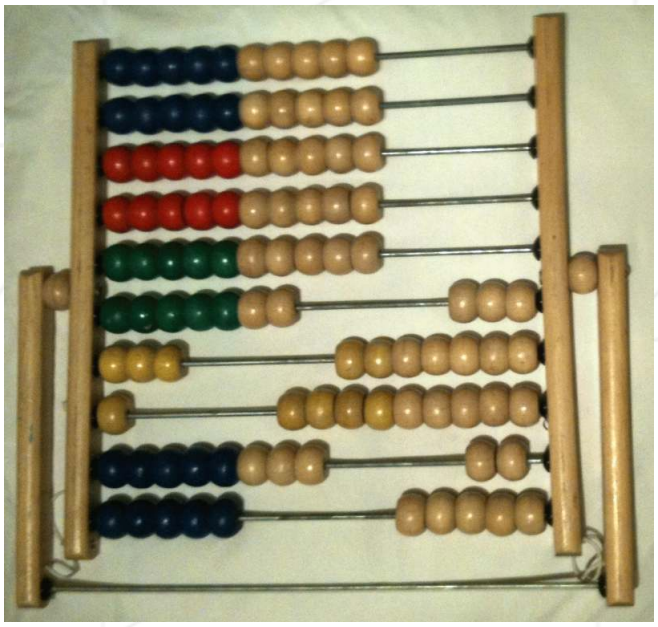
Summary: This document contains the subject for Day 00 for the „Piscine iOS Swift" from 42

Contents

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.

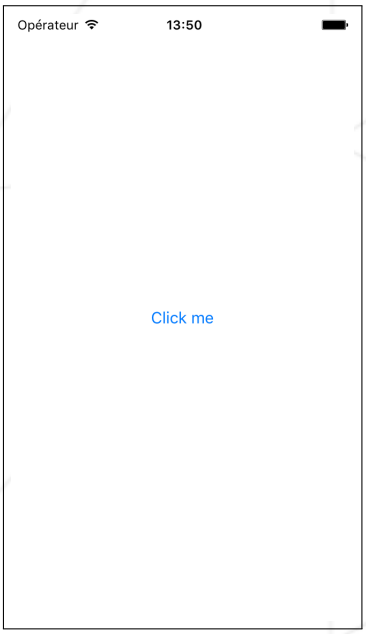
Chapter IV

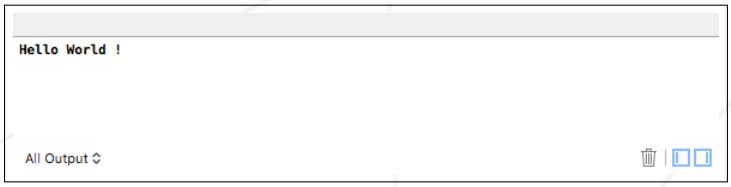
Exercise 00 : Hello World

	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

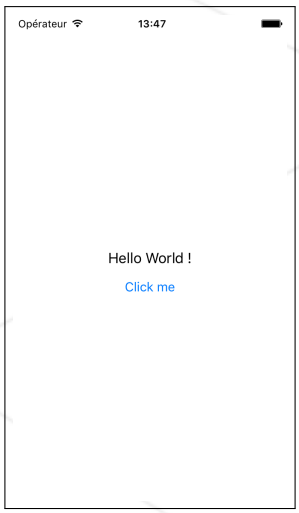
	Exercice : 01
Supersize me	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

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 horizontaly 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 **UIButtons** 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).

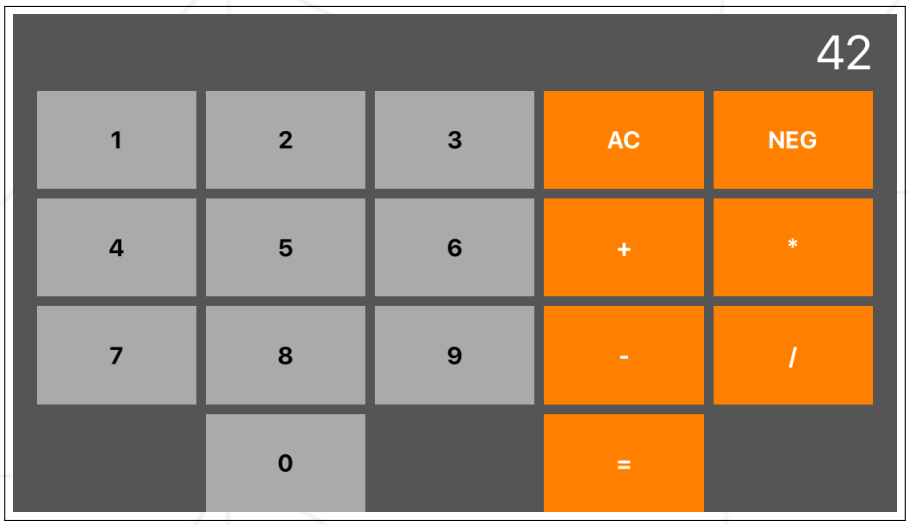
Chapter VII

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

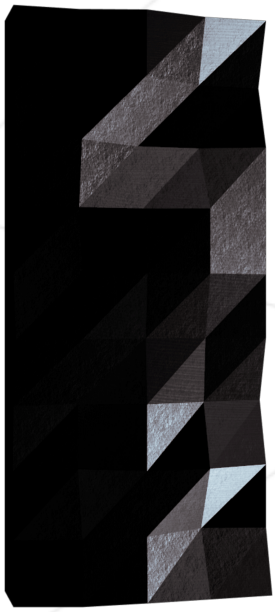
	Exercice : 04
Overflows	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

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**.

Rememeber 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!`



Piscine iOS Swift - Day 01

Card Game

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

Summary: This document contains the subject for Day 01 for „Piscine iOS Swift" from 42

Contents

Chapter I

Foreword

"Actually, e-sport, is considered a game of chance. This title depends of „Authority of online games regulations (fr. ARJEL) and the competitions could even be prohibited... However, given the legal vacuum on the e-sports, they are for the moment tolerated." Source

"With over 225 millions uniques spectators, the audience of eSports is considered as important as all professional sports leagues." Source

"Participation in these competitions promotes the team spirit, control and self-transcendence. Their broadcast mode, online, also encourages integration and the knowledges exchange between different cultures. In addition, the economic potential of these competitions, which revolves around entrance tickets, products and audiovisual rights, as well as tourism, is important. It is estimated that it will reach 800 milion euros in 2018. So it is an opportunity for France, socially and economically." Source

"The draft law on digital, presented by Axelle Lemaire has been accepted. The Government therefore disclosed the main points concerning this law, including recognition of e-Sport."" Source

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

Specific instructions of the day

This is a particular day. You will not compile an application, but you will do some exercises to discover Swift programming language.

- You have to create a new folder per exercise, at the root of your repository: *ex00 ex01 ex02...*
- You will compile the exercises with **swiftc**.
- You have to make your own test sets to prove that everything works correctly during defense. What is not tested, will not be evaluated.
- You can reuse files from previous exercises.

Chapter IV

Introduction

Swift is a multi-paradigm programming language, but mainly oriented to protocol, thanks to its two key words: `protocol, extension`. For a better understanding of this notions, first of all we must focus on object-oriented development.

Today we will focus on a classic 52 cards game through various small exercises that are meant to familiarize you with swift and make you use several notions:

The declarations: `var, let, type, weak, optional` to know how to type your variables.

Control structures: `loop, conditions, if let` to structure your code.

Classes: `class, func, overload, override, struct, enum, inheritance, extension, mutating` for object-oriented development.

Algo: `closures` for the anonymous functions.

This day will prepare you for the rest of the bootcamp. Try to go as far as possible.

Chapter V

Exercise 00 : Color and Value

	Exercise 00
Color and Value	
Turn-in directory : $ex00/$	
Files to turn in : `Color.swift, Value.swift, test file`	
Allowed functions : `None`	
Notes : `n/a`	

For the beginning we will define what is a colour and a value of a classic 52 cards game.

Create an enum **Color** with **String** type as the raw value that will represent the 4 colors. Add a constant static **allColors** of type [**Color**] which will represent all the possible colors of a card.

Now create an enum **Value** with the raw value of the type **Int** which will represent the values of the cards. Add a constant static **allValues** of type [**Value**] which will represent all the possible values of a card.

Chapter VI

Exercise 01 : Card

	Exercise 01
Card	
Turn-in directory : $ex01/$	
Files to turn in : `Color.swift, Value.swift, Card.swift, test file`	
Allowed functions : `None`	
Notes : `n/a`	

Create class **Card** which inherites from **NSObject** with :

- Properties **color** and **value**.
- A class constructor which takes a **Color** and a **Value** as parameters.
- An override of the property **var description: String** that allows you to describe the card.
- An override of the method **isEqual** of **NSPbject**

Overload the "==" operator to work on 2 **Card** which does pretty much the same thing as the **isEqual** method.

Here you have an example:

```
> let card1 = Card(c : Color.Spade, v : Value.Ace)
card1: Card = {
        ObjectiveC.NSObject = {
                isa = __lldb_expr_9.Card
        }
        color = Spade
        value = Ace
}
> print(card1)
(1, Spade)
> let card2 = Card(c : Color.Diamond, v: Value.Two)
card2: Card = {
        ObjectiveC.NSObject = {
        isa = __lldb_expr_9.Card
        }
        color = Diamond
        value = Two
}
> print(card2)
(2, Diamond)
> print(card1 == card2)
false
```

Chapter VII

Exercise 02 : Deck

	Exercise 02
Deck	
Turn-in directory : $ex02/$	
Files to turn in : `Color.swift, Value.swift, Card.swift, Deck.swift, a test file`	
Allowed functions : `Array's map method`	
Notes : `n/a`	

Create the class **Deck** inheritor of **NSObject**.
Add the static constants of type [**Card**] :

allSpades : which represents all `spades`

allDiamonds : which represents all `diamonds`

allHearts : which represents all `hearts`

allClubs : which represents all `clubs`

Add the static constant **allCards** of type [**Card**] which will be a list of all the possible cards of a game of 52.

Chapter VIII

Exercise 03 : Extension

	Exercise 03
Extension	
Turn-in directory : $ex03/$	
Files to turn in : `Color.swift, Value.swift, Card.swift, Deck.swift, a test file`	
Allowed functions : `arc4random_uniform`	
Notes : `n/a`	

The extensions are extremely useful for adding code to existing classes or structures.

For this exercise you will make an extension of **struct Array** in the **Deck.swift** file which adds a method that shuffles randomly the array.

Chapter IX

Exercise 04 : Board

	Exercise 04
Board	
Turn-in directory : $ex04/$	
Files to turn in : `Color.swift, Value.swift, Card.swift, Deck.swift, a test file`	
Allowed functions : `Toutes les méthodes de Array`	
Notes : `n/a`	

Add 3 properties of type [**Card**] at **Deck** class :

cards : which represents all the cards from the deck

disards : which represents the discard pile

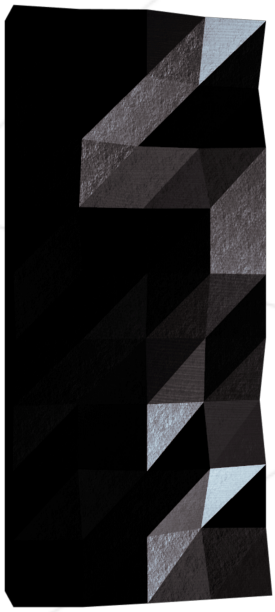
outs : which represent all the cards that are no longer in `cards` and not yet in `discard`

Create an class constructor which takes a **Bool** parameter which represents if the deck has to be sorted or mixed.

Override the property **var description: String** which returns all cards of **cards**.

Create the **draw () -> Card?** method that draws the first card of **cards** and place it in **outs**.

Create the **fold (c: Card)** method that places the card c in **discards** if it belongs to **outs**.



Piscine iOS Swift - Day 02

Death Note

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

Summary: This document contains the subject for Day 02 for „Piscine iOS Swift" from 42

Contents

Chapter I

Foreword

Thanks to Marmiton for this recipe:

I.1 Ramen (Japanese noodles) with fried shrimp

Preparation time : 5 minutes

Cooking time : 10 minutes

I.1.1 Ingredients (4 servings))

- 500 g of ramen (Japanese noodles, in the "exotic grocery" section of the supermarket)
- 2 carrots
- 2 small leeks
- 400 g of small peeled shrimps (frozen)
- 1 garlic clove
- 2 tablespoons of peanut oil
- Salt, pepper, soy sauce and sesame oil depending on yourtaste...

I.1.2 Preparing the recipe

Thinly slice the leeks and carrots with a razor, cut them into strips and cut them into "filaments" with a knife.

Chop the garlic clove.

Heat a tablespoon of peanut oil in a frying pan or skillet.

When it is very hot, throw the carrots and the leeks, and stir constantly for 2 min (they must remain crunchy).

Add the garlic clove and a pepper mill tower, then store in a dish.

Defrost shrimp by panning 2 min over high heat.

Add to vegetables by reserving 2 or 3 tablespoons of the liquid prepared.

Sauté the noodles in a spoonful of peanut oil for 1 min (They have to detach and start to color).

Add the water made by shrimp, salt, soy sauce, sesame oil (in small quantities: a few drops are sufficient in general, because it is very fragrant) and vegetables.

Stir another 1 min over high heat, and serve quickly.

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

Death Note Is a manga in which Light Yagami is the owner of a notebook of death. The person whose name is written in this book dies inevitably.

Today you will learn how to create a very simple Death Note simulation application. You will need to be able to display the name, date and description of the death of everybody in it and you will need to be able to add someone.

This day will allow you to understand how the **table views**, that are used extensively in iOS applications, works. It is a way of presenting similar data.

This day will also teach you how to change the view on an application and how to go back to an old view.

Chapter IV

Exercise 00 : TableView

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

For this first exercise you will create a **table view** and display the name and description of the death of 3 different people in the **table view** cells.

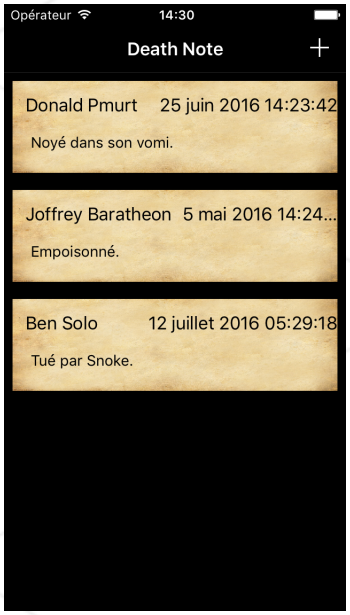
Chapter V

Exercise 01 : Turn the page

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

Add a **navigation bar** to your application. From now on you will have to put a title to all your views.

Add a button in the right slot of the **navigation bar** which allows you to go to another view (empty at the moment). We must be able to go backwards thanks to the **navigation bar**.



Chapter VI

Exercise 02 : Who will die?

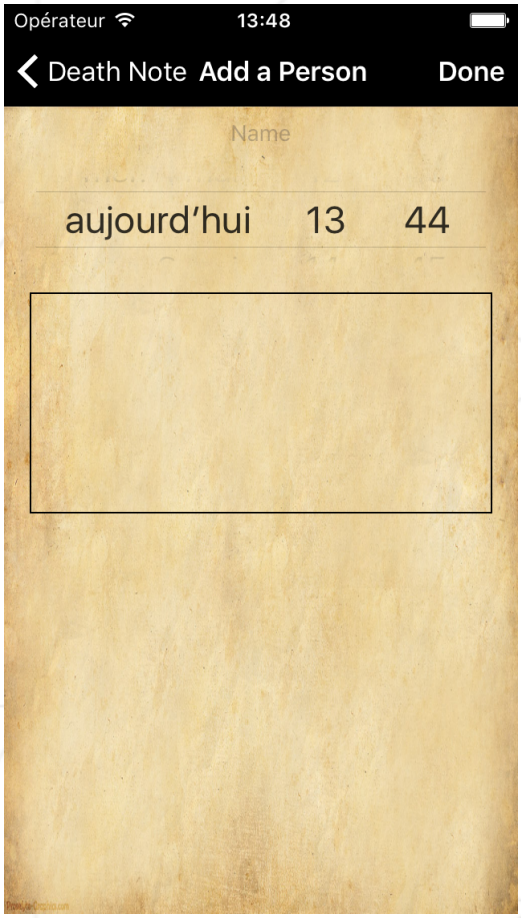
	Exercice : 02
Who will die?	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

We will now take care of the second view which will allow to add entries to our **table view**.

Add these fields :

- A field for writing the person name
- Another one for the description. It must be bigger and you have to be able to write on several lines.
- An **date picker** for the date of death. You should not be able to choose a past date.

Add an **Done** button on the right side of **navigation bar** which display in the console all the informations written in this form.



The **autolayout** must works correct on all devices!

Chapter VII

Exercise 03 : You will die

	Exercice : 03
You will die	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

Make sure that the **Done** button sends us back to the first view and displays the new person we just entered in the **table view**.

```
If the name field is empty, no one should be added.
If ONLY the description field is empty, the person is still added.
```

```
We have to RETURN to the first view and not instantiate it again.
```

Chapter VIII

Exercise 04 : Custom Death

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

Change the cell type of the **table view** to the type **Custom**. You will create your own cells as you wish, but they must contain the name, description and date of the death of all persons added in the second view.

`Do not forget to check that the application is correctly displayed on all devices.`

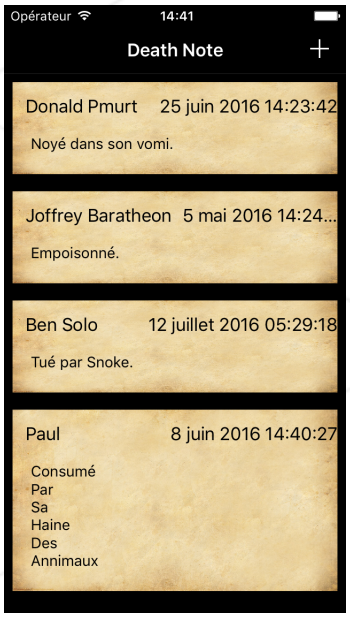
Chapter IX

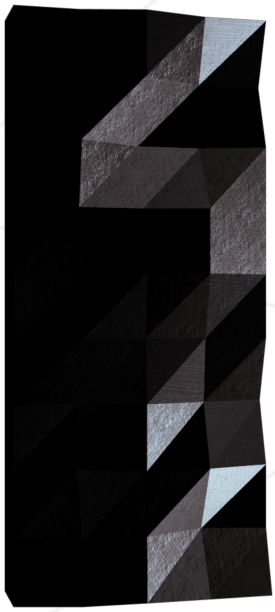
Exercise 05 : Unleash your power

	Exercice : 05
Unleash your power	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

Normally you should realize that the cells are not all adapted to their contents. Some are too large and some are too small depending on the number of lines contained in the description field.

You will solve this problem by making cells with dynamic heights.





Piscine iOS Swift - Day 03

APM

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

Summary: This document contains the subject for Day 03 for the „Piscine iOS Swift" from 42

Contents

Chapter I

Foreword

Here is an extract from Hubble's wikipedia page:

The Hubble Space Telescope (HST) is a space telescope that was launched into low Earth orbit in 1990 and remains in operation. Although not the first space telescope, Hubble is one of the largest and most versatile, and is well known as both a vital research tool and a public relations boon for astronomy. The HST is named after the astronomer Edwin Hubble, and is one of NASA's Great Observatories, along with the Compton Gamma Ray Observatory, the Chandra X-ray Observatory, and the Spitzer Space Telescope.

With a 2.4-meter (7.9 ft) mirror, Hubble's four main instruments observe in the near ultraviolet, visible, and near infrared spectra. Hubble's orbit outside the distortion of Earth's atmosphere allows it to take extremely high-resolution images, with substantially lower background light than ground-based telescopes. Hubble has recorded some of the most detailed visible light images ever, allowing a deep view into space and time. Many Hubble observations have led to breakthroughs in astrophysics, such as accurately determining the rate of expansion of the universe.

The HST was built by the United States space agency NASA, with contributions from the European Space Agency. The Space Telescope Science Institute (STScI) selects Hubble's targets and processes the resulting data, while the Goddard Space Flight Center controls the spacecraft.

Space telescopes were proposed as early as 1923. Hubble was funded in the 1970s, with a proposed launch in 1983, but the project was beset by technical delays, budget problems, and the Challenger disaster (1986). When finally launched in 1990, Hubble's main mirror was found to have been ground incorrectly, compromising the telescope's capabilities. The optics were corrected to their intended quality by a servicing mission in 1993.

Hubble is the only telescope designed to be serviced in space by astronauts. After launch by Space Shuttle Discovery in 1990, five subsequent Space Shuttle missions repaired, upgraded, and replaced systems on the telescope, including all five of the main instruments. The fifth mission was initially canceled on safety grounds following the Columbia disaster (2003). However, after spirited public discussion, NASA administrator Mike Griffin approved the fifth servicing mission, completed in 2009. The telescope is operating as of 2017, and could last until 2030–2040. Its scientific successor, the James Webb Space Telescope (JWST), is scheduled for launch in 2018.

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

The threads allow you to perform the instructions of a process by following their own call stack. Initially a process is started on a single thread, the **main thread**.

Using multiple threads allows you to parallelize the processing of multiple functions to run code in background. This point is extremely important on iOS to avoid blocking the user interface (UI) while the application is making calculations or waiting for a server to respond.

Today you will see several notions :

- How to use a **collection view**
- How to do a **multithread** on iOS
- How to create **alerts**
- How to use a **scroll view**

All ths application do, is to download images from the internet.

Chapter IV

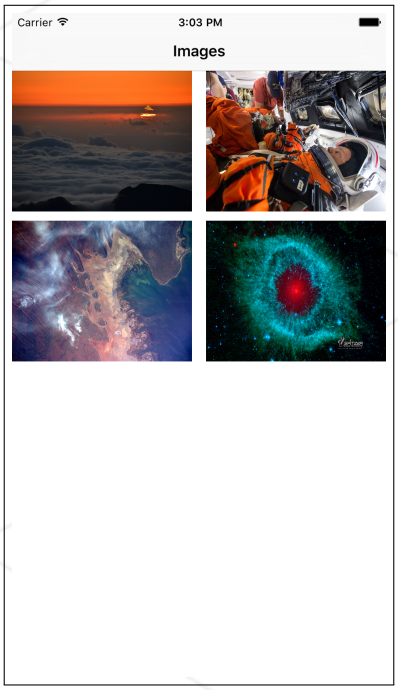
Exercise 00 : Pictures

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

The **collection view** is a tool that allows to display data, different from a **table view** but their use is almost identical.

Create a **collection view** that displays at least 4 photos of the web of your choice. The 4 photos must be displayed in full in the **collection view**.

Take big pictures so that downloads take time. You can search for pictures on the site of the nasa for example.



Chapter V

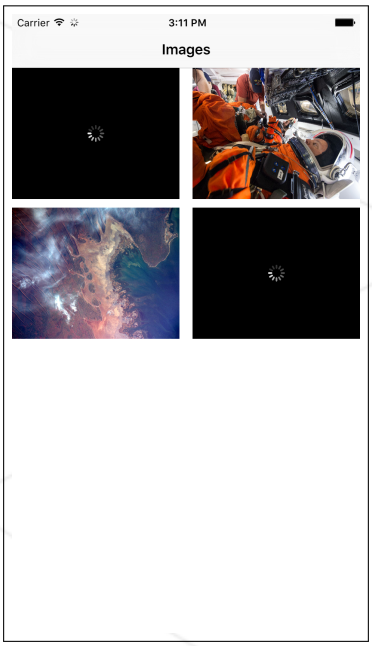
Exercise 01 : Multithreads

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

You noticed that the time the images are downloading, the UI is blocked and iOS does not respond. Calls on the **main thread** interfere with the user experience. To remedy this problem, you will make these calls asynchronous.

Also add an **activity monitor** to each view of the **collection view** that must rotate when the image is downloaded and disappears when the image is displayed.

You must also run the network activity indicator when the application uses the network and stop it when it no longer uses it.

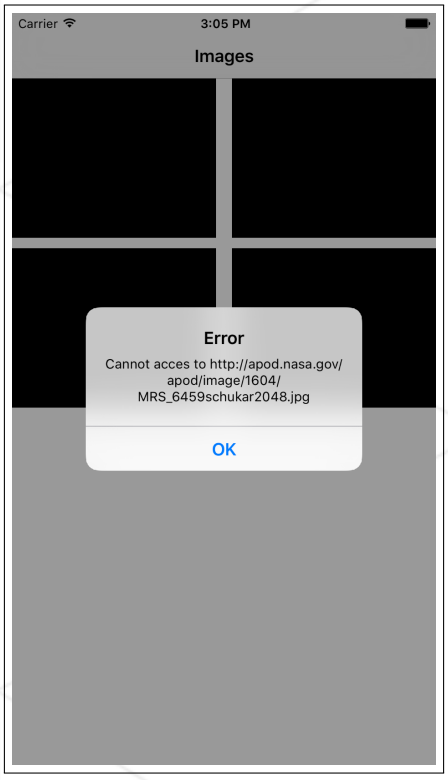


Chapter VI

Exercise 02 : Alerts

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

If there is a problem during the download of the photo, you have to make appear a simple **alert** which explains the problem with a "Ok" button to make disappear the **alert**.



Chapter VII

Exercise 03 : ScrollView

	Exercice : 03
ScrollView	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

Add a **navigation bar** with a title for each view.

Create a new view containing a **scroll view**. When you click on a cell in the **collection view**, you will need to display the **scroll view** with the large image. We must be able to move the image.

Chapter VIII

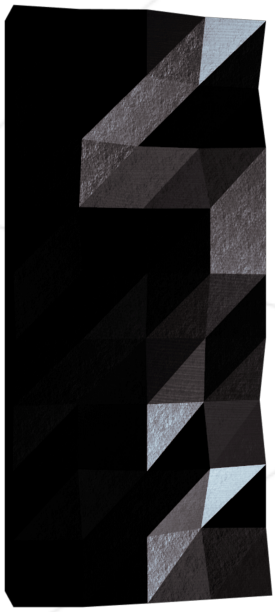
Exercise 04 : Zoom

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

To move the image is good, to be able to zoom is better. Make it possible to zoom in and zoom out of the image.

It is also necessary that the image holds perfectly in width with the maximum zoom out, and that whatever the device and its orientation!


Images



Piscine iOS Swift - Day 04

Tweets

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

Summary: This document contains the subject for Day 04 for the „Piscine iOS Swift" from 42

Contents

Chapter I

Foreword

Here is what wikipedia has to say about homing pigeon:

The homing pigeon is a variety of domestic pigeon (Columba livia domestica) derived from the rock pigeon, selectively bred for its ability to find its way home over extremely long distances. The wild rock pigeon has an innate homing ability, meaning that it will generally return to its nest, (it is believed) using magnetoreception. This made it relatively easy to breed from the birds that repeatedly found their way home over long distances. Flights as long as 1,800 km (1,100 miles) have been recorded by birds in competitive pigeon racing. Their average flying speed over moderate 640 km (400 miles) distances is around 80 km/h (50 miles per hour) but speeds of up to 140 km/h (90 miles per hour) have been observed in top racers for short distances.

Because of this skill, homing pigeons were used to carry messages as messenger pigeons. They are usually referred to as “pigeon post” or “war pigeon” during wars.

Homing pigeons are often incorrectly categorized as carrier pigeons, a breed of fancy pigeons selectively-bred for its distinctively rounded hard wattle.

I.1 History

The sport of flying homing pigeons was well-established as early as 3000 years ago. They were used to proclaim the winner of the Olympics. Messenger pigeons were used as early as 1150 in Baghdad and also later by Genghis Khan. By 1167 a regular service between Baghdad and Syria had been established by Sultan Nour-Eddin. In Damietta, by the mouth of the Nile, the Spanish traveller Pedro Tafur saw carrier pigeons for the first time, in 1436, though he imagined that the birds made round trips, out and back. The Republic of Genoa equipped their system of watch towers in the Mediterranean Sea with pigeon posts. Tipu Sultan used carrier pigeons. They returned to the Jamia Masjid mosque in Srirangapatna, which was his headquarters. The pigeon holes may be seen in the mosque's minarets to this day.

In 1818, a great pigeon race called the Cannonball Run took place at Brussels. In 1860, Paul Reuter, who later founded Reuters press agency, used a fleet of over 45 pigeons to deliver news and stock prices between Brussels and Aachen, the terminus of early telegraph lines. The outcome of the Battle of Waterloo was also first delivered by a pigeon to England. During the Franco-Prussian War pigeons were used to carry mail between besieged Paris and the French unoccupied territory. In December 1870, it took ten hours for a pigeon carrying microfilms to fly from Perpignan to Bruxelles.

Historically, pigeons carried messages only one way, to their home. They had to be transported manually before another flight. However, by placing their food at one location and their home at another location, pigeons have been trained to fly back and forth up to twice a day reliably, covering round-trip flights up to 160 km (100 mi). Their reliability has lent itself to occasional use on mail routes, such as the Great Barrier Pigeongram Service established between the Auckland, New Zealand, suburb of Newton and Great Barrier Island in November 1897, possibly the first regular air mail service in the world. The world's first "airmail" stamps were issued for the Great Barrier Pigeon-Gram Service from 1898 to 1908.

Homing pigeons were still employed in the 21st century by certain remote police departments in Odisha state in eastern India to provide emergency communication services following natural disasters. In March 2002, it was announced that India's Police Pigeon Service messenger system in Odisha was to be retired, due to the expanded use of the Internet. The Taliban banned the keeping or use of homing pigeons in Afghanistan.

To this day, pigeons are entered into competitions, with the winner receiving prize money at the end.

I.2 Navigation

Research has been performed with the intention of discovering how pigeons, after being transported, can find their way back from distant places they have never visited before. Most researchers believe that homing ability is based on a "map and compass" model, with the compass feature allowing birds to orient and the map feature allowing birds to determine their location relative to a goal site (home loft). While the compass mecha-

nism appears to rely on the sun, the map mechanism has been highly debated. Some researchers believe that the map mechanism relies on the ability of birds to detect the Earth's magnetic field. Birds can detect a magnetic field, to help them find their way home. Scientists have found that on top of a pigeon's beak large number of particles of iron are found which remain aligned to north like a man-made compass, thus it acts as compass which helps pigeon in determining its home. A light-mediated mechanism that involves the eyes and is lateralized has been examined somewhat, but developments have implicated the trigeminal nerve in magnetoception. Research by Floriano Papi (Italy, early 1970s) and more recent work, largely by Hans Wallraff, suggest that pigeons also orient themselves using the spatial distribution of atmospheric odors, known as olfactory navigation. Near their home lofts, in areas they have previously visited, pigeons probably are guided by visual landmarks.

Research by Jon Hagstrum of the US Geological Survey suggests that homing pigeons use low-frequency infrasound to navigate. Sound waves as low 0.1 Hz have been observed to disrupt or redirect pigeon navigation. The pigeon ear, being far too small to interpret such a long wave, directs pigeons to fly in a circle when first taking air, in order to mentally map such long infrasound waves.

Various experiments suggest that different breeds of homing pigeons rely on different cues to different extents. Charles Walcott at Cornell University was able to demonstrate that while pigeons from one loft were confused by a magnetic anomaly in the Earth it had no effect on birds from another loft 1.6 km (1 mile) away. Other experiments have shown that altering the perceived time of day with artificial lighting or using air conditioning to eliminate odors in the pigeons' home roost affected the pigeons' ability to return home.

Some research also indicates that homing pigeons navigate by following roads and other man-made features, making 90-degree turns and following habitual routes, much the same way that humans navigate.

GPS tracing studies also indicate that gravitational anomalies also play a role.

I.3 Roles

I.3.1 As postal carriers

When used as carrier pigeons in pigeon post a message is written on thin light paper and rolled into a small tube attached to the bird's leg. Pigeons can only go back to one "mentally marked" point that they have identified as their home, so "pigeon mail" can only work when the sender is actually holding the receiver's pigeons. White homing pigeons are used in release dove ceremonies at weddings, funerals, and some sporting events.

With training, pigeons can carry up to 75 g (2.5 oz) on their backs. The German apothecary Julius Neubronner used carrier pigeons to deliver urgent medication. In 1977 a similar carrier pigeon service was set up for the transport of laboratory specimens between two English hospitals. Every morning a basket with pigeons was taken from Plymouth General Hospital to Devonport Hospital. The birds then delivered unbreak-

able vials back to Plymouth as needed. The 30 carrier pigeons became unnecessary in 1983 because of the closure of one of the hospitals. In the 1980s a similar system existed between two French hospitals located in Granville and Avranche.

I.3.2 During war

A B-type bus from London converted into a pigeon loft for use in northern France and Belgium during the First World War Birds were used extensively during World War I. One homing pigeon, Cher Ami, was awarded the French Croix de guerre for her heroic service in delivering 12 important messages, despite having been very badly injured.

Crewman with homing pigeons carried in bombers as a means of communications in the event of a crash, ditching, or radio failure. During World War II, the Irish Paddy, the American G.I. Joe and the English Mary of Exeter all received the Dickin Medal. They were among 32 pigeons to receive this award, for their gallantry and bravery in saving human lives with their actions. Eighty-two homing pigeons were dropped into the Netherlands with the First Airborne Division Signals as part of Operation Market Garden in World War II. The pigeons' loft was located in London, which would have required them to fly 390 km (240 miles) to deliver their messages. Also in World War II, hundreds of homing pigeons with the Confidential Pigeon Service were airdropped into northwest Europe to serve as intelligence vectors for local resistance agents. Birds played a vital part in the Invasion of Normandy as radios could not be used for fear of vital information being intercepted by the enemy.

I.3.3 In computing

The humorous IP over Avian Carriers (RFC 1149) is an Internet protocol for the transmission of messages via homing pigeon. Originally intended as an April Fools' Day RFC entry, this protocol was implemented and used, once, to transmit a message in Bergen, Norway, on April 28, 2001.

In September 2009, a South African IT company based in Durban pitted an 11-month-old bird armed with a data packed 4 GB memory stick against the ADSL service from the country's biggest internet service provider, Telkom. The pigeon, Winston, took an hour and eight minutes to carry the data 80 km (50 miles). In all, the data transfer took two hours, six minutes, and fifty-seven seconds—the same amount of time it took to transfer 4% of the data over the ADSL.

I.3.4 Smuggling

Homing pigeons have been reported to be used as a smuggling technique, getting objects and narcotics across borders and into prisons. For instance, between 2009 and 2015, many pigeons have been reported to carry cell phones, SIM cards, phone batteries and USB cords into prisons on the Brazilian state of São Paulo.

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

Nowadays, the Internet is omnipresent in our daily life. It's even more true for people with a smartphone.

A lot of apps require an internet connection to send or receive data.

Developpers now require to think about designing API capable of communicating with several devices to centralise data. Some of these API are open to any developper, like the one from Twitter which we will use today.

Today you will learn to do HTTP requests to the Twitters's API on iOS with the goal of creating a client application for Twitter. It will have to display tweets in a **table view**.

Find here the official API Twitter documentation. You will need it to understand how to receive tweets.

Chapter IV

Exercise 00 : Tweet

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

To begin with, you will need a *model* to represent a **Tweet**.

Create a **Tweet struct** with :

- A **let name : String** property which will be the name of the person tweeting.
- A **let text : String** property which will be the content of the tweet.

The **Tweet struct** must implement the **CustomStringConvertible** protocol.

Chapter V

Exercise 01 : APITwitterDelegate

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

For this day you will need a *controller* allowing you to make requests to Twitter. This *controller* must call the right method of your **ViewController** when required. That is the reason why protocols are really important.

Create a **APITwitterDelegate protocol** with 2 methods:

- One to process received **Tweets** which will take [**Tweet**] as argument.
- Another one which will be called in case of error and will take **NSError** as argument.

Chapter VI

Exercise 02 : Requests

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

It is now time to create the *controller* mentionned in the last exercise.

Create an **APIController class** with at least the following:

- A **weak var delegate : APITwitterDelegate?** property which will be our **ViewController** later.
- A **let token : String** property which will be our twitter connexion token.
- A constructor which will take an **APITwitterDelegate?** as **delegate** and a **String** for the **token** argument.
- A method that will take a **String** argument and that will have to execute a request to twitter to get the last 100 english tweets containing **String** and call the corresponding **delegate** methods.

`At this stage you can request to twitter the last 100 english tweets containing "school 42" and display all of them in the debug console.`

Be careful about the URL encoding of your requests!

Chapter VII

Exercise 03 : Table View

	Exercice : 03
Table View	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

Let's now focus on the UI of your app.

Add a **navigation bar** and a **table view**, to your app. For obvious reasons, the **table view** must display the last 100 tweets containing "school 42".

For this to happen the *controller* linked to your **table view** must implement the **APITwitterDelegate protocol**.

Chapter VIII

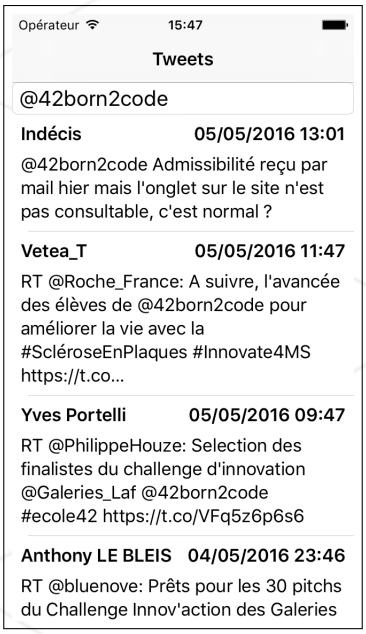
Exercise 04 : Research

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

This mini twitter client lacks a research field don't you think?

Add a **text field** to your **table view** to be able to search for something else than "school 42".

When pressing "enter" the keyboard must disappear and the request being done.



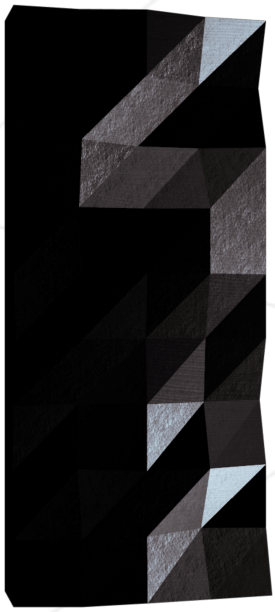
Chapter IX

Exercise 05 : Finishes

	Exercice : 05
Finishes	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

It is now time to make the client usable with the features you already implementer:

- Add **custom cell** for a more beautiful display.
- Dynamic cell sizes depending on the size of their content.
- Add **alerts** in case of errors.
- Display tweet dates.



Piscine iOS Swift - Day 05

Kanto

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

Summary: This document contains the subject for Day 05 from „Piscine iOS Swift” from 42

Contents

Chapter I

Foreword

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

Geolocation is an indispensable weapon of the good iOS developer kit, it is essential to know how to use it.

Apple makes available to you different frameworks like **MapKit** which allows you to use a very simple map or **CoreLocation** which allows you to manage the user's location.

These 2 frameworks will be your best friends for this day.

You will create a geolocation application of several places using:

a TabBarController : to organise your different views.

a MKMapView : for the map.

a CLLocationManager : to geolocate the user.

a SegmentedControlBar : to change the style of the map.

a MKAnnotationView : to customize the pop-up map.

Chapter IV

Exercice 00 : Tab Bar

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

Start by creating the project by choosing *Tabbed Application*.

Customize the icons of the 2 views already in the *MainStoryboard*.

One of the views will be the list of several places in a **table view**. The other will be the map on which the places will be displayed.

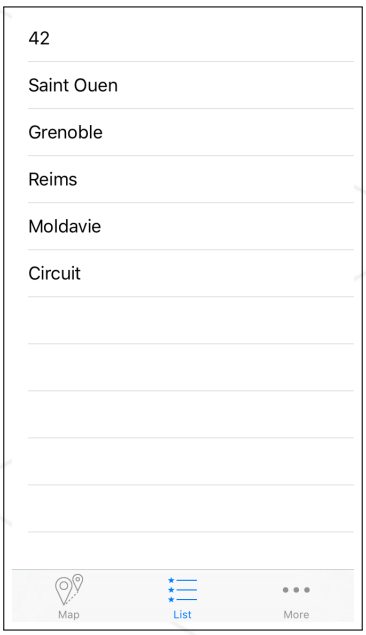
Chapter V

Exercice 01 : Table View

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

You will now add a **table view** to one of these 2 views. It must display at least 3 different places.

`Think about organizing your data to make the development more easy in future exercises. You always have the right to add other files to your project.`



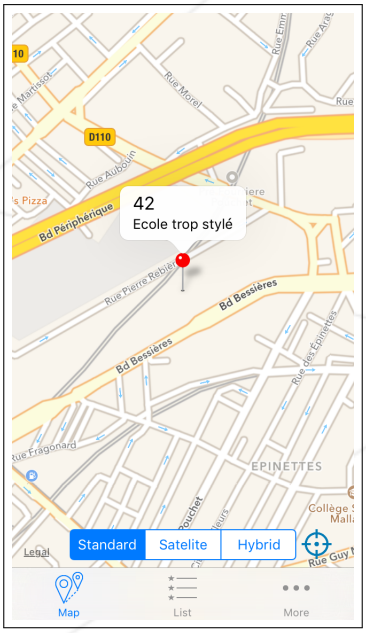
Chapter VI

Exercice 02 : MapKit

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

Let's get to the heart of the matter with this exercise, you will:

- Add a map to the second view.
- Display a *pin* to Ecole 42.
- Add a title and subtitle to this *pin*. This information must appear when you click on the pin.
- When it arrives on the map it must be zoomed on Ecole 42.


42
Ecole trop stylé
Bd Périphérique
Rue Audubon
Rue Morel
D110
Rue Pierre Rebière
Bd Bessières
Rue des Epinettes
EPINETTES
Standard
Satelite
Hybrid
Legal
Map
List
More

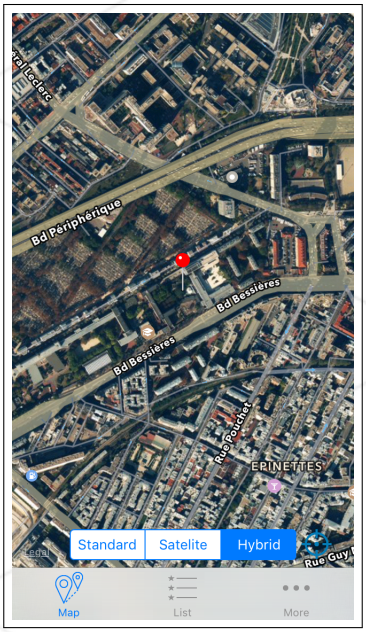
Chapter VII

Exercice 03 : SegmentControlBar

	Exercice : 03
SegmentControlBar	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, MapKit`	
Notes : `n/a`	

Now that you have managed to display the map as you want, you can add a **segmented control bar** to select the mode of the map.

There are 3 ways to display a map : *Hybrid*, *Satellite* or *Standard*. The **segmented control bar** allow to change the mode at any moment.



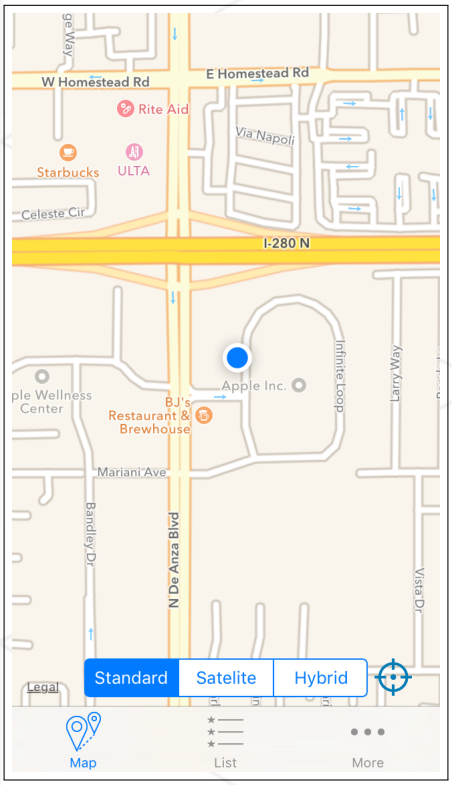
Chapter VIII

Exercice 04 : Geolocalisation

	Exercice : 04
Geolocalisation	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, MapKit, CoreLocation`	
Notes : `n/a`	

Now that your map is functional, it would be nice if you could have a button to get geolocalized.

Add a button that should refocus the map to your position by adjusting the scale of the display to zoom in on you.



Chapter IX

Exercice 05 : Location selection

	Exercice : 05
Location selection	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, MapKit, CoreLocation`	
Notes : `n/a`	

You must now make your list functional by passing variables between your views :

- All the places in your list must be present on your view of the map as *pin* with a title and a subtitle.
- A click on one of the places on your list must return to the view of the map and zoom to the selected place.

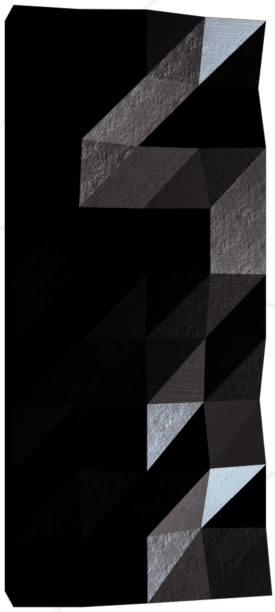
`Under no circumstances should you instantiate a controller and a view again.`

Chapter X

Exercice 06 : Pin's color

	Exercice : 06
Pin's color	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, MapKit, CoreLocation`	
Notes : `n/a`	

You will now customize the *pin* of the map giving them different colors.



Piscine iOS Swift - Day 06

MotionCube

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

Summary: This document contains the subject for Day 06 for the „Piscine iOS Swift" from 42

Contents

Chapter I

Foreword

Here is the world of **Poincaré** from the *Science and Hypothesis* book, chapter 4 :

Suppose, for example, a world enclosed in a large sphere and subject to the following laws:

—The temperature is not uniform; it is greatest at the centre, and gradually decreases as we move towards the circumference of the sphere, where it is absolute zero.

The law of this temperature is as follows: If R be the radius of the sphere, and r the distance of the point considered from the centre, the absolute temperature will be proportional to R^2-r^2.

Further, I shall suppose that in this world all bodies have the same co-efficient of dilatation, so that the linear dilatation of any body is proportional to its absolute temperature..

Finally, I shall assume that a body transported from one point to another of different temperature is instantaneously in thermal equilibrium with its new environment.

There is nothing in these hypotheses either contradictory or unimaginable.

A moving object will become smaller and smaller as it approaches the circumference of the sphere.

Let us observe, in the first place, that although from the point of view of our ordinary geometry this world is finite, to its inhabit ants it will appear infinite.

As they approach the surface of the sphere they become colder, and at the same time smaller and smaller. The steps they take are therefore also smaller and smaller, so that they can never reach the boundary of the sphere.

If to us geometry is only the study of the laws according to which invariable solids move, to these imaginary beings it will be the study of the laws of motion of solids deformed by the differences of temperature alluded to.

No doubt, in our world, natural solids also experience variations of form and volume due to differences of temperature.

But in laying the foundations of geometry we neglect these variations; for besides being but small they are irregular, and consequently appear to us to be accidental.

In our hypothetical world this will no longer be the case, the variations will obey very simple and regular laws. On the other hand, the different solid parts of which the bodies of these inhabitants are composed will undergo the same variations of form and volume.

Let me make another hypothesis: suppose that light passes through media of different refractive indices, such that the index of refraction is inversely proportional to R^2-r^2.

Under these conditions it is clear that the rays of light will no longer be rectilinear but circular.

To justify what has been said, we have to prove that certain changes in the position of external objects may be corrected by correlative movements of the beings which inhabit this imaginary world; and in such a way as to restore the primitive aggregate of the impressions experienced by these sentient beings. Suppose, for example, that an object is displaced and deformed, not like an invariable solid, but like a solid subjected to unequal dilatations in exact conformity with the law of temperature assumed above. To use an abbreviation, we shall call such a movement a non-Euclidean displacement.

If a sentient being be in the neighbourhood of such a displacement of the object, his impressions will be modified; but by moving in a suitable manner, he may reconstruct them. For this purpose, all that is required is that the aggregate of the sentient being and the object, considered as forming a single body, shall experience one of those special displacements which I have just called non-Euclidean. This is possible if we suppose that the limbs of these beings dilate according to the same laws as the other bodies of the world they inhabit.

Although from the point of view of our ordinary geometry there is a deformation of the bodies in this displacement, and although their different parts are no longer in the same relative position, nevertheless we shall see that the impressions of the sentient being remain the same as before; in fact, though the mutual distances of the different parts have varied, yet the parts which at first were in contact are still in contact. It follows that tactile impressions will be unchanged.

On the other hand, from the hypothesis as to refraction and the curvature of the rays of light, visual impressions will also be unchanged. These imaginary beings will therefore be led to classify the phenomena they observe, and to distinguish among them the “changes of position” which may be corrected by a voluntary correlative movement, just as we do.

If they construct a geometry, it will not be like ours, which is the study of the movements of our invariable solids; it will be the study of the changes of position which they will have thus distinguished, and will be “non-Euclidean displacements,” and this will be non-Euclidean geometry. So that beings like ourselves, educated in such a world, will not have the same geometry as ours.

The World of Four Dimensions.—Just as we have pictured to ourselves a non-Euclidean world, so we may picture a world of four dimensions.

The sense of light, even with one eye, together with the muscular sensations relative to the movements of the eyeball, will suffice to enable us to conceive of space of three dimensions.

The images of external objects are painted on the retina, which is a plane of two dimensions; these are perspectives.

But as eye and objects are movable, we see in succession different perspectives of the same body taken from different points of view. We find at the same time that the transition from one perspective to another is often accompanied by muscular sensations.

If the transition from the perspective A to the perspective B, and that of the perspective Aʹto the perspective Báre accompanied by the same muscular sensations, we connect them as we do other operations of the same nature.

Then when we study the laws according to which these operations are combined, we see that they form a group, which has the same structure as that of the movements of invariable solids.

Now, we have seen that it is from the properties of this group that we derive the idea of geometrical space and that of three dimensions. We thus understand how these

perspectives gave rise to the conception of three dimensions, although each perspective is of only two dimensions,—because they succeed each other according to certain laws. Well, in the same way that we draw the perspective of a three-dimensional figure on a plane, so we can draw that of a four-dimensional figure on a canvas of three (or two) dimensions. To a geometer this is but child's play.

We can even draw several perspectives of the same figure from several different points of view. We can easily represent to ourselves these perspectives, since they are of only three dimensions.

Imagine that the different perspectives of one and the same object to occur in succession, and that the transition from one to the other is accompanied by muscular sensations.

It is understood that we shall consider two of these transitions as two operations of the same nature when they are associated with the same muscular sensations.

There is nothing, then, to prevent us from imagining that these operations are combined according to any law we choose—for instance, by forming a group with the same structure as that of the movements of an invariable four-dimensional solid.

In this there is nothing that we cannot represent to ourselves, and, moreover, these sensations are those which a being would experience who has a retina of two dimensions, and who may be displaced in space of four dimensions. In this sense we may say that we can represent to ourselves the fourth dimension.

Conclusions.—It is seen that experiment plays a considerable rôle in the genesis of geometry; but it would be a mistake to conclude from that that geometry is, even in part, an experimental science.

If it were experimental, it would only be approximative and provisory.

And what a rough approximation it would be! Geometry would be only the study of the movements of solid bodies; but, in reality, it is not concerned with natural solids: its object is certain ideal solids, absolutely invariable, which are but a greatly simplified and very remote image of them.

The concept of these ideal bodies is entirely mental, and experiment is but the opportunity which enables us to reach the idea. The object of geometry is the study of a particular "group"; but the general concept of group pre-exists in our minds, at least potentially.

It is imposed on us not as a form of our sensitiveness, but as a form of our understanding; only, from among all possible groups, we must choose one that will be the standard, so to speak, to which we shall refer natural phenomena.

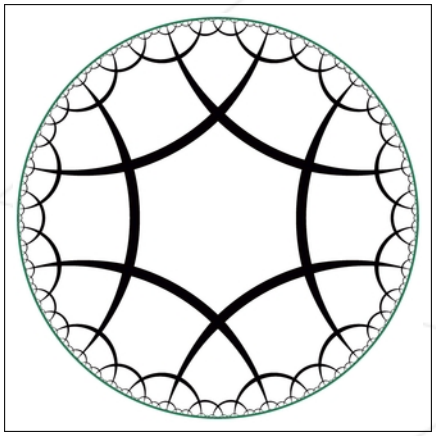
Experiment guides us in this choice, which it does not impose on us.

It tells us not what is the truest, but what is the most convenient geometry.

It will be noticed that my description of these fantastic worlds has required no language other than that of ordinary geometry.

Then, were we transported to those worlds, there would be no need to change that language.

Beings educated there would no doubt find it more convenient to create a geometry different from ours, and better adapted to their impressions; but as for us, in the presence of the same impressions, it is certain that we should not find it more convenient to make a change.



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

Have you ever heard “Don’t put your fingers on the screen!”? Of course today this sentence will have no sense. The disappearence of keyboard in favour of touch screens allows application to expend their interface to not be limited by keyboard keys anymore. As a matter of fact, when you used to play snake on a nokia 3310, we just need 4 keys to direct the snake, everything else is useless.

Apple also equiped their devices with different sensors such as an accelerometer, a gyroscope, proximity sensor or a barometer.

Therefore it is possible to retrieve user input or environment data thanks to the screen and the device’s sensors.

Today you will learn hos to use the **UIGestureRecognizer**, to retrieve user’s acctions on the screem as well as **CoreMotion** to get the orientation of the device in an application that will allow you to play with small squares and small circles. This shapes will be subject to the laws of physics like gravity, elasticity and collision using a**UIDynamicAnimator**.

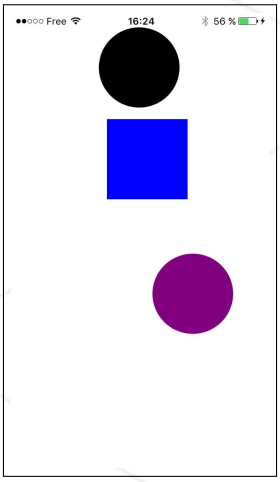
Chapter IV

Exercise 00 : TapGesture

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

The aim of this exercise is to be able to add a shape in a empty view by touching the screen. For this, you have to create a class that inherits the **UIView** class which will represent a shape.

- Its shape must be either a square either a circle.
- Its size must be 100 x 100.
- Its shape and color must be random.
- When touching the screen, the shape must appear under our finger.



Chapter V

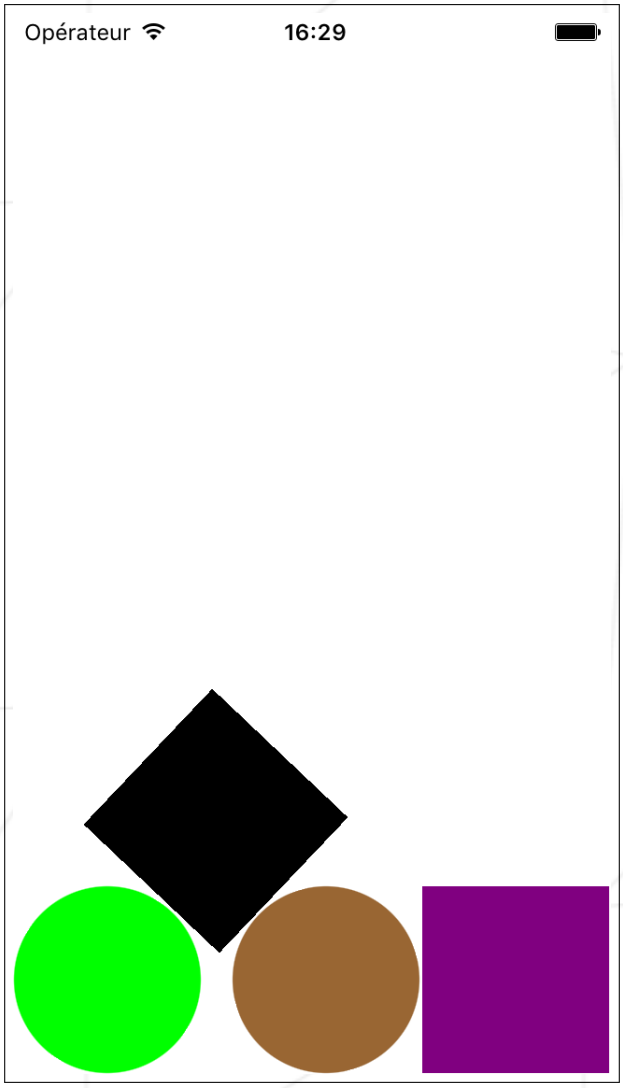
Exercise 02 : Dynamic Behavior

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

Add a little bit of physics to your shapes. They must be subject to several **UIDynamicBehavior** :

- Fall under the effect of gravity.
- Collision cannot make them go out of their “superview”.
- Have an elasticity effect, they must rebound.

`Shapes must rebound on the border of the screen, they cannot disappear.`


Opérateur
16:29

Chapter VI

Exercise 04 : Gestures

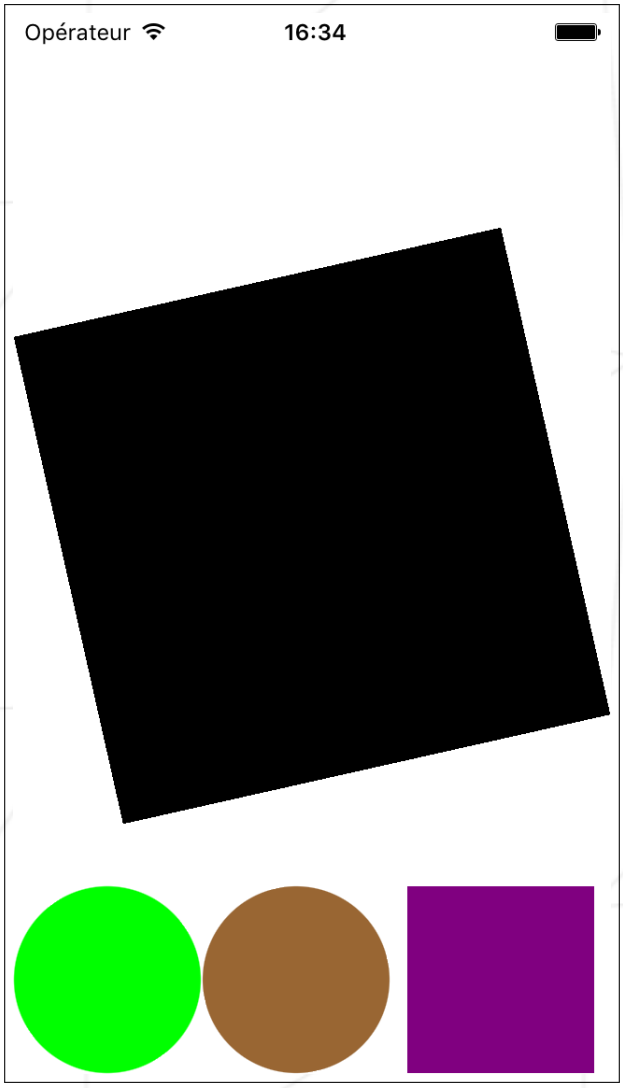
	Exercice : 04
Gestures	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

Interaction with those shapes would be fun!

Now add some **Gesture** to your shapes:

- **UIPanGestureRecognizer** to move them.
- **UIPinchGestureRecognizer** to enlarge or reduce them.
- **UIRotationGesture** to rotate them.

`When you interact with a shape, you must remove its gravity effect but not its collision or elasticity effect. When you release the shape, it must be subject to gravity again.`


Opérateur
16:34

Chapter VII

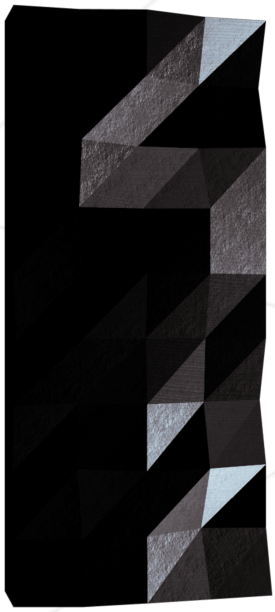
Exercise 06 : CoreMotion

	Exercice : 06
CoreMotion	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, CoreMotion`	
Notes : `n/a`	

In the project's settings, only allow portrait mode.

You now have to modify the gravity vector depending on the accelerometer data using the **CoreMotion**. Shapes must fall in the different directions depending on the orientation of the device.

The iOS simulator doens't allow you yet to simulate sensor's data therefore it is advised to test this app on a real Apple device. If you don't have one, don't panic, as to your peers to lend one to you. If you really couldn't find any Apple device, don't worry too much, this exercise is quite simple and p2p will not take it into consideration.



Piscine iOS Swift - Day 07

Siri

PE LIEB plieb@student.42.fr
Maxime LEMORT mlemort@student.42.fr
42 Staff pedago@42.fr

Summary: This document contains the subject for Day 07 for the „Piscine iOS Swift” from 42

Contents

Chapter I

Foreword

Voici la page wikipedia sur l'Intelligence Artificielle :

Artificial intelligence (AI, also machine intelligence, MI) is apparently intelligent behaviour by machines, rather than the natural intelligence (NI) of humans and other animals. In computer science AI research is defined as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of success at some goal. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning and problem solving".

The scope of AI is disputed: as machines become increasingly capable, tasks considered as requiring "intelligence" are often removed from the definition, a phenomenon known as the AI effect, leading to the quip "AI is whatever hasn't been done yet." For instance, optical character recognition is frequently excluded from "artificial intelligence", having become a routine technology. Capabilities generally classified as AI as of 2017 include successfully understanding human speech, competing at a high level in strategic game systems (such as chess and Go), autonomous cars, intelligent routing in content delivery networks, military simulations, and interpreting complex data.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. For most of its history, AI research has been divided into subfields that often fail to communicate with each other. However, in the early 21st century statistical approaches to machine learning became successful enough to eclipse all other tools, approaches, problems and schools of thought.

The traditional problems (or goals) of AI research include reasoning, knowledge, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence is among the field's long-term goals. Approaches include statistical methods, computational intelligence, and traditional symbolic AI. Many tools are used in AI, including versions of search and mathematical optimization, neural networks and methods based on statistics, probability and economics. The AI field draws upon computer science, mathematics, psychology, linguistics, philosophy, neuroscience, artificial psychology and many others.

The field was founded on the claim that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the nature of the mind and the ethics of creating artificial beings endowed with human-like intelligence, issues which have been explored by myth, fiction and philosophy since antiquity. Some people also consider AI a danger to humanity if it progresses unabatedly.

In the twenty-first century, AI techniques have experienced a resurgence following concurrent advances in computer power, large amounts of data, and theoretical understanding; and AI techniques have become an essential part of the technology industry, helping to solve many challenging problems in computer science.

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

You probably know that already but developpers are righty often and for most pointed as lazy: we will not reinvent the wheel every day if it already rolls perfectly. Today we will talk about pods thanks to Cocoapods !

But what is a pod exactly? A pos is not neither Planet of Death nor The Breeders. A pod is a **package** and is therefore managed by a **package manager**, **Cocoapods** in this case.

During the day we will learn how to use pods. Meaning how to install and use them. Today's objective will be to create a robot more commonly called **bot** which will give the weather forecast for a provided city.

We will use 2 distinct APIs coming from 2 services on which you will have to create an account.

- Recast.AI: An API bringing a une API **Artificial Intelligence** brick for you bot.
- Dark Sky: An open API that will allow you to get the forecast weather for a give latitude and longitude. (Formerly Forecast.IO)

Find below documentation you will need to finish the day:

- Cocoapods documentation
- Recast.AI Pod
- Dark Sky Pod
- JSQMessagesViewController Pod which will allow you to display your conversation in a messaging app way.

Chapter IV

Exercice 00 : Cocoapods Installation

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

To begin with you need to install **Cocoapods**.

Go take a look at the Cocoapods website and follow the procedure. Make sure that Cocoapods is properly installed by typing **'pod'** in the console:

```
>pod
Usage:

    $ pod COMMAND

      CocoaPods, the Cocoa library package manager.

Commands:

    + cache      Manipulate the CocoaPods cache
    + init       Generate a Podfile for the current directory.
    + install    Install project dependencies to Podfile.lock versions
    + ipc        Inter-process communication
    + lib        Develop pods
    + list       List pods
    + outdated   Show outdated project dependencies
    + plugins    Show available CocoaPods plugins
    + repo       Manage spec-repositories
    + search     Search for pods.
    + setup      Setup the CocoaPods environment
    + spec       Manage pod specs
    + trunk      Interact with the CocoaPods API (e.g. publishing new specs)
    + try        Try a Pod!
    + update     Update outdated project dependencies and create new Podfile.lock

Options:

    --silent     Show nothing
    --version    Show the version of the tool
    --verbose    Show more debugging information
    --no-ansi    Show output without ANSI codes
    --help       Show help banner of specified command
```

Chapter V

Exercice 01 : FirstViewController

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

For this day you will need a controller allowing you to request `Recast` & `Dark Sky`.

Create a controller containing:

- A **Bouton** to request Recast
- A **TextField** to write the text you will send
- A **Label** to display the answer

`Think about the Autolayout!`

Chapter VI

Exercice 02 : “RecastAI” pod

	Exercice : 02
“RecastAI” pod	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, Cocoapods`	
Notes : `n/a`	

Its now time to create an account on Recast.AI. Install the ’`RecastAI`’ pod to be able

to request and use the community Slackbot bot with the weather intention and the token. Make a request via the button by passing the **TextField** as a parameter.

The **label** must display the **Recast** return intention or “Error” if no intention is returned.

Chapter VII

Exercice 03 : “Dark Sky” pod

	Exercice : 03
“Dark Sky” pod	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, Cocoapods`	
Notes : `n/a`	

Now you will have to display the weather forecast depending on the geolocalisation returned. You will need to create an account on Dark Sky and obtain a token for this. Intall the **’Dark Sky’** pod then make a request to Dark Sky once the **Recast** request is done. The label must display the weather forecast returned by the Dark Sky API depending on the geolocalisation returned by the **Recast** API..

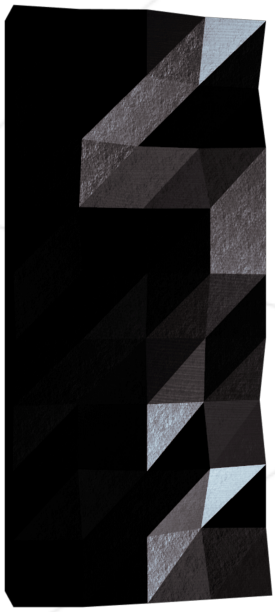
Chapter VIII

Exercice 04 : “JSQMessagesViewController” pod

	Exercice : 04
“JSQMessagesViewController” pod	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, Cocoapods`	
Notes : `n/a`	

Since everything is done you can now polish your interface. Let’s use the **’JSQMessagesViewController’** to display the requests in a the form of a conversation with the bot.

You will have to add a button to be able to make voice requests.



Piscine iOS Swift - Day 08

CoreData

Maxime LEMORT mlemort@student.42.fr
PE LIEB plieb@student.42.fr
42 Staff pedago@42.fr

Summary: This document contains the subject for Day 08 for the „Piscine iOS Swift” from 42

Contents

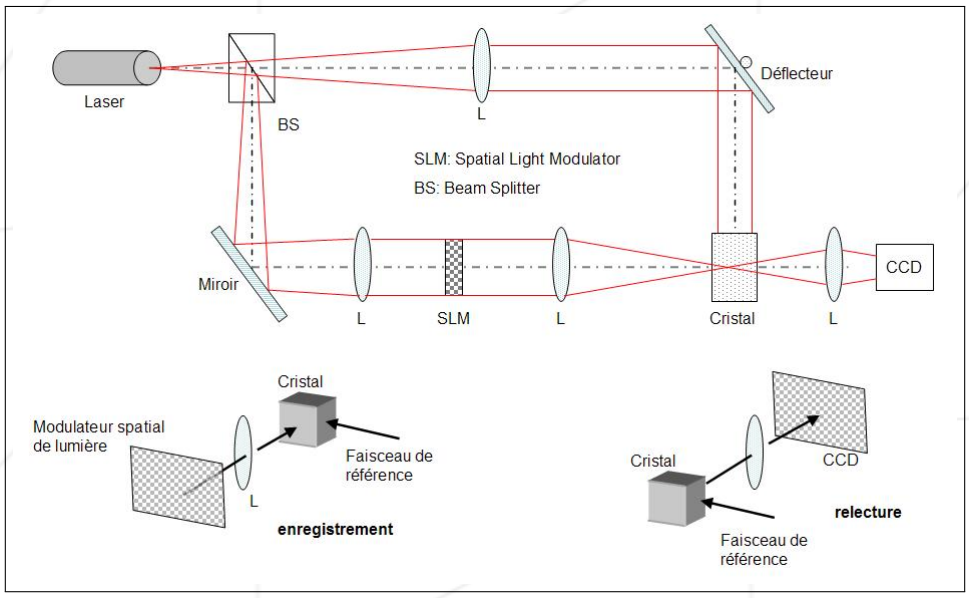
Chapter I

Foreword

Here is what the wikipedia page says about Holographic data storage:

Holographic data storage is a potential technology in the area of high-capacity data storage currently dominated by magnetic data storage and conventional optical data storage. Magnetic and optical data storage devices rely on individual bits being stored as distinct magnetic or optical changes on the surface of the recording medium. Holographic data storage records information throughout the volume of the medium and is capable of recording multiple images in the same area utilizing light at different angles.

Additionally, whereas magnetic and optical data storage records information a bit at a time in a linear fashion, holographic storage is capable of recording and reading millions of bits in parallel, enabling data transfer rates greater than those attained by traditional optical storage.



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

Now that you discovers about pods, let's learn how to create one. Still using Cocoapods !

If you already forgot what a pod is, consider is a package that will be generated by a **package manager Cocoapods** here.

Today's objective will be to create a package that will use the **CoreData** framework to learn how to use data persistance and models. The aim being to create an article manager that will serve as interface for the D09.

An important point: We are talking about integrating **CoreData** inside a pod and that's the challengin part. To continue with the next day you will have to finish this one so take your time to properly finish the Day 08!

Here are the documentation you will require to finish today's project:

- Doc Cocoapods
- CoreData

Chapter IV

Exercice 00 : Pod creation

	Exercice : 00
Pod creation	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Aucune`	
Notes : `n/a`	

You have to create a **Cocoapods** pod to begin with.

Go take a look on the following Cocoapods website and follow the procedure. You will find a lot of tutorials on the Internet. Your pod must be coded in **Swift**, contain an **example**, you don't need to implement **tests**. The name of your pod will be your **username followed by the current year** Faites un tour sur le site de Cocoapods et suivez la procédure. Vous trouverez aussi plein de tutoriels sur internet. Votre pod doit etre en **Swift**, contenir un **example**, vous n'avez pas besoin de **tests**. Le nom de votre pod sera votre **login suivi de l'année courante** (ex: **mlemort2016**).

Chapter V

Exercice 01 : Podspec

	Exercice : 01
Podspec	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Aucune`	
Notes : `n/a`	

Now that your pod is created you have to take care about its **podspec** file.

Your **podspec** file must contain:

- A **Description**
- A **Summary**
- A CoreData **Framework**

The only thing that you don't have to do is the github project url.

```
pod lib lint YOUR_POD !
```

Chapter VI

Exercice 02 : xcdatamodeld

	Exercice : 02
xcdatamodeld	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Aucune`	
Notes : `n/a`	

The time has come to create a data model for **CoreData** in your pod. Add a `article.xcdatamodeld` file and add:

- A **Title**
- A **Content**
- A **Language**
- An **Image**
- A **Creation Date**
- A **Modification Date**

Chapter VII

Exercice 03 : Class Article

	Exercice : 03
Class Article	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, CoreData`	
Notes : `n/a`	

Create now your **Article** class that extend **NSManagedObject**.
You class must contain the following attributes:

- A **Title** of type **String?**
- A **Content** of type **String?**
- A **Language** of type **String?**
- An **Image** of type **NSData?**
- A **Creation Date** of type **NSDate?**
- A **Modification Date** of type **NSDate?**
- An **Override** of **Description**

Chapter VIII

Exercice 04 : Class ArticleManager

	Exercice : 04
Class ArticleManager	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, CoreData`	
Notes : `n/a`	

Now that all of this is done you will now be able to create the **ArticleManager** class. This class must contain the following methods:

- **newArticle** that allow us to create a new article and returns it.
- **getAllArticles** that returns every stored articles.
- **getArticles(withLang lang: String)** that returns every stored articles in a specific language.
- **getArticles(containString str: String)** that returns every articles containing the following string give as a parameter.
- **removeArticle(article : Article)** that removes an article.
- **save** that saves every modification.

When creating your **NSManagedObjectContext** don't forget to use NSBundle(forClass: AnyClass) to change the right **Bundle**!

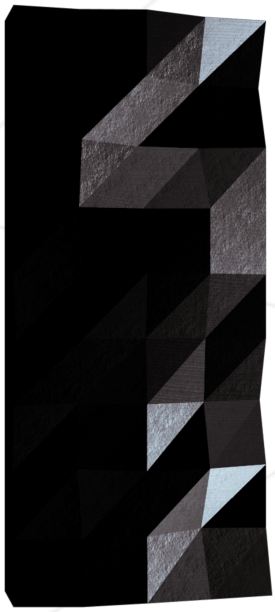
Chapter IX

Exercice 05 : ViewController

	Exercice : 05
ViewController	
Files to turn in : `.xcodeproj and all the necessary files`	
Allowed functions : `Swift Standard Library, UIKit, CoreData`	
Notes : `n/a`	

To conclude create several articles and the **ViewDidLoad** of your **ViewController** and display them in the debug at the launch of your app.

`When you launch the app several times old articles must persist.`



Piscine iOS Swift - Day 09

Personnal Diary

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

Summary: This document contains the subject for Day 09 for the „Piscine iOS Swift" from 42

Contents

Chapter I

Foreword

Here you have the summary of Victor Hugo's novel, The Last Day of a Condemned Man :

The novel presents itself as the journal of a man condemned to death, written during the last twenty-four hours of his existence, in which he recounts his experiences from the beginning of his trial to the time of his execution, about six weeks of his life. This story, a long inner monologue, is interspersed with anxious reflexions and memories of his other life, the "life before". The reader knows neither the name of this man nor for what he was condemned, apart from the sentence: " I, wretch, who has committed a true crime, who has shed blood ! ". The work is presented as a raw testimony, at the same time on the anguish of the condemned to death and his last thoughts, on the daily moral and physical sufferings that he undergoes and on the living conditions of the prisoners, for example in the chaining of convicts scene. He expresses his feelings regarding his previous life and his states of mind....

He will be executed under the clamor of people who see his death as a spectacle.

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

Dear Diary,

Today I will make a personal diary application ! And who says personal means secure. No one will be able to consult it except for me since it will only open with my digital fingerprint or my password thanks to the framework **LocalAuthentication** ! I will be able to write **Articles** with a title, a content, a date and even add photos taken with the camera ! And since I am nice, I will make it available in several languages.

It will be too good ! With this new application I could get rid of you and throw you on the chemney fire.

Chapter IV

Exercise 00 : The first brick

	Exercice : 00
The first brick	
Files to turn in : `.xcodeproj, the pod of d08 and all necessary files`	
Allowed functions : `Swift Standard Library, UIKit`	
Notes : `n/a`	

I'll just start by reusing the pod I did yesterday.

It is essential for my application since this pod allows me to store the **Articles** on my device and it even allows me to retrieve them !

It is quite possible to import a pod from a folder in my **Podfile**, so I will not deprive myself of it.

Chapter V

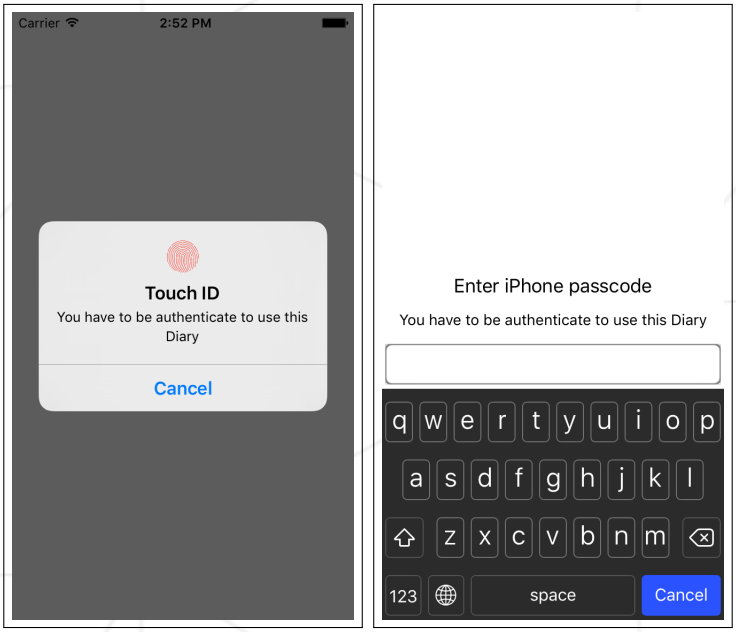
Exercise 01 : Paper Please

	Exercice : 01
Paper Please	
Files to turn in : `.xcodeproj, the pod of d08 and all necessary files`	
Allowed functions : `Swift Standard Library, UIKit, LocalAuthentication`	
Notes : `n/a`	

Now I will just make a first authentication view.

The application must ask me to authenticate myself with the **TouchID** if available or with the password if not. As long as authentication fails, it should start over.

Once authenticated, the view from the next exercise should appear.



Chapter VI

Exercise 02 : Turn the page

	Exercice : 02
Turn the page	
Files to turn in : `.xcodeproj, the pod of d08 and all necessary files`	
Allowed functions : `Swift Standard Library, UIKit, LocalAuthentication`	
Notes : `n/a`	

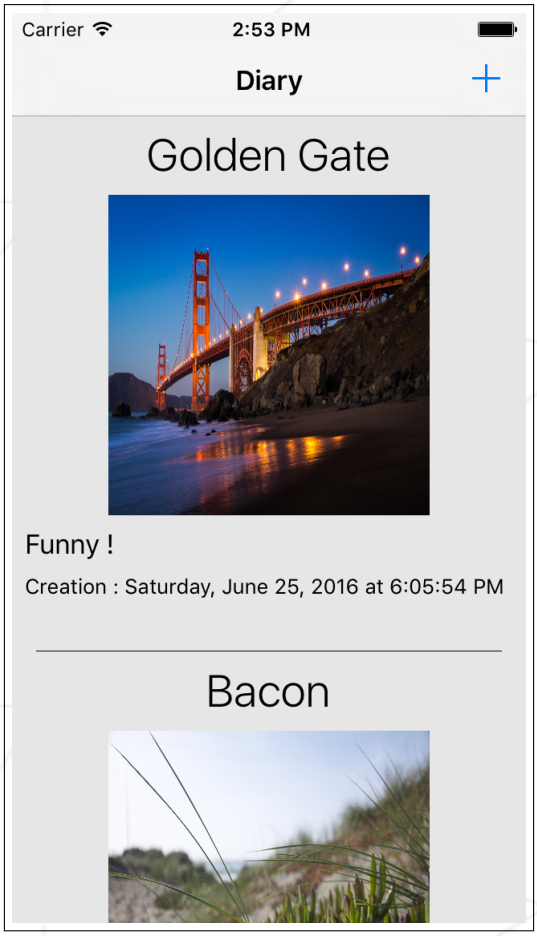
Now that I made sure I was the one using the application, I can display all the **Articles** present on the device.

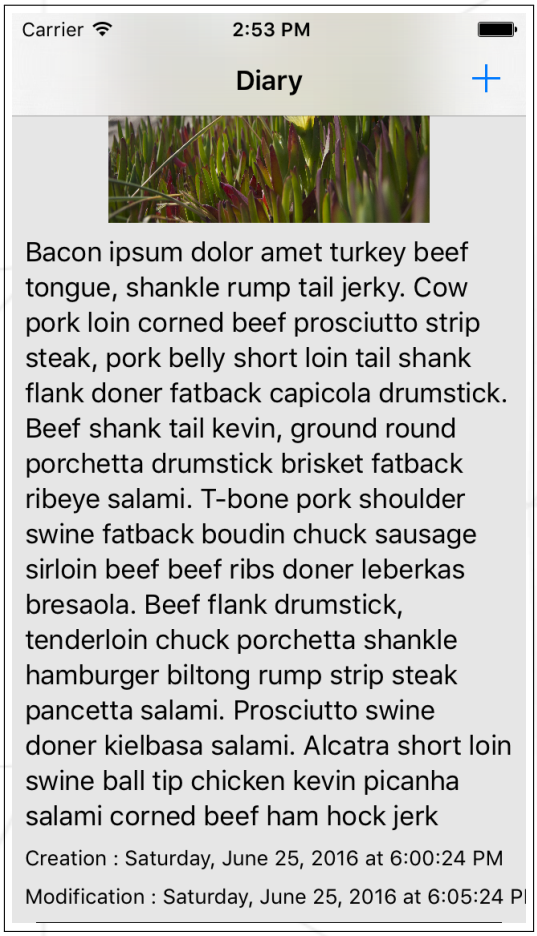
I already know the music : **TableView**, **CustomCells**, **Automatic Cell Height**, **blablabla**...

I just have to be careful not to forget anything :

- Title
- Photo
- Content
- Creation date
- Modification date if it is different from the creation date

`I have to display only the` **`Articles`** `saved in the current language of the device.`


Carrier
2:53 PM
Diary
Golden Gate
Funny !
Creation : Saturday, June 25, 2016 at 6:05:54 PM
Bacon


Carrier
2:53 PM
Diary
Bacon ipsum dolor amet turkey beef tongue, shankle rump tail jerky. Cow pork loin corned beef prosciutto strip steak, pork belly short loin tail shank flank doner fatback capicola drumstick. Beef shank tail kevin, ground round porchetta drumstick brisket fatback ribeye salami. T-bone pork shoulder swine fatback boudin chuck sausage sirloin beef beef ribs doner leberkas bresaola. Beef flank drumstick, tenderloin chuck porchetta shankle hamburger biltong rump strip steak pancetta salami. Prosciutto swine doner kielbasa salami. Alcatra short loin swine ball tip chicken kevin picanha salami corned beef ham hock jerk
Creation : Saturday, June 25, 2016 at 6:00:24 PM
Modification : Saturday, June 25, 2016 at 6:05:24 PM

Chapter VII

Exercise 03 : RTVA

	Exercice : 03
RTVA	
Files to turn in : `.xcodeproj, the pod of d08 and all necessary files`	
Allowed functions : `Swift Standard Library, UIKit, LocalAuthentication`	
Notes : `n/a`	

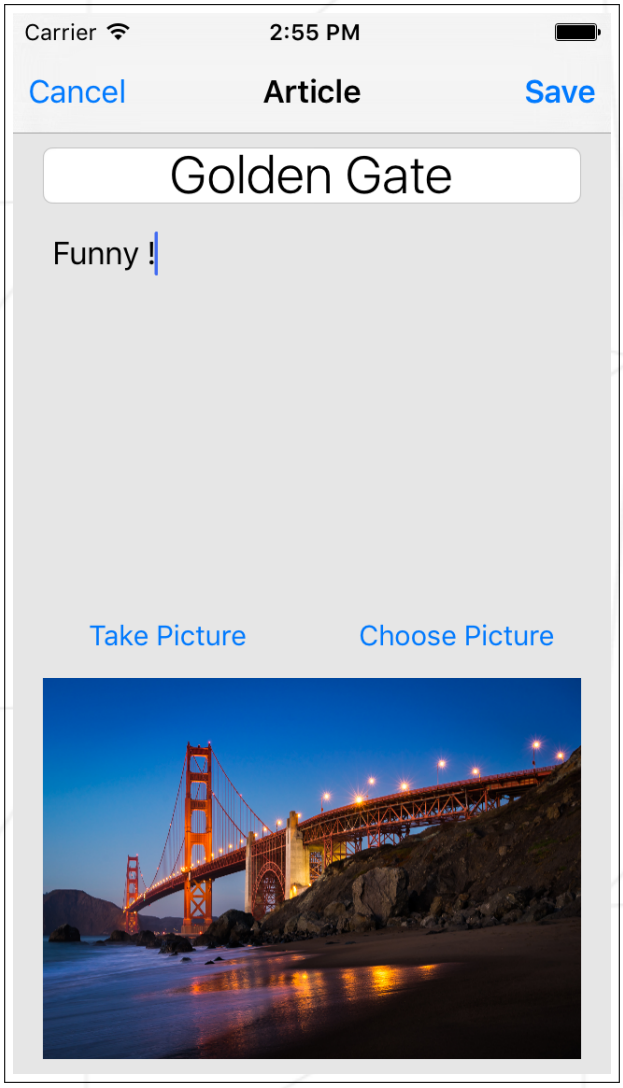
It's good to be able to view the **Articles**, but I have to find a way to add new ones.

I know ! I will add a **NavigationBar** with a title and a **BarButton** that will send me to a view to add an **Article**.

In this new view, all the elements needed to add a new **article** will be present, plus a button for saving and returning to the previous view.

For the Photo, I should have the option to either choose one from the photos on the phone or to take one with the camera.

`Well, I obviously have to check that the articles are saved on the device. When I exit the application and restart it, all the articles should be there.`


Carrier
2:55 PM
Cancel
Article
Save
Golden Gate
Funny !
Take Picture
Choose Picture

Chapter VIII

Exercise 04 : Oups

	Exercice : 04
Oups	
Files to turn in : `.xcodeproj, the pod of d08 and all necessary files`	
Allowed functions : `Swift Standard Library, UIKit, LocalAuthentication`	
Notes : `n/a`	

During testing I made a mistake in one **article** I saved !

I'll have to implement the ability to edit an **article** in the application.

I found something really good ! When I click on an **article** in the **TableView** the same view from the previous exercise is loaded with all the fields to be filled. I just have to correct my errors and save. I didn't even have to add another view with code.

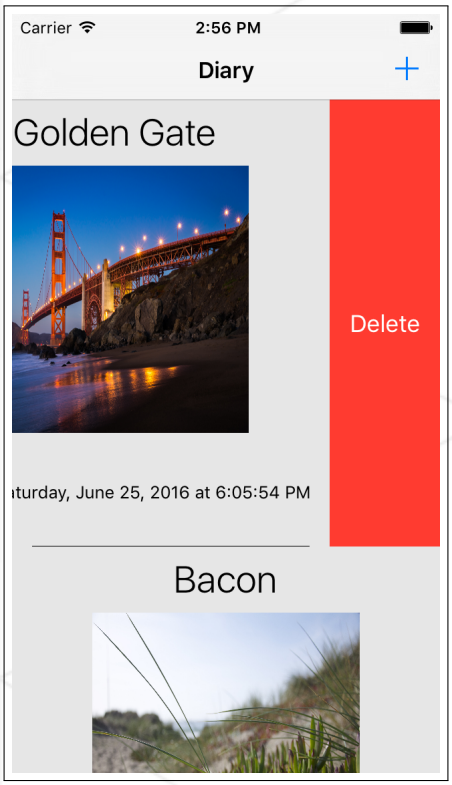
Chapter IX

Exercise 05 : Tear the page

	Exercice : 05
Tear the page	
Files to turn in : `.xcodeproj, the pod of d08 and all necessary files`	
Allowed functions : `Swift Standard Library, UIKit, LocalAuthentication`	
Notes : `n/a`	

No, finally this **article** does not please me.

In some applications, I saw that one could **Swipe** the Cells to make a menu appear. I'll use that to delete an **article** !



Chapter X

Exercise 06 : Translation

	Exercice : 06
Translation	
Files to turn in : `.xcodeproj, the pod of d08 and all necessary files`	
Allowed functions : `Swift Standard Library, UIKit, LocalAuthentication`	
Notes : `n/a`	

Now that I made the application in English, I just translated it into French for myself.

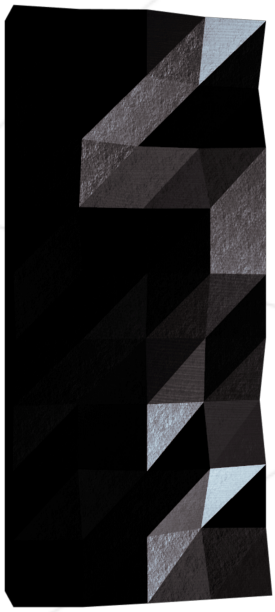
`I have to be careful to change the language of the simulator in order to check that all the Strings are translated, except for the use entries.`

Chapter XI

Exercise 07 : Ninja

	Exercice : 07
Ninja	
Files to turn in : `.xcodeproj, the pod of d08 and all necessary files`	
Allowed functions : `Swift Standard Library, UIKit, LocalAuthentication`	
Notes : `n/a`	

If I haven't closed my application and it keeps running in the background, some one else could open it and have access ! I have to find a way for the application to not run in the background.



Piscine iOS Swift - Rush 00

Forum

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

Summary: This document contains the subject for Rush00 of the „Piscine iOS Swift" from 42

Contents

Chapter I

Foreword

Here is what wikipedia can tell us about the site 4chan :

4chan, whose name comes from the Japanese Yotsuba Channel, is an anonymous English speaking forum, consisting of a network of image boards, modeled after the popular Japanese sites 2channel (created in 1999 by Hiroyuki Nishimura) and Futaba Channel. It has been online since October 1, 2003.

When it was created, one of the main objectives of 4chan (Yotsuba Channel) was to be a medium for discussions about anime, manga, doujinshi and the whole Japanese culture. However, the topics addressed by the users currently cover a much broader range, mainly due to the diversity of topics covered by the image boards present on the site. The French daily newspaper Le Monde describes it as "the troublemaker of the Web", "particularly controversial" for its functioning without information moderation, while Libération sees it as a site "that rubs shoulders with the worst (racism, homophobia, sexism) and the least worst (Internet meme, Anonymous)" – the latter being born on the site.

In 2012, it surpasses the Futaba Channel in terms of traffic. In the 2014 Alexa Internet ranking, 4chan is 995th worldwide and 492nd in the US.

In September 2015, after the departure of Christopher Poole in January, 4chan is taken over by the creator of 2channel, the Japanese Hiroyuki Nishimura.

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

This weekend you have to work in groups to do this rush. Try to work together and share your knowledge to move forward. You can, of course, split the tasks in order to go faster.

You will make a client application for the 42 forum. 42 provides you with an API that allows you (among other things) access to the school forum.

Your application should function globally like all other forums, it should allow consulting messages by topic, creating, deleting and editing messages etc...

Here is the documentation of the API.

You have 2 days to do this rush, so make an application that is functional, ergonomic and pretty !

Chapter IV

Part 1 : Connection

For part 1, your application must contain these features :

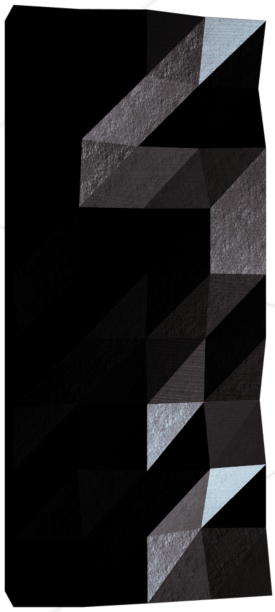
- Connecting to the **42 API** with your intranet **username** and **password**.
- Being able to disconnect.
- Viewing the **topics** of the forum with the **dates** and **authors**.
- Viewing the **messages** and **reponses** with the **dates** and **authors**.

Chapter V

Part 2 : Modification

For part 2, your application must contain these features :

- Being able to **add** messages.
- Being able to **edit** messages.
- Being able to **delete** messages.
- Being able to **add** topics.
- Being able to **edit** topics.
- Being able to **delete** topics.



Piscine iOS Swift - Rush 01

Plan 42

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

Summary: This document contains the subject for Rush01 of the „Piscine iOS Swift" from 42

Contents

Chapter I

Intro

Les voyages en train - *Grand Corps Malade*
I believe that love stories are like train travel,
And when I see all these travelers sometimes I'd like to be one of them,
Why do you think so many people are waiting on the station platform,
Why do you think we're so afraid to arrive late.

Trains often start when least expected,
And the love story takes you under the helpless eyes of witnesses,
The witnesses are your friends who say goodbye on the platform,
They watch the train go away with a worried smile,
You too wave at them and imagine their comments,
Some think that you are wrong and that you don't have your feet on the ground,
Each one of them has his prognosis on the length of the trip,
For most, the train will derail from the first storm.

Great love necessarily changes your behavior,
From the first day, you have to choose your compartment,
Aisle or window seat, you have to find the right place,
You choose if it's a first or second class love story.

For the first kilometers, you only have eyes for her face,

You don't take into account the parade of landscapes outside the window,
You feel alive, you feel light, you do not see the time pass,
You feel so good that you almost want to hug the conductor.

But the magic only lasts for a while and your story is on the rocks,
You tell yourself that you have nothing to do with it and that it is her own fault,
The roar of the train makes you drunk and every turn sickens you,
You must get up and walk to strech your heart.

And the train slows down and it is already the end of your story,
Besides, you were a jerk, your friends remained at the other station,
You say goodbye to the one whom you will now call your ex,
In her diary, she'll put a little white-out on your name.

It's true that love stories are like train travel,
And when I see all these travelers sometimes I'd like to be one of them,
Why do you think so many people are waiting on the station platform,
Why do you think we're so afraid to arrive late.

For many, life comes down to trying to get on the train,
To know what love is and to discover each other full of enthusiasm,
For many the goal is to arrive at the right time,
To have a successful trip and access to happiness.

It's easy to catch a train, still one needs to take the good one,
I boarded two or three trains but it wasn't the right wagon,
Because the trains are whimsical and some are inaccessible,
And I don't think it's always possible with the French national railway company.

There are those for whom the trains are always on strike,
And their love stories only exist in their dreams,
And then there are those who rush into the first train without paying attention,
But inevitably they will get off disappointed at the next station,
There are those who are terrified of commitment because they are too emotional,
For them, it's too risky to cling to the locomotive,
And then there are the adventurers who take trip after trip,
Once a story is over they attack another page.

After my only real trip, I suffered for months,
We parted by mutual agreement, but she was more in agreement than I,
Since I hang out at the platform, I look at the trains departing,
There are doors that open, but in a train station I feel left out.

It seems that train travel ends badly in general,
If that's the case for you hang on and be strong,
For one thing is certain, there will always be a terminus,
Now you are warned, the next time you will take the bus.

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

This weekend you have to work in groups to do this rush. Try to work together and share your knowledge to move forward. You can, of course, split the tasks in order to go faster.

You will make a GPS application.

Your application should function globally like any other GPS device: geolocation, address search, route to the desired location.

You can use any pods you want.

You have 2 days to do this rush, so make an application that is functional, ergonomic and pretty !

Chapter IV

Instructions

Your application must contain these features :

- Geolocating the mobile phone
- Finding an address and displaying its location on a map
- Displaying the route on the map if desired. With the choice of departure and destination.
- Your application must be fun to use. The user must know what is happening at all times.