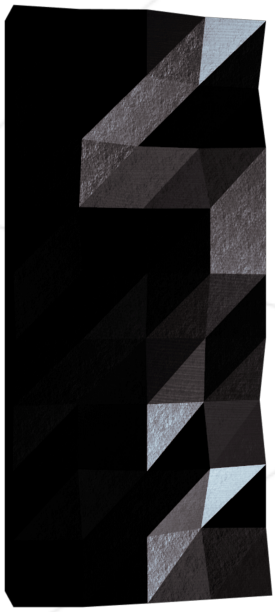

Piscine iOS Swift - Day 01

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**.