



Piscine iOS Swift - Day 01

Card Game

Résumé: This document contain the subject for Day 01 for „Piscine iOS Swift” from [42](#)

Table des matières

I	Préambule	2
II	Instructions	3
III	Specific rules of the day	4
IV	Introduction	5
V	Exercise 00 : Color and Value	6
VI	exercice 01 : Card	7
VII	Exercise 02 : Deck	9
VIII	Exercise 03 : Extension	10
IX	Exercise 04 : Board	11

Chapitre I

Préambule

„Actually, https://fr.wikipedia.org/wiki/Sport_electroniquee-sport, is considered a game of chance. This title depends of „Authority of online games regulations” (fr. AR-JEL) 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 od 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](#)

Chapitre II

Instructions

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



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

Chapitre III

Specific rules 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 for the exercise, at the root of your directory :
ex00 ex01 ex02...
- You will compile the exercises with **swiftc**
- You have to make your own test sets to prove that everything works correctly at defense. What is not tested it will not be considered.
- You can reuse files from previous exercises

Chapitre 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 card 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 piscine. Try to go as far as possible.

Chapitre V

Exercise 00 : Color and Value

	Exercice : 00
Color and Value	
Dossier de rendu : <i>ex00/</i>	
Fichiers à rendre : <code>Color.swift</code> , <code>Value.swift</code> , <code>test file</code>	
Fonctions Autorisées : Aucune	


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

Chapitre VI

exercice 01 : Card

	Exercice : 01
Card	
Dossier de rendu : <i>ex01/</i>	
Fichiers à rendre : <code>Color.swift</code> , <code>Value.swift</code> , <code>Card.swift</code> , <code>test file</code>	
Fonctions Autorisées : <code>Aucune</code>	

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

- Properties **color** and **value**
- A class constructor which takes a **Color** and a **Value** as parameters
- An override of the property `description: String` that allow you to write the map
- An override of the method **isEqual** of **NSObject**


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
```

Chapitre VII

Exercise 02 : Deck

	Exercice : 02
Deck	
Dossier de rendu : <i>ex02/</i>	
Fichiers à rendre : <code>Color.swift</code> , <code>Value.swift</code> , <code>Card.swift</code> , <code>Deck.swift</code> , a test file	
Fonctions Autorisées : La methode <code>map</code> des instances d' <code>Array</code>	


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

Chapitre VIII

Exercise 03 : Extension


	Exercice : 03
Extension	
Dossier de rendu : <i>ex03/</i>	
Fichiers à rendre : <code>Color.swift</code> , <code>Value.swift</code> , <code>Card.swift</code> , <code>Deck.swift</code> , a <code>test file</code>	
Fonctions Autorisées : <code>arc4random_uniform</code>	

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

At this exercise you will make an extension of **struct Array** in the **Deck.swift** file which adds a method that mixes the table randomly.

Chapitre IX

Exercise 04 : Board

	Exercice : 04
Board	
Dossier de rendu : <i>ex04/</i>	
Fichiers à rendre : <code>Color.swift</code> , <code>Value.swift</code> , <code>Card.swift</code> , <code>Deck.swift</code> , a test file	
Fonctions Autorisées : Toutes les méthodes de <code>Array</code>	

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

card : which represents all the cards from the deck

discards : which represents all the fake cards

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

Create an class constructor which takes an **Bool** parameter which represents if the deck have 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 map of **cards** and place it in **outs**.

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