



Piscine iOS Swift - Day 01

Card game

Summary: Here is the content of the iOS Swift Piscine's Day 01 de [42](#)

Contents

I	Preamble	2
II	Instructions	3
III	Today's specific rules	4
IV	Introduction	5
V	Exercise 00: Color and Value	6
VI	Exercise 01: Card	7
VII	Exercise 02: Deck	9
VIII	Exercise 03: Extension	10
IX	Exercise 04: Board	11

Chapter I

Preamble

"The [e-sport](#) is currently considered a gambling activity in France. Thus, it is accountable to the Online Gambling Commission and competitions might be prohibited in the country. However, regarding the relative novelty of esports, they're still tolerated." Translated from [Source](#)

"With an audience of more than 225 millions viewers, the eSport is considered the biggest professional sports league." Translated from [Source](#)

"These competitions advocate team spirit, self control and self excellence. Their online streaming modes also supports integration and intercultural exchange. Moreover, the competition's economic prospectives based upon tickets sales, audiovisual rights and indirect touristic benefits is important. It was valued at 800 millions euros in 2018. Hence, it's an opportunity for France, both economically and socially." Translated from [Source](#)

"The act of law on the digital proposed by Axelle Lemaire has been voted. The government has unveiled the main focuses of the act and among others, the acknowledgement of eSports." Translated from [Source](#)

Chapter II

Instructions

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



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



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

Chapter III

Today's specific rules

This is a special day. You will not turn-in an application but you will complete a few exercises to discover Swift.

- You will create a folder for each exercise at the root of your repo: *ex00 ex01 ex02...*
- You will compile your exercises with **swiftc**
- You must turn-in your own game tests to prove everything works properly during the evaluation. Be exhaustive. A game not tested will be considered invalid.
- You can use the files from the previous exercises.

Chapter IV

Introduction

Swift is a multi-paradigm programming language with a focus on protocol thanks to two tags: `protocol`, `extension`. To understand these notions, you will have to understand the object development.

Today, we're gonna work on a classic 52 cards game through various exercises that will allow you to acquaint yourself with Swift and make you use several notions:

Declarations: `var`, `let`, `type`, `weak`, `optional` so you can properly type your variables.

Control structures: `loop`, `conditions`, `if let` so you can structure your code.


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

Algo: `closures` for anonymous functions.

This day should get you ready for the rest of the piscine. Try to go as far as possible.

Chapter V

Exercise 00: Color and Value

	Exercise 00
Color and Value	
Turn-in directory : <i>ex00/</i>	
Files to turn in : Color.swift , Value.swift , a test file	
Allowed functions : Aucune	


For a starter, we're going to define what's the color and a value of a card from a classic 52 cards game.

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

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

Chapter VI

Exercise 01: Card

	Exercise 01
Card	
Turn-in directory : <i>ex01/</i>	
Files to turn in : Color.swift , Value.swift , Card.swift , a test file	
Allowed functions : Aucune	

Create the **Card** class that inherits **NSObject** with :

- The properties **color** and **value**
- A builder that takes a **Color** and a **Value**
- An override with the property **var description: String** that allows to write the card.
- An override of the **isEqual** method of **NSObject**


Overloading the operator "==" so it works on 2 **Card** that remotely works like the **isEqual** method.

Here is 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 : <i>ex02/</i>	
Files to turn in : <code>Color.swift</code> , <code>Value.swift</code> , <code>Card.swift</code> , <code>Deck.swift</code> , a test file	
Allowed functions : <code>The array instances method</code>	

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

allSpades: that represents all the **spades**

allDiamonds: that represents all the **diamonds**


allHearts: that represents all the **hearts**

allClubs: that represents all the **clubs**

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

Chapter VIII

Exercise 03: Extension


	Exercise 03
Extension	
Turn-in directory : <i>ex03/</i>	
Files to turn in : <code>Color.swift</code> , <code>Value.swift</code> , <code>Card.swift</code> , <code>Deck.swift</code> , a test file	
Allowed functions : <code>arc4random_uniform</code>	

Extensions are very useful to add code to some class or structure that already exist.

In this exercise, you will make an extension of the **struct Array** in the **Deck.swift** file that adds a method randomly mixing the table.

Chapter IX

Exercise 04: Board

	Exercise 04
Board	
Turn-in directory : <i>ex04/</i>	
Files to turn in : <code>Color.swift</code> , <code>Value.swift</code> , <code>Card.swift</code> , <code>Deck.swift</code> , a test file	
Allowed functions : All the Array methods	

Add 3 [**Card**] type properties to the **Deck** class:

cards: that represents all the deck's cards.

discards: that represents all the cards that have been discarded.

outs: that represents all the cards that are not in **cards** anymore and not yet in **discards**.

Create a builder that takes a **Bool** in parameter that shows if the deck must be sorted out or mixed.

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

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

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