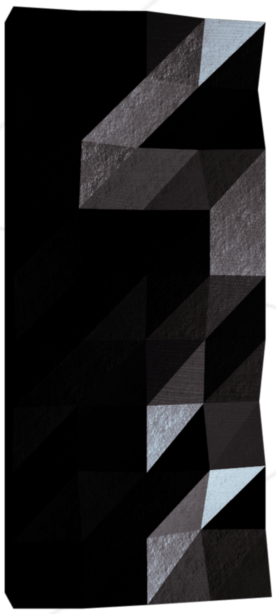

Piscine iOS Swift - Day 01

Card game

Summary: Here is the content of the iOS Swift Piscine's Day 01 de 42

Contents

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

Consignes

Sauf contradiction explicite, les consignes suivantes seront valables pour tous les jours de cette Piscine.

- Seul ce sujet sert de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices. L'url de votre dépot `GIT` pour cette journée est disponible sur votre intranet.
- Vos exercices seront évalués par vos camarades de Piscine.
- En plus de vos camarades, vous pouvez être évalués par un programme appelé la Moulinette. La Moulinette est très stricte dans sa notation car elle est totalement automatisée. Il est donc impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les mauvaises surprises.
- Les exercices shell doivent s'éxcuter avec `/bin/sh`.
- Vous ne devez laisser aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices dans votre dépot de rendu.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Toutes les réponses à vos questions techniques se trouvent dans les `man` ou sur Internet.
- Pensez à discuter sur le forum Piscine de votre Intra et sur Slack !
- Lisez attentivement les exemples car ils peuvent vous permettre d'identifier un travail à réaliser qui n'est pas précisé dans le sujet à première vue.
- Réfléchissez. Par pitié, par Thor, par Odin !

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 programing 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 : *ex*00/	
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 : *ex*01/
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 : *ex*02/	
Files to turn in : `Color.swift, Value.swift, Card.swift, Deck.swift, a test file`	
Allowed functions : `The array instances method`	

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 : *ex*03/	
Files to turn in : `Color.swift, Value.swift, Card.swift, Deck.swift, a test file`	
Allowed functions : `arc4random_uniform`	

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 : *ex*04/	
Files to turn in : `Color.swift, Value.swift, Card.swift, Deck.swift, 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**.