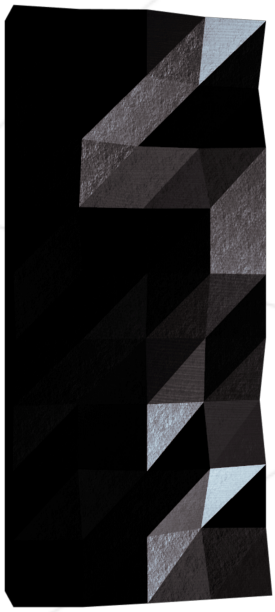

# Piscine iOS Swift - Day 01

## Jeu de cartes

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

*Résumé: Ce document contient le sujet du Day 01 de la piscine iOS Swift de 42*

# Table des matières

# Chapitre I

# Préambule

"L'e-sport est actuellement considéré comme une jeu de hasard. A ce titre, il dépend de l'Autorité de régulation des jeux en ligne (ARJEL) et les compétitions pourraient même être interdites... Cependant, compte tenu du vide juridique sur l'e-sport, elles sont pour le moment tolérées." Source

"Avec plus de 225 millions de spectateurs uniques, l'audience eSport est considérée comme la plus importante de toutes les ligues de sport professionnel." Source

"Ces compétitions participent à la promotion de l'esprit d'équipe, le contrôle et le dépassement de soi. Leur mode de diffusion, en ligne, encourage aussi l'intégration et l'échange entre les cultures. En outre, le potentiel économique de ces compétitions, qui tourne autour des billets d'entrées, des produits dérivés et des droits audiovisuels mais aussi des retombées touristiques indirectes, est important. Il est estimé à 800 Meuros en 2018. C'est donc une chance pour la France, socialement et économiquement." Source

"Le projet de loi sur le numérique présenté par Axelle Lemaire a été accepté. Le gouvernement a donc dévoilé les principaux points concernant cette loi dont la reconnaissance de l'e-Sport." Source

# Chapitre II

# Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Lisez attentivement l'integralité du sujet avant de commencer.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Vos exercices seront corrigés par vos camarades de piscine.
- Le sujet fait foi, ne vous fiez pas toujours à la lettre aux demos qui peuvent contenir des ajouts supplémentaires non demandés.
- Vous devrez rendre une app par jour (sauf pour le Day 01) sur votre depot git, rendez le dossier du projet Xcode.
- Voici le manuel officiel de Swift
- Voici le manuel officiel de Swift Standard Library
- Il est interdit d'utiliser d'autres librairies, packages, pods... avant le Day 07
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Pensez à discuter sur le forum Piscine de votre Intra !
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Réfléchissez. Par pitié, par Odin ! Nom d'une pipe.

L'intra indique la date et l'heure de fermeture de vos dépots. Cette date et heure correspond également au début de la période de peer-evaluation pour le jour de piscine correspondant. Cette période de peer-evaluation dure exactement 24h. Une fois ces 24h passées, vos notes peer manquantes seront complétées par des 0.

# Chapitre III

# Règles spécifiques de la journée

Ce jour est particulier. Vous ne rendrez pas une application mais vous allez faire quelques exercices pour découvrir le Swift.

- Vous allez créer un dossier par exercice à la racine de votre dépot : *ex00 ex01 ex02...*
- Vous compilerez vos exercices avec **swiftc**
- Vous devrez rendre vos propres jeux de tests pour prouvez que tout fonctionne correctement en soutenance, soyez exhaustif. Ce qui n'est pas testé est considéré comme faux
- Vous pouvez réutiliser les fichiers des exercices précédents

# Chapitre IV

# Introduction

Swift est un langage de programmation multi-paradigme mais surtout orienté protocole grâce à ses deux mots clés : `protocol, extension`. Pour bien comprendre ces notions il faut d'abord être au point sur le développement orienté objet.

Aujourd'hui nous allons nous focaliser sur un jeu classique de 52 cartes au travers de différents petits exercices qui ont pour but de vous familiariser avec swift et vous faire utiliser plusieurs notions :

**Les déclarations :** `var, let, type, weak, optional` pour bien savoir typer vos variables

**Les structures de contrôle :** `loop, conditions, if let` pour structurer votre code

**Classes :** `class, func, overload, override, struct, enum, inheritance, extension, mutating` pour le dévelppement orienté objet

**Algo :** `closures` pour les fonctions anonymes.

Cette journée doit vous armer pour le reste de la piscine, essayez d'aller le plus loin possible.

# Chapitre V

# Exercice 00 : Color et Value

|  | Exercice : 00 |
|---|---|
| Color et Value | |
| Dossier de rendu : $ex00/$ | |
| Fichiers à rendre : `Color.swift, Value.swift, un fichier de test` | |
| Fonctions Autorisées : `Aucune` | |
| Remarques : `n/a` | |

Pour commencer nous allons définir ce qu'est une couleur et une valeur d'une carte d'un jeu classique de 52 cartes.

Créez un enum **Color** avec comme valeur brute le type **String** qui représentera les 4 couleurs. Ajoutez lui une constante static **allColors** de type [**Color**] qui représentera l'ensemble des couleurs possibles d'une carte.

Créez maintenant un enum **Value** avec comme valeur brute le type **Int** qui représentera les valeurs des cartes. Ajoutez lui une constante static **allValues** de type [**Value**] qui représentera l'ensemble des valeurs possibles d'une carte.

# Chapitre VI

# Exercice 01 : Card

|  | Exercice : 01 |
|---|---|
| Card | |
| Dossier de rendu : $ex01/$ | |
| Fichiers à rendre : `Color.swift, Value.swift, Card.swift, un fichier de test` | |
| Fonctions Autorisées : `Aucune` | |
| Remarques : `n/a` | |

Créer la class **Card** qui hérite de **NSObject** avec :

- Les properties **color** et **value**
- Un constructeur qui prend une **Color** et une **Value**
- Un override de la property **var description : String** qui permet de décrire la carte
- Un override de la méthode **isEqual** de **NSObject**

Overloader l'opérateur "==" pour qu'il fonctionne sur 2 **Card** qui fait sensiblement la même chose que la méthode **isEqual**.

Voici un exemple :

```
> 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

# Exercice 02 : Deck

|  | Exercice : 02 |
|---|---|
| Deck | |
| Dossier de rendu : $ex02/$ | |
| Fichiers à rendre : `Color.swift, Value.swift, Card.swift, Deck.swift, un fichier de test` | |
| Fonctions Autorisées : `La méthode map des instances d'Array` | |
| Remarques : `n/a` | |

Créer la class **Deck** héritant de **NSObject**.
Ajoutez les constantes static de type [**Card**] :

**allSpades :** qui représente tous les `piques`

**allDiamonds :** qui représente tous les `carreaux`

**allHearts :** qui représente tous les `coeurs`

**allClubs :** qui représente tous les `trèfles`

Ajouter la constante static **allCards** de type [**Card**] qui sera la liste de toutes les cartes possibles d'un jeu de 52.

# Chapitre VIII

# Exercice 03 : Extension

|  | Exercice : 03 |
|---|---|
| Extension | |
| Dossier de rendu : $ex03/$ | |
| Fichiers à rendre : `Color.swift, Value.swift, Card.swift, Deck.swift, un fichier de test` | |
| Fonctions Autorisées : `arc4random_uniform` | |
| Remarques : `n/a` | |

Les extensions sont extrêmement pratiques pour ajouter du code à des class ou structures déjà existantes.

Dans cet exercice vous allez faire une extension de la **struct Array** dans le fichier **Deck.swift** qui ajoute une méthode qui mélange le tableau aléatoirement.

# Chapitre IX

# Exercice 04 : Board

|  | Exercice : 04 |
|---|---|
| Board | |
| Dossier de rendu : $ex04/$ | |
| Fichiers à rendre : `Color.swift, Value.swift, Card.swift, Deck.swift, un fichier de test` | |
| Fonctions Autorisées : `Toutes les méthodes de Array` | |
| Remarques : `n/a` | |

Ajoutez 3 properties de type [**Card**] à la class **Deck** :

**cards :** qui représente toutes les cartes du deck

**discards :** qui représente toutes les cartes défaussées

**outs :** qui représente toutes les cartes qui ne sont plus dans `cards` et pas encore dans `discards`

Créer un constructeur qui prend un **Bool** en paramètre qui représente si le deck doit être trié ou mélangé.

Overridez la property **var description : String** qui retourne toutes les cartes de **cards**.

Créer la méthode **draw () -> Card ?** qui pioche la première carte de **cards** et la place dans **outs**.

Créer la méthode **fold(c : Card)** qui place la carte c dans **discards** si elle appartient bien à **outs**.