Les bases du langage Swift (M623)

LP SMIN, IUT1, UGA

Gwen Salaün

Organisation du M623

- 34h de cours / TP / projet

- Contenu en 3 parties :
  - Apprentissage du langage Swift
  - Bases de la programmation iOS
  - Projet : développement d'une application iOS

- Évaluation sur le projet (rendu et présentation)

=> Les supports de cours / ressources sont sur moodle

Qu’est ce que Swift ?

- Swift est un nouveau langage de programmation à la fois puissant et intuitif, créé par Apple pour l’élaboration d’applications iOS et Mac

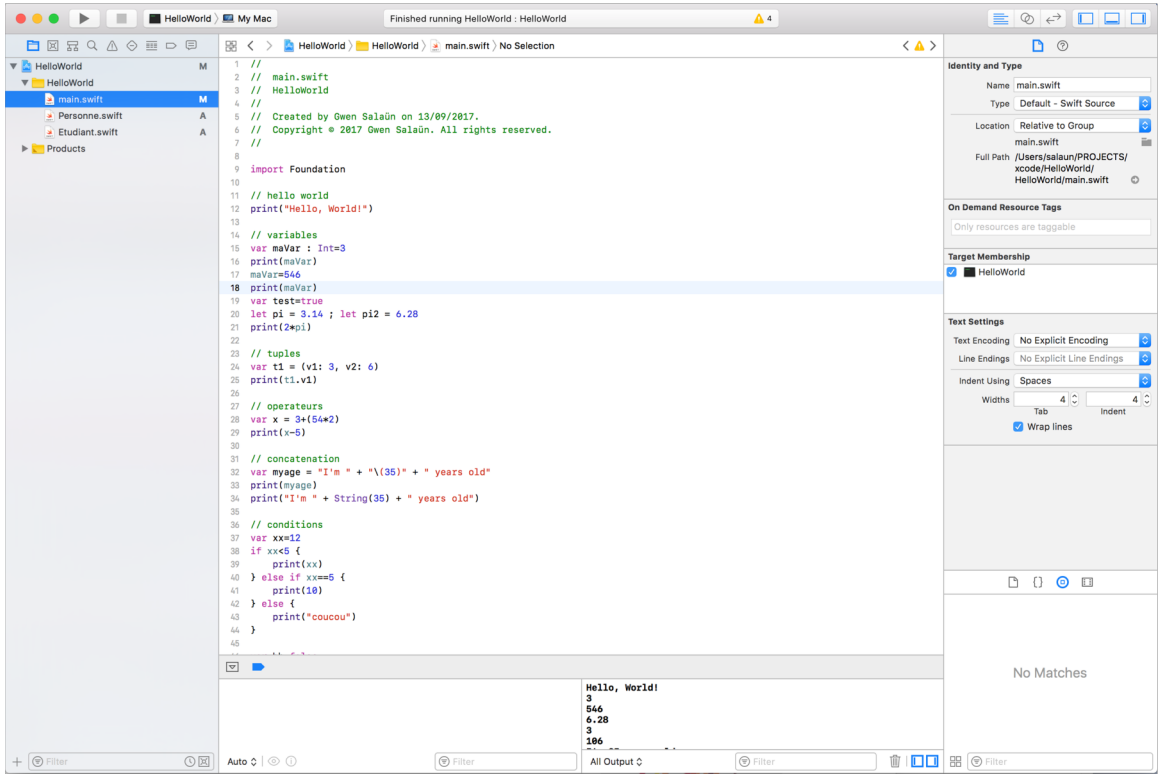
- Swift a pour ambition de succéder à Objective-C et est beaucoup plus rapide, plus concis, plus simple que l'Objective-C

- Première version de Swift date de 2014.

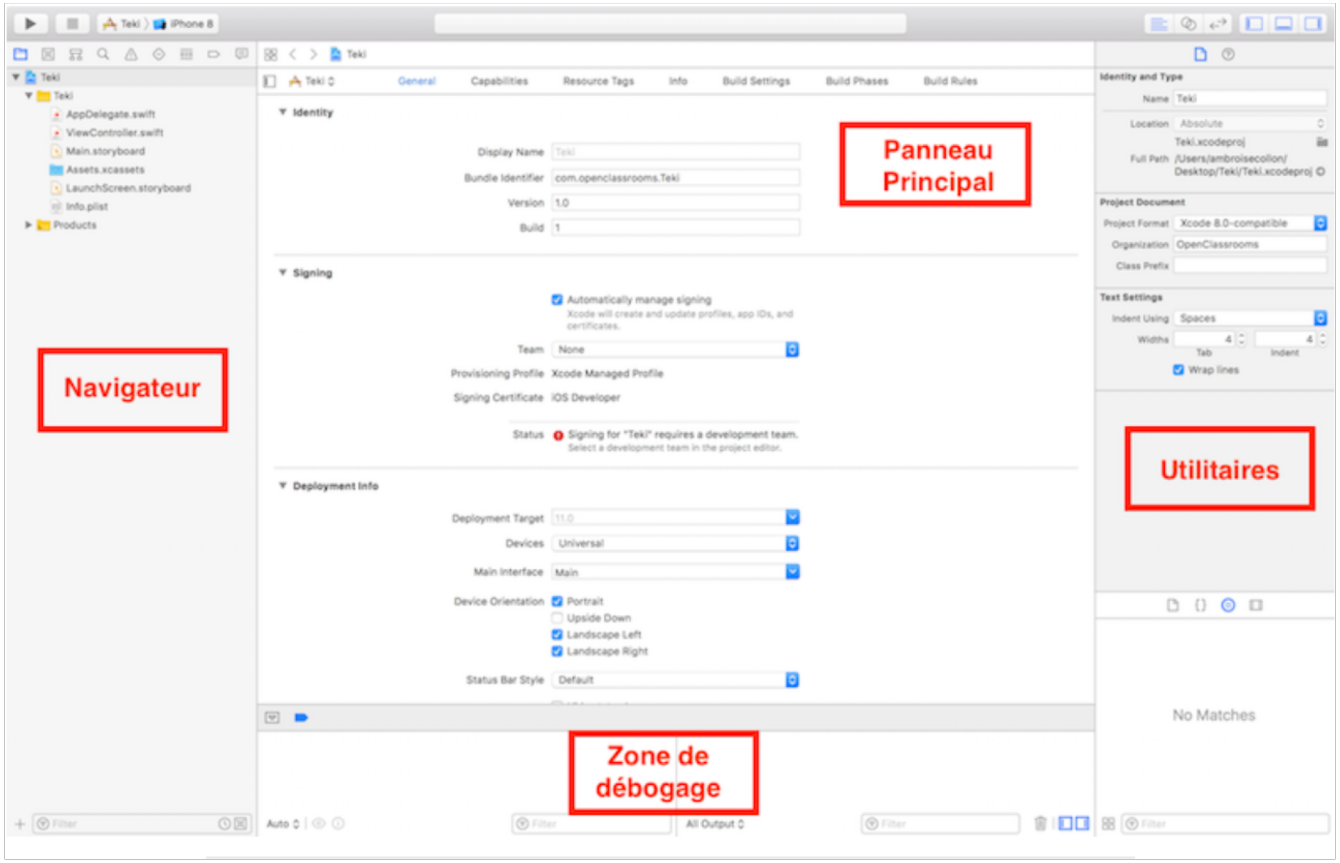
- Version courante : Swift 4.2 (octobre 2018)

Outil de développement

Programmer en Swift avec Xcode (dernière version : 10.1, octobre 2018)



Interface de Xcode



Commentaires

- Qu'est-ce qu'un commentaire ? Il s'agit d'un texte, d'un code, de tout ce que vous voulez qui ne sera pas interprété par votre programme Swift.

- Les commentaires sont cruciaux si vous voulez écrire du code compréhensible et réutilisable

```
1 // Commentaire sur une seule ligne
2 print("Hello, World !") // Affichera "Hello, World !" à l'écran
```

```
1 /* Voici un exemple
2 d'un commentaire
3 sur plusieurs lignes */
4 print("Hello, World !") /* Affiche "Hello, World !" à l'écran */
```

Variables

- Utilisation du mot clé ‘var’ (attention à la casse!)

```
1 var nombreDeFreres = 2
2 var nombreDeSoeurs = 3
3 nombreDeFreres = 3
4 nombreDeSoeurs = 4
```

- Le typage est implicite par défaut mais peut être expliciter

```
1 var nombreDeFreres: Int
2 var pi: Double
3 var phrase: String
4 var information: Bool
```

- Définition de constante avec ‘let’

```
1 let pi = 3.1415
```

Tuples

- Un tuple peut contenir plusieurs valeurs

```
let tuple = (valeur1, valeur2, ...)
// Ou
var tuple = (valeur1, valeur2, ...)
```

- Vous pouvez aussi directement donner un nom à vos valeurs lorsque vous déclarez un tuple

```
let erreur404Http = (code: 404, message: "Not Found")
```

```
let valeur1 = erreur404Http.code
let valeur2 = erreur404Http.message
```

Opérateurs

Signe	Signification
+	Addition
-	Soustraction
*	Multiplication
/	Division

```
let division = 100 / 11 // Dans 100 combien de fois 11 ?
let modulo = 100 % 11 // Quel est le reste de 100 / 11 ?

/*
    Dans 100 il y a 9 fois 11
    Il reste alors 1
*/
print(division) // Affiche 9
print(modulo) // Affiche 1
```

Concaténation et conversion

- Pour concaténer deux variables on utilise ‘+’

```
1  let hello = "Hello,"
2  let world = " World !"
3  let helloWorld = hello + world
4
5  print(helloWorld)
6  print(hello + world) // On aurait aussi pu écrire directement comme ceci
7
8  let nombre1 = 1
9  let nombre2 = 2
10 // Astuce pour concaténer deux entiers
11 let nombre = "\(nombre1)" + "\(nombre2)"
12 print(nombre)
```

- Conversion de type

```
1 let nombre = 72
2 let monTexte = "Mon nombre choisi est le nombre : " + String(nombre)
3 print(monTexte)
```

Conditions (1/2)

- Opérateurs de comparaison

Symbole	Signification
==	est égal à
>	est supérieur à
>=	est supérieur ou égal à
<	est inférieur à
<=	est inférieur ou égal à
!=	est différent de

- Condition avec un ‘if’

```
1 let age = 20
2
3 if age >= 18 {
4     print("Vous êtes majeur !")
5     // Exécutez ici tout ce que vous voulez pour une personne majeure !
6 }
```

Conditions (2/2)

- Combiner des conditions

Signe	Signification	Vrai quand ?
&&	et	Vrai quand les deux valeurs sont vraies.
||	ou	Vrai quand au moins une des deux valeurs est vraie.

- Condition avec un ‘if’ et une condition double

```
1 let age: Int = 32
2 let nationalite: String = "USA"
3
4 if age >= 21 && nationalite == "USA" {
5     print("Vous êtes Américain et en plus vous êtes majeur.")
6 }
```

Priorité des opérateurs

Signification	Opérateur
Parenthèses	(,)
Non	!
Multiplication, division, modulo	*, /, %
Addition soustraction	+, -
Inférieur, inférieur ou égal, supérieur, supérieur ou égal	<, <=, >, >=
Égal, différent de	==, !=
Et	&&
Ou	||

Structure ‘if .. else’

- Un exemple

```
let age: Int = 18

if age >= 21 {
    print("Vous êtes majeur, et même aux Etat-Unis !")
} else if age >= 18 {
    print("Vous êtes majeur, mais pas aux Etat-Unis. :(")
} else {
    print("Vous êtes mineur.")
}
```

- Lorsqu’il y a beaucoup de conditions, utiliser le ‘switch’ !

```
let note = 19

switch note {
case 0...9:
    print("Vous n'avez pas la moyenne, vous n'avez donc pas de mention.")

case 10...12:
    print("Vous avez la moyenne, mais vous n'obtenez pas de mentions.")

case 12...14:
    print("Vous avez obtenu la mention assez bien.")

case 14...15:
    print("Vous avez obtenu la mention bien.")

case 16...18:
    print("Vous avez obtenu la mention très bien.")

case 18...20:
    print("Vous avez les félicitations du jury !")

default:
    print("Navré, il faut avoir une de ces notes pour avoir une mention.")
}
```

La condition ternaire

- Elle permet de faire la même chose qu'un 'if .. else' mais en compact

```
1  let note = 12
2  var moyenne: Bool
3
4  // Cette partie
5  if note < 10 {
6      moyenne = false
7  } else {
8      moyenne = true
9  }
10
11 // Est identique à cette partie
12 moyenne = note < 10 ? false : true
```

Boucle ‘while’

- Répétition d’une séquence d’instructions tant qu’une condition est vérifiée

- Exemple :

```swift
var nbDeLignes: Int = 1

while nbDeLignes <= 1000 {
    print("\(nbDeLignes). Je dois apprendre mes leçons en cours de Swift.")
    nbDeLignes += 1
}
```

Boucle ‘repeat .. until’

- C’est une variante du ‘while’
- Différence : elle s’exécute au moins une fois
- Exemple :

```swift
var nbDeLignes: Int = 1

repeat {
    print("\(nbDeLignes). Je dois apprendre mes leçons en cours de Swift.")
    nbDeLignes += 1
} while nbDeLignes <= 1000
```

Boucle ‘for .. in’

- La boucle ‘for’ est utile lorsqu’on sait exactement quand est ce que l’on veut s’arrêter
- Exemple :

```
1 var nbDeLignes: Int
2
3 for nbDeLignes in 1...1000 {
4     print("\(nbDeLignes). Je dois apprendre mes leçons en cours de Swift.")
5 }
```

Tableaux

- Déclaration et accès

```
let prenoms = ["Camille", "Maxime", "Antoine", "Lucie", "Mathilde"]

print(prenoms[2])
```

- Ajout

```
var prenoms = ["Chloé", "Damien", "Antoine", "Lucie", "Mathilde"]

// On ajoute en fin de tableau
prenoms = prenoms + ["Jean"]
print(prenoms)

// Ou bien encore en début du tableau
prenoms = ["Jean"] + prenoms
print(prenoms)
```

- Modification

```
prenoms[0] = "Rudy"
```

Dictionnaires

- Déclaration et ajout

```
1 var personne = ["Nom": "Durand", "Prénom": "Maxime", "Adresse": "94 rue machin",
  "Ville": "Lille"]
2 personne["Commentaire"] = "Personne super cool !"
3
4 print(personne)
```

- Modification

```
1 var personne = ["Nom": "Durand", "Prénom": "Maxime", "Adresse": "94 rue machin",
  "Ville": "Lille"]
2 personne["Adresse"] = "12 rue bidule"
3
4 print(personne)
```

Parcours de tableau / dict.

- Utilisation de la boucle ‘for .. in’ sans indice

```
1 let prenoms = ["Camille", "Maxime", "Antoine", "Lucie", "Mathilde"]
2
3 for prenom in prenoms {
4     print(prenom)
5 }
```

```
1 let personne = ["Nom": "Durand", "Prénom": "Maxime", "Adresse": "94 rue machin",
  "Ville": "Lille"]
2
3 for (cle, valeur) in personne {
4     print(cle + " - " + valeur)
5 }
```

Fonction

- Une fonction est une portion de code qui vous permet d'exécuter une suite d'instructions et qui va vous retourner ou non une valeur

```
1 func disBonjour(prenom: String, nom: String) {
2     print("Bonjour " + prenom + " " + nom + " !")
3 }
4
5 // Pour l'utiliser
6 disBonjour(prenom:"Jean", nom: "Dupont")
7 disBonjour(prenom:"Robert", nom: "Durand")
```

Fonction avec retour

- Le type de retour est explicité dans la signature de la fonction

```
 7 func calculePerimetreRectangle(longueur: Double, largeur: Double) -> Double {
 8     let perimetre = (longueur + largeur) * 2
 9
10     return perimetre
11 }
```

```
17 let largeur = 2.5
18 let longueur = 5.0
19 let perimetre = calculePerimetreRectangle(longueur: longueur, largeur: largeur)
20 print("Le périmètre du rectangle est de \(perimetre) cm.")
```

Fonction en paramètre

- Il est possible de passer une fonction en paramètre

```swift
func premierPlusGrandQueDeuxieme(nb1: Int, nb2: Int) -> Bool {
    return nb1 > nb2 // nb1 > nb2 retourne un booléen si c'est vrai ou non
}

func maFonction(maFonctionParametre: (Int, Int) -> Bool) {
    if maFonctionParametre(4, 3) {
        print("Condition validée.")
    }
}

// On a juste à fournir le nom de notre fonction
// A condition qu'elle respecte les paramètres et type de retour
maFonction(maFonctionParametre: premierPlusGrandQueDeuxieme)

// Affichera "Condition validée"
```

Fonction en retour

- Il est possible de retourner une fonction en précisant le type de retour

```swift
func hello(debutMessage: String) -> (String) -> String {

    func nestedHello(finMessage: String) -> String {
        return "\(debutMessage) \(finMessage)"
    }
    return nestedHello
}

print(hello(debutMessage: "Hello, ")("World !"))
// Affichera "Hello World !"
```

Exercice 1

- Écrire un programme Swift qui calcule les nombres premiers jusqu'à une certaine borne, en utilisant le crible d'Ératosthène (voir wikipedia)

L'algorithme:

```
Construire une liste de tous les entiers supérieurs à 1 et inférieurs ou égal à n.
Supprimer les multiples de tous les premiers inférieurs ou égal à la racine carrée de n,
ensuite, les nombres qui restent sont premiers.
```

En pseudo code (en fait en code Scriptol):

```
const int n=50
array a = [0]
for int i in 1..n let a[i] = 1

int m = int(sqrt(n))

for int i in 2..m
   if a[i]
      for int j in 2.. n
         a[i * j] = false
      /for
   /if
/for
```

Afficher les nombres premiers:

```
for int x in 0 .. n
   if a[x] print x
/for
```

Exercice 2

- Écrire un programme Swift qui étant donné un dictionnaire de couples *(prénom, note)* définit les fonctions effectuant les traitements suivants :
  - la recherche de la note d'une personne
  - la modification de la note d'une personne
  - l'ajout d'un couple *(prénom, note)*
  - la recherche de la meilleure note
  - le calcul de la moyenne
  - l'affichage des couples par ordre alphabétique des prénoms
  - l'affichage des couples par ordre croissant des notes
  - la suppression de tous les couples ayant une note donnée
  - …

Exercice 3

- Écrire une fonction en swift qui encode le tri à bulles

```
tri_à_bulles(Tableau T)
    pour i allant de taille de T - 1 à 1
        pour j allant de 0 à i - 1
            si T[j+1] < T[j]
                échanger(T[j+1], T[j])
```

Ressources

- Cours Swift sur OpenClassrooms

https://openclassrooms.com/en/courses/1823851-apprenez-a-programmer-en-swift

- Tutoriel apple

https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/index.html#//apple_ref/doc/uid/TP40014097-CH3-ID0