IF26 - développement d'applications mobiles

TP07 - iOS 1 - prise en main de l'environnement de développement

Préparations

➢ Objectifs

- L'objectif de ce TP concerne la découverte et l'apprentissage du langage Swift proposé par Apple.

➢ Connaissances et compétences

- Savoir installer l'environnement de développement sur son poste de travail
- Concevoir des programmes en Swift
- Appréhender les principes de la programmation orientée objet avec Swift

➢ Ressources

- Développement iOS en Swift : Cours et tutos : Exercice 1 - Fonctions
  - https://youtu.be/Rqmm6S00rjA
- Développement iOS en Swift : Cours et tutos : Exercice 2 - Fonctions (2m18s)
  - https://youtu.be/b5F9byYn2Ok
- Développement iOS en Swift : Cours et tutos : Exercice 3 - Chaînes
  - https://youtu.be/C-clPhcWK68
- Swift classes
  - https://youtu.be/2HwnLD0Xxro
- Protocol
  - https://youtu.be/KzyplO_QoMw

Exercice 1 : Environnement Xcode et langage Swift

1. Lancez l'application Xcode et effectuez éventuellement les mises à jour.
2. Sélectionnez le premier item : Get Started with a playground.
   a. Sélectionnez iOS parmi la liste iOS, tvOS et macOS
   b. Sélectionnez blank
   c. Sauvegardez votre fichier sous le nom if26_tp07_exo1.playground dans un répertoire IF26_tp07.
3. Type, variables et affectations en swift
   a. Définir une variable x ayant pour valeur 2.
   b. Quelle est la différence entre les instructions var et let. Montrer par un exemple l'intérêt de l'instruction let.
   c. Recherchez sur Internet la liste des types de base en swift (Int, Double, ...).
   d. Définir une variable prix de type Double ayant une valeur de 12,56.
   e. Ajouter une instruction print pour afficher uniquement la variable prix.
   f. Ajouter une instruction print pour afficher un message "prix = " et la variable prix.
4. Chaînes de caractères et fonctions
   a. Définissez deux chaînes de caractères nom et prénom et proposez des valeurs.
   b. Concaténez les deux chaînes avec un espace entre les deux et affichez le résultat.

   c. Rédigez une fonction echo1 possédant un paramètre de type chaîne de caractères qui permet l'affichage à l'écran du paramètre. Application avec nom et prénom.
   d. Rédigez une fonction echo2 possédant un paramètre de type chaîne qui renvoie la longueur de la chaîne. Application avec nom et prénom.

5. Définition de tupes (liste)
   a. Définissez une liste de sigles de modules du programme ISI (exemple LO07 NF19 IF26 ...).
   b. Affichez le second élément de la liste.
   c. Définissez une nouvelle liste de modules du programme ISI sous la forme d'une liste de couples (clé / valeur). La clé correspond au sigle et la valeur à l'intitulé du module.
   d. Affichez l'intitulé du module correspondant à la clé IF26.
   e. Remarque : ne pas confondre une liste définie avec des parenthèses avec un tableau défini avec des crochets.
6. Définir des types
   a. Il est possible de définir des identifiants de type à l'aide de l'instruction typealias. Définissez un nouveau type Point composé de 2 entiers (Int) représentant ses coordonnées x et y.
   b. Créez une constante point1 ayant pour coordonnées 4 et 6 et affichez cette constante. Même chose avec un point2 (10,0).
   c. Définissez une fonction fpoint ayant deux paramètres de type Point et retournant un Point dont les coordonnées x et y correspondant aux plus grandes valeurs de x et de y. Application avec les deux points définis précédemment.

Exercice 2 : Swift est objet (classes, héritage, protocole, ...)

1. Créez un nouveau fichier if26_tp07_exo1.playground qui regroupera l'ensemble des codes de l'exercice 2.
2. Rédigez une classe Personne possédant trois attributs nom, prénom et âge.
3. Ajoutez un constructeur par défaut init().
4. Ajoutez un deuxième constructeur avec trois paramètres. Recherchez la syntaxe sur Internet.

   ```
   init (nom: String, prenom: String, age: Int)
   ```
5. Ajoutez une méthode isAdult() qui retourne True si l'étudiant est majeur.
6. Ajoutez une propriété calculée description qui joue le rôle de la méthode toString() en Java.
7. Créez une instance p1 de la classe Personne.
8. Affichez la description de cette instance p1.
9. Créez une classe Etudiant dérivant de la classe Personne disposant d'un attribut supplémentaire nocarte (String).
10. Ajoutez un constructeur init(...) avec quatre paramètres. Vous devez faire appel à la méthode init() de la classe Personne.
11. Ajoutez un attribut description qui joue le rôle de la méthode toString() en Java. Vous devrez ajouter le mot clé override avant la déclaration de la variable description.
12. Créez un Etudiant p2 et affichez sa description
13. Créez un protocol (une Interface en Java) nommé Utt et contenant la description de deux méthodes :

    ```
    func juryResultat() -> String
    func GPA() -> Int
    ```
14. Créez une classe Doctorant dérivant de la classe Etudiant et implémentant le protocole Utt.
15. Comment reconnaît-on la classe dérivée des protocoles dans l'écriture d'une classe ?

16. Proposez un code arbitraire pour les deux méthodes
17. Créez une instance p3 de la classe Doctorant et faite appel à la méthode juryResultat().

Exercice 3 : Traduire les classes Java Resultat, Module et Cursus en Swift

1. Créez un nouveau fichier if26_tp07_exo3.playground qui regroupera l'ensemble des codes de l'exercice 3.
2. Quels sont les intérêts de définir une énumération ?
3. Recherchez la syntaxe pour la définition d'une énumération (enum) en Swift.
4. Recherchez la syntaxe pour la définition d'une énumération associée à une valeur en Swift. Différences avec la définition en Java ?
5. Application avec la réécriture de la classe Java Resultat en Swift.
6. Rédigez une classe Module en Swift avec :
    a. Sigle, catégorie, crédit et résultat comme attributs
    b. une méthode init() qui attribue des valeurs par défaut aux 4 attributs
    c. une méthode init(...) avec 4 paramètres pour une initialisation complète d'un objet Module
    d. une variable descriptor
7. Créez un objet m1 avec les données de IF26 et affichez son descriptor pour valider votre travail.
8. Rédigez une classe Cursus en Swift avec :
    a. un seul attribut profil correspondant à un tableau d'objet Module
    b. une méthode ajoute(m: Module) qui ajoute un module dans le profil
    c. une méthode init() qui initialise le profil avec au moins 4 objets Module.
    d. une méthode getSigles() qui retourne un tableau des sigles des modules
    e. une méthode getModule(p: Int) qui retourne le module situé à la position p dans le profil.
9. Créez un objet monCursus
10. Affichez le résultat de la méthode getSigles() et de la méthode getModule(2).

Exercice final : devoir

1. Pour chaque étudiant, proposez soit une question supplémentaire à l'un des exercices de ce TD/TP, soit un commentaire, soit un QCM ou soit une ressource disponible sur Internet.
2. Vous transmettrez votre proposition via le devoir moodle sur le site e-learning du module.