Cours n°2 - Swift et Xcode

Partie 1 - Swift

Le but de ce cours est de vous introduire au langage Swift.

Le langage a tout d'abord été conçu par Apple pour le développement natif d'applications iOS mais est aujourd'hui utilisé pour développer des projets de différentes natures et pouvant être développés ou exécutés sous macOS, Linux ou Windows.

Swift est un langage compilé (comme le C ou le Java) ce qui signifie qu'il faut compiler vos fichiers avant de les exécuter. Pour cela, dans un terminal ou une invite de commande sous Windows, vous pouvez naviguer grâce à la commande `cd` dans le dossier où se trouve votre fichier et taper « `swiftc` » suivi d'un espace et du nom de votre fichier puis valider en appuyant sur « Entrée ». Un fichier exécutable portant le même nom que votre fichier (sans l'extension) sera alors créé. Vous pouvez l'exécuter pour voir le résultat en tapant le nom du fichier (sans l'extension) précédé d'un point et d'un slash (c.f. diapositive 5).
Vous pouvez écrire votre code Swift avec n'importe quel éditeur de code (Sublime Text, Atom, etc.).

Vous pouvez aussi passer par le mode interactif (appelé REPL) pour exécuter du code Swift à la volée (c.f. diapositive 6).

En principe, le code des diapositives parle de lui-même. Il y a évidemment quelques différences avec des langages comme le C ou le Java mais le mode de pensé reste le même pour les types et structures de contrôle de base (diapositives 7 à 15). La seule différence notable avec les langages auxquels vous êtes habitués est que le type d'une variable peut-être détecté automatiquement lorsque vous fournissez une valeur (exemple à la diapositive 7).

On notera toute fois sur les diapositives 14 et 15 deux particularités : par défaut, l'appel d'une fonction en Swift requiert que l'on passe le nom de chaque argument. Il est possible de revenir à quelque chose de plus usuel en plaçant un signe underscore suivi d'un espace devant la définition de chaque argument (c.f. diapositive 15). Aussi, on notera sur cette diapositive que Swift rend le mot clé `return` optionnel ; la dernière ligne d'une fonction est considérée comme la valeur de retour si `return` n'est pas utilisé.

Le code concernant les classes (diapositives 16 à 20) est censé parler de lui-même également.

En ce qui concerne les protocoles (diapositive 21), rien de nouveau si ce n'est le nom : on retrouve le même concept que les interfaces en Java, c'est à dire que l'on définit un « contrat » que chaque classe qui y souscrit devra honorer (c'est à dire, en fournissant l'implémentation de chaque méthode du protocole).

Les énumérations sont semblables à ce que l'on peut trouver dans les langages objets également si ce n'est qu'elles sont utilisées pour la définition des erreurs dans vos programmes (c.f. diapositives 22 à 24). On notera aussi à la dernière ligne de la diapositive 21 que Swift fournit un raccourci pratique lorsque le type d'une variable est déjà connu : seul le nom du champ de l'énumération est requis. Au niveau de la diapositive 24, le mot clé `guard` n'a rien de magique ; vous pouvez considérer ça comme l'équivalent d'un `if`.

Le seul concept que vous n'avez peut-être pas l'habitude de rencontrer (bien qu'il existe en Java sous le nom de « Lambda ») est celui des closures (diapositives 25 à 28). Basiquement, une closure est une fonction qui n'a pas de nom. C'est un bloque de code que l'on peut passer en paramètre d'autres fonctions.

Par exemple, la diapositive 26 montre la définition d'une closure se contentant d'afficher avec la fonction `print` un élément qui lui est passé en paramètre. Une closure peut-être stockée dans une variable comme n'importe quel autre élément ; ici, cela permet de l'appeler comme une fonction classique.

Simple raccourci que vous pourrez rencontrer en lisant du code Swift (diapositive 27) écrit par quelqu'un d'autre : Swift permet d'accéder aux paramètres d'une closure par rapport à leur position (`$0` étant le premier paramètre, `$1` le deuxième, etc.).

Les closures sont notamment utilisées pour parcourir ou traiter des ensembles d'éléments comme le montre la diapositive 28 avec dans un premier temps une closure stockée dans une variable et la même closure mais passée à la volée.

Partie 2 - Xcode

Les diapositives concernant Xcode ne vous seront utiles que si vous êtes sous macOS ou si nous sommes présents physiquement à la fac.

Xcode est l'IDE fourni par Apple qui permet de faire du développement Swift que ce soit pour de simples programmes ou des applications iOS. Le panneau qui s'affiche à l'ouverture vous permet de créer un nouveau projet complet, nous y reviendrons plus tard.

Dans le cadre des premiers exercices, Xcode propose un outil extrêmement pratique qui s'appelle « Playground » et qui permet de tester du code Swift simplement en ayant un retour visuel.