CPOA : Amphi 12

Langages modernes

Il existe plusieurs langages de programmation car chacun permet de répondre à une problématique donnée. De plus, les communautés de développeurs poussent les langages à évoluer et donc à provoquer l'apparition de nouveaux langages ou de nouvelles versions de ceux existant déjà.

Les langages de programmation ne sont pas figés dans le temps. Ils évoluent de manière constante pour être de plus en plus efficace pour exprimer ce qu'on souhaite faire.

Par exemple, depuis le début de ce cours, en septembre, ces langages ont évolués : Java 9, C++ 17, Kotlin 1.2, Clojure 1.9

Les langages modernes proposent également de **nouvelles fonctionnalités**.

En effet, dans de nombreux langages, il n'est plus nécessaire de déclarer un type pour chaque variable. A la compilation, le type de la variable est **déduit automatiquement** en fonction de son contenu dans le programme.

Des **expressions lambdas** apparaissent également dans le but de simplifier davantage la syntaxe pour le développeur.

Les littéraux sont de plus en plus utilisés pour les collections car ils permettent d'alléger considérablement la syntaxe.

Exemple d'utilisation des littéraux :

{'a' : x, 'b' : y} = f()

est un littéral permettant de passer les paramètres x et y dans la fonction f et de récupérer leurs valeurs par un simple affichage de ces 2 variables (x et y).

Le **REPL (Read Eval Print Loop)** consiste à proposer une console où on saisit de simples instructions qui seront exécutées directement les unes après les autres (ex. : interpréteur en console de Python).

Typage statique vs dynamique

Types statiques : Chaque expression est affectée à un type lors de la phase de compilation.

Types dynamiques : Les types des expressions sont déterminés lors de l'exécution.

Remarque : C++ est un langage purement statique. Les types sont affectés uniquement lors de la phase de compilation.

Les expressions lambdas

Les expressions lambdas permettent de définir des fonctions anonymes.

En Java, une fonction anonyme est une classe anonyme qui implémente une interface avec une seule méthode. Les expressions lambdas sont disponibles à partir de Java 8.

Exemple de lambda en Java :

```
Integer difference = (x, y) -> x-y ;
```

En termes de performance, la création d'une expression lambda ne nécessite pas l'instanciation d'un objet, donc pas besoin de demander à la JVM de créer un objet qui sera ensuite nettoyé par le garbage collector.

Les lambdas permettent de simplifier la création des méthodes callback (réaction à un évènement via un observer).

Python

Python est un **langage orienté objet** (on peut créer des classes) qui **se focalise sur la lisibilité du code**. Ce langage est compatible avec la programmation fonctionnelle et utilise le principe du REPL via sa console d'interprétation.

Le typage est dynamique mais il n'y a pas de restrictions d'accès : toutes les **méthodes** d'une classe sont toujours **publiques**.

Kotlin

Kotlin est le nouveau langage pour la programmation d'application Android.

Il s'agit d'un langage **POO** qui s'exécute sur la JVM. Kotlin est compatible avec Java (et y ressemble d'ailleurs au niveau de la syntaxe). Ce langage est cependant statiquement typé et la **nullabilité** est **explicite** : il faut explicitement indiquer que la valeur de la variable peut prendre la valeur nulle.

Clojure

Clojure est un **langage purement fonctionnel** et également compatible avec Java.

Exemple d'instruction sur Clojure : (réaliser l'opération 3 + 2 = 5)

Il faut saisir (+3 2) pour obtenir 5.

Il réutilise aussi beaucoup de types de Java et est **dynamiquement typé**. Toutes les données sont immuables, seules les références changent.

Ce langage est actuellement très utilisé pour le développement d'application web.

Conclusion du cours

Durant le module de conception et de programmation avancée, nous avons étudié différents concepts de programmation.

Nous avons commencé par déterminer ce qu'est un langage de programmation à proprement dit avant de se consacrer davantage sur la programmation orientée objet et les concepts liées (immuabilité, design patterns, collections…). Nous avons ainsi pu comprendre comment les objets interagissent entre eux afin de réaliser une application bien structurée et performante.

Nous avons essentiellement mis en œuvre ces concepts généraux, communs à de nombreux langages, sur Java. Ensuite, nous avons étudié plus précisément le fonctionnement de la JVM et la manière dont elle exécute les programmes multithreads, permettant de paralléliser différents traitements (développement d'un serveur avec prise en charge réseau permettant à plusieurs clients d'interagir avec lui simultanément).

Enfin, nous nous sommes intéressés aux langages modernes, en répertoriant les dernières évolutions de différents langages comme Kotlin, Java 8/9, Python, Clojure, C++ 17, etc…