Bases Java

# I/ Présentation

Tourne de la même façon sur tous les OS. Le code produit est indépendant de tout OS.

1ere version en 1995.

LA JVM connait Java, Scala, Groovy.

Différentes versions importantes : Java 2 (bases + collections) / 5 (génériques) / 8 (expressions lambdas : comme les fonctions anonymes de JS) / 9 (modules) / 13 (n = switch(p){}

Caractéristiques essentielles du langage :

* Java vient du C/C++ (on peut réutiliser la syntaxe)
* Langage objet (polymorphisme)
* Sécurité : Exceptions
* Multithread

Portabilité :

* Source : le code source permet de créer un fichier du nom de la classe : Fichier.java par exemple.
* Compilation : javac Fichier.java (commande shell comme gcc) -> résultat : Fichier.class
* Interprétation : java Fichier (commande shell) grâce à la JVM qui possède un garbage collector qui tourne en tache de fond pour la gestion de mémoire, on n’y a pas accès. Il y a aussi un chargeur de classe qui va chercher les classes de Java en local ou sur internet, cette bibliothèque de classe est pilotée grâce à classpath (paramètre -cp). Il y a aussi un vérificateur de code (permet de ne pas écrire sur le disque dur d’un client)

Se définit comme :

* Un langage
* Une API
* Commandes : javac, java, javadoc (documentation, génère du code html pour visualiser les ifnos sur une page web), jar (déploiement fabrique les librairies)

# II/ Exemples

* Affichage des premières classes :

public class Example {

public static void main(String []arg) {

int i ;

for(i=0 ; i<10 ; i++)

System.out.println(i + ‘’ ‘’ + (i\*i)) ;

La bibliothèque java.lang est déjà importée, pas besoin de import.

Package : - 1 ensemble de classes = 1 répertoire = 1 jar

* Nom package = nom de classe

Ex : java.util.Scanner scan :

Scan = new java.util.Scanner()

* package mylib ; (permet de créer un package pour que la classe du fichier devienne un package)

args[0] désigne le premier paramètre

# III/ Syntaxe et instructions

* Nommage :
  + Nom de classe : NomDeClasse (pas de chiffres)
  + Variables/fonctions : nomDeVariable / nomDeFonction
  + Constantes : NOM\_DE\_CONSTANTE
  + Package : nomdepackage.ssrep1.ssrep2
* Instructions : if/switch/for/while/dowhile/break/continue
* Fonctions :

public static type idf(types paramètres){

return ;

}

* Variables : 2 types
  + Veux du C = types primitifs : char, int, long, short, float, double
  + Interdit de déclarer des méthodes par un point sur des types primitifs :

Char c = ‘a’ ;

c.toUpperCase(c) ;

* + Types classes

X objet = null ;

objet = new X(…) ;

Exemple : int => Interger

float => Float

Integer value = 10 -> autoboxing

int i = value ; -> unboxing

Maxvalue et min value -> limites des types (constantes)

Méthode CompareTo qui existe dans toutes les classes (comme le strcmp) pour comparer 2 la taille de 2 objets.

* + Classes de chaines :

String

* + - Length, charAt, indexOf, lastIndexOf, equals, compareTo, substring, toUpperCase, toLowerCase

StringBuffer : objet partagé (pour le multithread)

StringBuilder : objet non partagé

stringBuilder sb = new StringBuilder(‘’esgi’’);

String value = sb.toString();

# Les tableaux

Tableaux statiques : []

type[] array ;

array = new type[TAILLE] ;

ex : int[] array ;

Array.length

Ex : for (i = 0 ; i < array.length ; i++) {

sop(array[i]) ;

}

for (i nt value : array) {

sop(array[i]) ;

}

Arrays.sort(array)

Tableau dynamiques : ArrayList

Collections.sort

ArrayList<Double> list ;

list = new ArrayList<Double> () ;

list.add(12.5) ;

list.size() ;

list.get(i) ;

# IV/ Exceptions

Def = mécanisme de détection d’erreurs à l’exécution à l’exécution d’un programme (POO).