Java

Les éléments de base du langage

Java : Les éléments de base du langage

Introduction au langage Java

Historique de Java

1990 Initialisation du projet chez **Sun Microsystems.**
(Langage conçu pour gérer des appareils électroniques)

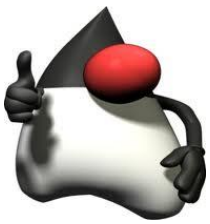
1993 L'arrivée **d'Internet** réoriente le projet.

1996 Versions successives : **Java** 1.0.2 , **Java** 1.1.5 (1997)
Java 1.2 nommée **Java 2** .
Logiciel développement appelé J.D.K.

2005 **J.D.K. 5**
2011 Java Standard Edition : **Java SE 7**

- En complément des outils disponibles avec **J**ava **D**evelopment **K**it, deux E.DI. facilitent le développement : **Eclipse** et **NetBeans.**

- **Java SE 7** est le socle de toutes les A.P.I.s Java y compris **Java EE** 7.



Caractéristiques de Java :

- Un langage résolument **objet** : Très inspiré de **C++ :** (**Encapsulation**, **Héritage**, **Interface**, ...). Certains écueils de **C++** ont disparu.

- des mécanismes d'implémentation efficaces (Typage fort, gestion des **exceptions**, multithreading, *Garbage Collector*).

- Gestion des connexions **réseaux** , connectivité des **bases de données**.

- La compilation génére du ***bytecode*** interprété par une machine virtuelle d'exécution (**JVM**) .

- Des applications autonomes (applications ***stand-alone***) ou intégrées dans des pages **xHTML** (***applets***) et des **applications d'entreprise** déployées sur un serveur .

- Une **portabilité** exceptionnelle. (Le compilateur Java génère des instructions en ***bytecode*** totalement indépendant de toute architecture particulière.)

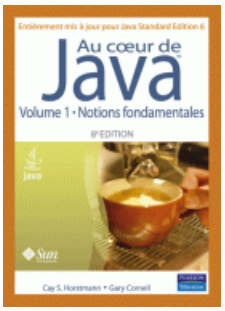
- **Allocations dynamiques** de mémoire gérées par la machine virtuelle.

- **Libération automatique** de la mémoire dès la libération de l'objet.

- La gestion des **E/S** facilitée et universelle grâce **aux flux .**

- Un langage robuste et **sécurisé**.

Bibliographie de référence :



Au cœur de Java – Notions fondamentales (Horstmann & Cornell) Editions : Campus Press (Sun Microsystems)

Au cœur de Java - Fonctions avancées (Horstmann & Cornell) Editions : Campus Press (Sun Microsystems)

Programmer en Java (Claude Delannoy)
Editions : Eyrolles.

Le livre de Java – premier langage (Anne Tasso)
Editions : Eyrolles.

Un premier programme Java

- **Visualisons le programme suivant :**

```
public class PremiereAplication {

        // Une application stand alone est représentée par une classe
        public static void  main(String[] args) {

        /* En Java, toutes les fonctions sont membres .
                On ne trouve que des méthodes.
                La classe PremiereApplication  définit une seule méthode qui
                        représente le code à exécuter
        */

        System.out.println (" Bonjour tout le monde ! " );
    }  // fin de main

} // fin de la définition de la classe
```

Nous obtenons à l'écran :

Bonjour tout le monde !

Le langage Java :

- Langage Orienté Objet .

- Les notions à l'origine de nombreuses erreurs en C++ **ont disparu en Java** . (Allocation et libération de mémoire manuelles , arithmétiques des pointeurs , plus de confusions pour le signe '=' , plus d'héritage multiple ...)

- L'unité de code est la **classe .**

- La bibliothèque d'exécution est **indépendante** de la plate-forme.
  Java est un langage "à objets".(**Toutes les fonctions sont des méthodes**)

-
-

- Une **classe** CCC se définit par la structure suivante :

```
class CCC {
    … /* variables et méthodes */
}
```

- Dans une classe, une méthode *mmm* se définit par :

```
[ modificateurs … ] typeRenvoyé mmm ( liste des paramètres ) {
. . .  /* instructions */
}
```

En résumé :

Toute **application** Java doit contenir **au moins une classe** :

- portant le **même nom** que le fichier source dans lequel elle est enregistrée (c'est elle qui lui donne son nom).

- possédant au moins une méthode :
  - nommée ***main****(…)*
  - de type ***public*** et ***static***
  - admettant un paramètre de type String []
  - renvoyant *void*

Eléments de base du langage

Les types primitifs

- Entiers avec signe :
  byte *(*8 bits), ***short*** *(*16 bits), ***int*** *(*32 bits), ***long*** (64 bits).

- Réels (représentation en virgule flottante) :
  float (32 bits), ***double*** (64 bits)

- Caractères :
  char (16 bits, **Unicode**)

- Valeurs logiques *true* et *false*
  boolean

Domaines de valeurs pour les types numériques

Type	domaine de valeurs
byte	-128 à 127
short	-32768 à 32767
int	-2147483648 à 2147483647
long	-9223372036854775808 à 9223372036854775807
float	de -1,4039846e45 à 3,40282347e+38 *(pour les valeurs positives)*
double	de -4,940856e-324 à 1,79779e+308 *(pour les valeurs positives)*

Déclaration d'une variable

Une **déclaration de variable** précise le type, l'identificateur et éventuellement une ou plusieurs valeurs d'initialisation :

type identificateur [= valeur(s)d'initialisation] ;

Un **identificateur** est une suite de caractères formée avec des lettres, des chiffres ainsi que les caractères '_' (souligné) et '$'.

Le premier caractère ne peut être un chiffre. Les minuscules sont distinguées des majuscules.

Instructions, portée d'une variable

- Instructions

Une instruction **Java** est :

• une **opération (ordre) simple** terminée par un point-virgule et pouvant être librement écrite sur plusieurs lignes :

• une **opération composée**, encore appelée **bloc**, qui rassemble des instructions encadrées par des accolades :

```
{
  instruction 1 ;
  instruction 2 ;
  … ;
  instruction n ;
}
```

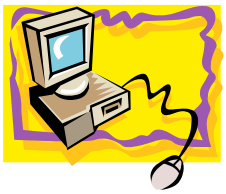
- Portée d'une variable

• La portée d'une variable, et donc son **domaine d'utilisation**, va de la ligne de sa déclaration jusqu'à la fin du bloc dans lequel elle est définie.

```
{
  int rayon  ;
  …

}
```

Structures de contrôle



Structures conditionnelles

- **if** (condition)
  instruction ;

- **if** (condition) *instruction ;*
  else *instruction ;*

- **switch** (*expressionEntière)* {
  case *Constante1 : instruction ;*
  case *Constante2 : instruction ;*
  …
  default : *instruction ;*
  }

Structure itératives

- **while** (condition) {
  instructions ;
  }

- **do** {
  instructions ;
  } **while** (condition) ;

- **for** (expression1 ; expression2 ; expression3) {
  instructions ;
  }

- Ruptures de séquence

- Le mot-clé ***break*** permet de sortir d'un ***switch*** ou d'une itération .
- Le mot-clé ***continue*** ne peut être utilisé que dans les itérations : il arrête l'exécution de l'itération et transfère le contrôle au test de sortie.

Affectation et opérateurs arithmétiques

- L'opérateur d'affectation

Le résultat d'une affectation est **typé** et renvoie la valeur de la variable affectée. On peut ainsi écrire des affectations en cascade :

```
a = b = c = d = expression ;
```

- Opérateurs d'affectation combinés

L'affectation peut être combinée avec un opérateur arithmétique pour former un opérateur combiné. Les affectations suivantes sont équivalentes :

```
x1 = x1 + 5 ;          // x1  est une variable entière
x1 += 5 ;
```

De façon générale, si ***op*** est un opérateur arithmétique et ***expG*** une expression admise à gauche de l'opérateur d'affectation , les deux écritures :

```
expG = expG  OP  expression ;
expG OP=  expression ;
```

sont équivalentes.

- Opérations arithmétiques

Les types de base ***byte***, ***short*** , ***int*** , ***long*** , ***char*** , ***float*** et ***double*** admettent comme opérateurs : + - * / % , dont les priorités et l'associativité sont celles des autres langages.

La division par zéro crée un incident (**exception**) .

- Les opérations de casting

Le **transtypage** (ou cast) impose la conversion : c'est le développeur qui assume alors le risque de perte d'information.

Tous les transtypages ne sont pas licites :

```
boolean trouve = true;
int entier = (int) trouve;
```

*// erreur de compilation : aucune conversion admise de **boolean** vers **int***

L'opérateur *cast* a une grande priorité :

```
short   i = 10;
short val = (short  i * 2;            // Erreur de compilation
short val = (short) (i * 2);          // Ok à la compilation
```

Autres opérateurs

- Incrémentation et décrémentation

Les opérateurs ++ et -- peuvent être **postfixés** ou **préfixés** :

```
int x1 =  3 ;
int x2  = x1++;                 //  x1 vaut 4 et x2 vaut 3
int x3 = ++x1 ;                 // x1 vaut 5 et x3 vaut 5
```

- Comparaisons

Les opérateurs relationnels dont le résultat est de type ***boolean*** sont les opérateurs classiques : < , <= , == , >= , > , !=

- L'opérateur ternaire

L'opérateur **ternaire ?** : a le même comportement qu'en C/C++ .

condition ? e1 : e2 prend la valeur *e1* si vrai et *e2* dans le cas contraire .

- Opérateurs logiques

Ils produisent tous un résultat ***boolean*** . Ils se notent :

!	*non*
&	*et*
|	*ou*
^	*ou exclusif*

&& : *et conditionnel* : l'évaluation de la condition est finie dès qu'elle devient fausse.

|| : *ou conditionnel* : l'évaluation de la condition est finie dès qu'elle devient vraie.

```
if ( ( n != 0 ) & ( m / n > p ) ) … // invalide si n vaut zéro
if ( ( n != 0 ) && (  m/n > p ) ) … //      valide
```

Opérations bit à bit

Les opérateurs suivants s'appliquent à des valeurs entières. Ils réalisent des opérations bit à bit et produisent des **résultats entiers**.

&	réalise un *et* bit à bit
|	réalise un *ou* bit à bit
^	réalise un *ou exclusif* bit à bit
>>	réalise un décalage à droite, en gardant le signe
<<	réalise un décalage à gauche, en gardant le signe
>>>	réalise un décalage à droite sans garder le signe

Ecriture de constantes

- Entières

décimale 125
octale 011
hexadécimale 0x15
type long 30l (ou 30L)

- Réelles

type ***float*** 1.0f 1e-5f .5f
type ***double*** -3.2e2l 1.35

- Caractères

'8' '\n' '\u0031' '\u06f1'

derrière \u : caractère **UNICODE** il faut 4 chiffres **hexadécimaux** .

- Constantes chaînes de caractères

"Texte : \"Java\"

"Liste des chiffres : 012345"

Priorité des opérateurs

Opérateurs	Associativité
[ ] . ()	gauche à droite
! ~ ++ -- + unaire - unaire	droite à gauche
new (type)	droite à gauche
* / %	gauche à droite
+ -	gauche à droite
<< >> >>>	gauche à droite
< > <= >= instanceof	gauche à droite
== !=	gauche à droite
&	gauche à droite
^	gauche à droite
|	gauche à droite
&&	gauche à droite
||	gauche à droite
? :	gauche à droite
= *= /= %= += -= &=	gauche à droite

Copyright

Chef de projet (responsable du produit de formation)

PERRACHON Chantal, DIIP Neuilly-sur-Marne

Ont participé à la conception

COULARD Michel, CFPA Evry Ris Orangis

Réalisation technique

COULARD Michel, CFPA Evry Ris Orangis

Crédit photographique/illustration

Sans objet

AFPA Février 2014

Association nationale pour la Formation Professionnelle des Adultes
13 place du Général de Gaulle – 93108 Montreuil Cedex

www.afpa.fr