Mémento Java

- Le langage
- Structure générale
- expressions
- Les tableaux
- Instructions
- Fonctions
- Classes et objets

Le langage

Le langage de programmation Java (nom dérivé de Kawa) a vu le jour en 1995. C'est un langage

- fortement typé
- orienté objet
- compilé-interprété

Le fichier source d'un programme est *compilé* en un langage intermédiaire (*byte code*) indépendant de la machine cible. Le byte code est interprétable sur toute machine possédant une *machine virtuelle Java.*

Le langage Java vient avec un ensemble important de classes prédéfinies (autour de 500) qui encapsulent des mécanismes de base comme

- des strucures de données : vecteurs, tables de hachage, grands nombres.
- des outils à la base des communications, comme les URL
- un ensemble de composants d'interfaces graphiques

Structure générale

Un programme Java est constitué

- d'un ensemble de directives d'importation comme

  ```
  import java.io.*;
  ```

- d'un ensemble de déclaration de classes.

Chaque classe contient

- des déclarations de variables;
- des déclarations de fonctions (*méthodes*)
- ces attributs sont soit
  - des attributs de classe (`static`)
  - des attributs d'objet (ou d'instance)

Les fonctions contiennent des déclarations de variables locales et des instructions.

Une fonction spéciale est appelée à l'exécution :

```
public static void main(String[] args) {...}
```

Tout identificateur a un type. Chaque fonction et chaque donnée font partie d'une classe.

- une fonction ou donnée de classe autre que la classe courante est appelée en préfixant son nom du nom de la classe.

  `Math.cos()`   `Math.PI`

- Une fonction d'objet est appelée en la préfixant du nom de l'objet.

La classe courante et l'objet courant sont sous-entendus.

Exemple:

```
System.out.println()
```

Ici `out` est une donnée statique de la classe `System`. C'est un objet de la classe `PrintStream` et `println` est une méthode d'objet.

Expressions

Types primitifs

`byte` : un octet.

`boolean` vaut `true` ou `false`

`char` caractère sur 2 octets (en unicode)

`short, int, long` entier sur sur 2, 4 ou 8 octets

`float, double`nombre réel sur 4 ou 8 octets

Une variable de ces types se déclare en donnant d'abord son type.

```
int i, j;
float re, im;
boolean termine;
```

Une variable `static` est membre de classe. Une variable `final` est une constante.

Les tableaux

Un tableau se *déclare*, se *construit* et s'*utilise*.
Déclaration :

```
int[] a  //vecteur d'entiers+
double[][] m;  // matrice de doubles
```

La création d'un tableau par `new` :

```
a = new int[N];
m = new double[N][P];
```

On s'en sert par

```
x = m[i][j];
for (i = 0; i < a.length; i++)
  System.out.print( a[i] );
```

On peut fusionner déclaration et initialisation :

```
int N = 7;
String[] jours = {"lundi", "mardi","mercredi",
 "jeudi", "vendredi", "samedi", "dimanche"};
```

Instructions

Affectation

```
x = 1; y = x = x+1;
```

Instructions conditionnelles

```
if (E) S
if (E) S else T
```

on peut aussi traiter plus de cas :

```
switch (chiffre) {
  case 0: return "zero";
  case 1: return "un";
  ...
}
```

Itérations

```
while (E) S
do S while (E)
for (E; F; G) S
```

Une instruction `break;` fait sortir du bloc où elle se trouve.

Fonctions

Chaque classe contient une suite de fonctions non emboitées. La déclaration commence par : type de la valeur nom de la fonction(type et nom des arguments)

Une fontion qui ne retourne pas de valeur a pour type de retour le type `void`.

L'instruction

```
return E;
```

donne la valeur de E comme valeur de la fonction et interrompt l'exécution.

Un même identificateur peut désigner des fonctions différentes pourvu que les signatures des arguments soient différentes.

Classes et objets

Une classe est à la fois

- la déclaration d'une suite de fonctions et de variables.
- la déclaration d'un type non primitif.

Les *constructeurs* sont des fonctions particulières permettant de créer des objets de la classe.

```
class Gauss {
  int re;
  int im;

  Gauss (int x, int y) { re = x; im = y; }

  public static void main(String[] args) {
    Gauss a = new Gauss(3,5);
    ...
  }
}
```

Les méthodes d'objet s'appliquent à l'objet appelant.

```
void imprimer() {
  System.out.println(re + " " + im);
}
```

On s'en sert dans

```
a.imprimer();
```

De manière équivalente :

```
static void simprimer(Gauss a) {
  System.out.println(a.re+ " "+ a.im);
}
```

s'utilise par

```
simprimer(a);
```

Objets et types primitifs

Une variable d'un type primitif désigne une valeur de ce type.
Une variable d'un type non primitif désigne l'emplacement où se trouve l'objet après sa création.
On dit que c'est une *référence*.

```
Gauss a; //creation d'une variable
int[] m;
a = new Gaus(); // creation de l'objet
m = new int[10];
```

A la création, une variable d'objet contient `null`.
C'est la référence vide.

Copie de références et copie d'objets.

```
Gauss b;
b = a; //copie de reference : un seul objet
```

`a` et `b` désignent le même objet. Une copie véritable doit être programmée.