Algorithmique et langage de programmation – JAVA

De bons cours complet et gratuit se trouve ici pour les plus courageux !

https://openclassrooms.com/courses/apprenez-a-programmer-en-java

La programmation orienté objet :

Une **classe** est un modèle à partir du quelle les **objets** sont créés, on regroupe des objets avec une classe lorsqu'on veut qu'ils aient les même **attributs** (Par exemple : on peut définir un classe peinture dont les objet sont des pot de peinture qui ont pour attribut leur couleur).

1 classe == 1 conteneur de méthodes (fonctions dans le monde objet)

→ Tout est défini de cette manière
→ Dans chaque classe on retrouvera :

1. Un constructeur du même nom que la classe de type **void** qu'on appel via « **new** » (voir p2)
2. La définition des attributs (leur type)
3. D'une méthode « main » si l'on veut pouvoir exécuter des lignes de commande.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("HelloWorld!");
    }
}
```

Ici on définit la classe HelloWorld :
- On ne lui donne pas d'attribut ;
- On définit un main qui ne renvoit rien (**VOID**) accessible par tous (**PUBLIC**) et exécutable sans faire appel à « new » (**STATIC**) ;
- On s'attend à avoir un tableau de texte (type **String[]**) en entrée et quelque soit celle-ci on va imprimer « Hello World ! ».

Caractéristiques de JAVA :

- langage déclaratif : déclaration des variables avant leur utilisation
- typage fort : nécessitant des conversions de types
- types primitifs non classes
- tailles et espaces de valeurs définis par la machine virtuelle Java

Taille et espace de valeurs des Types Primitifs

- types entiers :
  - **byte** : 1 octet (-128 à +127),
  - **short** : 2 octets (-32 768 à 32 767),
  - **int** : 4 octets environ($-2 * 109$ à $2 * 109$)
  - **long** : 8 octets environ ($-9 * 1018$à$9 * 1018$)
- types à virgule flottante : norme IEEE754-1985
  - **float** : 4 octets avec 24 bits pour la mantisse et 8 pour l'exposant (1.4E-45, à 3.4E38)
  - **double** : 8 octets avec 53 bits de mantisse et 11 pour l'exposant
- **boolean** (true/false),
- **char** (2 octets)

Instructions et instructions de contrôle

- Instructions de contrôle : **if, switch, for, while, do while**
- Opérateurs : tous les opérateurs habituels par exemple, **+ - / *,**
- Affectation =
- Blocs, instructions simples : **{}, « ; »**

Tests en Java

- **if**(*expression booléene*) instruction
- **if**(*expression booléene*) instruction **else** instruction
- switch (Case en shell)

```
switch(choix){
        case 1: System.out.println("choix␣1−");break;
        case 2: System.out.println("choix␣2−");break;
}
```

Constructeurs en Java

- Méthode particulière :
  - Nom identique à la classe
  - Aucun type de retour
  - Appelé directement à travers **new**
- Si aucun constructeur n'est spécifié, le compilateur en fabrique un par défaut qui initialise les attributs à 0 (correspondant à leur type)
- La surcharge des méthodes (overloading) permet à une classe d'avoir plusieurs constructeurs qui diffèrent par le nombre et le type de leurs arguments

Visibilité des méthodes (Il faut protéger les attributs)

- Les méthodes publiques (**public**) d'une classe définissent son interface publique
- Les méthodes privées (**private**), ne sont accessibles que depuis l'intérieur de la classe
- Les méthodes protégées (**protected**), sont accessibles depuis l'intérieur de la classe et de l'intérieur de ses classes dérivées

On peut définir des sous classes, celles-ci hérite des attributs de la classe parent. Cela permet de séparer les objets en sous groupe qui diffèrent par certains de leurs attributs.

- Une classe ne peut hériter que d'une autre classe (héritage simple)
- Une classe hérite d'une autre par l'utilisation du mot réservé extends
- Le mot réservé final utilisé devant le mot clé class interdit toute spécialisation de la classe sur laquelle il est utilisé (On ne peut plus le modifier !)
- Appel dans le constructeur de la classe dérivée d'un des constructeurs de la classe parente par utilisation du mot réservé **super()**
  - ♦ En première ligne du constructeur de la classe enfant
  - ♦ Si aucun appel ⇒ appel au constructeur sans argument de la classe parente

Si on veut relever des exceptions et renvoyer quelque chose on utilise **throw** :

exemple :

```
private static void positif (final double val) throws NumException {
        if (val<0.0) throw new NumException (val,"Valeur positive attendue.");
        return;
 }
```