Définir et Instancier une classe en Java

Déclaration et Implémentation d'une classe

En algorithmique (comme en C++ la plupart du temps), l'écriture du corps des méthodes (implémentation) se fait après la déclaration de la classe. En java au contraire, l'implémentation et la déclaration de la classe ne peuvent pas être séparées.

Etudions la syntaxe de la définition d'une classe en Java à travers l'exemple compte:

```
class Compte
{
   //attributs
   private int numero;
   private String nom;
   private double solde;

   //définition des méthodes
   public void Init(int UnNumero, String UnNom)
   {
         numero = UnNumero;
         nom = unNom;
         solde = 0;
   }
   public void Crediter(double Montant)
   {
         solde = solde + Montant;
   }
   public void Debiter(double Montant)
   {
         solde = solde – Montant;
   }
   public double GetSolde( )
   {
         return solde;
   }
}        //fin de la classe
```

Chaque membre (attribut ou méthode) est précédé par un modificateur d'accès private ou public

- **private** veut dire que le membre est encapsulé, inaccessible de l'extérieur de l'objet
- **public** veut dire que le membre fait partie de l'interface , accessible de l'extérieur

(Remarque: il existe d'autres modificateurs que l'on verra plus tard)

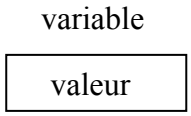
Passage de paramètres:
en Java, les variables de type primitif sont toujours passées par valeur et les objets sont toujours passés par référence. Il n'y a pas de signe indiquant la nature du passage de paramètre.

Gestion des variables simples et des objets

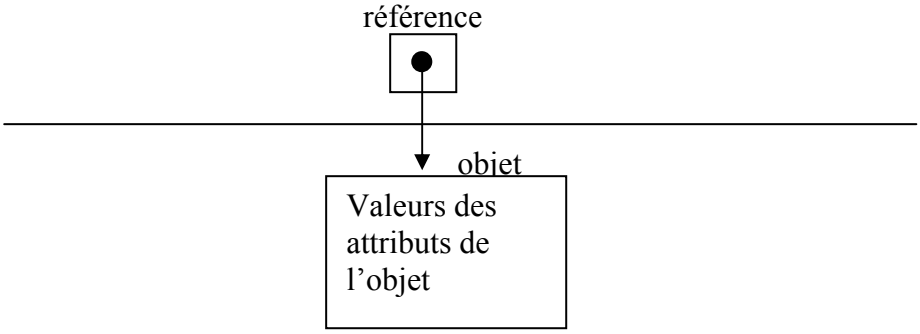
En java, il existe trois types de variables :

- les variables primitives qu'on qualifiera « simples » qui sont de l'un des 8 types primitif (char, short, int, byte, float, long, double, boolean)
- les objets, c'est-à-dire les variables dont le type est une classe
- les références aux objets, qui sont des sortes de pointeurs

Les variables simples contiennent directement leur valeur. Elles sont déclarées simplement dans la pile. (Remarque: en Java, on ne peut pas construire de variables simples dans le tas.)



Les objets en revanche sont contenu dans le tas (impossible de les déclarer dans la pile). Ils ne sont pas déclarés mais alloués dynamiquement. Pour y accéder, il faut passer par un pointeur spécial qu'on appelle référence. Une référence est donc un pointeur qui désigne un objet et qui permet de le manipuler. (Remarque : il est impossible de définir une référence à une variable simple ou à une autre référence, contrairement au C++). Une référence est déclarée dans la pile.



Au niveau de leur manipulation, références et pointeurs sont totalement différents. Utiliser une référence est bien plus simple que d'utiliser un pointeur. En fait, une fois l'instanciation réalisée, **une référence peut quasiment être confondue avec l'objet qu'elle désigne.**

- on ne peut pas accéder à l'adresse mémorisée dans une référence (pas d'opérateur &, pas d'arithmétique sur les pointeurs)
- pour accéder à l'objet référencé, il n'y a pas de déréférencement à réaliser (pas d'opérateur * ni ->). L'accès à l'objet est direct par sa référence.

C'est une des raisons majeures pour lesquelles Java est plus simple que C++.

Instanciation

Rappelons qu'un objet est une instance de classe et que l'instanciation consiste à créer un objet à partir d'une classe (comme faire un gâteau à partir d'un moule). Une classe peut bien entendu servir à créer plusieurs instances (objets).

Comme les objets sont contenus dans le tas, on ne peut pas déclarer un objet comme on déclare une variable simple. **Un objet en Java doit toujours être créé dynamiquement avec l'opérateur new**. Il faut d'abord déclarer une référence à un objet du type voulu puis créer (construire) l'objet grâce à l'opérateur new, comme en C++.

Ex:

Soit une classe Personne
Voilà comment créer un objet toto de type Personne.

```
Personne toto;        //toto est une référence sur un objet de classe Personne
toto = new Personne( );     //un nouvel objet de type Personne est créé
                            //et l'adresse du nouvel objet est affectée à toto
```

On pourrait simplifier l'écriture en une seule ligne :

```
Personne toto = new Personne( );
```

En C++, on déclarerait un pointeur pour créer dynamiquement un objet:

```
Personne * toto;
toto = new Personne( );              //ou en une ligne : Personne* toto = new Personne( ) ;
```

Voyez la grande ressemblance entre les deux écritures: la seule différence c'est qu'en C++, la variable toto est précédée d'une * pour spécifier que c'est un pointeur sur une Personne. Et ensuite, en C++, il faudrait déréférencer le pointeur (par * ou ->) afin d'accéder à l'objet. Rien de tel en Java.

En Java, il n'y a pas d'opérateur spécial pour signifier que la variable déclarée est une référence. C'est le compilateur qui reconnaît s'il s'agit d'une variable simple ou d'une référence :

- si le type est primitif, la variable déclarée est « simple » et contenue dans la pile
- si le type est une classe, automatiquement la variable déclarée est une référence sur un objet de la classe et il faudra ensuite construire un objet qui sera désigné par cette référence.

Nous avons dit qu'une fois instanciés, un objet peut être confondu avec sa référence. En réalité, il faut toujours avoir en tête que ce que l'on manipule, ce sont des références à des objets et pas directement les objets. Ainsi il faut se méfier, notamment lorsqu'on veut copier un objet dans un autre objet. Si on réalise une simple affectation, on ne va pas créer une copie mais on va faire pointer une autre référence sur le même objet!

Ex:

```
Compte UnCompte;
Compte UnAutreCompte;
UnCompte = new Compte;
UnAutreCompte = UnCompte;
```

Les constructeurs

Rappel:
Un constructeur est une méthode particulière qui est appelée automatiquement à la création d'un objet et qui permet d'initialiser correctement cet objet.

En java, les objets ne peuvent être crées que par allocation dynamique. Le constructeur est appelé lors de l'allocation de l'objet par new (et non lors de la déclaration de la référence).

Un constructeur se reconnaît facilement en Java:

- il **porte le même nom que la classe**
- il **n'a pas de type de retour (même pas void)**

On peut évidemment définir plusieurs constructeurs pour une même classe, s'ils ont une signature différente.

Modification de la classe compte : la méthode Init est remplacée par deux constructeurs:

```
class Compte
{
   //attributs
   private int numero;
   private String nom;
   private double solde;

   //définition des méthodes
   public Compte(int UnNumero, String UnNom)          //premier constructeur
   {
         numero = UnNumero;
         nom = unNom;
         solde = 0;
   }

   public Compte (int UnNumero, String UnNom, double SoldeInitial)   //deuxième constructeur
   {
         numero = UnNumero;
         nom = unNom;
         solde = SoldeInitial;
   }
   public void Crediter(double Montant)
   {
         solde = solde + Montant;
   }
   public void Debiter(double Montant)
   {
         solde = solde – Montant;
   }
   public double GetSolde( )
   {
         return solde;
   }
}        //fin de la classe
```

Pour créer un compte, on a deux manières

```
Compte cpte1 = new Compte(135, "Dupont");        //instanciation avec appel du premier constructeur
Compte cpte2 = new Compte(159, "Martin", 1500);        //instanciation avec appel du 2° constructeur
```

Si aucun constructeur n'est défini explicitement, le compilateur en créé un quand même. Ce constructeur défini par défaut ne comporte pas de paramètres et initialise tous les attributs à leur valeur nulle (0 pour les variables numériques, NULL pour les références, "" pour les chaînes de caractère, …).
Mais dès lors qu'un constructeur est défini explicitement, le compilateur n'en créé pas d'autre. L'utilisateur est obligé, à l'instanciation, de passer les paramètres requis par l'un des constructeurs explicite. Mais le concepteur de la classe peut créer son propre constructeur par défaut (c'est-à-dire sans paramètres).

Même lorsque l'instanciation ne comporte aucun paramètre (appel du constructeur par défaut s'il existe), il faut mettre les parenthèses.

Ajoutons un constructeur par défaut à la classe Compte

```
…
 public Compte()          // constructeur par défaut
   {
         numero = 0;
         nom = "inconnu";
         solde = 0;
   }
…
```

On peut alors instancier un objet de type Compte de la façon suivante:

```
Compte un_compte = new Compte( );
```

Instanciation d'un objet =
déclaration de la référence + allocation en mémoire + construction (appel d'un constructeur)