Les Variables : projet *Choco*

Introduction

Il est facile de concevoir ce que sont les variables. Prenons comme exemple des boites de chocolat. Ces boite sont remplies avec plusieurs catégories de chocolats : les Truffes, les MonChéri, les Cœurs et les Mokas. Ce sont les 4 pauvres variétés de chocolats que vend le chocolatier. Vous pouvez choisir par exemple 8 Truffes et 20 MonChéri dans votre première boite et 4 Cœurs, 10 Mokas et 14 MonChéri dans votre seconde boite. Imaginons maintenant en programmation : on crée des objets *boite* (cf. chapitre sur la POO pour les objets) avec pour *variables* : Truffes, MonCheri, Cœurs, Mokas. Dans le cas de la première boite, on va associer la valeur 8 à la *variable* Truffe, et la valeur 20 à la *variable* MonCheri.

Types de variables

Il y a différents types de variables en Java, qui ne stockent pas les mêmes valeurs. Regardez le tableau ci-dessous regroupant les variables les plus utilisées :

NOM	PLACE EN BITS	VALEURS STOCKEES	COMMENTAIRES
byte	8	De –128 à 127	Vous ne l'utiliserez pas souvent
short	16	-32 768 à 32 767	
int	32	$-2x10^9$ à $2x10^9$	Appelée INTEGER (entier en français), très utilisée
long	64	$-9x10^{18}$ à $9x10^{18}$	
boolean	1	booléennes	Valeurs binaires : *true* ou *false*
String		chaînes	Considéré à ce stade comme une variable bien que cela n'en soit pas une. Ne pas oublier la majuscule.
BigInteger	+ de 64	?	Importer java.math.BigInteger
double			Pratique pour les décimales. Très utilisé.
float		Virgule flottante	Pi, e …

Note : vous pouvez créer vos propres types de variables, nous verrons plus tard…

Déclaration de variables

La création d'une variables est appelée « déclaration ». On va donc déclarer nos variables en chocolat, elles stockeront le nombre de chocolats de chaque type contenus dans une boite. Elles seront du type *integer*.

- int Truffes ;
- int MonCheri ;
- int Moka ;
- int Cœur ;

On peut également déclarer toutes ces variables d'un seul coup car elles appartiennent au même type, c'est à dire les entiers (*integer)*

- int Truffes, MonCheri, Moka, Cœur ;

Initialisation des variables

Initialiser une variable signifie lui donner une valeur de départ (cette valeur peut changer au cours du programme, normal, c'est variable !). Prenons le cas de la première boite de chocolats :

- int Truffes = 8;
- int MonCheri = 20;

Note : ceci déclare les variables avant de las initialiser. Ces lignes regroupent donc chacune deux opération en une seule.

Opérateurs d'affectation

On a une variable x, déclarée avec une valeur bien précise. On veut lui ajouter la valeur de y (qui est aussi déclarée et initialisée). On écrit alors : *x=x+y*. Et x prend alors une nouvelle valeur (son ancienne ajoutée à celle de y, qui, elle ne change pas). On peur aussi multiplier, diviser… :

OPERATION A REALISER	OPERATION DEVELOPPEE	OPERATION SIMPLIFIEE
Addition	X = x + y	X += y
Soustraction	X = x – y	X -= y
Multiplication	X = x * y	X *= y
Division	X = x / y	X /= y

Incrémentation et décrémentation

Incrémenter est augmenter de 1 la valeur d'une variable et décrémenter, diminuer de 1. Ces procédés sont particulièrement utiles dans les *boucles*. (cf. chapitre sur les boucles). On incrémente à l'aide de ++ et on décrémente à l'aide de --.

```
int x = 7 ;      // déclaration de x, initialisée avec 7 pour valeur
x = x++ ;      // on incrémente x, x a maintenant le valeur 8
```

Comparaison de variables

Le titre est assez parlant… on regarde si deux variables sont égales, différentes, si la première est supérieure à le seconde, si la seconde est inférieure ou égale à la première…

OPERATEUR	DESCRIPTION
==	Egal à
!=	Différent de
<	Inférieur à
>	Supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à

Note : ceci ne fonctionne qu'avec les variables et pas avec les objets (dans ce cas le programme compare si c'est le même objet, pas s'il a la même valeur). Cela ne fonctionne donc pas avec *String* qui est un objet…

Les Constantes

Les constantes sont des variables constantes (!) (si, si, je vous jure…). En fait elles sont utilisées pour les mêmes choses que les variables ; stocker des valeurs, mais une fois initialisées, la valeur ne peut plus changer… Elles sont déclarées avec le mot clé *final*.

```
final float pi = 3.14159 ;
```

Vous n'allez quand même pas essayer de changer la valeur de pi ? ;-)

Blocs et portée

En Java, les programmes sont très structurés. Ils sont organisée en blocs. Par exemple tout ce qui est contenu dans la méthode *main()* est un seul et même bloc (souvent subdivisé en autres petits blocs). De même, les instructions de la méthode *paint()* d'une applet sont dans un seul et même bloc. Pour les reconnaître : ils commencent par une accolade ouvrante { et finissant par une fermante }.
Les variables classiques ont une portée dans les bloc qui les contient. Si vous faites référence dans un bloc à une variable déclarée dans un autre bloc, le compilateur génèrera une erreur. (ne dites pas « bug » pour l'instant…il y a une grosse différence entre un bug et l'erreur décrite ci-dessus). Mais il est possible en Java de créer des

variables spéciales qui ont pour portée tout le programme (ou la *classe*, c'est un synonyme de *programme*). Ces variables bien pratiques sont appelées *variables de classe*. On les déclare à l'aide du mot clé *static*.

- static int nombreDeBilles = 200 ; // ce garçon a beaucoup de billes !
- static String surnom = « Drizzt » ; /* comme il s'appelle Djkelimbfastlichtz, on le surnomme Drizzt, la valeur de type String (chaîne de caractères) de la variable «surnom» est donc Drizzt*/

Application : le programme Chocos

Le programme Chocos va déclarer plusieurs variables. Les valeurs de ces variables vont changer puis affichées sur le périphérique de sortie, ligne de commande.
On utilisera : System.out.println() pour afficher la valeur des variables.

```
class Chocos { // début du programme
   static String mangeur = "Gargantua"; //variable statique. portée = tout le programme


   public static void main(String args[]) { // début de la méthode main()

     int nombreDeChocolats = 40; //variable non-statique. portée = ce bloc main()

     System.out.println(mangeur + " a " + nombreDeChocolats + " chocolats");
     System.out.println("Il en mange 3 d'un coup");

     nombreDeChocolats = nombreDeChocolats - 3; // changement de valeur de la variable
     System.out.println("Il en reste:"+nombreDeChocolats);
     System.out.println("Il renverse la boite, 15 sont perdus");

     nombreDeChocolats = nombreDeChocolats - 15;
     System.out.println("Il en reste:"+nombreDeChocolats);
     System.out.println("Pour se venger," + mangeur + " finit la boite");

     nombreDeChocolats = nombreDeChocolats - 22;
     System.out.println("Il en reste:"+nombreDeChocolats);
     System.out.println("Mais " + mangeur + " a encore une boite...");
  }
}
```