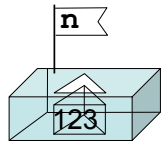
Les variables

Une variable possède 3 caractéristiques:

- Son **identificateur**, qui est le nom par lequel la donnée est désignée;
- Son **type**, qui définit de quel « genre » est la donnée contenue dans la variable;
- Sa **valeur**. Par exemple, si la donnée est un nombre, sa valeur pourra être 123 ou 3.14




```
class ExempleVariable
{
  public static void main(String[] args)
  {
    int n = 4;
    int nCarre;

    nCarre = n * n;

    System.out.println("La variable n contient " + n);
    System.out.println("Le carre de " + n + " est " + nCarre + ".");
    System.out.println("Le double de n est " + 2 * n + ".");
  }
}
```

Déclarations de variables

Les lignes:

type de la variable

identificateur, ou nom, de la variable.
Il est choisi par le programmeur et permet de référer la variable créée

```
int n = 4;
int nCarre;
```

sont des **déclarations de variables**.

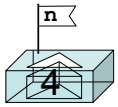
Une déclaration de variable permet de créer une variable.

Initialisation

En même temps qu'elle est déclarée, une variable peut être initialisée, c'est-à-dire lui donner une valeur avant de l'utiliser.

La ligne:

```
int n = 4;
```



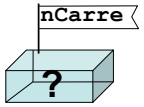
déclare donc une variable appelée `n` et lui donne la valeur 4.

Initialisation

Une variable non initialisée ne peut être utilisée.

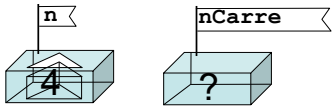
Par exemple:

```
int nCarre;
System.out.println(nCarre);
```



donne une erreur à la compilation:

```
variable nCarre might not have been initialized
   System.out.println(nCarre);
```

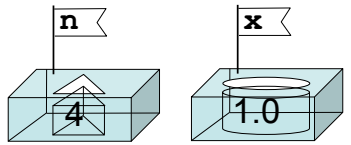



```
int n = 4;
int nCarre;
```

type pour les valeurs entières: la variable `n` ne peut contenir que des valeurs entières

```
int n = 4;
double x = 1.0;
```

type pour les valeurs décimales: la variable `x` ne peut contenir que des valeurs décimales



Déclaration de variables

De façon générale, une déclaration de variable suit le schéma:

```
type identificateur = valeur_initiale;
```

ou éventuellement:

```
type identificateur;
```

N'oubliez pas le point-virgule ; à la fin

Une fois défini, le type de la variable ne peut plus changer.

Déclaration de variables

D'autres exemples de déclaration:

```
int m = 1;
int p = 1, q = 0;
double x = 0.1, y;
```

on peut déclarer plusieurs variables simultanément.
Ne pas en abuser

Noms de variables

Règles pour nommer les variables:

- Le nom peut être constitué uniquement de lettres, de chiffres, et des deux seuls symboles autorisés: _ (*underscore*) et `$`. Pas d'espace !
- Le premier caractère est nécessairement une lettre ou un symbole;
- Le nom ne doit pas être un mot-clé réservé par le langage Java;
- Les majuscules et les minuscules sont autorisées mais ne sont pas équivalentes. Les noms `ligne` et `Ligne` désignent deux variables différentes.

Exemples de noms valides:

`nCarreTotal` `sousTotal98`

Exemples de noms invalides:

`n carre` Contient des espaces;
`n-carre` Contient le symbole - (moins);
`1element` Commence par un chiffre.

Conventions en Java pour les noms de variables

En Java, bien que ce ne soit pas requis par le compilateur, la convention est d'écrire le nom des variables en commençant par une minuscule, et commencer les mots suivants par une majuscule.

Par exemple, on utilisera

`nombreDePoints`

plutôt que

`NombreDePoints`

ou

`nombre_de_points`

Types de variables

Les principaux types élémentaires sont:

- `int`, pour les valeurs entières (pour *integer*, entiers en anglais);
- `double`, pour les nombres à virgule, par exemple 0.5

et aussi:

- `float`: aussi pour les nombres à virgule, moins précises mais occupant moins de mémoire que les `double`s;
- `char`: pour les caractères (A..Z etc.);
- ...

Affectations

La ligne:

```
nCarre = n * n;
```

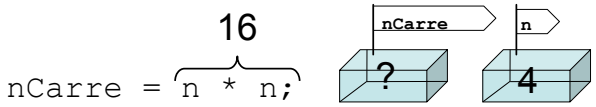
est une **affectation**.

Attention, ce **n'est pas** une égalité mathématique: Une affectation est une instruction qui permet de **changer** une valeur à une variable.

Affectations

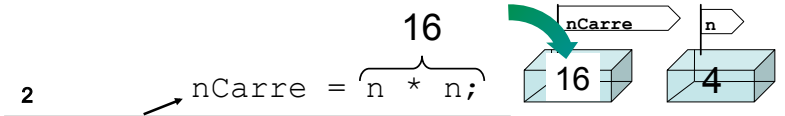
L'exécution d'une affectation se décompose en deux temps :

1 L'expression à droite du signe = est évaluée:
`n * n` est évaluée avec la valeur de `n` au moment de l'exécution.
L'étoile `*` représente la multiplication, `n * n` vaut donc 4 × 4 = 16



Affectations

L'exécution d'une affectation se décompose en deux temps :



2 la valeur de l'expression est stockée dans la variable à gauche du signe =
L'ancienne valeur de `nCarre` est perdue.

Affectations

De façon plus générale, une affectation suit le schéma:

```
nom_de_variable = expression;
```

N'oubliez pas le point-virgule `;` à la fin

Une expression calcule une valeur, qui doit être de même type que la variable.

Exemples d'expression:

- `4`
- `n * n`
- `n * (n + 1) + 3 * n - 2`

Nous reviendrons sur les expressions un peu plus loin.

Attention: Ne confondez pas une affectation avec une égalité mathématique.

Toutes les deux utilisent le signe égal `=`, mais l'affectation est un mécanisme dynamique.

Par exemple, les deux instructions:

```
a = b;
b = a;
```

copie la valeur de `b` dans `a`

copie la valeur de `a` dans `b`

ne sont pas identiques.

En mathématiques:

$$b = a + 1$$

signifie que tout au long des calculs, a et b vérifieront toujours cette relation. Autrement dit, quel que soit a, b sera toujours égal à $a+1$

En Java:

```
a = 5;
b = a + 1;
a = 2;
```

donne à `b` la valeur de `a+1`, c'est-à-dire 6.

donne à `a` la valeur 2, sans que `b` ne soit modifiée!
`b` contient donc toujours 6.

On peut écrire aussi des affectations telles que:

```
a = a + 1;
```

Ce type d'affectation, où une variable apparaît de chaque côté du signe `=` permet de résoudre de nombreux problèmes.

Cette affectation signifie:

« calculer l'expression de `a + 1` et ranger le résultat dans `a`. Cela revient à augmenter de `1` la valeur de `a` »

Nous reviendrons sur ce point dans la suite.

Déclaration de constantes

Il peut arriver que la valeur d'une variable ne doive pas changer après l'initialisation. Dans ce cas, il faut ajouter le mot-clé `final` devant la déclaration:

```
final type identificateur = valeur_initiale;
```

Par exemple:

```
final double VITESSE_DE_LA_LUMIERE = 299792.458;
```

Dans ce cas, on ne peut plus modifier la variable:

```
VITESSE_DE_LA_LUMIERE = 100; // erreur !
```