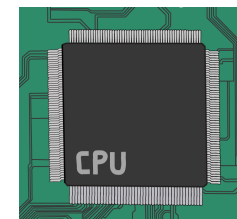
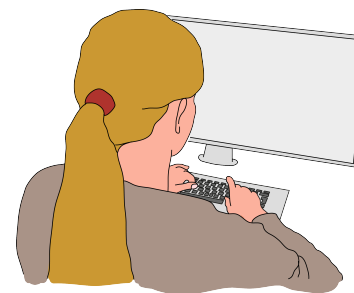
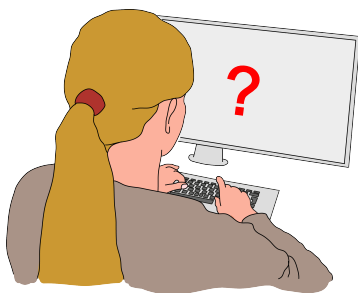
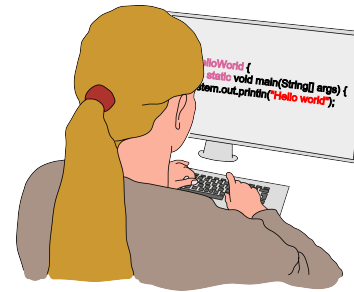
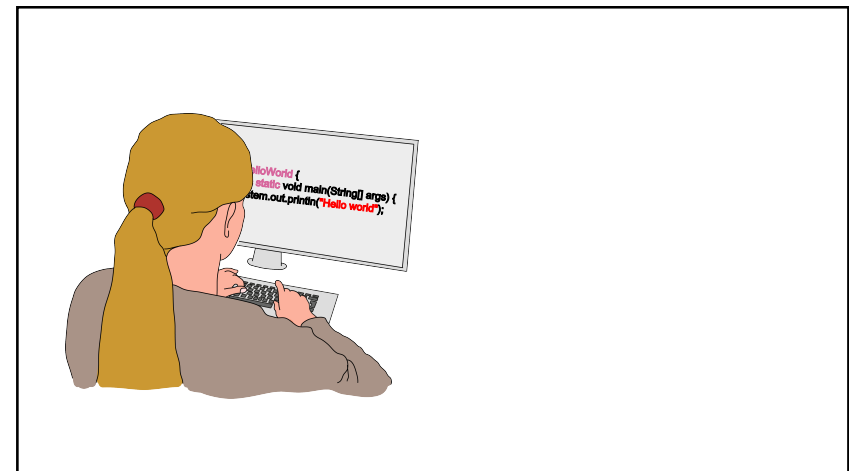
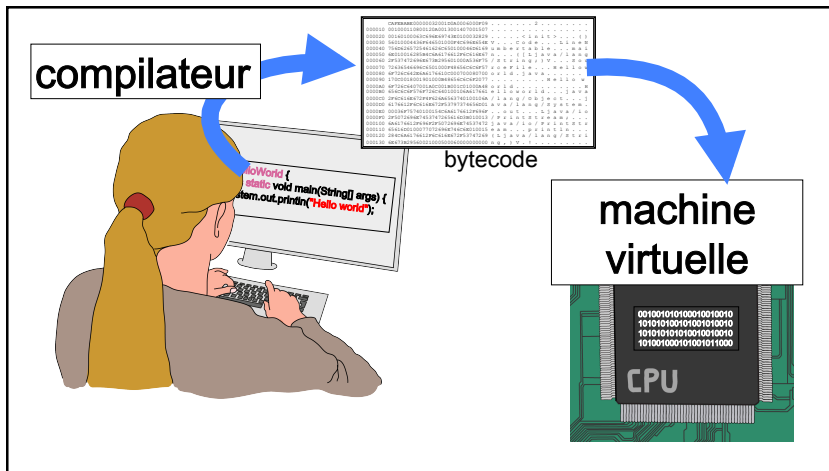
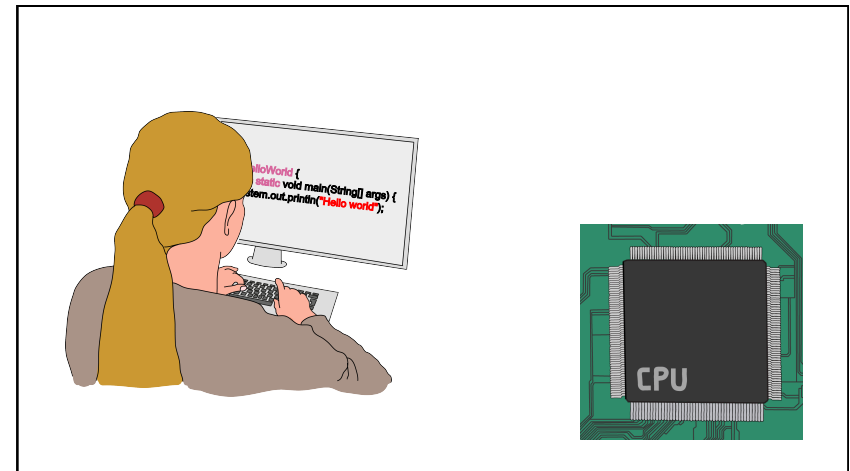
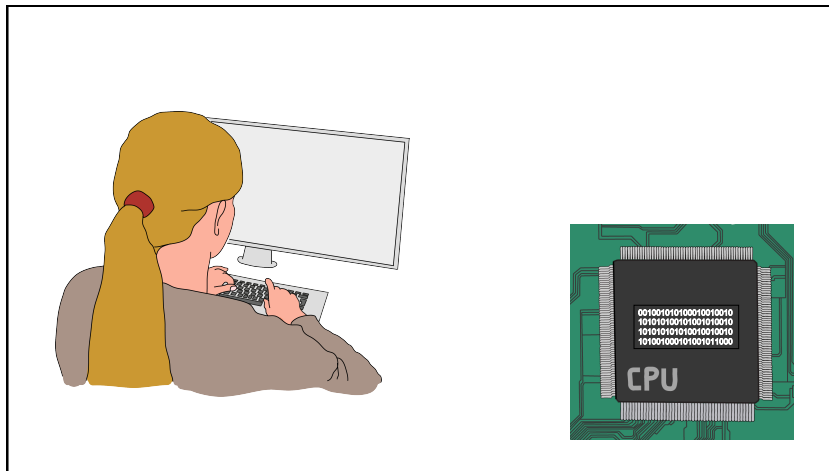
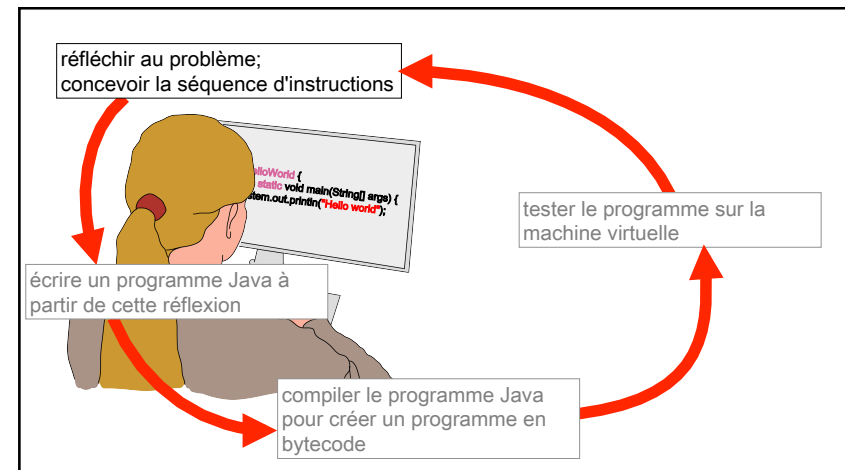
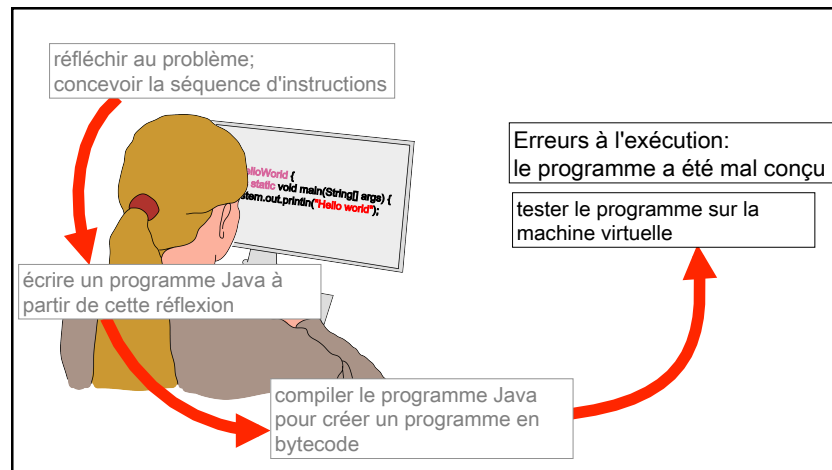
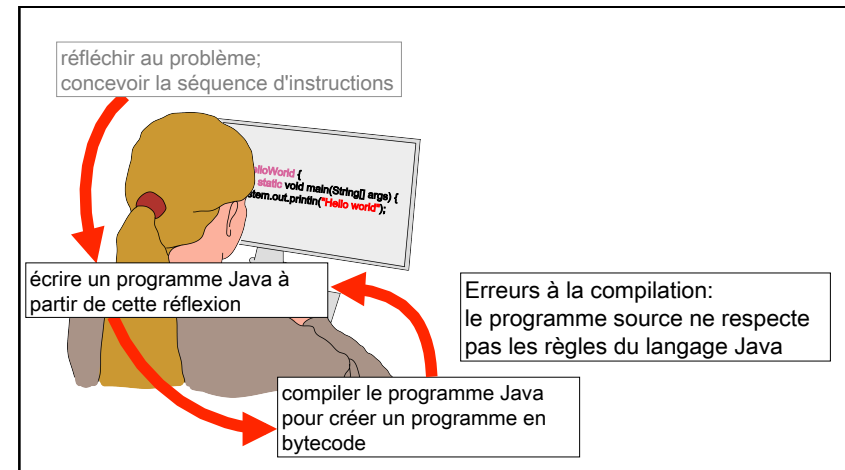
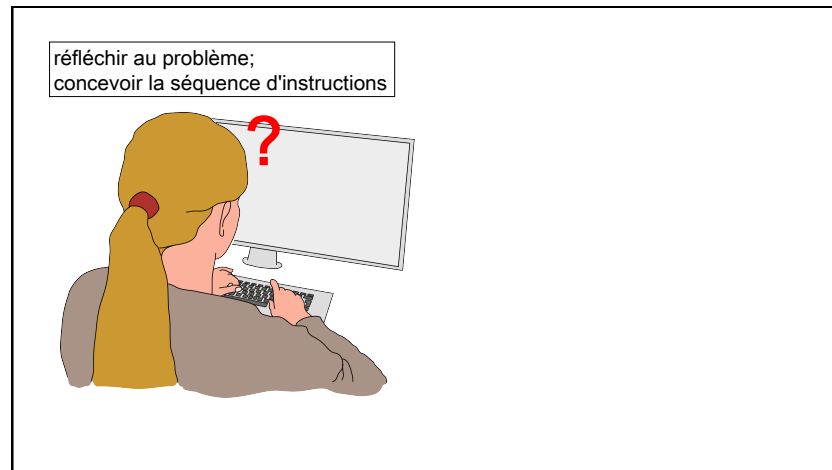


Introduction



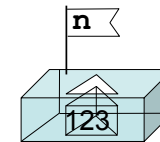




Les variables

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**.

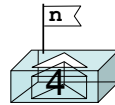
Une déclaration de variable permet de créer une variable.

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;
```



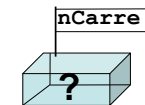
déclare donc une variable appelée `n` et lui donne la valeur 4.

Initialisation

Une variable non initialisée ne peut être utilisée.

Par exemple:

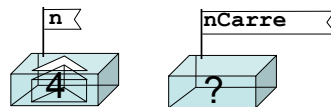
```
int nCarre;  
System.out.println(nCarre);
```



donne une erreur à la compilation:

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

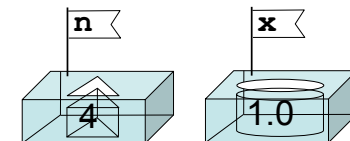
```
int n = 4;  
int nCarre;
```



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

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 `doubles`;
- `char`: pour les caractères (A..Z etc.);
- ...

Affectations

La ligne:

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

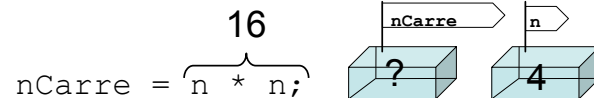
est une **affectation**.

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 $4 * 4 = 16$

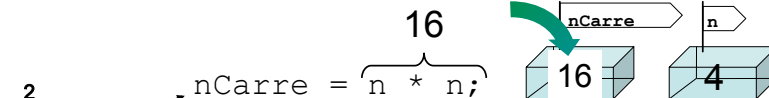


Affectations

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;` ← copie la valeur de b dans a
`b = 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; ← donne à b la valeur de a+1, c'est-à-dire 6.  
a = 2; ← 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 !
```

Les variables : lecture / écriture

Pour écrire à l'écran

fonction d'affichage
sur le terminal

affiche la *valeur* de la variable n
(et non pas la lettre n)

ce qui est entre guillemets
(") est affiché littéralement

```
System.out.println("La variable n contient " + n + ".");
```

affiche:

La variable n contient 4.

les différents éléments à
afficher doivent être
séparés par le symbole +

Pour écrire à l'écran

fonction d'affichage
sur le terminal

affiche la *valeur* de la variable n
(et non pas la lettre n)

ce qui est entre guillemets
(") est affiché littéralement

```
System.out.println("La variable n contient " + n + ".");
```

la fonction `System.out.println` fait un "retour à la ligne": le prochain affichage se fera sur la ligne suivante de la fenêtre Terminal.

Il existe une variante qui ne fait pas de retour à la ligne: `System.out.print`

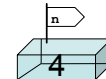
On peut afficher uniquement la valeur d'une variable en faisant simplement:

```
System.out.println(n);
```

On peut aussi afficher la valeur d'une expression:

```
System.out.println("Le double de " + n + " est " + 2 * n + ".");
```

expression



Attention, si on veut afficher une somme:

```
System.out.println("n plus nCarre = " + (n + nCarre));
```

il faut ajouter des parenthèses pour éviter une ambiguïté.

Déroulement du programme pas-à-pas

```
→ 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 + ".");
```

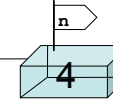
Ce qui s'affiche :

|

```
→ 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 + ".");
```

Ce qui s'affiche :

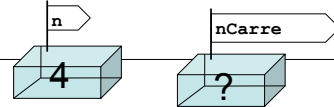
|



```
→ 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 + ".");
```

Ce qui s'affiche :

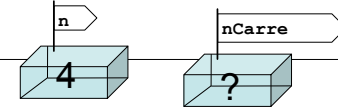
|

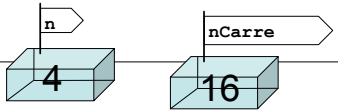


```
→ int n = 4;  
   int nCarre;  
  
   nCarre = n * n;  
  
   System.out.println("La variable n contient " + n);  
   System.out.println("Le carre de " + n + " est " + nCarre + ".");  
   System.out.println("Le double de n est " + 2 * n + ".");
```

Ce qui s'affiche :

|





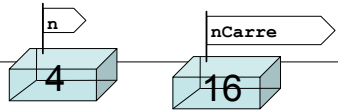
```
int n = 4;
int nCarre;

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 + ".");
```

Ce qui s'affiche :

|



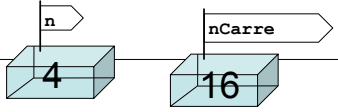
```
int n = 4;
int nCarre;

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 + ".");
```

Ce qui s'affiche :

|



```
int n = 4;
int nCarre;

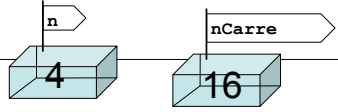
nCarre = n * n;

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 + ".");
```

Ce qui s'affiche :

La variable n contient 4.

|



```
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 + ".");
```

Ce qui s'affiche :

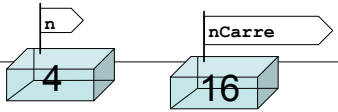
La variable n contient 4.
Le carre de 4 est 16.

|

```
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 + ".");
```



Ce qui s'affiche :

La variable n contient 4.
 Le carre de 4 est 16.
 Le double de n est 8.

Qu'affiche ce programme ?

```
int a = 2;
int b = 1;

b = a * (b + 2);

System.out.println(a + ", " + b);
```

A: a, b

B: 1, 2

C: 2, 1

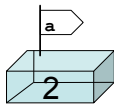
D: 2, 6

?

```
→ int a = 2;
int b = 1;

b = a * (b + 2);

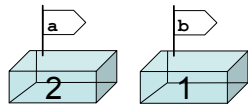
System.out.println(a + ", " + b);
```



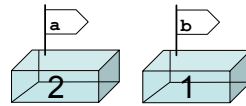
```
→ int a = 2;
int b = 1;

b = a * (b + 2);

System.out.println(a + ", " + b);
```

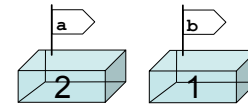


```
int a = 2;
int b = 1;
→ b = a * (b + 2);
System.out.println(a + ", " + b);
```



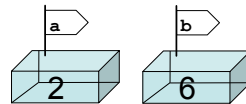
```
int a = 2;
int b = 1;
→ b = a * (b + 2);
System.out.println(a + ", " + b);
```

2 * (1 + 2) = 2 * 3 = 6

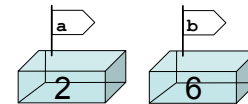


```
int a = 2;
int b = 1;
→ b = a * (b + 2);
System.out.println(a + ", " + b);
```

2 * (1 + 2) = 2 * 3 = 6



```
int a = 2;
int b = 1;
→ b = a * (b + 2);
System.out.println(a + ", " + b);
```



Affiche:
2, 6

Qu'affiche ce programme ?

```
int a = 5;  
int b = a + 3;  
  
a = 1;  
  
System.out.println(a + ", " + b);
```

A: 5, 4

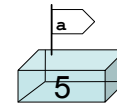
B: 1, 1

C: 1, 4

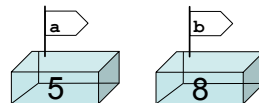
D: 1, 8

?

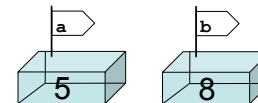
```
int a = 5;  
int b = a + 3;  
  
a = 1;  
  
System.out.println(a + ", " + b);
```



```
int a = 5;  
int b = a + 3;  
a = 1; 5 + 3 = 8  
  
System.out.println(a + ", " + b);
```



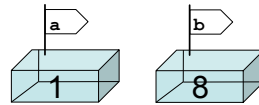
```
int a = 5;  
int b = a + 3;  
  
a = 1;  
  
System.out.println(a + ", " + b);
```



```
int a = 5;
int b = a + 3;

a = 1;

System.out.println(a + ", " + b);
```

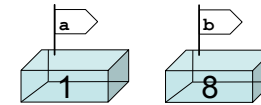


```
int a = 5;
int b = a + 3;

a = 1;

System.out.println(a + ", " + b);
```

Affiche:
1, 8



Qu'affiche ce programme ?

```
int a = 1;
int b = 2;

a = b;
b = a;

System.out.println(a + ", " + b);
```

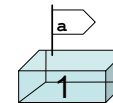
- A: 1, 1
- B: 1, 2
- C: 2, 1
- D: 2, 2

?

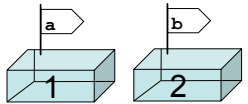
```
int a = 1;
int b = 2;

a = b;
b = a;

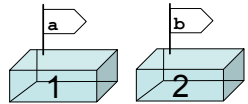
System.out.println(a + ", " + b);
```



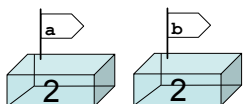

```
int a = 1;  
→ int b = 2;  
  
a = b;  
b = a;  
  
System.out.println(a + ", " + b);
```



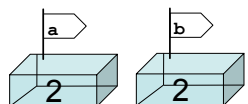
```
int a = 1;  
int b = 2;  
  
→ a = b;  
b = a;  
  
System.out.println(a + ", " + b);
```



```
int a = 1;  
int b = 2;  
  
→ a = b;  
b = a;  
  
System.out.println(a + ", " + b);
```



```
int a = 1;  
int b = 2;  
  
a = b;  
→ b = a;  
  
System.out.println(a + ", " + b);
```

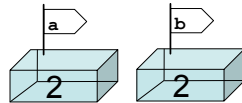


```
int a = 1;  
int b = 2;
```

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

→ `System.out.println(a + ", " + b);`

Affiche:
2, 2



Supposons qu'on ait déclaré et initialisé deux variables a et b.

Comment échanger leurs valeurs ?

Les instructions:

```
a = b;
```

```
b = a;
```

ne **marchent pas**, comme le montre l'exercice précédent.

Solution: utiliser une variable intermédiaire:

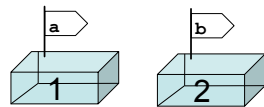
```
int a = 1;
```

```
int b = 2;
```

```
int temp = a;
```

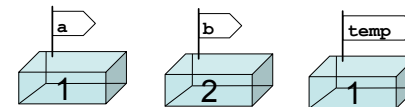
```
a = b;
```

```
b = temp;
```



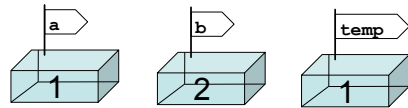
→ `int a = 1;`
`int b = 2;`
`int temp = a;`

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



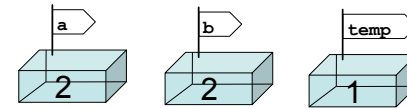
→ `int a = 1;`
`int b = 2;`
`int temp = a;`

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



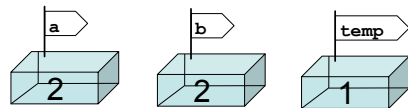
```
int a = 1;  
int b = 2;  
int temp = a;
```

```
→ a = b;  
  b = temp;
```



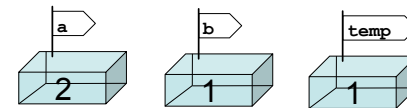
```
int a = 1;  
int b = 2;  
int temp = a;
```

```
→ a = b;  
  b = temp;
```



```
int a = 1;  
int b = 2;  
int temp = a;
```

```
  a = b;  
→ b = temp;
```



```
int a = 1;  
int b = 2;  
int temp = a;
```

```
  a = b;  
→ b = temp;
```

```

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 + ".");
    }
}

```

```

import java.util.Scanner;

class DemanderVariable
{
    public static void main(String [] args)
    {
        Scanner keyb = new Scanner(System.in);
        System.out.println("Entrez une valeur pour n");
        int n = keyb.nextInt();

        int 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 + ".");
    }
}

```

Lire une valeur au clavier

Il faut d'abord importer la classe `Scanner` pour la rendre visible au compilateur:

```
import java.util.Scanner;
```

On peut maintenant créer un objet `Scanner`, par exemple:

```
Scanner keyb = new Scanner(System.in);
```

`keyb` est une variable qu'on peut utiliser pour demander des valeurs au clavier.

Par exemple:

```
int n = keyb.nextInt();
```

Lire une valeur au clavier

```
int n = keyb.nextInt();
```

Cette instruction

1. Arrête le programme momentanément;
2. Attend que l'utilisateur entre une valeur au clavier et appuie sur la touche `return`;
3. Affecte la valeur entrée par l'utilisateur à la variable `n`, puis le programme continue.

`nextInt()` est une méthode de l'objet `Scanner`. Il existe une autre méthode pour demander une valeur de type `double`:

```
double x = keyb.nextDouble();
```

Déroulement du programme pas-à-pas

```
Scanner keyb = new Scanner(System.in);
System.out.println("Entrez une valeur pour n");
int n = keyb.nextInt();

int nCarre = n * n;

System.out.println("La variable n contient " + n);
```

Ce qui s'affiche:

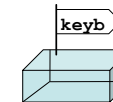
|

Déroulement du programme pas-à-pas

```
Scanner keyb = new Scanner(System.in);
System.out.println("Entrez une valeur pour n");
int n = keyb.nextInt();

int nCarre = n * n;

System.out.println("La variable n contient " + n);
```



Ce qui s'affiche:

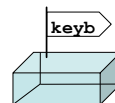
|

Déroulement du programme pas-à-pas

```
Scanner keyb = new Scanner(System.in);
System.out.println("Entrez une valeur pour n");
int n = keyb.nextInt();

int nCarre = n * n;

System.out.println("La variable n contient " + n);
```



Ce qui s'affiche:

Entrez une valeur pour n

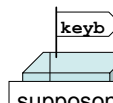
|

Déroulement du programme pas-à-pas

```
Scanner keyb = new Scanner(System.in);
System.out.println("Entrez une valeur pour n");
int n = keyb.nextInt();

int nCarre = n * n;

System.out.println("La variable n contient " + n);
```



Ce qui s'affiche:

Entrez une valeur pour n

2

|

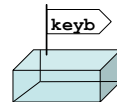
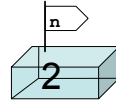
supposons que l'utilisateur entre la
valeur 2 au clavier (il faut presser
return une fois la valeur tapée)

Déroulement du programme pas-à-pas

```
Scanner keyb = new Scanner(System.in);
System.out.println("Entrez une valeur pour n");
→ int n = keyb.nextInt();

int nCarre = n * n;

System.out.println("La variable n contient " + n);
```



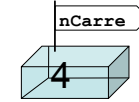
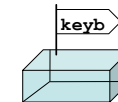
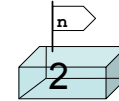
Ce qui s'affiche:

```
Entrez une valeur pour n
2
|
```

Déroulement du programme pas-à-pas

```
Scanner keyb = new Scanner(System.in);
System.out.println("Entrez une valeur pour n");
int n = keyb.nextInt();
→ int nCarre = n * n;

System.out.println("La variable n contient " + n);
```



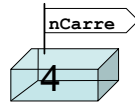
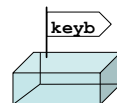
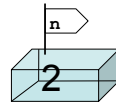
Ce qui s'affiche:

```
Entrez une valeur pour n
2
|
```

Déroulement du programme pas-à-pas

```
Scanner keyb = new Scanner(System.in);
System.out.println("Entrez une valeur pour n");
int n = keyb.nextInt();

int nCarre = n * n;
→ System.out.println("La variable n contient " + n);
```



Ce qui s'affiche:

```
Entrez une valeur pour n
2
La variable n contient 2
|
```

Lecture d'une ligne complète

```
Scanner keyb = new Scanner(System.in);
String s = keyb.nextLine();
System.out.println("Vous avez saisi : " + s);
```

s est une chaîne de caractères. Après l'exécution de l'instruction:

```
String s = keyb.nextLine();
```

s contient tous les caractères tapés par l'utilisateur, jusqu'à l'appui de *return*.

Attention avec `nextLine()` !

```
int i = keyb.nextInt();  
String s1 = keyb.nextLine();  
String s2 = keyb.nextLine();
```

Si on tape:

25 francs

23 francs

→ i contient 25, s1 contient francs, s2 contient 23 francs

Si on tape:

14

euros

43

→ i contient 14, s1 est **vide**, s2 contient euros !

car `nextLine()` lit ce qui suit 14 jusqu'à la fin de la ligne, c'est-à-dire rien !

Expressions

Expression et Opérateurs

A droite du signe égal dans une affectation se trouve une **expression**:

```
nom_de_variable = expression;
```

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

Une expression peut être simplement une **valeur littérale**:

```
4  
3.14
```

ou une formule qui met en oeuvre des **opérateurs**:

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

Les valeurs littérales et leurs types

- `1` est de type `int`;
- `1.0` est de type `double`;
- `1.` est équivalent à `1.0`, et donc de type `double`. On peut écrire:
`double x = 1.;`
au lieu de
`double x = 1.0;`

Il vaut mieux écrire `1.0` plutôt que `1.` puisque c'est plus lisible
- On peut utiliser la notation scientifique, par exemple écrire `2e3` pour 2×10^3 , c'est-à-dire 2000.
De façon générale: `aeb` vaut $a \times 10^b$. Par exemple:
`double x = 1.3e3;`
→ `x` vaut $1.3 \times 10^3 = 1.3 \times 1000 = 1300$
`double y = 1.3e-3;`
→ `y` vaut $1.3 \times 10^{-3} = 1.3 \times 0.001 = 0.0013$

Opérateurs

On dispose des 4 opérateurs usuels:

- + pour l'addition;
- pour la soustraction;
- * pour la multiplication;
- / pour la division.

Attention: si la division se fait entre `int`, il s'agit de la division entière.

Par exemple:

`1 / 2` vaut 0

`5 / 2` vaut 2

mais

`1 / 2.0` vaut bien 0.5

On dispose aussi des opérateurs `+=`, `-=`, `*=`, `/=`

Par exemple:

```
a += 5;  
est équivalent à  
a = a + 5;
```

```
b *= a;  
est équivalent à  
b = b * a;
```

Opérateurs relatifs au type `int`

Dans le cas des `int`, il existe aussi:

- un opérateur **modulo**, noté `%`, qui renvoie le **reste** de la division entière:
`11 % 4` vaut `3`
(la division de 11 par 4 a pour reste 3 car $11 = 2 * 4 + 3$).

```
0 % 4 vaut 0 car 0 = 0 * 4 + 0  
1 % 4 vaut 1 car 1 = 0 * 4 + 1  
2 % 4 vaut 2 car 2 = 0 * 4 + 2  
3 % 4 vaut 3 car 3 = 0 * 4 + 3  
4 % 4 vaut 0 car 4 = 1 * 4 + 0  
5 % 4 vaut 1 car 5 = 1 * 4 + 1  
...
```

Opérateurs relatifs au type `int`

Dans le cas des `int`, il existe aussi:

- deux opérateurs notés `++` et `--`, qui permettent respectivement d'incrémenter et de décrémenter, c'est-à-dire d'ajouter et de soustraire 1 à une variable.

Par exemple, l'instruction:

```
++i;  
est équivalente à :  
i = i + 1;
```

Ces deux opérateurs sont souvent utilisés avec l'instruction `for`, que nous verrons par la suite.

Affectation d'une valeur décimale à une variable entière

En Java, il est **impossible** d'affecter une valeur décimale par exemple de type `double` à une variable de type `int`:

Exemple:

```
double x = 1.5;  
int n = 3 * x; // Erreur !!!
```

Le compilateur produit le message d'erreur suivant :

```
error: possible loss of precision  
    n = 3 * x;  
      ^  
required: int  
found: double
```

Affectation d'une valeur entière à une variable décimale

En revanche, il est possible d'affecter une valeur de type `int` à une variable de type décimale, par exemple `double`.

Exemple:

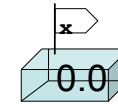
```
int n = 3;  
double x = 2 * n;
```

La division entière

```
double x;
```

```
x = 1 / 2;
```

l'expression `1 / 2` est d'abord évaluée
elle vaut 0
la valeur 0 est affectée à `x`



La division entière

Le problème peut se poser par exemple quand on calcule la moyenne de deux valeurs entières:

```
int note1 = 4;  
int note2 = 5;  
  
double moyenne = (note1 + note2) / 2;
```

Dans ce cas, `moyenne` vaut 4 au lieu de 4.5

La division entière

Une solution possible:

```
int note1 = 4;  
int note2 = 5;  
  
double moyenne = (note1 + note2);  
moyenne /= 2;
```

Fonctions mathématiques

Java fournit les fonction mathématiques usuelles, ainsi que des constantes comme Pi.

Ces fonctions et constantes s'utilisent en suivant la notation:

```
Math.nomFonction(expression1, expression2, ...);  
Math.nomConstante
```

Par exemple:

```
class ExempleMathematique  
{  
    public static void main(String[] args) {  
  
        double angle = 10 * Math.PI / 180;  
        double s = Math.sin(angle);  
    }  
}
```

Fonctions mathématiques

L'ensemble des fonctions disponibles est documenté dans:

<http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html>

```
import java.util.Scanner;  
  
class ExempleAngle  
{  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.println("Entrez un angle en degres: ");  
        double angleEnDegres = scanner.nextDouble();  
        double angleEnRadians = Math.PI * angleEnDegres / 180;  
        System.out.println("Sa valeur en radians est " + angleEnRadians);  
        System.out.println("Son cosinus vaut " + Math.cos(angleEnRadians));  
    }  
}
```

$$\text{angle en radians} = \frac{\pi \text{ angle en degrés}}{180}$$