

# Cours Introduction à la Programmation Java I (IP1 Java)

Arnaud Sangnier  
[sangnier@irif.fr](mailto:sangnier@irif.fr)

Mercredi 13 Septembre 2017  
INFO et MATHS-INFO

# But du cours

- Apprendre les bases de la programmation
- Être capable de comprendre des programmes
- Être capable d'écrire des programmes simples
- Langage utilisé : **Java**
- **Points positifs**
  - Cours sans difficulté théorique
  - Savoir programmer est un atout important
- **Points 'négatifs'**
  - Travail régulier nécessaire
  - Fort taux d'échec à l'examen
  - Besoin de beaucoup de rigueur dans l'écriture des programmes

# Organisation des enseignements

- **Cours**

- 6 cours
- Un mercredi sur deux de 15h45 à 17h45, amphi 1A
- **13/09 – 27/09 – 11/10 – 25/10 – 15/11 – 29/11**

- **Cours/td**

- 2h par semaine
- Début : semaine du 18 septembre

- **Tp**

- 2 fois 2h par semaine
- Début : semaine du 18 septembre

- **Tutorat**

- Accès libre
- Tous les jours entre 12h et 14h (horaires et salles à préciser)
- Début : semaine du 25 septembre

# Organisation des enseignements

- **Cours**

- 6 cours
- Un mercredi sur deux de 15h45 à 17h45, amphi 1A
- **20/09 – 4/10 – 18/10 – 8/11 – 15/11**

- **Cours/td**

- 2h par semaine
- Début : semaine du 18

- **Tp**

- 2 fois 2h par semaine
- Début : semaine du 18 septembre

- **Tutorat**

- Accès libre
- Tous les jours entre 12h et 14h (horaires et salles à préciser)
- Début : semaine du 25 septembre

**Important :**  
**Respecter votre groupe**

# Évaluation

- **Contrôle continu cours/td**
  - 2 épreuves
- **Contrôle continu Tp**
  - 2 épreuves
- **Partiel**
  - Un partiel de 2h le **samedi 4 novembre de 12h30 à 14h30**
- **Examen**
  - Un examen de 3h en décembre

# Note

**Td : Résultats des cours/td**

**Tp : Résultat des tp**

**P : Résultat du partie**

**E : Résultat de l'examen**

- Note finale Cc :  $(Td + Tp) / 2$
- Note écrit Ne :  $\text{Max}(E, (E + P) / 2)$
- **Note session 1 :  $(3*Ne + Cc) / 4$**
- **Remarques :**
  - Absence aux CC : 0
  - Absence au partiel : 0
  - Absence à l'examen : pas de note
  - **Une mauvaise note au partiel est rattrapable**

# Points sur le contenu

- Les bases de la programmation seront présentés en cours/td
  - Des supports vous seront distribués
- Les Tp servent à mettre en pratique ces bases
  - Les énoncés seront sur Moodle
- En amphi :
  - Présentation de concepts généraux
  - Aide à la compréhension de ce qui est vu en td/tp
  - Correction partielle des examens de l'année précédente

# Communication

- N'hésitez pas à communiquer avec vos chargés de cours/td et tp
- Vous pouvez aussi m'écrire : [sangnier@irif.fr](mailto:sangnier@irif.fr)
- Nous lisons tous nos mails régulièrement
  - Respecter cela dit les règles de courtoisie dans vos mails
    - N'oubliez pas de **signer votre mail**, d'**écrire sans faute d'orthographe**, de préciser votre groupe etc
- N'hésitez pas à refaire les exercices chez vous et à vous adresser à vos encadrants en cas de doute
- **Évitez d'envoyer un programme tapé dans un mail ou dans un document Word !!!!**
- **Page Moodle du cours** (toutes les infos y sont données) :
  - <https://moodlesupd.script.univ-paris-diderot.fr/course/view.php?id=2848>
- **Page internet du cours** (pour les supports)
  - <https://www.irif.fr/~sangnier/enseignement/ip1-java.html>



# Programmer

- Pour les TPs, il vous faut un **login** et un **mot de passe** pour pouvoir vous connecter (donnés au moment de votre inscription)
- Comment travailler vos cours :
  - Écrire les programmes sur feuille sans les tester n'est pas suffisant
  - Il faut écrire des programmes chez vous ou en salle de TP et tester qu'ils fonctionnent bien
  - La voie vers le succès pour ce cours : **programmer encore et encore**

# Qu'est ce qu'un programme ?

- Un programme est une **suite d'instructions** qui pourra être 'exécutée' par la machine
- Quelles sont les instructions disponibles
  - Faire un calcul arithmétique (par ex.  $12 * 5$ )
  - Afficher une chaîne de caractères
  - Déplacer la souris
  - Lancer un autre programme
  - Manipuler des données
  - Jouer un son
  - etc

# Où trouve-t-on les programmes ?

- Tout ce que vous utilisez sur une machine telle qu'un ordinateur, une tablette ou un smartphone est un programme
  - Les applications
  - Les logiciels
  - Mais aussi le système qui fait fonctionner votre appareil (android, IOS, Windows, Linux,...)

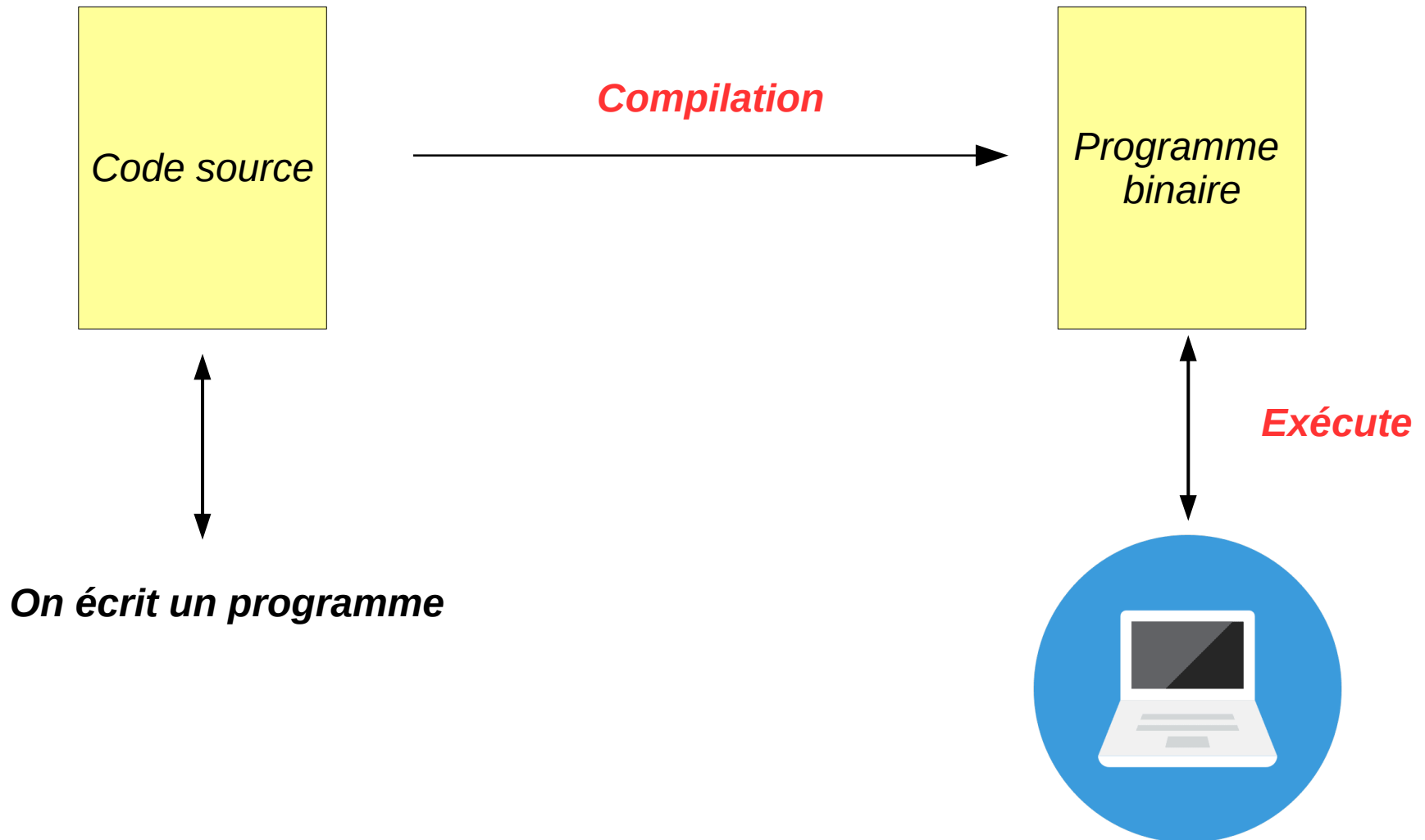
# Comment écrit-on un programme ?

- Un programmeur écrit un programme dans un langage de programmation
- Il existe plusieurs langages de programmation et plusieurs familles de langage de programmation
  - **Langages Orientée Objets :**
    - Ex : **Java**, Python
  - **Langage Impératif :**
    - Ex : C
  - **Langage Fonctionnelle :**
    - Ex : CAML, OCAML,

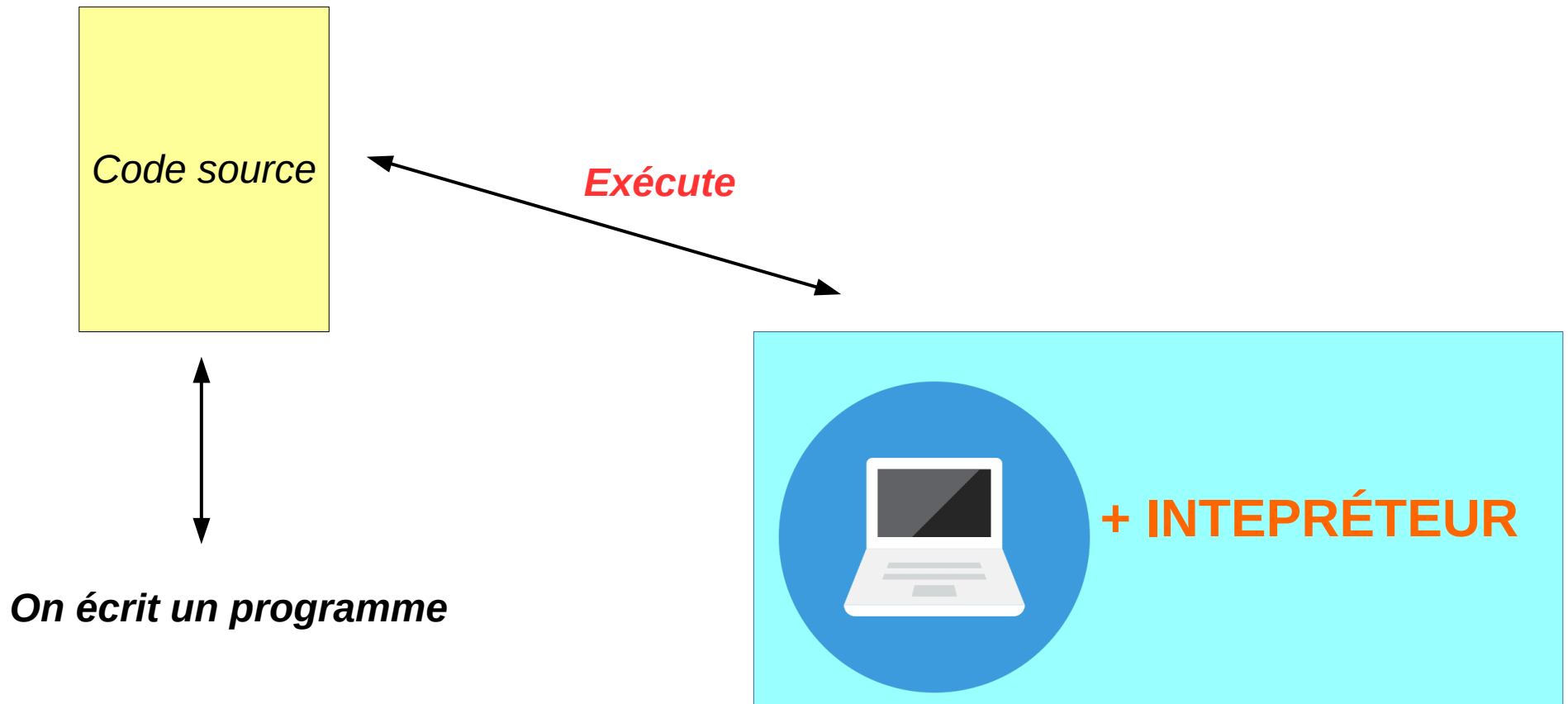
# Comment la machine comprend tous les langages ?

- Le langage de programmation est un langage 'compréhensible' par les humains
- Les instructions sont un mélange d'anglais et d'opérations mathématiques, plus certaines instructions spécifiques à chacun des langages
- Le programme écrit par le programmeur est contenu dans un fichier, on parle de **code source**
- Le code source est ensuite soit **traduit** vers un langage compréhensible par la machine (langage binaire), on parle de **compilation**, soit il est interprété par un **interpréteur** qui exécute ces instructions (langage interprété)
- Pour pouvoir exécuter un programme, il faut donc soit avoir le compilateur (Java) ou l'interpréteur (Python, OCaml)
- **Remarques : L'interpréteur et le compilateur sont eux-mêmes des programmes**

# Schéma d'exécution d'un code source avec compilation



# Schéma d'exécution d'un code interprété



# Langage étudié

- **JAVA**
- En fait un **sous langage**
  - Réalisation de calculs arithmétiques
  - Manipulation de chaînes de caractères
  - Lire/écrire/modifier des variables
  - Boucler sur des instructions
  - Tester des valeurs
  - Écrire et appeler des fonctions
  - Manipulation de tableaux



# Données manipulées

- Un programme manipule des données
- Ces données peuvent être de différentes sortes
- On parle en fait de **type**
  - **int** : il s'agit des données entières (par exemple : 3, 4, 5000, etc.)
  - **double** : il s'agit des nombres réels avec virgules (par exemple : 3.5, 2.4, etc)
  - **String** : il s'agit de chaînes de caractères (par exemple "Hello World", "345", "Un message !", etc)
  - On verra qu'il existe d'autres types

# Quelques remarques sur les données

- Il faut éviter de mélanger les données de type différent
- Il faut toujours avoir en tête quel type de données on manipule
  - **La division entière  $2 / 4$  donne  $0$  alors que la division réelle  $2.0 / 4$  donne  $0.5$**
- À quoi sert le type **String** ?
  - Typiquement à stocker des données correspondant à des chaînes de caractères, mais aussi des messages que l'on souhaite afficher
- **ATTENTION :**
  - Une machine n'a pas une précision infinie
  - Ainsi, on ne peut pas compter jusqu'à l'infini
  - Pour les nombres réels, on ne dispose pas d'une précision infinie
    - Par exemple :  $1 / 3$  est interprété en Java comme  $0.3333333333333333$
    - Il n'y a pas un nombre infini de chiffres après la virgule !

# Opération sur les données

- Un programme peut faire des opérations sur les données
- Sur les données entières, comme un calculatrice :
  - addition ( $2 + 5$ ), soustraction, division entière ( $3 / 4$ ), multiplication, etc
- Sur les chaînes de caractères : concaténation
  - "Hello" + "World !" donne la chaîne "Hello World !"
- **ATTENTION :**
  - ne mélanger pas les types dans les opérations en faisant
  - Par exemple : "Hello" + 3 marche mais n'est pas recommandée, ni  $3 * \text{"Bob"}$

# Opération sur les données

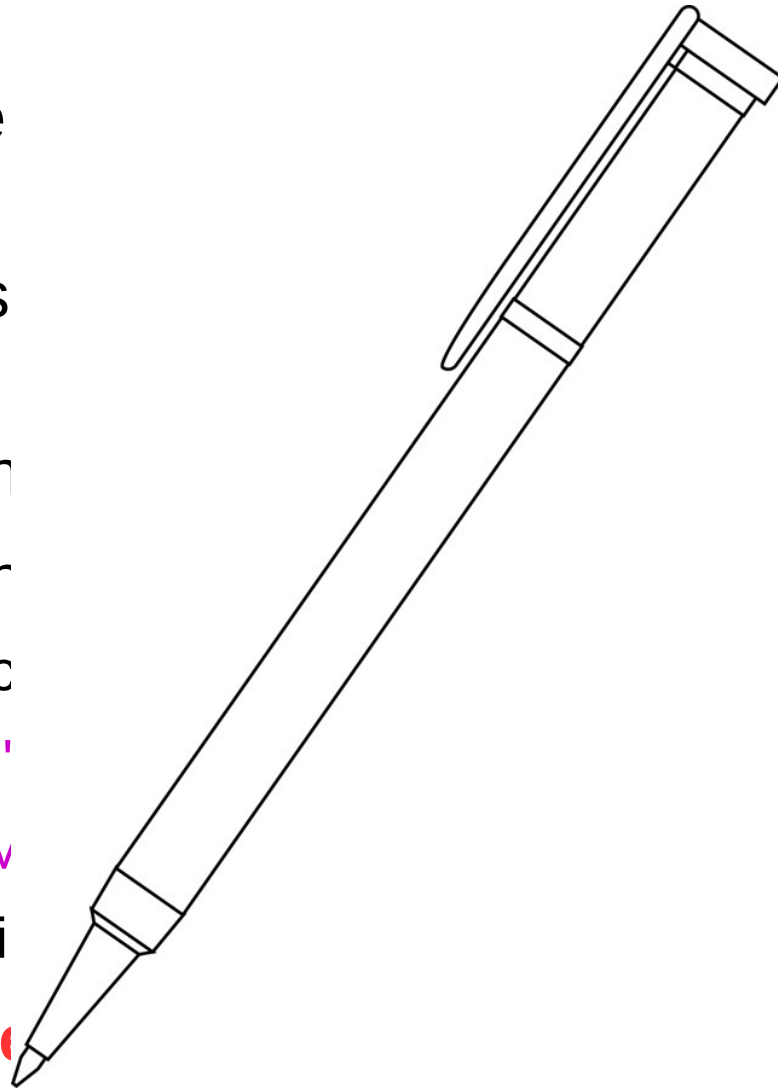
- Un programme peut faire des opérations sur les données
- Sur les données entières, comme un calculatrice :
  - addition ( $2 + 5$ ), soustraction, division entière ( $3 // 4$ ), multiplication, etc
- Sur les chaînes de caractères : concaténation
  - "Hello" + "World !" donne la chaîne "Hello World !"
- **ATTENTION :**
  - ne mélanger pas les types dans les opérations
  - Par exemple :  $3 * \text{"Bob"}$  n'existe pas, ni  $3 + \text{"Bob"}$

# Voir le résultat d'une opération

- Un programme qui fait des opérations le fait **silencieusement** (on ne voit pas l'effet)
- Pour voir le résultat d'une opération, on peut demander au programme de l'afficher
- On utilise la fonction **System.out.print** ou **System.out.println**
- L'argument donné est affiché sur le terminal, par exemple
  - `System.out.println (6*7)` affiche 42
  - `System.out.println ("Hello ! ")` affiche Hello !
  - `System.out.println("Un" + "Message")` affiche UnMessage
  - **Attention** : `System.out.println("6*7")` affiche 6\*7 (et pas 42)
- **Un programme n'affiche rien si on ne lui demande pas**

# Voir le résultat d'une opération

- Un programme (on ne voit pas)
- Pour voir le résultat d'une opération, on demande au programme de
- On utilise la fonction `print()`
- L'argument doré est le résultat de l'opération
  - `print(6*7)` affiche 42
  - `print("Hello ! ")` affiche Hello !
  - `print("Un" + "Nuit")` affiche UnNuit
  - **Attention** : `print(6*7)` affiche 42, mais `print(6*7)` affiche 42
- **Un programme qui demande pas**



ait **silencieusement**

peut demander au

al, par exemple

**si demande pas**

# Les variables

- Un programme peut stocker les données
  - pour faciliter leur manipulation
  - pour abstraire leur valeur
  - pour les réutiliser plus tard
  - pour faire des calculs complexes
- Il dispose de sa mémoire (pensez à un ensemble de cases)
- Une variable indique un endroit de la mémoire où est stocké une donnée
- Une variable a un **nom**, par exemple x, y, z, var, z3
- Une variable a un **type**
- Pour utiliser la variable, on utilise son nom
- Opérations sur les variables : **Déclaration**, **Affectation**, **Lecture** et **Modification**

# Opération sur les variables

- **Déclaration et affectation** (Donner le type de la variable et mettre une donnée)
  - Attention on utilise le symbole = , mais qui ne veut pas dire égalité
  - On déclare qu'une seule fois le type
  - Par exemple : `int x = 3`
- **Lecture** (lire la donnée d'une variable)
  - On utilise le nom de la variable à la place de la donnée
  - Par exemple : `System.out.println (x + 2)` affiche 5
- **Modification** (modifier la valeur d'une variable)
  - Comme l'affectation : `x = 8`



# Opération sur les variables

- Le programme suivant

```
int x = 3 ;  
int y = 2 ;  
int z = x + 1 ;  
X = 6 ;  
y = 2 * x ;  
System.out.println(x)  
System.out.println(y)  
System.out.println(z)
```

- Affiche

```
6  
12  
4
```

# Quelques règles de bonne conduite

- Toujours initialiser une variable, par exemple au début du programme
- Ne pas déclarer deux fois variables
  - Par exemple : on ne peut pas faire
    - `int x = 2` et ensuite `int x = 3` - > **ERREUR**
    - ni non plus `int x = 1` et `int z = int x` - > **ERREUR**
- **Interdit** de mettre à gauche de = une valeur et à droite une variable
  - Par exemple : ~~2 = x~~

# Que fait la machine ?

- Si par exemple on a une ligne  $z = (x * x) + 2$ 
  - 1) Va chercher la valeur de la variable x
    - Si x n'a pas de valeur → **Erreur**
  - 2) Calcule  $(x * x) + 2$
  - 3) Stocke la valeur obtenue dans la variable z
- On calcule d'abord ce qui se trouve à droite du symbole =

# Exemple - I

- Le programme suivant

```
int x = 3 ;  
int y = 2 ;  
int z = x * x ;  
y = 3 * z ;  
System.out.println (x) ;  
System.out.println (y) ;  
System.out.println (z) ;
```

- Affiche

```
3  
27  
9
```

## Exemple - II

- Le programme suivant

```
int x = 3 ;  
x = x + x ;  
x = x - 1 ;  
System.out.println (x) ;
```

- Affiche

5

# Plusieurs valeurs possibles mais un seul programme

- Une clef essentielle de la programmation est d'écrire des programmes qui vont marcher pour différentes valeurs possibles
- Par exemple :
  - Un programme qui calcule  $n!$  (factorielle de  $n$ )
  - Un programme qui calcule la somme de deux entiers  $a$  et  $b$
  - Un programme qui calcule le pgcd de deux entiers  $a$  et  $b$
- Ici les valeurs de  $n$ ,  $a$  et  $b$  ne sont pas connus à l'avance, on sait juste qu'il s'agit d'entier

# Les fonctions

- Une fonction d'un programme est une liste d'instructions
- Elle peut être appelée plusieurs fois
- Elle peut prendre des valeurs en entrée
  - Il s'agit des arguments
- Elle peut calculer une valeur et la renvoyer
  - Il s'agit de la valeur de retour
- On lui donne un nom (le nom de fonctions) et on précise **toujours** le **type de retour** et le **type des arguments**
- Exemple : la fonction `System.out.println (String s )` qui ne retourne pas de valeur mais qui affiche à l'écran la chaîne de caractères donnée en arguments

# Exemple de fonctions

```
public static int f (int x){  
    return (2*x) ;  
}  
public static void main(String[] args) {  
    int a = f(3) ;  
    int b = f(5) ;  
    System.out.println (a) ;  
    System.out.println (b) ;  
}
```

Définition de la fonction f

Appel de la fonction f

- Affiche

```
6  
10
```

- Si on enlève les `System.out.println`, le programme n'affiche rien



# Que fait la machine ?

```
public static int f (int x){  
    return (2*x) ;  
}
```

- Si par exemple on a une ligne  $z = f(4)$ 
  - 1) Elle remplace la valeur x de f par 4
  - 2) Elle calcule  $2*4$
  - 3) Elle renvoie la valeur 8
  - 4) Elle stocke cette valeur dans z

# Exemple

```
public static int f (int x){  
    return (2*x);  
}  
public static void main(String[] args) {  
    int z = 10;  
    z = f(z);  
    System.out.println(z);  
}
```

- Affiche

20

# Exemple

```
public static int f (int x){  
    return (2*x) ;  
}  
  
public static void main(String[] args) {  
    int x = 10 ;  
    int z = f(x) ;  
  
    System.out.println (x) ;  
    System.out.println (z) ;  
}
```

**ATTENTION :**  
Les deux x ne sont pas  
la même variable

- Affiche

```
10  
22
```

# Exemple

```
public static int f (int x) {  
    x = x + 1 ;  
    return (2 * x) ;  
}  
  
public static void main(String[] args) {  
    int x = 10 ;  
    int z = f(x) ;  
    System.out.println(x) ;  
    System.out.println(z) ;  
}
```

- Plus sûr en écrivant :

```
public static int f (int x) {  
    x = x + 1 ;  
    return (2 * x) ;  
}  
  
public static void main(String[] args) {  
    int x = 10 ;  
    int z = f(x) ;  
    System.out.println(x) ;  
    System.out.println(z) ;  
}
```

**On évite ainsi les confusions possibles**

# Instructions conditionnelles et boucles

- Vous verrez aussi des instructions pour tester la valeur de variable

```
if (x == 5) :  
    System.out.println ("AH");  
else :  
    System.out.println("OH");
```

Affiche AH si  
x vaut 5  
et OH sinon

- Ou pour répéter un certain nombre de fois une instruction

```
for(int i=1;i<100;i=i+1){  
    System.out.println("Hello") ;  
}
```

Affiche 100 fois  
Hello