

Initiation à la programmation en Java

JC Damichel

Lycée Eiffel

2 novembre 2016



Sommaire

- 1 Langage Java
- 2 Exemple : résolution d'une équation du premier degré
- 3 Les erreurs
- 4 Les types
- 5 Remarques sur Java

Présentation de Java

- Langage choisi en ISN : Java
- Java = terme d'argot américain signifiant café
- Logo :



- Développé par Sunmicrosystems, apparu en 1995, racheté par Oracle en 2009
- Utilisation libre

Caractéristiques de Java

- Orienté objets $\Rightarrow$ version simplifiée avec Java's Cool et Processing
- Robuste et sûr
- Indépendant de la machine employée pour l'exécution : portable sur tout type d'ordinateur, mobile... si JVM (Java Virtual Machine) installée sur l'appareil
- Langage interprété (contrairement à compilé) :
 - ▶ Précompilation qui génère du bytecode (ce n'est pas du langage machine)
 - ▶ La machine virtuelle compile à la volée le bytecode en langage machine
- Syntaxe proche du C/C++
- Typage statique



Algorithme en langage naturel

```
1  VARIABLES
2    a EST_DU_TYPE NOMBRE
3    b EST_DU_TYPE NOMBRE
4    solution EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6    LIRE a
7    LIRE b
8    SI (a!=0) ALORS
9      DEBUT_SI
10     AFFICHER "Il y a une solution : "
11     solution PREND_LA_VALEUR -b/a
12     AFFICHER solution
13   FIN_SI
14   SINON
15     DEBUT_SINON
16     SI (b==0) ALORS
17       DEBUT_SI
18       AFFICHER "Tous les réels sont solution"
19     FIN_SI
20   SINON
21     DEBUT_SINON
22     AFFICHER "Il n'y a pas de solution"
23   FIN_SINON
24   FIN_SINON
25 FIN_ALGORITHME
```

Code source Java

```

/*
 * Ce programme résout une équation du premier degré
 * de la forme  $ax+b=0$ .
 * L'utilisateur saisit a puis b.
 */

//Solution représente la solution si elle existe
double a, b, solution;

void main(){
    //L'utilisateur saisit a et b
    a=readDouble();
    b=readDouble();
    //Si a est différent de 0, il y a une unique solution:  $-b/a$ 
    if (a!=0) {
        solution=-b/a;
        println("Il y a une solution : "+solution);
    }
    else {
        //Si a=0 et b=0, tous les réels sont solution
        if (b==0) {
            println("Tous les réels sont solution.");
        }
        else {
            //Si a=0 et b différent de 0, il n'y a pas de solution
            println("Il n'y a pas de solution.");
        }
    }
}

```

Code source Java

- Java est sensible à la casse
- Chaque instruction se termine par ;
- Respecter la syntaxe =grammaire (ensemble de mots clés+assemblage de ces mots)
- Commenter le code source (sans surcharger)
- Indenter le code
- On déclare les variables : Java est un langage à typage statique
- La fonction *main* (fonction principale) doit être présente : elle est exécutée quand on exécute le bytecode
- On écrit le code dedans
- **Par convention**, les noms de variables commencent par une minuscule et chaque nouveau mot par une majuscule (exemple *a = nomDeVariable*)



Trois types d'erreur

■ Erreurs de syntaxe

- ▶ On détecte ces erreurs à la compilation
- ▶ Le message d'erreur n'est pas toujours clair

■ Erreurs à l'exécution

- ▶ Par exemple, division par zéro
- ▶ Ces erreurs peuvent dépendre des conditions initiales $\Rightarrow$ difficiles à détecter
- ▶ Batterie complète de tests par le programmeur
- ▶ Exemple d'équation du 1er degré : le programmeur doit penser à tester tous les cas et à gérer le cas $a = 0$.

■ Erreurs sémantiques (erreurs de logique)

- ▶ Le programme s'exécute sans erreur mais le résultat est faux (toujours ou dans certains cas)
- ▶ Batterie complète de tests par le programmeur



Les types élémentaires (liste non exhaustive)

■ Entiers : *int*

Ce sont des entiers signés et représentés en complément à 2. Ils sont codés sur 32 bits. Ils sont compris entre - 2147483648 et 2147483647.

■ Réels : *double*

Ils sont codés sur 64 bits. Ils sont compris entre $\pm 4.9E - 324$ et $\pm 1.7976931348623157E308$.

Exemple : 123.456 34d 34.0 34. 56e34 45.67e2

■ Booléens : *boolean*

Ils prennent deux valeurs : **true** et **false**. Ils sont codés sur 1 bit.

■ Caractères : *char*

Ils sont codés sur 16 bits. Ils sont notés entre deux apostrophes.

Exemple : 'a' 'ç' '\n' (passage à la ligne) '\\ ' (backslash) '\"' (apostrophe)



Les types élémentaires : initialisation

- Lors de la déclaration (par exemple *int a*), case mémoire réservée (32 bits)
- Lors de l'affectation, (par exemple *a=5*), case mémoire remplie par des bits représentant 5 (101)
- Ces deux étapes sont nécessaires

Les objets de base (liste non exhaustive)

■ Chaînes de caractères : *String*

Les constantes littérales chaînes de caractères sont notées entre guillemets. Une chaîne, une fois créée, ne peut pas être modifiée : par exemple, si on veut supprimer une lettre de la chaîne, il faut créer une autre chaîne.

- ▶ L'opérateur `+` permet de concaténer des chaînes.
- ▶ Pour obtenir la longueur d'une chaîne `s` : `s.length()`
- ▶ Pour obtenir un caractère dans une chaîne `s` : `s.charAt(i)` où `i` est l'indice du caractère. L'indice varie de 0 à `s.length()-1`
- ▶ Pour comparer deux chaînes `s` et `r` : `s.equals(r)`. Attention à ne pas confondre avec `s == r` qui compare en fait les adresses de `s` et `r` dans la mémoire.
- ▶ Pour avoir d'autres fonctions sur les `String` : taper *javadoc 7 string* dans la barre de recherche d'un navigateur.



Les objets de base (liste non exhaustive)

■ tableaux :

Un tableau est une liste d'éléments de même type. Pour utiliser un tableau :

- 1 On le déclare : `int[] nomTableau` ; pour un tableau d'entiers ou `String[] nomTableau` ; pour un tableau de String. Attention : la déclaration ne remplit pas le tableau et ne définit pas sa taille.
- 2 On le crée en définissant sa taille : `nomTableau=new int[10]` ; pour un tableau de 10 entiers. La taille du tableau est fixée définitivement et le tableau ne peut pas grandir pendant l'exécution. Pour obtenir la taille d'un tableau :
`nomTableau.length`
- 3 On le remplit : `nomTableau[0]=100` ; `nomTableau[1]=50` ; ...

Les objets de base (liste non exhaustive)

■ tableaux :

Remarque :

- ▶ On peut déclarer et créer un tableau et le remplir ensuite :
`int[] nomTableau=new int[10];`
...
`for (int i=0;i<nomTableau.length;i++) nomTableau[i]=i+2;`
- ▶ Lorsque c'est possible, on peut déclarer, créer et remplir un tableau en une ligne : `int[] nomTableau={4, 5, 6};`
- ▶ On peut aussi déclarer un tableau et le créer/remplir ensuite :
`int[] nomTableau;`
...
`nomTableau= new int[] {4, 5, 6};`

Les objets de base (liste non exhaustive)

■ tableaux :

Initialisation :

- ▶ Lors de la déclaration (par exemple `int[] nomTableau;`), case mémoire réservée pour contenir la référence du tableau
- ▶ Lors de la création, (par exemple `nomTableau=new int[10];`), 10 cases mémoire réservées pour contenir 10 entiers et la référence représentant la position de la première de ces cases est stockée dans la case du point précédent
- ▶ Lors du remplissage, les entiers sont stockés dans les cases réservées
- ▶ Attention, la variable `nomTableau` ne contient pas les valeurs du tableau mais l'adresse où elles se trouvent. On dit que `nomTableau` est un *handle*.

Les objets de base (liste non exhaustive)

■ tableaux à deux dimensions :

Ce sont des tableaux de tableaux :

- 1 Déclaration : `int[][] nomTableau` ; pour des entiers.
- 2 Création : `nomTableau=new int[5][10]` ; pour 5 lignes et 10 colonnes.
- 3 Remplissage :

```
for (int i=0;i<nomTableau.length;i++)  
  for (int j=0;j<nomTableau[i].length;j++)  
    nomTableau[i][j]=i+j;
```

Si possible, on peut déclarer, créer et remplir un tableau à 2 dimensions en une seule ligne :

```
int[][] nomTableau={{1, 2, 3}, {4, 5, 6}};
```

Astuces à connaître

- **division entière** : si on divise deux entiers, ou deux **variables de types entiers**, le résultat est un entier (résultat tronqué)
- Pour comparer deux chaînes s et r : $s.equals(r)$. Attention à ne pas confondre avec $s == r$ qui compare en fait les adresses de s et r dans la mémoire car s et r sont des handles.
- Si $tab1$ et $tab2$ sont des tableaux, alors $tab1 == tab2$ compare les adresses et pas les contenus des tableaux.
- Pour les tableaux, les indices varient entre 0 et longueur du tableau -1