Langages avec des objets : JAVA

Robert Cori

Buts de ce cours:

1. Comprendre les principes de la programmation
2. Connaître les développements modernes de logiciels et avoir une idée de leur fonctionnement
3. Maîtriser l'écriture de programme.
4. Découvrir quelques problèmes actuels de l'informatique.

Vos enseignants

- Jean-Claude Ville

  Groupe 1 : mardi 9h 30 -> 12h 20

  Groupe 2 : lundi 15h30 -> 18h 30

Vos enseignants

- Anne Vialard .  Groupe 3 : Jeudi 14h -> 16h 50

- David Auber Groupe 4 : Vendredi 14h -> 16h 50 

Menu du Jour (cours)

1. Organisation de l'enseignement.
2. Introduction à Java.
   - Généralités: mise en œuvre
   - Classes
   - Constantes
   - Variables et Types
   - Affectations
   - Structures de contrôle
   - Boucles
   - Fonctions
   - Tableaux
   - Chaînes de caractères (String)

Organisation

Contenu du cours

1. Mise à niveau Java

   Ensuite:

2. Classes
3. Modularité
4. Types abstraits
5. Héritage

Informations Pratiques

- TDs
- Devoir Surveillé
- Epreuve écrite
- Rattrapage en jury

A propos de Java

- Un langage utilisé pour le WWW
- Une version simplifiée de C++
- Langage avec des objets *(object oriented)*
- Avantages: typé, relativement simple, proche de C, assez répandu.
- Inconvénients: instructions parfois lourdes, difficultés de la programmation avec des objets.

Premières caractéristiques de Java

- Un programme est constitué de classes.
- Une classe contient des constantes, des variables, des fonctions ...
- Premiers programmes : une seule classe.
- La fonction qui a pour nom `main` dans la classe `Exclasse` est exécutée lors de la commande

```
java Exclasse
```

Premier programme

Programme qui affiche des valeurs à l'écran :

```
public class Prog1 {
//On compare pi et 22/7

    static final double u = 22./7;
    public static void main (String[] arg){

        System.out.println(u);
        System.out.println(Math.PI);
    }
}
```

Compilation, Interprétation

A mettre dans un fichier `Prog1.java`, ensuite pour compiler

```
javac  NomClasse.java
```

Enfin pour exécuter (interpréter le byte-code)

```
java  NomClasse
```

```
% javac Prog1.java
 % java Prog1
3.142857142857143
3.141592653589793
 %
```

Constantes

- On déclare les constantes dans une classe par: `final static` suivi du type, du nom et de la valeur avec = comme pour une affectation
- Ne jamais mettre des valeurs constantes au milieu d'un programme

Exemple :

```
final static int n = 100;
final static float taux = 0.30;
```

Variables, Types

- Langage fortement typé: toute variable a un type qu'il faut déclarer
- Un type définit des valeurs et des opérations que l'on peut effectuer sur celles-ci.

Les types

- Un type désigne un ensemble de valeurs et un ensemble d'opérations sur ces valeurs
- Exemple, les types primitifs: entiers, flottants, booléens, caractères
- Le type tableau de type `typ` est un nouveau type
- En Java les autres types sont définis par des classes: (objets et méthodes à voir plus tard)

Types primitifs

1. Les octets (8 bits) `byte`
2. Les entiers courts (16 bits) `short`
3. Les entiers (32 bits) `int`
4. Les entiers longs (64 bits) `long`
5. Les réels (32 bits) `float`
6. Les réels longs (64 bits) `double`
7. Les caractères (16 bits) `char`
8. Les booléens `boolean`: constantes : `true, false`.

Affectation

```
x = E;
```

- L'expression `E` est évaluée
- La variable `x` est affectée de la valeur obtenue

- On peut convertir implicitement un entier `int` en un entier `long` ou un `short` en `int`, de même pour les `float` en `double`. On peut aussi convertir un entier en `float` ou en `double`

```
short i = 23;
int j = i;
long u = j;
float u = (float) i;
int k = (int) u;
```

Incrémentation

```
i++;
++i;
x = i++;
x = ++i;
```

Expressions arithmétiques

Variables, constantes, opérations mathématiques

Calculer la différence des surfaces d'un cercle de rayon 2.75 et d'un carré de coté 5:

```
int a = 5;
float r = (float) 2.75;
double pi = Math.PI;
float s = pi * r * r - a * a;
System.out.println(s);
```

Calculer le reste de la division de 3125 par 713:

```
int a = 3125 % 713;
```

Afficher le résultat :

```
System.out.prinln(a);
```

Expressions logiques

Expression évaluées à `true` ou `false`

```
x > 8 ;
x == 7;
x != 9;
```

Attention à la différence entre = affectation, et == comparaison

Connecteurs logiques

- `&&`, signifie *et*, `||` signifie *ou* et `!` signifie *non*.
- Ordre de calcul évite les erreurs

```
(a > 8) && (b <16);
! (a == 9);
c != 0;
(a > 8) && (b/c == 1);
(b/c == 1) || (a == 8);
boolean res  = true;
```

Instructions Conditionnelles

```
if (E)  I
```

```
if (E) I  else  J
```

```
if (E) {I1  I2 . . . IP} else {J1  J2  ... Jq}
```

```
static int absDiff (int a, int b){
    if (a > b) return a - b;
        else return b - a;
}
```

Boucle `while`

```
  while (E) I

  while (E) {

     I1 I2 ....Ip
   }

 static int divEntiere (int a, int b){
     int r = 0;
     while (a >=  0) {
        a = a  - b;
        r++;
     }
     return r;
}
```

Boucle `for`

```
for (Init E; Incr) I
```

Equivalent à :

```
Init
while(E) {
 I
 Incr
 }
```

Itérations, Boucles: Exemple

```
class SommeCube  {
     static final int n = 12;

     public static void main (String[]  arg){
          int s = 0;
          for (int i = 0; i < n; i++)
               s = s + i * i * i;
          System.out.println (s);
          }
}
```

Fonctions

- Une fonction a des paramètres et obtient un résultat
- Le type de la fonction est donné par le type de son résultat
- Déclarer une fonction, par son type et le type des paramètres (signature)
- Le résultat est rendu à l'appelant par `return E`

```
static int sommeMod2 (int a, int b){
  return (a + b) % 2 ;
}
```

- Le calcul d'une fonction est appelé par `nom(par1, par2, .. park)`

```
int z = sommeMod2 (x,y);
```

Les tableaux

- Un tableau est un ensemble de variables de même type
- Se déclare par `typ[] tab;` ou `typ tab[];`
- De préférence `typ[] tab;`
- Les éléments sont des variables de type `typ`
- Les noms des variables sont `tab[0], tab[1], ... tab[p-1]`
- Le nombre d'éléments du tableau `tab` est donné par `tab.length`

Déclaration et initialisation d'un tableau

Pour désigner la variable `alpha` formellement comme un tableau d'entiers

```
int[] alpha;
```

Pour réserver de place en mémoire pour la variable `alpha` comme un tableau composé de n entiers

```
alpha = new int[n];
```

on peut faire les deux en même temps

Exemple:

```
int n = 7;
int[] p1  = new int[n];
for (i = 0; i < n ; ++i)
     if (i < 3) p1[i] = 0;
     else p1[i] = 1;
```

ou de manière équivalente:

```
int[] p1  = {0, 0, 0, 1, 1, 1, 1};
System.out.println(); p1.length;
```

Fonctions sur les tableaux

Une fonction peut avoir pour résultat un tableau :

```
static int[]  tabConstant (int a, int n) {
int[] res = new int[n];
        for (int i = 0 ; i < n ; ++i)
            res[i] = a;
        return res;
    }
```

Chaînes de caractères

- Se déclare par:

```
String u;
```

- Initialisée par:

```
u = ``Bonjour tout le monde'';
```

- Ne peut pas être modifiée On ne peut pas remplacer un caractère par un autre.
- Ceci est possible avec `StringBuffer`

Chaînes de caractères : fonctions utiles

```
int l = u.length();
```

Donne le nombre de caractères de la chaîne

```
u.compareTo(v);
```

Donne pour résultat 0 si les deux chaînes sont égales, négatif si u vient avant v dans l'ordre lexicographique, positif sinon

Chaînes de caractères (fin)

- On imprime une chaîne par

```
String u = `` Universite Bordeaux 1'';
System.out.println(u);
```

- D'ailleurs tous les arguments de System.out.println sont transformés automatiquement en chaînes de caractères
- La concaténation se réalise par +

```
String x = ``Bonjour'';
String y = ``les amis'';
String z = x + y;
```

- L'affectation

```
String x = ``bonjour'';
String y = x;
```

ne recopie pas "bonjour" mais effectue une référence sur cette chaîne.

L'opération +

L'opération + désigne à la fois l'addition (entiers, flottants) et la concaténation de chaînes

```
class Plus{

public static void main(String[] arg){
    String x = " Bonjour ", y = "les amis ";
    String z = x + y;
    x = "Au revoir";
    System.out.println(z);
    System.out.println(x);
    int a = 3, b = 5;
    System.out.println(a + z + b);
    System.out.println(a + b + z);
    System.out.println(z + a + b);
  }
}
```

```
java Plus
        Bonjour les amis
        Au revoir
        3 Bonjour les amis 5
        8 Bonjour les amis
        Bonjour les amis 35
```

Les arguments de main

- C'est la variable déclarée dans `main(String[] arg)`
- Ainsi `arg` est un tableau de chaînes de caractères.
- A l'exécution de `java Nomclasse abc xyz`
- Le nombre d'arguments est `arg.length`
- Le premier argument est `arg[0]` le deuxième `arg[1]` etc ..
- Ces arguments sont de type `String`
- Il faut parfois les transformer en des nombres entiers par `Integer.parseInt`

Exemple d'entrées avec des arguments pour `main`

```
public class Additionner{
    public static void main (String[] arg){
        int a = Integer.parseInt(arg[0]);
        int b = Integer.parseInt(arg[1]);
        int res  = a + b;
        System.out.println (c);
    }
}
```