

Informatique 01

Programmation en Java

Tris élémentaires

François Bourdoncle

`Francois.Bourdoncle@ensmp.fr`

`http://www.ensmp.fr/~bourdonc/`

Tronc commun d'Informatique

<http://w3.edu.polytechnique.fr/informatique/inf01.html>

- Contenu
 - Principes de la programmation
 - Algorithmes fondamentaux
- Langages de programmation : Java, CAML
 - Java
 - CAML
- Organisation
 - 10 PC + 10 TD
 - HC + CC
 - Projet (1000 lignes environ)
- Equipement
 - Stations de travail Unix
 - Ordinateurs personnel
 - Réseau interne – Internet
- Supports
 - Poly Cori-Lévy
 - Bibliothèque
- Délégué

Majeures (2ème année)

- [(M1)] Mathématique et Informatique (Jean Vuillemin)
 - Calculs et Preuves
 - Architecture du calcul
 - Codes correcteurs
 - Analyse et synthèse d'images
 - Arithmétique et Cryptographie
 - Calculs symboliques
 - Composants programmables
 - Images : analyse et synthèse
 - Interprète Scheme réparti
- [(M2)] Informatique (Robert Cori)
 - Langages et compilation
 - Structures de la programmation
 - Algorithmique et problèmes NP
 - ⇒ Systèmes d'exploitation et réseaux d'ordinateurs
 - Bases de données
 - Calcul parallèle
 - Programmation logique par contraintes

Activité de recherche

- Laboratoires
 - Digital Equipment Corporation
Laboratoire de Recherches de Paris
 - École des Mines de Paris
Centre de Mathématiques Appliquées
- Interprétation abstraite
- Langages orientés-objet (langage “Jazz”)
- Recherche documentaire (“Refine” d’AltaVista)
- `Francois.Bourdoncle@ensmp.fr`
- `http://www.ensmp.fr/~bourdonc`

Le langage Java

- Sun Microsystems
- Syntaxe à la C++
- Fortement typé
- Orienté-objet
- Gestion automatique de la mémoire
- Portable
- Bibliothèques standard
- Internet (applets, etc.)

Premier programme Java

```
class TD1 {
    public static void main(String[] args) {
        final int N = 10;
        int i;

        i = 0;
        while (i < N) {
            write("i -> " + i);
            if ((i % 2) == 0) {
                writeln(" [pair]");
            } else {
                writeln(" [impair]");
            }
            i = i + 1;
        }
    }
}
```

```
i -> 0 [pair]
i -> 1 [impair]
i -> 2 [pair]
i -> 3 [impair]
i -> 4 [pair]
i -> 5 [impair]
i -> 6 [pair]
i -> 7 [impair]
i -> 8 [pair]
i -> 9 [impair]
```

Syntaxe de base

- Instruction terminées par un point-virgule
- Groupe d'instructions entre accolades
- Déclaration de variable

```
    TYPE VARIABLE;  
    TYPE VARIABLE = VALEUR;  
final TYPE VARIABLE = VALEUR;  
    TYPE[] VARIABLE = new TYPE[TAILLE];
```

- Affectation

```
VAR = EXPR;  
VAR[INDICE] = EXPR;
```

- Instruction de contrôle

```
if (E) { INST... } else { INST... }  
for (INIT ; COND ; SUIV) { INST... }  
while (EXPR) { INST... }  
do { INST... } while (EXPR);
```

- Affichage

```
write(CHAINE);  
writeln(CHAINE);
```

Opérateurs

- Affectation : =
- Arithmétiques : + - * / %
- Comparaison : == != < <= > >=
- Booléens : && || !
- Concaténation de chaînes : +
- Comparaison de chaînes : `s1.equals(s2)`
- Accès aux caractères : `s.charAt(i)`
- Points délicats
 - Confusion entre = et ==
 - Confusion entre comparaison des *références* et comparaison de la *valeur* des objets.

Tris élémentaires

- Exercice : classement d'élèves par ordre alphabétique
- Découpage en modules fonctionnels
- Structures de données
- Algorithmes, complexité

Tri (variables globales)

```
class Tri {  
    // Tableau à trier  
    static String[] noms;  
  
    // Fonctions de tri  
    static void trier() { ... }  
  
    // Fonction d'impression  
    static void imprimer() { ... }  
  
    // Fonction principale  
    public static void main(String[] args) {  
        // Initialisation du tableau de noms  
        noms = new String[3];  
        noms[0] = "Annie";  
        noms[1] = "Julie";  
        noms[2] = "Armand";  
  
        trier();  
        imprimer();  
    }  
}
```

Tri (variables locales)

```
class Tri {
    static void trier(String[] t)
        { ... }

    static void imprimer(String[] t)
        { ... }

    public static void main(String[] args) {
        String[] noms = new String[] {
            "Annie",
            "Julie",
            "Armand"
        };

        trier(noms);
        imprimer(noms);
    }
}
```

Tri (ligne de commande)

```
% java Tri Annie Julie Armand
```

```
class Tri {  
    static void trier(String[] t)  
        { ... }  
  
    static void imprimer(String[] t)  
        { ... }  
  
    public static void main(String[] args) {  
        trier(args);  
        imprimer(args);  
    }  
}
```

Tri (élèves)

```
class Eleve {
    String nom;
    float note;
}

class Tri {
    static void trier(Eleve[] eleves)
        { ... }

    static void imprimer(Eleve[] eleves)
        { ... }

    public static void main(String[] args) {
        Eleve[] eleves = new Eleve[3];
        eleves[0] = new Eleve();
        eleves[0].nom = "Annie";
        eleves[0].note = 13.0;
        eleves[1] = new Eleve();
        eleves[1].nom = "Julie";
        eleves[1].note = 7.0;
        eleves[2] = new Eleve();
        eleves[2].nom = "Armand";
        eleves[2].note = 18.0;

        trier(eleves);
        imprimer(eleves);
    }
}
```

Tri (constructeurs)

```
class Eleve {
    String nom;
    float note;

    Eleve(String s, float x) {
        nom = s;
        note = x;
    }
}

class Tri {
    static void trier(Eleve[] eleves)
        { ... }

    static void imprimer(Eleve[] eleves)
        { ... }

    public static void main(String[] args) {
        Eleve[] eleves = new Eleve[] {
            new Eleve("Annie", 13.0),
            new Eleve("Julie", 7.0),
            new Eleve("Armand", 18.0)
        };

        trier(eleves);
        imprimer(eleves);
    }
}
```

Tri par sélection

```
static boolean meilleur(Eleve e1, Eleve e2) {  
    return (e1.note > e2.note);  
}
```

```
static void echanger(Eleve[] eleves,  
                    int i1, int i2) {  
    Eleve e = eleves[i1];  
    eleves[i1] = eleves[i2];  
    eleves[i2] = e;  
}
```

```
void trier(Eleve[] eleves) {  
    final int N = eleves.length;  
  
    for (int i = 0 ; i < N - 1 ; ++i) {  
        int m = i;  
        for (int j = i + 1 ; j < N ; ++j) {  
            if (meilleur(eleves[m], eleves[j])) {  
                m = j;  
            }  
        }  
        echanger(eleves, i, m);  
    }  
}
```

- Ordre du tri ?
- Complexité dans le cas le pire ?

Tri à bulles

```
static void trier(Eleve[] eleves) {
    final int N = eleves.length;

    for (int i = N - 1 ; i >= 0 ; --i) {
        for (int j = 1 ; j <= i ; ++j) {
            if (meilleur(eleves[j - 1], eleves[j])) {
                echanger(eleves, j - 1, j);
            }
        }
    }
}
```

- Ordre du tri ?
- Complexité dans le cas le pire ?
- Complexité en moyenne (comparaisons, échanges) ?

Tri par insertion

```
static void trier(Eleve[] eleves) {
    final int N = eleves.length;

    for (int i = 1 ; i < N ; ++i) {
        Eleve e = eleves[i];
        int j = i;
        while (j > 0 && meilleur(eleves[j - 1], e)
        {
            eleves[j] = eleves[j - 1];
            --j;
        }
        eleves[j] = e;
    }
}
```

- Ordre du tri ?
- Complexité en moyenne (comparaisons) ?
- Sentinelle (plus mauvais élève)

Exercices

- Tri en temps linéaire (tri du mécanographe)
- Animation des algorithmes de tri
- Complexité théorique maximale du tri
- Mesure effective de la complexité moyenne du tri par insertion, par tirage aléatoire