Les petits devant, les grands derrière

Jean-Jacques.Levy@inria.fr

http://para.inria.fr/~levy

tel: 01 39 63 56 89

Catherine Bensoussan

cb@lix.polytechnique.fr

Aile 00, LIX

tel: 34 67

http://w3.edu.polytechnique.fr/informatique/

Plan

1. Classes
2. Objets
3. Tris élémentaires
4. Recherche en table élémentaire
5. Graphique

Classes

- **Une classe représente un ensemble de variables et de fonctions**
- `class` *identificateur* { *liste_d'instructions* }
- $C.f$ **pour désigner la fonction f de la classe C**
  `Math.random, Integer.parseInt`
- $C.x$ **pour désigner la variable x de la classe C**
  `Integer.MIN_VALUE, Integer.MAX_VALUE, Float.NaN,`
  `Byte.MIN_VALUE, Byte.MAX_VALUE,`
  `System.out, System.err, System.in`
- **le nom de la classe est facultatif pour référencer une variable ou une fonction de la classe courante.**

Visibilité des noms dans les classes

- **par défaut, on peut accéder à tous les champs des classes figurant dans le même répertoire ("paquetage courant"),**
- **on ne peut accéder qu'aux champs publics d'une classe figurant dans un autre paquetage (`public`).**
- **il y a des restrictions d'accès pour les champs déclarés privés (`private`).**

Objets

- Les classes sont aussi des types de données
- Les objets sont les valeurs de type classe.

```
class Date {

  int    j; // jour
  int    m; // mois
  int    a; // annee
}

class BogueVirtuel {
  public static void main (String[ ] args) {

    Date an0 = new Date();
    an0.j = 1; an0.m = 1; an0.a= 1970;

    Date a2k = new Date();
    a2k.j = 1; a2k.m = 1; a2k.a= 2000;

    System.out.println (an0.j + "/" + an0.m + "/" + an0.a);
    System.out.println (a2k.j + "/" + a2k.m + "/" + a2k.a);
  }
}
```

Objets – 2

- $C\ \ o;$ **pour déclarer l'objet o de classe C**
  `String s; Date s;`

- $o.f$ **pour désigner la méthode f de l'objet o**
  `s.charAt, s.compareTo, System.out.println, System.err.println`

- $o.x$ **pour désigner le champ x de l'objet o**
  `d.j, d.m, d.a`

Objets – 3

- **un champ `static` n'existe qu'en un seul exemplaire pour toute la classe.**
- **un champ non statique correspond à une case mémoire différente pour tout objet instance de cette classe. On peut l'initialiser à la création de l'objet avec un constructeur.**
- **un constructeur est une fonction anonyme (non statique) que l'on peut déclarer dans toute classe.**

Exemple

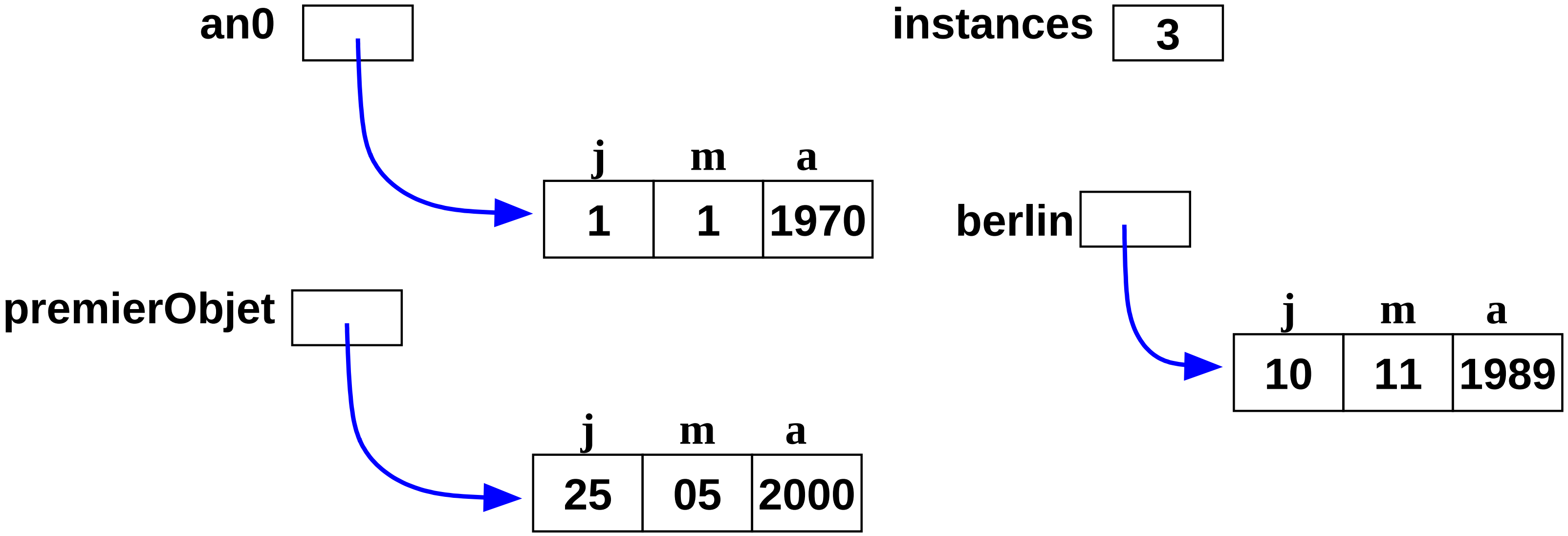
```
class Date {
  int    j,  m,  a;
  static int instances = 0;

  Date (int jour, int mois, int annee) {
    j = jour; m = mois; a = annee; ++instances;
  }

  static Date an0 = new Date (1, 1, 1970);
}
```

Objets – 4

```
class Test {
  Date premierObjet = new Date (25, 05, 2000);
  Date berlin = new Date (10, 11, 1989);
}
```



Objets – 5

Deux méthodes utiles à définir dans une classe

```
class Date {
  ...
  public boolean equals (Date d) {
    return j == d.j && m == d.m && a == d.a;
  }

  public String toString () {
    return j + "/" + m + "/" + a;
  }
```

Par convention, `toString` est appelé par + quand il a le sens de la concaténation des chaînes de caractères, et

```
System.out.print (x);
```

équivaut à

```
System.out.print (x.toString());
```

On peut donc écrire `System.out.println (d)` dans le cas des dates.

Recherche d'un minimum dans un tableau

```
static int minimum (int[ ] a) {
    r = Integer.MAX_VALUE;
    for (i=0; i < a.length; ++i)
        if ( a[i] < r )
            r = a[i];
    return r;
}
```

Ca marche pour le cas vide. (Il faut toujours y penser!).

- n **comparaisons (si** n **est la longueur du tableau)**
- **nombre total d'opérations est** $O(n)$
  (algorithme linéaire en temps),
  Il a une complexité linéaire.

Tri sélection

Donné: un tableau a de n éléments.

Résultat: le même tableau trié en ordre croissant, c'est à dire $a_i \leq a_j$ pour $0 \leq i < j < n$.

Algorithme:

- **On cherche l'emplacement du minimum de a,**
- **On échange l'élément de tête de a avec le minimum**
- **On recommence sur le tableau moins le premier élément, . . . tant que le tableau à traiter n'est pas vide.**

Echange de deux éléments

- **pour échanger les valeurs de deux variables, on effectue une permutation circulaire avec une troisième variable (auxiliaire).**

```
int x = 1, y = 2;

int aux = x;
x = y;
y = aux;
```

Peut-on faire mieux?

Tri sélection – le programme

```
static void triSelection (int[ ] a) {
    int   i_min, t;

    for (int i = 0; i < a.length - 1; ++i) {
        i_min = i;
        for (int j = i+1; j < a.length; ++j)
           if (a[j] < a[i_min])
               i_min = j;
        t = a[i_min]; a[i_min] = a[i]; a[i] = t;
    }
}
```

- n^2 comparaisons (si n est la longueur du tableau)
- nombre total d'opérations est $O(n^2)$
  (algorithme quadratique en temps),
  Il a une complexité quadratique.

Tri par insertions

Comme pour trier un paquet de cartes

```
static void triInsertion (int[ ] a) {
  int   j, v;

  for (int i = 1; i < a.length; ++i) {
      v = a[i]; j = i;
      while (j > 0  &&  a[j-1] > v) {
          a[j] = a[j-1];
          --j;
      }
      a[j] = v;
  }
}
```

- $I(a)$ **comparaisons (le nombre d'inversions dans** a**)**
- **nombre total d'opérations est** $O(n^2)$ **dans le cas pire (algorithme quadratique en temps),**
  Il a une complexité quadratique.
- **bon quand** a **est presque trié.**

Le tri Shell [D. L. Shell 1959]

```
static void triShell (int[ ] a) {

  int h = 1; do h = 3*h + 1; while ( h <= a.length );
  do {
      h = h / 3;
      for (int i = h; i < a.length; ++i)
          if (a[i] < a[i-h]) {
              int   v = a[i], j = i;
              do {
                  a[j] = a[j-h];
                  j = j - h;
              } while (j >= h  &&  a[j-h] > v);
              a[j] = v;
          }
  } while ( h > 1);
}
```

Pas plus que $O(n^{3/2})$ comparaisons !!
Bon pour de gros fichiers (celui du noyau Maple)

Recherche en table

Une table est une ensemble de paires (k, d) où k est une clé et d une donnée. Par exemple, k est un nom de personne, d est son numéro de téléphone (ou son adresse e-mail).

On cherche une clé dans une table pour obtenir la donnée correspondante.

Recherche séquentielle

```
static int recherche (String x, String[ ] nom, int[ ] tel) {

  for (int i = 0; i < nom.length; ++i)
      if (x.equals(nom[i]))
          return tel[i];
  return -1;
}
```

Recherche en table – bis

nom	tel
Robert	05 56 84 60 88
Jean-Jacques	01 39 63 56 89
Philippe	01 69 33 34 89
Sam	01 69 33 46 15
Catherine	01 69 33 34 67
Guillaume	03 83 59 30 21

Recherche dichotomique en table

On suppose la table ordonnée en ordre croissant.

```
static int rechercheDichotomique (String x, String[ ] nom, int[ ] tel) {
  int   i, cmp, g = 0, d = nom.length-1;
  do {
      i = (g + d) / 2;
      cmp = x.compareTo(nom[i]);
      if (cmp == 0)
          return tel[i];
      if (cmp < 0)
          d = i - 1;
      else
          g = i + 1;
  } while (g <= d);
  return -1;
}
```

Complexité? Peut-on aller plus vite?
interpolation, hachage

Quelques primitives graphiques

```
// variables globales
static Grafport gc;
static int background, foreground, hilite, hilite2;
static int epsilonX, epsilonY, pasX, pasY, x0, y0;

static void ouvrirGraphique (int n, int valeurMax) {

  g = new Grafport();

  int sizeX = Math.min(500, (n+5) * 10),
      sizeY = Math.min(400, (valeurMax+5) * 5);

  Rect r = new Rect();
  r.left = 10; r.right = (short)(r.left + sizeX);
  r.top = 10; r.bottom = (short)(r.bottom + sizeY);
  g.setDrawingRect(r);
```

Graphique en Java – suite

```
    foreground = g.redColor;   background = g.Whitecolor;
    hilite = g.blueColor;   hilite2 = g.yellowColor;

    epsilonX = Math.max(7, sizeX / (n + 3));
    epsilonY = Math.max(20, sizeY / valeurMax);
    pasX = Math.max (1, (sizeX - 2 * epsilonX) / n);
    pasY = Math.max (1, (sizeY - epsilonY) / valeurMax);
    x0 = epsilonX; y0 = sizeY - epsilonY;
  }

  static void tracerElement (int[ ] a, int i, int color) {
    int xi = x0 + i * pasX;
    int yi = y0 - v[i] * pasY;
    g.foreColor (color);

    Rect r = new Rect();
    g.setRect(r, xi, yi, xi+pasX, y0);
    g.paintRect (r);
  }
```

Graphique en Java – suite

La documentation sur Maclib se trouve en `Informations diverses` sur la page Web du cours
`http://w3.edu.polytechnique.fr/informatique/`

Cette bibliothèque est une version simplifiée de AWT. En outre, elle est compatible avec la même bibliothèque en C, Pascal ou ML; elle existe sur stations Unix et sur MacIntosh.

Modules graphiques

`MacLib` est un sous-ensemble de AWT suffisant pour notre cours (écrit par Ph. Chassignet)

```
blackColor                      blueColor
redColor                        cyanColor
greenColor                      magentaColor
whiteColor                      yellowColor
patCopy                         patXor
addPt(Point, Point)             backColor(Color)
backColor(int)                  button()
buttonState()                   charWidth(char)
drawChar(char)                  drawLine(int, int, int, int)
drawString(String)              drawText(char[], int, int)
emptyRect(Rect)                 equalPt(Point, Point)
equalRect(Rect, Rect)           eraseArc(Rect, int, int)
eraseOval(Rect)                 erasePolygon(Point[], int)
eraseRect(Rect)                 eraseRoundRect(Rect, int, int)
foreColor(Color)                foreColor(int)
frameArc(Rect, int, int)        frameOval(Rect)
framePolygon(Point[], int)      frameRect(Rect)
frameRoundRect(Rect, int, int)  getClick(Point)
getDrawingRect(Rect)            getKey()
getMouse(Point)                 getPen(Point)
getPort()                       getTextRect(Rect)
hideAll()                       initMacLib()
initQuickDraw()                 insetRect(Rect, int, int)
```

Modules graphiques

```
invertArc(Rect, int, int)        invertCircle(int, int, int)
invertOval(Rect)                 invertPolygon(Point[], int)
invertRect(Rect)                 invertRoundRect(Rect, int, int)
keyPressed()                     line(int, int)
lineTo(int, int)                 move(int, int)
moveTo(int, int)                 newPointArray(int)
offsetRect(Rect, int, int)       paintArc(Rect, int, int)
paintCircle(int, int, int)       paintOval(Rect)
PaintOval(Rect)                  paintPolygon(Point[], int)
paintRect(Rect)                  paintRoundRect(Rect, int, int)
penMode(int)                     penNormal()
penSize(int, int)                pt2Rect(Point, Point, Rect)
ptInRect(Point, Rect)            RGBBackColor(RGBColor)
RGBForeColor(RGBColor)           sectRect(Rect, Rect, Rect)
setDrawingRect(Rect)             setPort(GrafPort)
setPt(Point, int, int)           setRect(Rect, int, int, int, int)
setText(Frame)                   setTextRect(Rect)
showDrawing()                    showText()
stringWidth(String)              subPt(Point, Point)
sysBeep(int)                     textWidth(char[], int, int)
tickCount()                      trackMouse(Point)
unionRect(Rect, Rect, Rect)      waitClickDown()
waitClickUp()
```