Chapitre VI

Divers, tableaux, conversion de types, noms

Types énumérés

- Exemple:
  - `enum Couleur {PIQUE, CŒUR, CARREAU, TREFLE,}`
  - définit des constantes énumérées (champs static de la classe)
  - on peut définir des méthodes dans un enum
  - des méthodes pour enumeration E
    - public static E[] values() retourne les constantes dans l'ordre de leur énumeration
    - public static E valueOf(String nom) la constante associé au nom
  - un type enum étend implicitement java.lang.Enum (aucune classe ne peut étendre cette classe)

Tableaux

- collection ordonnée d'éléments,
- les tableaux sont des Object
- les composants peuvent être de types primitifs, des références à des objets (y compris des références à des tableaux),

Tableaux

- int [] tab= new int [3];
  - déclaration d'un tableau d'int
  - initialisé à un tableau de 3 int
- indices commencent à 0
- contrôle de dépassement
  - ArrayIndexOutOfBoundException
- length donne la taille du tableau

Tableaux

- un tableau final: la référence ne peut être changée (mais le tableau référencé peut l'être)
- tableaux de tableaux:
- exemple:

```
public static int[][] duplique(int[][] mat){
        int[][] res= new int[mat.length][];
        for(int i=0;i<mat.length;i++){
            res[i]=new int[mat[i].length];
            for (int j=0;j<mat[i].length;j++)
                res[i][j]=mat[i][j];
        }
        return res;
    }
```

Tableaux

- exemple:

```
public static void affiche(int [][] tab){
        for(int[] d:tab){
            System.out.println();
            for(int v:d)
                System.out.print(v+" ");
        }
    }
```

Iterable

- `interface Iterable<T>`
  - une seule méthode:

    `Iterator<> iterator()`

    (méthode par défaut forEach en java 8)
- Si w est un objet qui implémente Iterable<T>, d'une classe permet les boucles « for » sous la forme:

  ```
  for(T v: w){…}
  ```
  - correspondant à:

    ```
    Iterator<T> iter=w.iterator();
    while(iter.hasNext()){v=iter.next();…}
    ```

Tableaux

- `public static void arraycopy(Object src, int srcPos,Object dest, int destPos, int l)`

  copie les l éléments du tableau source à partir de srcPos vers dest à partir de destPos (dans System)

- la classe java.util.Arrays contient de nombreuses méthodes permettant de rechercher un élément (binarySearch) de copier (copyOf, copyOfRange) tester l'égalité (equals, deepEquals) trier (sort) etc.

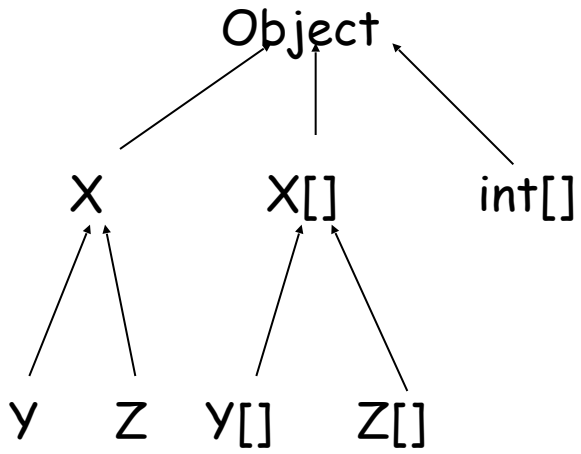
Initialisations

```
static int[][] pascal={
        {1},
        {1,1},
        {1,2,1},
        {1,3,3,1},
    };
String[] nombre= {"un", "deux", "trois", "quatre"};
```

Exemple

```
for(int i=1;i<p.length;i++){
     p[i]=new int[i+1];
     p[i][0]=1;
     for(int j=1; j<i;j++){
         p[i][j]=p[i-1][j-1]+p[i-1][j];
     }
     p[i][i]=1;
}
```

Tableau et héritage



- `Y[] yA=new Y[3];`
- `X[] xA=yA; //ok`
- `xA[0]=new Y();`
- `xA[1]=new X(); //non`
- `xA[1]=new Z(); //non`

Noms

- il y a 6 espaces de noms
  - package
  - type
  - champs
  - méthode
  - variable locale
  - étiquette

Noms!?

```
package divers;
class divers{
    divers divers(divers divers){
        divers:
            for(;;){
            if (divers.divers(divers)==divers)
                break divers;
            }
            return divers;
    }
}
```

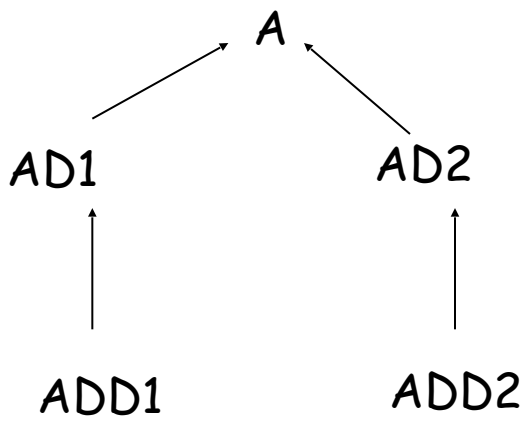
Trouver la bonne méthode

- Il faut pouvoir déterminer une seule méthode à partir d'une invocation avec des paramètres
- problèmes: héritage et surcharge
  - (en plus des problèmes liés à la généricité)
- principe trouver la méthode "la plus spécifique"

Règles

1. déterminer dans quelle classe chercher la méthode (uniquement par le nom)
2. trouver les méthodes dans la classe qui peuvent s'appliquer
    1. sans "boxing" sans nombre variable d'arguments
    2. avec boxing
    3. avec un nombre variable d'arguments
3. si une méthode a des types de paramètres qui peuvent être affectés à une autre des méthodes de l'ensemble -> la supprimer
4. s'il ne reste qu'une méthode c'est elle (sinon ambiguïté sauf s'il s'agit de méthodes abstraites)

Exemple




```
void f(A a,AD2 ad2)//un
void f(AD1 ad1,A a)//deux
void f(ADD1 add1, AD2 s)//trois
void f(A … a)//quatre

f(Aref, AD2ref);//a
f(ADD1ref, Aref);//b
f(ADD1ref, ADD2ref);//c
f(AD1ref, AD2ref);//d
f(AD2ref, AD1ref);//e
```

Exemple (suite)

- (a) correspond exactement à (un)
- (b) correspond à (deux)
- (c) peut correspondre aux trois premiers mais (un) est moins spécifique que (trois) idem entre (un) et (trois) d'où résultat (trois)
- (d) (un) et (deux) ne peuvent s'éliminer
- (e) uniquement (quatre)