CH 10 Les conteneurs, ou structures collectives en Java

Stocker ensemble des objets similaires. On parle d’une collection d’objets.

Ex : utilisateurs inscrits, pièces d’un jeu, …

Les opérations courantes :

-ajout, suppression, recherche, tri, …

1) Les tableaux : simples mais limités

Eléments rangés en séquence. Accès à 1 élément par son indice direct.

Syntaxe aisée avec []. La classe Arrays fournit des méthodes.

Les tableaux peuvent manquer de souplesse :

* Taille choisie à l’instanciation et non modifiable ensuite
* Parcours par un indice = source de bugs
* ….

Le tableau peut suffire souvent. Si ce n’est pas le cas, on étudiera d’autres collections Java.

N.B. Charger les classes de ce chapitre par import java.util.\* ;

2) La classe Arrays pour manipuler les tableaux

Elle contient des méthodes statiques prenant un tableau en paramètre et réalisant des opérations telles que :

* Tri par la méthode sort(…)
* Comparaison du contenu : equals(…)
* Copie d’un tableau : copyOf(…)
* Recherche d’un élément dans un tableau trié : binarySearch(…)

Ex : pour trier un tableau tab : Arrays.sort(tab) ;

3) Les comparateurs

Trier un tableau => ses éléments disposent d’une relation d’ordre naturelle. Mais les types objets ?

Il faut décrire explicitement une relation d’ordre au moyen d’une objet comparateur.

2 façons d’implanter un comparateur

3.1) Votre objet implante l’interface Comparable

Cette interface oblige à implanter une méthode nommée compareTo(Object o)

Elle doit retourner un entier :

* Négatif si this est plus petit que o
* Nul si this est égal à o
* Positif si this est plus grand que o

Ex : une classe Paire (de 2 entiers) qu’on trie par rapport à la somme des 2 entiers

Public class Paire implements Comparable{

Public int x,y ;

Public int compareTo(Object o){

Return (this.x + this.y) – (( (Paire) o).x + ((Paire) o).y) ;

}

}

Soit tabPaires un tableau de Paire

Paire [] tabPaires = new Paire[10] ;

tabPaires[0] = new Paire(5,8) ;…………..

Pour le trier :

Arrays.sort(tabPaires) ;

3.2) Une classe séparée implante l’interface Comparator

On peut décrire autant de ces classes que de relations d’ordre souhaitées. (ex : par ordre croissant, ou décroissant)

Dans Comparator ,2 méthodes :

Boolean equals(Object o)

Int compare(Object o1, Object o2) (retourne une entier < 0 si o1 < o2, > 0 si o1 > o0 ou 0 si o1 = o2)

Public class DecroissantPaire implements Comparator{

Public int compare(Object o1, Object o2){

Paire p1 = (Paire) o1 ;

Paire p2 = (Paire) o2 ;

Return (p2.x + p2.y) – (p1.x + p1.y) ;

}

}

Pour l’utiliser :

Arrays.sort(tabPaire, new DecroissantPaire()) ;

Trie les paires par ordre décroissant

4) Conteneurs en Java

Le package java.util définit plusieurs conteneurs – Il y’en a 2 types

* Collection : groupement d’objets individuels
* Map : -------------------------d’associations clé / valeur

4.2) ArrayList : le conteneur à tout faire

Genre de tableau de taille variable. Il implante Collection => possède toutes les méthodes de Collection

Mais il dispose en plus d’indices, comme un tableau : ATTENTION !! syntaxe [] interdite

Exemple :

Add(Object o) ajoute l’objet o à la fin

Add(Object o , int i) ajouts l’objet o à la place i (et décale les objets suivants d’une case)

Remove(int i) libere la place i sans laisser de trou

Get (int i) : retourne l’élément d’indice i

Set(int i, Object o) : écrase la case i et la remplace avec o

indiceOf(Object o) : indice de o si présent

….

Consultez l’API JAVA pour voir ttes les méthodes de ArrayList

Souvent l’ArrayList suffit.

Ex : ArrayList Al = new ArrayList()

Al.add(new Rectangle(2,3)) ;

Al.add (new Cercle(8)) ;

Al.add(new Rectangle(5,5)) ;

Al.remove(2) ;

…

Exercice :

Implantez une file (FiFo = First in Fisrt out) avec un ArrayList

4.3) Autres contenues de type Collection

Vector : versin obselète de ArrayList

LinkedList : concept de liste chainée

HashSet : table de hachage

TreeSet : structure arborescente

5)Les itérateurs

Ce sont des objets du type java Iterator (définis dans java.utils) servant à parcourir les collections

Un itérateur est à un moment donnée

* Positionné sur un élément
* Peut retourner cet élément et passer au suivant (next())
* Supprimer le dernier élément renvoyé (remove())
* Reste-il des éléments ? (hasNext())

Pour obtenir un itérateur sur une collection (nommée par exemple col) : invoquer la méthode itérator()

Iterator it = col.Iterator() ;

Exemple :

Public class Iter{

Public static void main(String [] args){

ArrayList tab = new ArrayList() ;

For(int i=0 ; i<10 ; i++){

Tab.add(new A()) ;

}

Iterator it = tab.iterator() ;

While(it.hasNext()){

System.out.println(it.next()) ;

}

}

}

Autre exemple avec la liste al de tout à l’heure

Iterator itForme = al.iterator() ;

While(itForme.hasNext()){

System.out.println(((Forme) itForme.next()).perimetre()) ;

}

CH 11 Polymorphisme

Un objet peut exister sous plusieurs formes : soit comme une instance de sa vraie classe, soit come une instance de sa super-classe.

1. Transtypage ascendant et descendant
   1. Inclusion de types : transtypage ascendant

Soit n € N et x € R

X <- n ne pose pas de problèmes car N c R

Par contre n <- w=x pose problème de précision : x est plus précis que n.

Sue faire de la partie décimale de x ?

En java

Int n=5 ;

Double =x ;

X=n ; //pas de problèmes

Int « c » double : on réalise un transtypage ascendant.

Transtypage ascendant = toujours valide et autorisée, pas besoin de l’expliciter.

Ça marche aussi avec des objets définis par héritage.

Fille f = new File() ;

Mere m = f ; //OK transtypage

Syntaxiquement m est de type Mere, physiquement m est de type Fille .

Transtypage descendant

C’est dans l’autre sens :

Float x =3.14 ;

Int n= x ; // ne compile pas, il faut transtyper (caster) explicitement

Int n = (int) x ; //tronque les décimales

Version objet :

Mere m = new Mere() ;

Fille f = m ; //ne marche pas

Fille f = (Fille) m ; //transtypage descendant

Transtypage descendant doit être directement déclaré.

Ça garantit que le code puisse être compilé.

Ça ne garantit pas que ça marche à l’éxecution.

Exemple :

Mere m = new Mere() ;

Pairep = (Paire) m ; //Ok compilation

Mais à l’éxecution ça plantera.