Les collections

Plan du cours

- Généralités sur les collections
- Collections et itérateurs
- Maps
- Utilitaires : trier une collection et rechercher une information dans une liste triée

Définition

- Une collection est un objet qui contient d'autres objets
- Par exemple, un tableau est une collection
- Le JDK fournit d'autres types de collections sous la forme de classes et d'interfaces
- Ces classes et interfaces sont dans le paquetage java.util

Généricité

- Avant le JDK 5.0, il n'était pas possible d'indiquer qu'une collection du JDK ne contenait que des objets d'un certain type ; les objets contenus étaient déclarés de type Object
- A partir du JDK 5.0, on peut indiquer le type des objets contenus dans une collection grâce à la généricité: List<Employe>
- Ce cours privilégie les collections génériques

Les interfaces: Collection

- Des interfaces dans 2 hiérarchies principales :
  - List<E>
  - Map<K,V>
- List correspond aux interfaces des collections proprement dites
- Map correspond aux collections indexées par des clés ; un élément de type V d'une Map est retrouvé rapidement si on connaît sa clé de type K(comme les entrées d'un dictionnaire ou les entrées de l'index d'un livre)

Classes étudiées

- Nous étudierons essentiellement les classes ArrayList et, HashMap comme classes d'implémentation de List et de Map
- Elles permettront d'introduire des concepts et informations qui sont aussi valables pour les autres classes d'implantation

Collections du JDK 1.1

- Les classes et interfaces suivantes, fournies par le JDK 1.1,
  - Vector
  - HashTable

existent encore mais il vaut mieux utiliser les nouvelles classes du JDK 1.2

- Il est cependant utile de les connaîtres car elles sont utilisées dans d'autres API du JDK

La classe Vector

- **Un vecteur est un tableau dynamique dont les éléments sont des références d'objets**
- est un tableau d'objets dont la taille peut varier au fil du temps

public Vector(): construit un vecteur vide

public final int size() :nombre d'élément du vecteur

public final void addElement(Object obj): ajoute l'objet référencé par obj au vecteur

public final Object elementAt(int index): référence de l'objet n° index du vecteur - les indices commencent à o

public final Enumeration elements(): l'ensemble des éléments du vecteur sous forme d'énumération

La classe Vector

- **public final Object firstElement():** référence du premier élément du vecteur

  public final Object lastElement(): référence du dernier élément du vecteur

  public final boolean isEmpty(): rend vrai si le vecteur est vide

  public final void removeElementAt(int index): enlève l'élément d'indice index

  public final void removeAllElements(): vide le vecteur de tous ses élément

Les listes : ArrayList

- La gestion des ArrayList est similaire à la gestion des tableaux puisque le programme crée une liste par ajout de données au fur et à mesure des besoins de l'utilisateur.
- Les donnés sont enregistrées dans l'ordre d'arrivée (indice)
- Les données enregistrées dans une ArrayList sont en réalité rangées dans un tableau interne créé par l'interpréteur. La taille du tableau interne est gérée automatiquement par Java.
- Lorsque la liste des éléments à ajouter dépasse la taille du tableau interne, un nouveau tableau est créé et les anciennes valeurs y sont copiés.

Manipulation d'une liste

- Déclaration : ArrayList liste=new ArrayList();
- Les méthodes :

✓ add(objet) // Ajoute un élément objet en fin de liste

✓ add(indice, objet) // insère un élément objet à l'indice donné

✓ get(indice) // retourne stocké à l'indice donné

✓ set(indice,objet) // remplace l'élément situé en position indice par l'objet

✓ size() // retourne le nombre d'élément dans la liste

✓ remove(indice) // supprime l'objet dont l'indice est donné par paramètre

Suite : ArrayList

- removeRange(i,j) // supprime tous les éléments compris entre les indices (i valeur comprise) et (j valeur comprise)
- clear() // supprime tous les éléments de la liste
- indexof(objet) // retourne l'indice dans la liste du premier objet donné ou -1 si objet n'existe pas
- lastIndexof // retourne l'indice dans la liste dernier objet donné ou -1 si objet n'existe pas
- contains (objet) // retourne true ou false si l'objet existe ou non dans la liste
- Iterator iterator() // étudié plus loin dans le cours

Remarque importante

- Une liste a la capacité de mémoriser n'importe quel type d'objet , il est donc nécessaire d'indiquer au compilateur à quel type correspond l'objet extrait.
- Object get(indice) // retourne un objet de type Object (la classe racine)
- Utilisation de Cast (transtypage)
- Les types génériques (la remède)

Exemple : liste des Points

```
import java.util.*;

Public class ListPoint {

private ArrayList liste;

public ListPoint (){  liste-=new ArrayList(); }

public void  ajouteUnPoint(){ liste.add(new Point()) ;}

public void afficheLePoints(){  Int nbPoint=liste.size();

if (nbPoint>0){ Point tmp;

for(int i=0; i<nbPoint; i++){

tmp=(Point) liste.get(i);

tmp.afficheUnPoint();} }

 else    System. out.println(« il n'y a pas e point dans cette liste }}
```

Les classes associées aux types intrinsèques

- Pour chaque type simple correspond une classe

✓ int → Integer

✓ char → Character

✓ double → Double

✓ float → Float

✓ long → Long

✓ short → Short

Passage de type simple à une classe correspondante

- Passage de int à Integer
- Integer x=new Integer(6);
- Passage de Integer à int
- int ix=x.intValue();

ArrayList : finale

- Les méthodes de la classe ArrayList permettent aussi la recherche d'un élément dans la liste grâce à la méthode indexof(objet), qui retrouve l'indice de l'objet donnée
- Dans le cas où la liste mémorise des objets complexe (tels que les données caractéristique d'un Point) la recherche n'est pas simple. En fait, la méthode indexof(objet) ne retrouve l'objet donné qu'à la seule condition qu'il soit totalement identique à celui stocké dans la liste

Les dictionnaires → HashMap<K,V>

- Organiser les données, non plus par rapport à un indice, mais par rapport à une clé explicite .
- La recherche dans un objet dans la liste s'effectue, non plus sur l'ensemble des données qui le composent mais sur une clé unique qui lui associée
- L'organisation de données, par association d'une clé à un ensemble de données, est appelée un dictionnaire.
- Une clé repère une et une seule valeur

Manipulation d'un dictionnaire

- Le type HashMap proposé par le langage java permet de réaliser simplement l'association clé-élément.
- Utilisation : HashMap listeclassee=new HashMap();
- Les méthodes :

✓ put(clé,objet) // place dans le dictionnaire l'association clé-objet

✓ get(clé) // retourne l'objet associé à la clé donné

✓ remove(clé) // supprime dans le dictionnaire l'association clé-objet à partir de la clé donné

✓ size() // taille du dictionnaire

✓ Iterator iterator()

Dictionnaire de Points

- Définir une clé d'association

- création du dictionnaire

Iterator : parcourir une collection

- Utilisation des plusieurs techniques pour parcourir et afficher les éléments d'une collection.
- Java propose un outil facile et efficace.
- Le parcours dans une collection d'objets, s'effectue par l'intermédiaire d'un objet de type Iterator que l'on applique à la collection.

Manipulation : Iterator

- Utilisation : Iterator it=(collection).iterator();

Les méthodes :

- hasNext() // détermine s'il existe encore des éléments dans la collection sur laquelle est appliquée l'itérateur
- next() // permet l'accès à l'élément suivant dans la collection.