STRUCTURES DE DONNÉES

A quoi ça sert ?

- Stocker des données en mémoire
- Ordonner / Classer
- Les structures peuvent être :
  - Statiques
  - Dynamiques

Différents types de structures

	Ajout/suppr	Accès	Classes en Java
Tableaux	-	++	ArrayList EnumSet EnumMap
Listes chaînées	++	-	LinkedList HashSet LinkedHashSet
Tables de hachage	++	++	HashSet HashMap
Arbres	+	+	TreeSet TreeMap

L'interface Iterable

- Une seule méthode :

Iterator<E> iterator()

- L'interface Iterator

  boolean hasNext()

  E next()

  void remove()

Comment s'en servir ?

- Avant Java 5

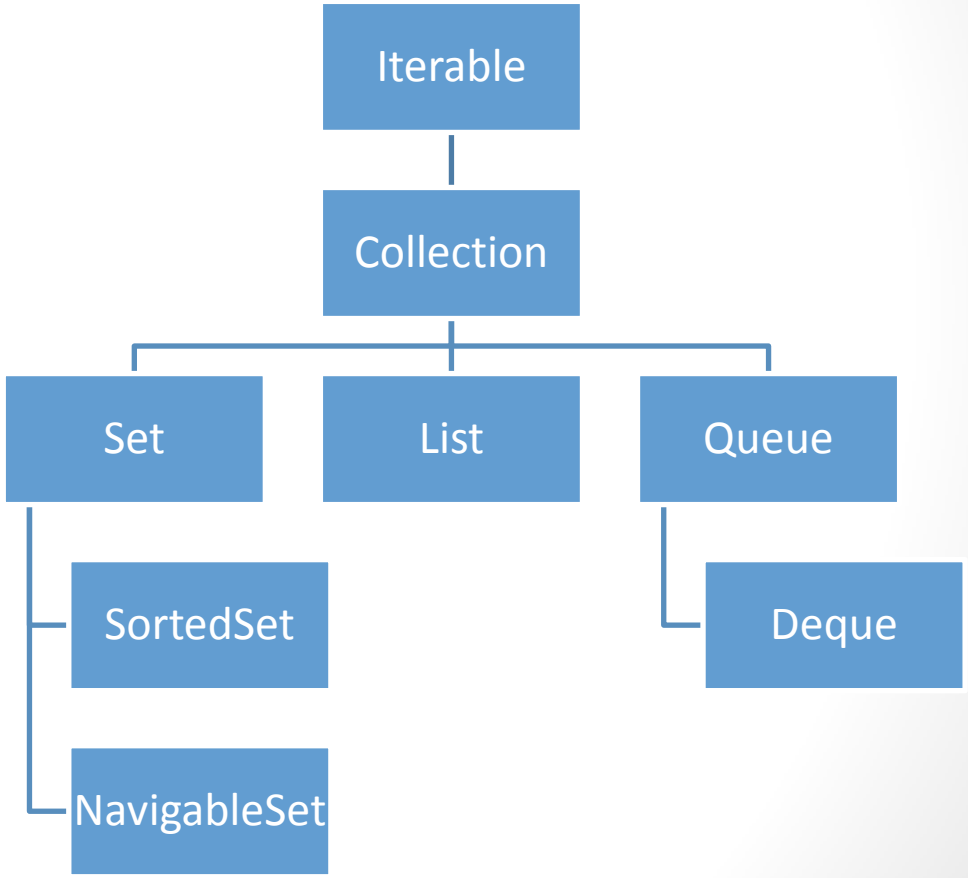
```
for(Iterator it = c.iteractor();itr.hasNext();)
    System.out.println(it.next());
```

- Depuis Java 5

```
for(Object o : c)
    System.out.println(o);
```

L'interface Collection

- Hérite de l'interface Iterable
- Pas d'association



Ses principales méthodes

- boolean **add**(E e)
- boolean **addAll**(Collection<? extends E> c)
- boolean **contains**(Object o)
- boolean **equals**(Object o)
- boolean **isEmpty**()
- Iterator<E> **iterator**()
- boolean **remove**(Object o)
- int **size**()
- Object[] **toArray**()

L'interface Set

- Pas de méthode spécifique
- L'ordre n'a pas d'importance
- Accepte les doublons

- SortedSet
  - Trie automatiquement et les retourne dans l'ordre
  - NavigableSet
    - Méthodes permettant de trouver les élements les plus proche d'un élément donné

- AbstractSet l'implémente
  - TreeSet, EnumSet et HashSet hérite de AbstractSet

La classe TreeSet

- implemente SortedSet
  - SortedSet<E> **subSet**(E fromElement, E toElement)
  - SortedSet<E> **headSet**(E toElement)
  - SortedSet<E> **tailSet**(E fromElement)
  - E **first**()
  - E **last**()

- implémente NavigableSet
  - E **lower**(E e)
  - E **floor**(E e)
  - E **ceiling**(E e)
  - E **higher**(E e)
  - NavigableSet<E> **descendingSet**()
  - Iterator<E> **descendingIterator**()

Besoin d'un Comparator

- Interface
  - int **compare**(T o1, T o2)
    - Nombre négatif si o1 est inférieur à o2
    - 0 si o1 = o2
    - Nombre positif si o1 est supérieur à o2
  - boolean **equals**(Object obj)
    - Vrai si obj est égal à l'objet courant
    - Faux sinon

L'interface Queue

- Éléments ajoutés à la queue d'une file
  - boolean **add**(E e)
  - boolean **offer**(E e)
  - E **remove**()
  - E **poll**()
  - E **element**()
  - E **peek**()

- Deque
  - Ajoute et supprime des éléments en tête ou en queue

L'interface List

- Accepte les doublons
- Ordre a une importance
- Accès aux éléments via un indice
- La classe qui l'implémente : AbstractList
- Les classes qui en héritent :
  - LinkedList
  - ArrayList
  - Vector
  - Stack

Caractéristiques

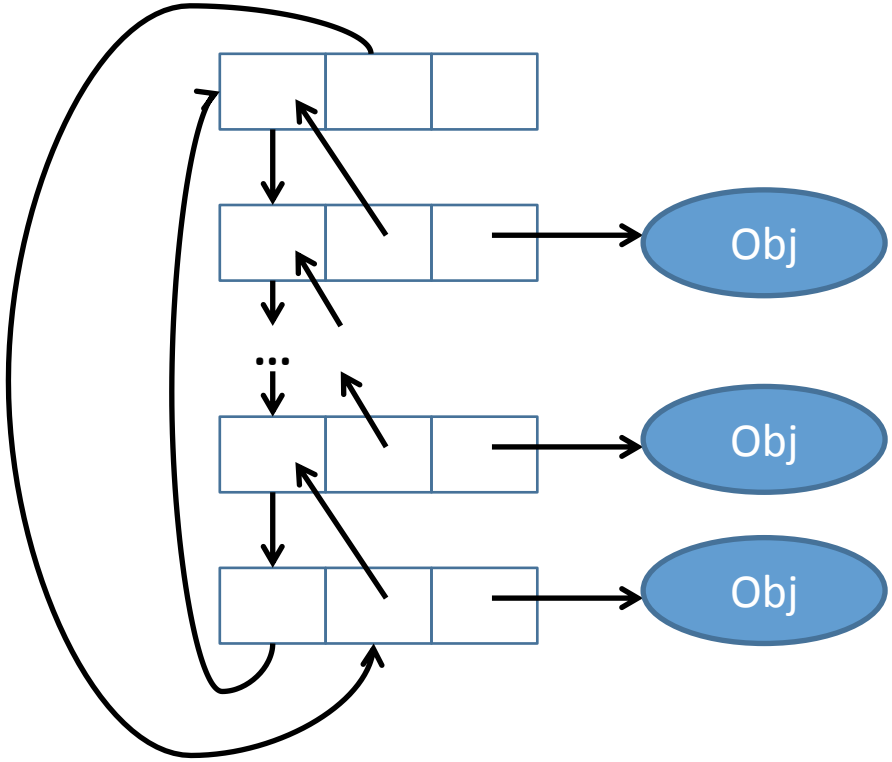
- Ses méthodes :
  - void **add**(int index, E element)
  - E **get**(int index)
  - int **indexOf**(Object o)
  - int **lastIndexOf**(Object o)
  - E **remove**(int index)
  - E **set**(int index, E element)

La classe ArrayList

- Tableau d'objet
  - Accède à un élément via un indice
  - Peut grandir et rétrécir à la demande

La classe LinkedList

- Liste doublement chainée circulaire



La classe stack

- Dernier entré, Premier sorti !

- Les méthodes spécifiques
  - boolean **empty**()
  - E **peek**()
  - E **pop**()
  - E **push**(E item)
  - int **search**(Object o)

L'interface Map

- Association clé-valeur
- SortedMap
  - Tri les valeurs dans l'ordre croissant des clés
  - NavigableMap
    - Méthodes permettant de trouver les élements les plus proche d'un élement donné
- Classe qui l'implémente : AbstractMap
- Classes qui héritent de AbstractMap
  - EnumMap
  - HashMap
  - TreeMap (implémente SortedMap et NavigableMap)

Principales méthodes

- boolean **containsKey**(Object key)
- boolean **containsValue**(Object value)
- boolean **equals**(Object o)
- V **get**(Object key)
- boolean **isEmpty**()
- V **put**(K key, V value)
- V **remove**(Object key)
- int **size**()
- Collection<V> **values**()