Structures de données

Christophe Damas
José Vander Meulen

Organisation

- 4 parties:
  - Collections & Maps (rappel) : 1 séance ½
  - Files de priorité : ½ séance
  - Graphes : 1 séance + projet (3 séances)
  - Arbres : 3 séances
  - Code de Huffmann : 2 séances
- Evaluation
  - En juin
    - 10 % projet intégré
    - 90 % examen sur machine
  - En septembre
    - 100 % examen sur machine

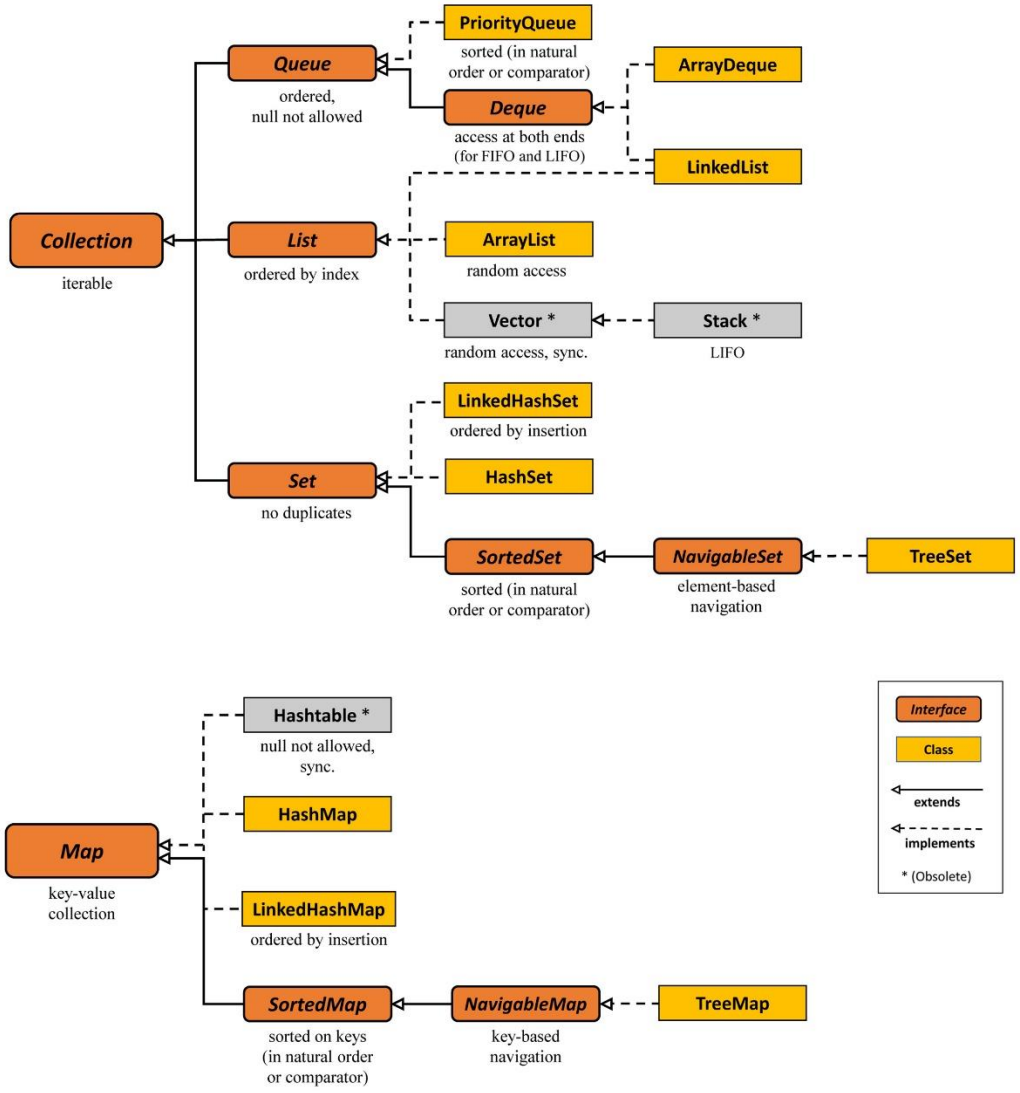
COLLECTIONS, MAPS: RAPPEL

Attention, cette présentation n'est qu'un bref rappel.
Pour plus d'informations, référez-vous à la javadoc.

vinci
HAUTE ÉCOLE LÉONARD DE VINCI

institut paul lambin

Java 8 - Collections & Maps



Structures de données JAVA utilisées dans ce cours

- List
  - ArrayList
- Queue
  - PriorityQueue (on en parlera la semaine prochaine)
- Deque
  - ArrayDeque
  - LinkedList
- Set
  - HashSet
  - TreeSet
- Map
  - HashMap
  - SortedMap

Interface List

- Séquence ordonnée

```
public interface List<E>{
        boolean isEmpty();
        int size();
        E get(int index);
        void add(int index, E element);
        void add(E element);
        E remove(int index);
        boolean contains(Object o);
        String toString();
}
```

- Implémentation:
  - Basée sur un tableau: ArrayList
  - Basée sur une liste doublement chainée: LinkedList

  Complexité ?

Interface Deque

- Collection linéaire qui permet l'ajout et la suppression des deux cotés
- Permet d'implementer une pile (LIFO), une file (FIFO),…

```
public interface Deque<E>,…{
        boolean isEmpty(); int size();
        boolean contains(Object o); String toString();

        //operation pour implementer une pile
        void push(E item);
        Object pop();
        Object peek();

        //operation pour implementer une file
        void add(E e);
        E poll();

        … (addFirst, addLast, removeFirst, removeLast, …)
}
```

- 2 implémentations
  - Basée sur un tableau: ArrayDeque
  - Basée sur une liste doublement chainée: LinkedList

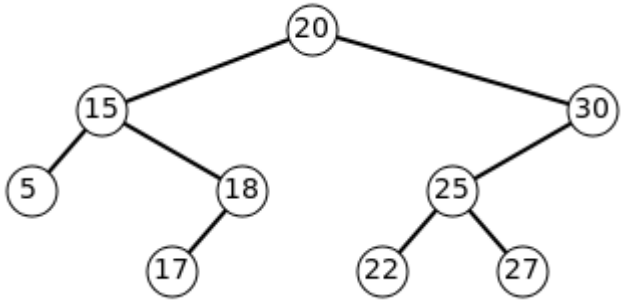
Interface Set

- Collection qui ne contient pas de doublons

```
public interface Set<E>{
    boolean isEmpty() ;
    int size();
    boolean add(E e) ;
    boolean contains(Object o) ;
    boolean remove(Object o) ;
    String toString() ;
}
```

Ensemble: implémentation

- Ensemble non trié
  - table de hashage: HashSet
    - Operations en O(1)
- Ensemble trié
  - implémente l'interface SortedSet
    - Méthodes supplémentaires first(), last()
  - arbre binaire: TreeSet
    - Operations en O(log(N))



Dictionnaire

- Collection qui associe des clés et des valeurs

```
public interface Map <K,V>{
    boolean isEmpty() ;
    int size();
    boolean put(K key,V value);
    boolean containsKey(K key);
    Object remove(Object key);
    Object get(Object key);
}
```

Dictionnaire: Implémentation

- Dictionnaire non trié: HashMap
  - Table de hashage
    - Opérations: O(1)
- Dictionnaire trié: TreeMap
  - Utilisation arbre binaire
    - Operations: O(log(N))
  - Méthodes supplémentaires pour obtenir les clés les plus basses/hautes