

Collections : tables associatives

Pratique de la programmation orientée-objet
Michel Schinz – 2019-03-11

Collections étudiées

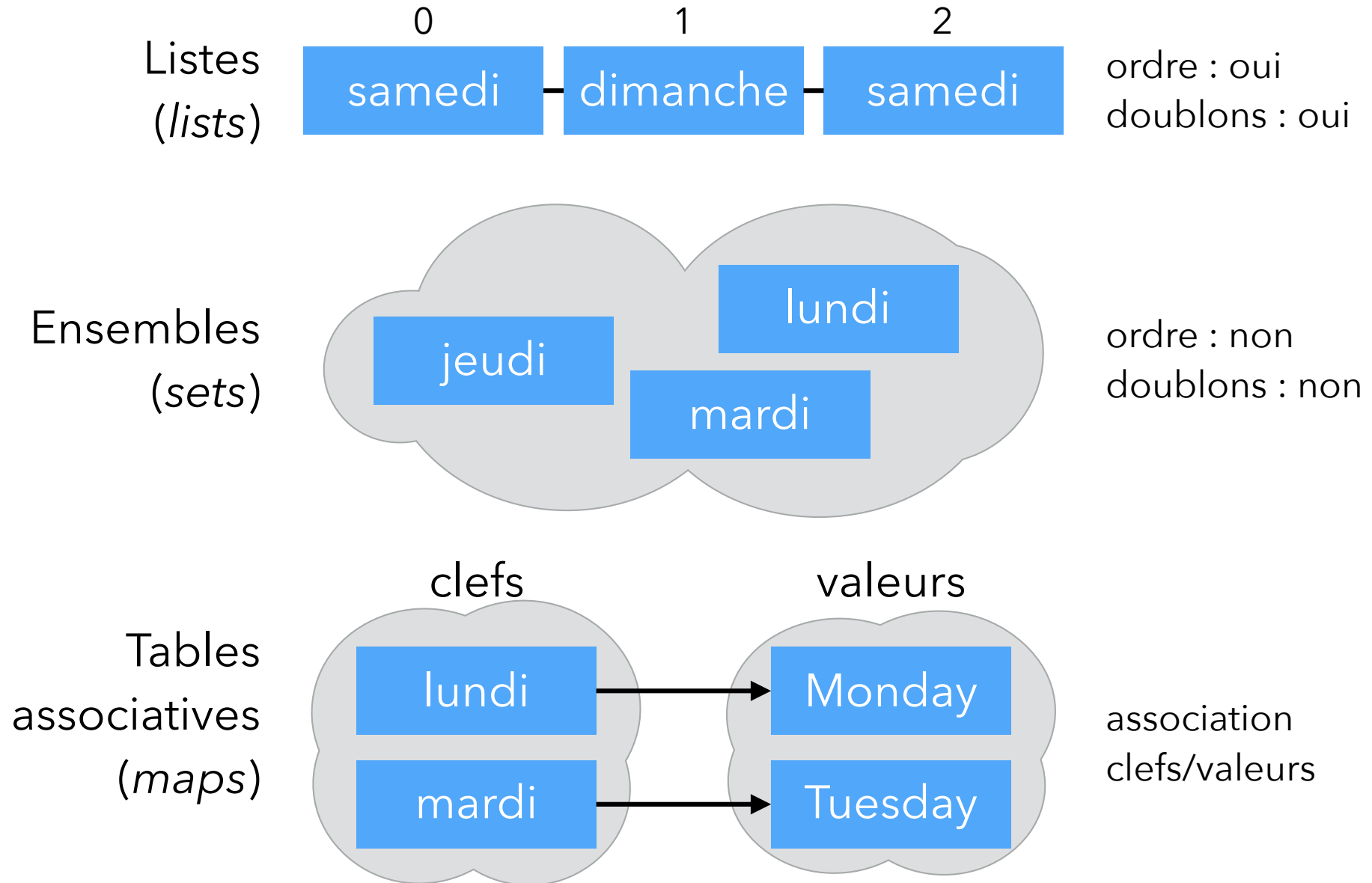


Table associative

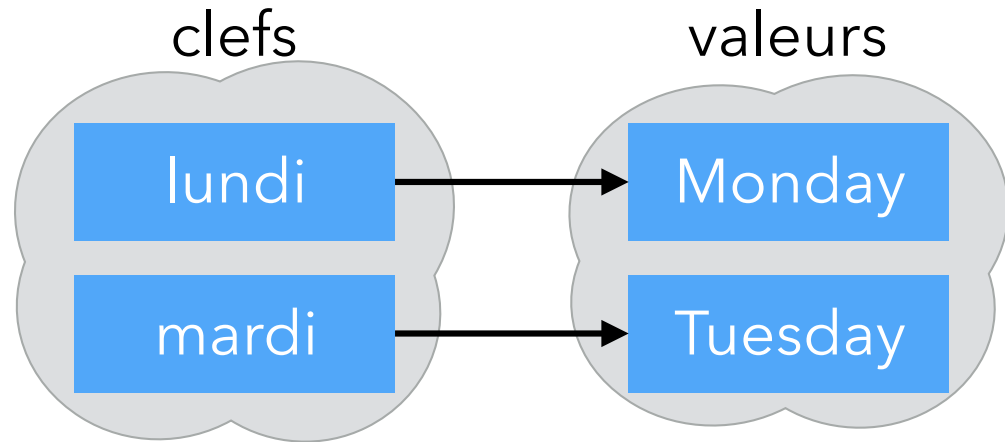
Une table associative associe des valeurs à des clefs. Par exemple, la table ci-dessous associe leur représentation en morse aux lettres (et chiffres) de l'alphabet latin.

Code morse international	
<div>1. Un tiret est égal à trois points.</div> <div>2. L'espacement entre deux éléments d'une même lettre est égal à un point.</div> <div>3. L'espacement entre deux lettres est égal à trois points.</div> <div>4. L'espacement entre deux mots est égal à sept points.</div>	
A	• —
B	— • • •
C	— • — •
D	— • •
E	•
F	• • — •
G	— — •
H	• • • •
I	• •
J	• — — —
K	— • —
L	• — • •
M	— —
N	— •
O	— — —
P	• — — •
Q	— — • —
R	• — •
S	• • •
T	—
U	• • —
V	• • • —
W	• — —
X	— • • —
Y	— • — —
Z	— — • •
1	• — — — —
2	• • — — —
3	• • • — —
4	• • • • —
5	• • • • •
6	— • • • •
7	— — • • •
8	— — — • •
9	— — — — •
0	— — — — —

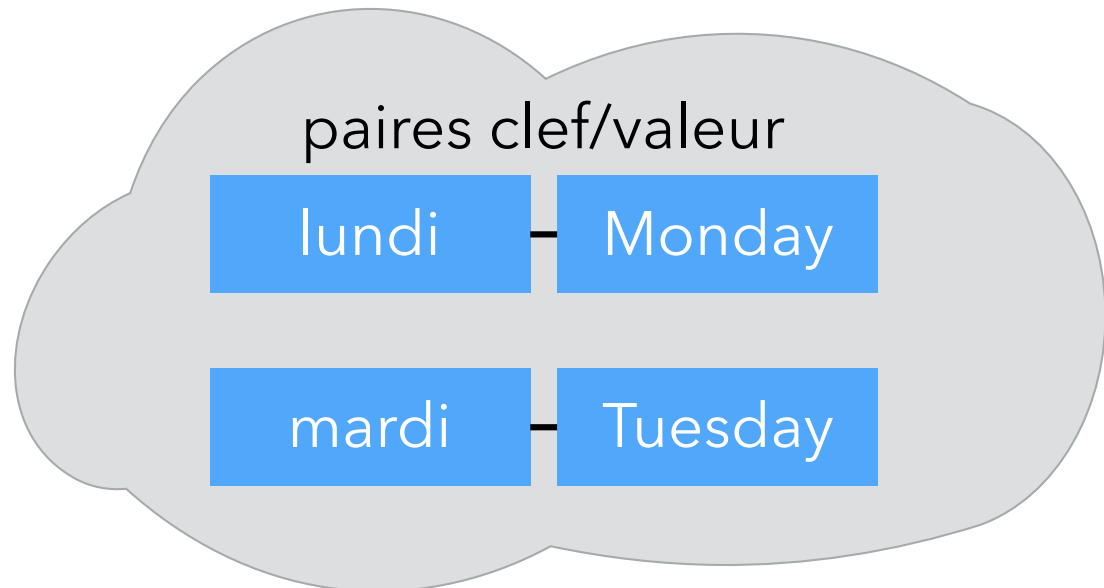
Table associative

Une table associative peut être vue comme...

...un ensemble de
clefs, et un ensemble
de valeurs, liés entre
eux, ou

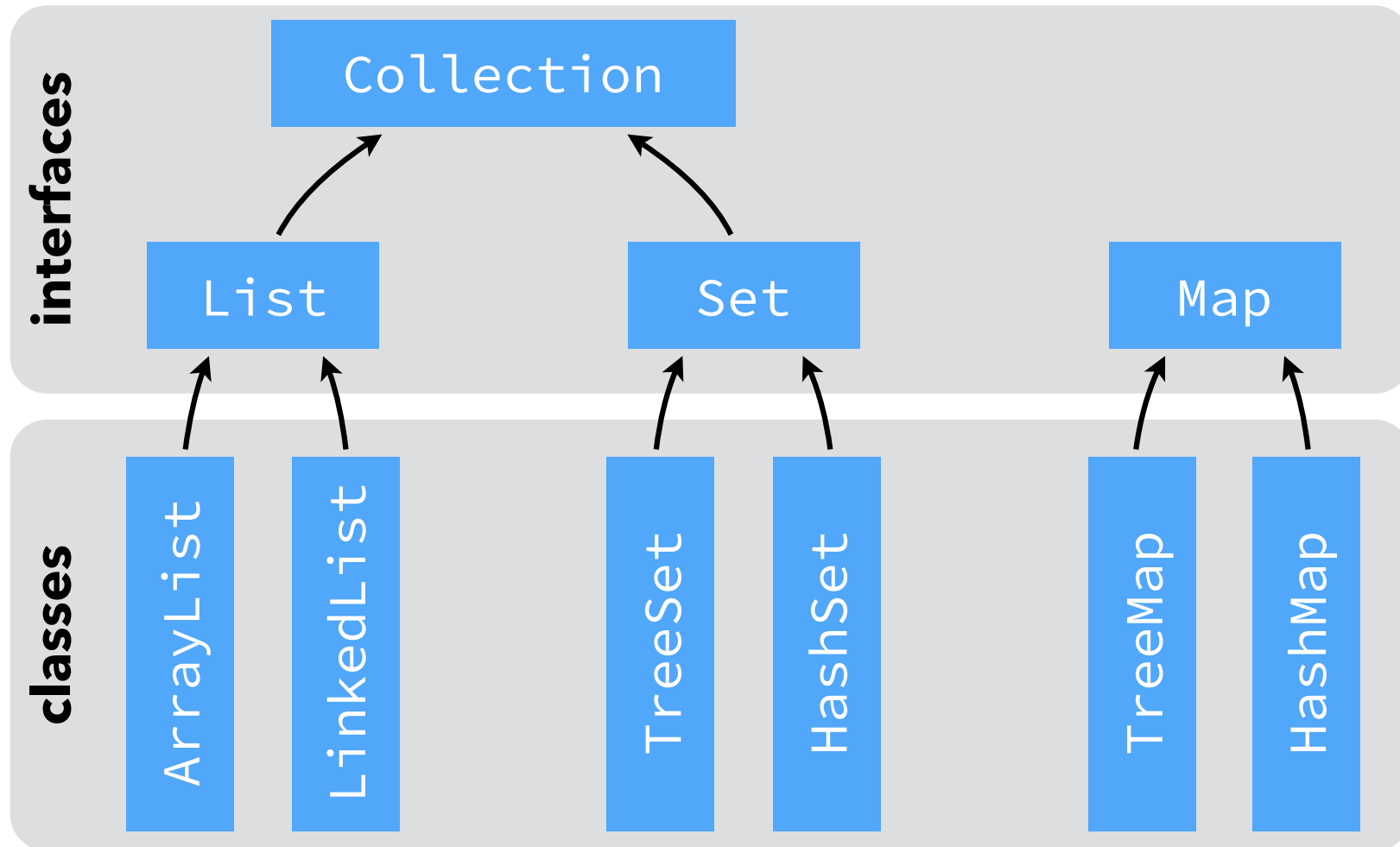


...un ensemble de
paires clef/valeur.



Collections de l'API Java

(vue très partielle et simplifiée du paquetage `java.util`)



+ classes utilitaires **Collections** et **Arrays**

Mises en œuvre de Map

Opération	TreeMap	HashMap
ajout (put), consultation (get), etc.	$O(\log n)$	$O(1)$

Différence importante :

- TreeMap stocke ses clefs de manière triée,
- HashMap stocke ses clefs dans un ordre quelconque.

L'interface Map (1/4)

```
public interface Map<K, V> {  
    // méthodes de consultation  
    boolean isEmpty();  
    int size();  
  
    boolean containsKey(Object k);  
    V get(Object k);  
    V getOrDefault(Object k, V d);  
  
    // ... à suivre  
}
```

L'interface Map (2/4)

```
public interface Map<K, V> { // ... suite
    // méthodes d'ajout/remplacement
    V put(K k, V v);
    V putIfAbsent(K k, V v);
    void putAll(Map<K, V> m);

    // ... aussi : compute, computeIfAbsent, ...

    // ... à suivre
}
```


L'interface Map (3/4)

```
public interface Map<K, V> { // ... suite
    // méthodes de remplacement
    V replace(K k, V v);
    boolean replace(K k, V v1, V v2);

    // méthodes de suppression
    void clear();
    V remove(Object k);
    boolean remove(Object k, Object v);

    // ... à suivre
}
```

L'interface Map (4/4)

```
public interface Map<K, V> { // ... suite
    // vues
    Set<K> keySet();
    Collection<V> values();
    Set<Map.Entry<K, V>> entrySet();

    // paire clef/valeur
    public static interface Entry<K, V> {
        K getKey();
        V getValue();
        void setValue(V v);
    }
}
```

Ensembles / tables assoc.

Ensembles et tables associatives sont très similaires :

$\text{Set}\langle E \rangle \approx \text{Map}\langle E, \text{Void} \rangle$

$\text{Map}\langle K, V \rangle \approx \text{Set}\langle \text{Map.Entry}\langle K, V \rangle \rangle$

Conséquences :

1. tout ce qui a été dit sur les éléments des ensembles s'applique aux clefs des tables associatives,
2. les uns peuvent être mis en œuvre au moyen des autres (en pratique en Java : `TreeSet` mis en œuvre au moyen de `TreeMap`, idem pour `HashSet/HashMap`).