Collections : ensembles

Pratique de la programmation orientée-objet

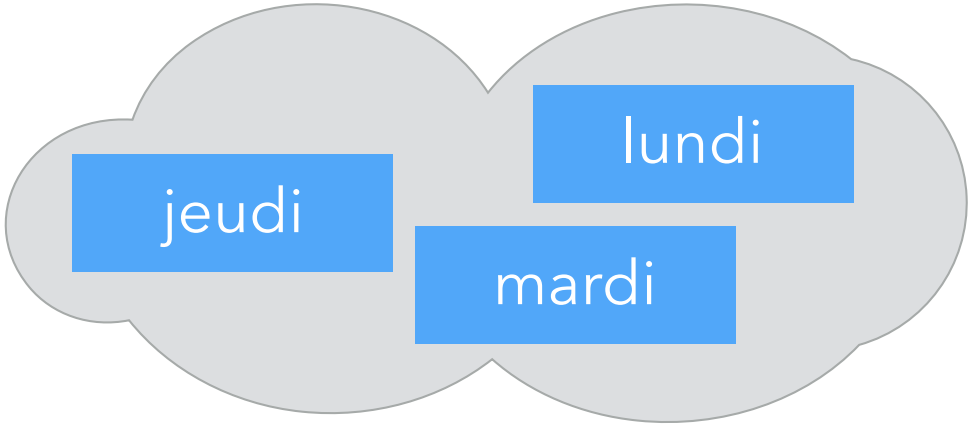
Michel Schinz – 2019-03-11

Collections étudiées

Listes
(*lists*)

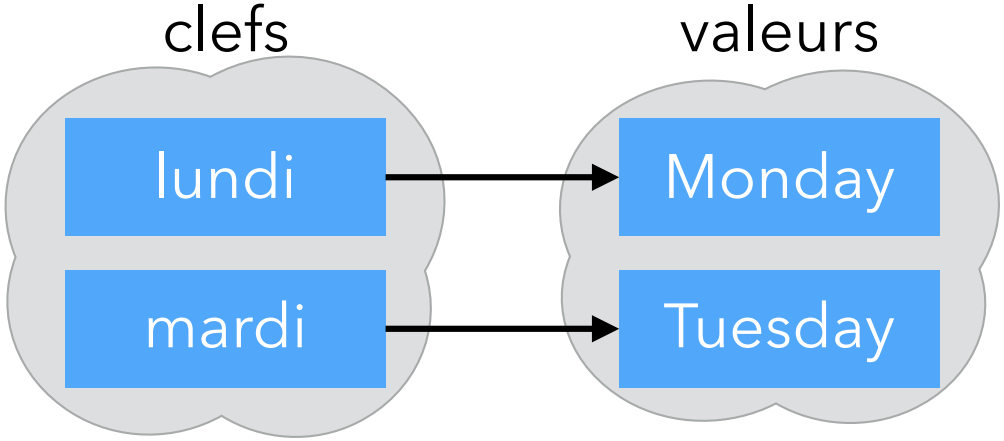
ordre : oui
doublons : oui

Ensembles
(*sets*)



ordre : non
doublons : non

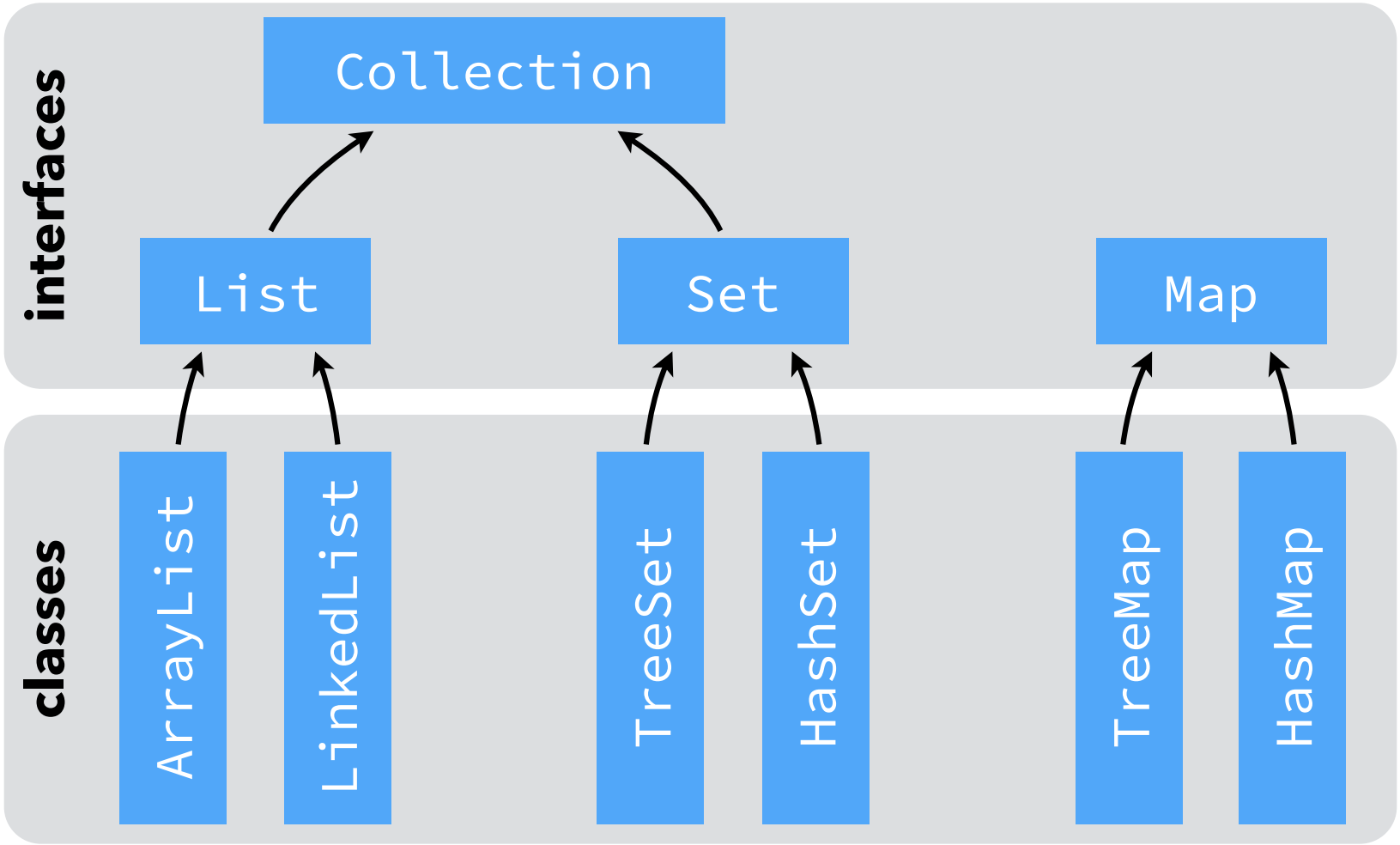
Tables
associatives
(*maps*)



association
clefs/valeurs

Collections de l'API Java

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



\+ classes utilitaires `Collections` et `Arrays`

L'interface Set

```
public interface Set<E>
    extends Collection<E> {
  // n'ajoute aucune méthode !
}
```

Mises en œuvre de `Set`

Opération	`TreeSet`	`HashSet`
ajout (`add`), test (`contains`), etc.	O(log *n*)	O(1)

Différence importante :

- `TreeSet` stocke ses éléments de manière triée,
- `HashSet` stocke ses éléments dans un ordre qui semble quelconque.

Egalité, comparaison, hachage

Comparable

L'interface `java.lang.Comparable` est implémentée par les classes dont les instances savent se comparer entre elles :

```
public interface Comparable<T> {
  int compareTo(T that);
}
```

Comparator

L'interface `java.util.Comparator` est implémentée par les objets sachant comparer des objets d'un autre type.

```
public interface Comparator<T> {
  public int compare(T v1, T v2);
}
```

Comparable/Comparator

Ne confondez pas `Comparable` et `Comparator` !

```
// Implémentée par les classes dont les
// instances « savent » se comparer.
// (p.ex. String, Integer, etc.)
public interface Comparable<T> {
  int compareTo(T that);
}
// Implémentée par des classes dont le seul
// but est de comparer des objets d'un
// autre type.
public interface Comparator<T> {
  public int compare(T v1, T v2);
}
```

hashCode

Contrairement à la notion d'ordre, le hachage est prédéfini en Java, et donc disponible pour tous les objets :

```
public class Object {
  // … autres méthodes
  int hashCode();
}
```

La mise en œuvre par défaut utilise l'identité du récepteur.

hashCode / equals

Les méthodes `hashCode` et `equals` doivent toujours être **compatibles**, c-à-d que la propriété suivante doit être vraie pour toute paire d'objets o_1 et o_2 :

$$o_1.\texttt{equals}(o_2) \Rightarrow o_1.\texttt{hashCode}() == o_2.\texttt{hashCode}()$$

(Deux objets égaux ont la même valeur de hachage.)

Attention, l'implication inverse n'est pas vraie (en général) !

$$o_1.\texttt{hashCode}() == o_2.\texttt{hashCode}() \not\Rightarrow o_1.\texttt{equals}(o_2)$$

(Deux objets ayant la même valeur de hachage ne sont pas forcément égaux.)

Eléments d'une collection

Collection	Contraintes
`List`	*aucune*
`Set`	`equals`
`TreeSet`	`Comparable.compareTo` ou `Comparator.compare`
`HashSet`	`hashCode` + `equals`