

Développement en JAVA

Les collections

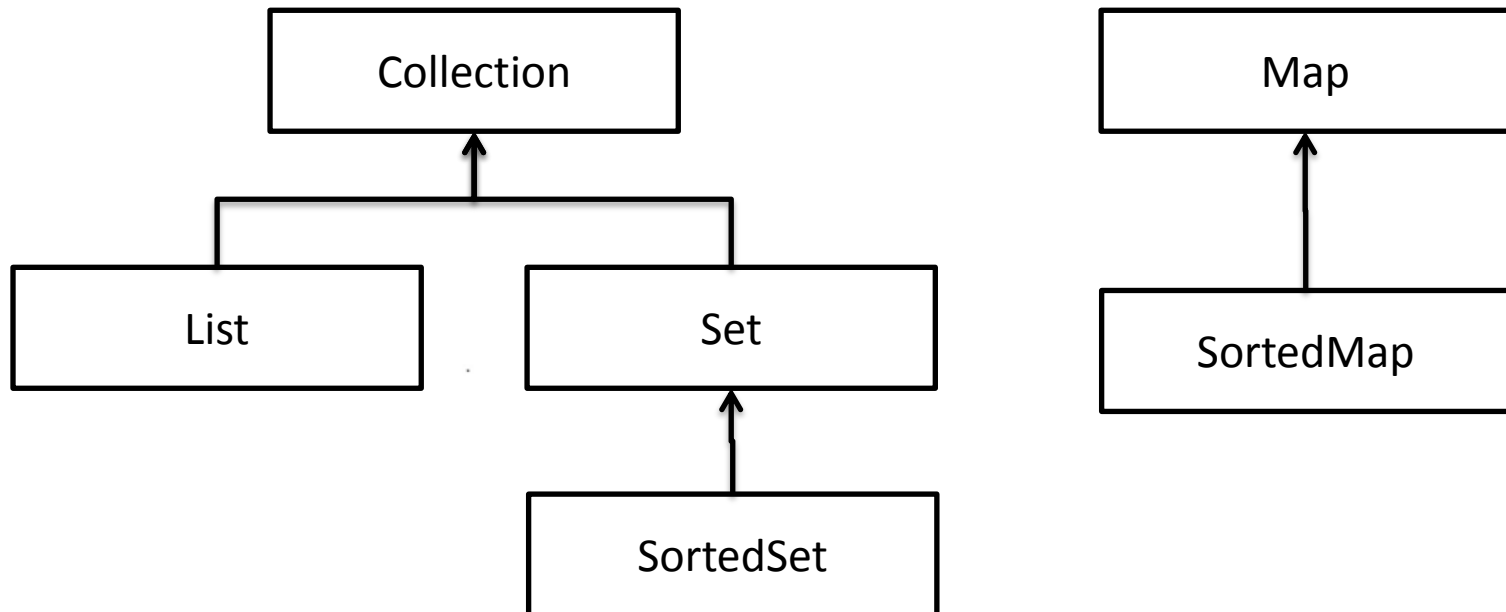


Agenda

- Les interfaces Collections, List et Map
- La classe ArrayList
- La classe LinkedList
- La classe HashMap
- La classe Queue
- La classe Stack

Les collections

- L'API collections de Java offre des classes génériques pour pouvoir gérer des ensembles d'objets



L'interface Collection

- Une Collection est un ensemble d'objets
- L'interface Collection implémente l'interface Iterable
- Toute collection doit être paramétrisé en fonction d'une Classe
- Elle ne peut donc contenir des objets que d'un seul type ou de ses types dérivés

```
// Une collection d'utilisateurs  
Collection<Utilisateur> utilisateurs;  
  
// Une collection pouvant stocker n'importe quel objet  
Collection<Object> objets;
```

L'interface List

- Hérite de l'interface Collection
- Représente une collections ordonnée d'éléments
- Un élément peut être inséré ou retiré à une position précise dans la liste
- Une collection de type List peut être converti en tableau à l'aide de la méhtode toArray()

Iterator

- Un iterator est un objet qui permet de parcourir une collection
- Un iterator possède notamment les méthodes
 - hasNext()
 - next()
 - remove()

Iterator

```
List<Objet> uneList = methodeQuiRetourneUneListe();  
  
Iterator<Objet> it = uneListe.iterator();  
  
while(it.hasNext()){  
    System.out.println(it.next());  
}
```

La boucle foreach

- La boucle foreach offre une manière simple de parcourir l'ensemble des éléments d'une collection de type List<?>

```
List<Objet> uneList = methodeQuiRetourneUneListe();  
  
for(Objet obj: uneList){  
    System.out.println(obj);  
}
```


ArrayList

- Correspond à une structure de données de type tableau
- Caractéristiques
 - Accès à une valeur rapide $O(1)$
 - Ajout d'une valeur en fin de liste rapide $O(1)$
 - Ajout d'une valeur en milieu de liste lente $O(n)$



LinkedList

- Correspond à une structure de données de type liste chaînée
- Caractéristiques:
 - Lent en lecture $O(n)$
 - Insertion rapide à n'importe quel endroit de la liste $O(1)$

Exemple

```
List<Concert> concerts = new ArrayList<Concert>();  
  
for(Concert c : concerts){  
    System.out.println("Concert de " + c.chanteur + " le " +  
c.date);  
}
```

Map

- Une Map est un dictionnaire de type <clé, valeur>
- Une Map ne peut pas être parcourue avec une boucle foreach
- Les principales implémentations sont
 - HashMap
 - Hashtable
 - TreeMap

Map

```
Map<String, User> usersMap = new HashMap<String, User>();  
  
// Ajout d'un élément  
usersMap.add(user1.name, user1);  
  
// Recherche d'un élément existant, peut renvoyer null  
User user2 = usersMap.get("test");
```

Set

- Un Set est une Map n'acceptant pas les doublons
- Contrairement à une liste les éléments ne sont pas ordonnés, il n'est donc possible que de:
 - Ajouter un élément
 - Supprimer un élément
 - Parcourir l'ensemble des éléments du Set
- L'utilisation d'un Set n'est pas pertinent en l'absence de redéfinition des méthodes equals et hashCode

Set

```
Set<User> usersSet= new HashSet<String, User>();

// Ajout d'un élément
if(usersSet.add(user1)){
    // L'utilisateur a bien été ajouté au Set
}else{
    // Sinon c'est qu'il est déjà présent
}
```