Programmation Orientée Objet Java

Collection

v1.1 - 28.04.2023

Plan

- Introduction
- Generics
- Collection
  - List
  - Set
  - Map
  - Queue
- Iterator
- Classe Collections

Introduction

Introduction

Le framework Java Collection est un incontournable dans le monde Java.

Il contient les principales interfaces :

- java.util.Collection
- java.util.List
- java.util.Set
- java.util.Map
- java.util.Queue

Generics

Generics 

Les **generics** permettent de définir un typage spécifique à une structure de données (pour une collection par exemple).
Cela permet de vérifier les types à la compilation et d'éviter les cast (à l'exécution).
Pour définir un **generic**, il faut définir un type objet (classe) encadré des symboles < et > (< > est appelé le diamond operator).

Exemple :

- Avant **Java 5** : List maListe = new ArrayList();
  La variable maListe correspond à une liste de données qui peuvent être de types hétérogènes 👎
- Depuis **Java 5** : List<Integer> maListe = new ArrayList<Integer>();
  La variable maListe correspond à une liste d'Integer 👍
- Depuis **Java 7** : List<Integer> maListe = new ArrayList<>();
  La variable maListe correspond à une liste d'Integer 👍

Collection

Collection

L’interface Collection est l’interface de base dont beaucoup d’interfaces dépendent.

On peut citer les principales :

- List
- Set
- Map
- Queue

Collection : List

L'interface List est l'interface de base pour gérer des listes d'éléments ordonnés.

Contrairement aux tableaux, les List ont une taille dynamique.

Les principales classes qui implémentent List sont :

- ArrayList : tableau dynamique
- LinkedList : liste doublement chainée
- Vector : l'ancêtre de la classe ArrayList
  - Stack : mode LIFO (Last In First Out)

Collection List : méthodes courantes

Voici quelques méthodes courantes de List :

- add(Object o) et add(int index, Object o)
- clear()
- boolean contains(Object o)
- Object get(int index)
- int indexOf(Object o)
- boolean isEmpty()
- iterator() et listIterator()
- remove(Object o) et remove(int index)
- set(int index, Object o)
- int size()
- Object[ ] toArray()

TD06.01-List

Exercice : Reprenez votre TD Fibonacci pour l'améliorer

1. Calculez un nombre aléatoire (entre 10 et 30) de nombres de Fibonacci
2. Stockez ces nombres dans une ArrayList<Integer>
3. Parcourez cette liste pour afficher les nombres :
   a. À l'aide d'une boucle de type "for i"
   b. À l'aide d'une boucle de type "for each"
4. Supprimez les nombres pairs
5. Modifiez le 3ème nombre pour le rendre négatif
6. Insérez le chiffre 0 à la 5ème position
7. Affichez cette liste mise à jour dans la console

Collection : Set

L’interface Set est l’interface de base pour gérer des ensembles d’éléments.

Contrairement aux List, la notion d’index n’existe pas avec les Set et les doublons y sont interdits.

Les principales classes qui implémentent Set sont :

- EnumSet (classe Abstraite)
- HashSet
- SortedSet (interface)
    - TreeSet

Collection Set : méthodes courantes

Voici quelques méthodes courantes de Set :

- add(Object o)
- clear()
- boolean contains(Object o)
- boolean isEmpty()
- iterator()
- remove(Object o)
- int size()
- Object[ ] toArray()

TD06.02-Set

Exercice :

1. Dans votre IDE, créez un nouveau projet Java nommé par exemple “Loto”
2. Définissez un nom de package
3. Créez une classe et y ajouter la méthode public static void main(String[ ] args) mais laissez-la vide
4. Créez une autre classe nommée Ball qui représentera une boule de loto.
   Chaque boule de loto devra avoir un numéro unique
5. Créez une autre classe nommée Lottery qui va :
   a. Créer et contenir 30 boules de Loto
   b. Procéder au tirage, c’est à dire renvoyer 5 boules
6. Dans la méthode main, instanciez un objet Lottery et lancez le tirage afin d’afficher dans la console les numéros “gagnants”

Collection : Map

L'interface Map est l'interface de base pour gérer des données de type clé/valeur.
A chaque **clé** est associée une **valeur/objet unique**.
Les **clés** sont toutes d'un même type. Les **valeurs** sont toutes d'un même type.
Impossible d'avoir deux clés identiques !

Les principales classes qui implémentent Map sont :

- EnumMap (classe Abstraite)
- HashMap, Hashtable
- SortedMap (interface)
  - TreeMap

Collection Map : méthodes courantes

Voici quelques méthodes courantes de Map :

- clear()
- boolean containsKey(K key)
- boolean containsValue(V value)
- V get(K key)
- boolean isEmpty()
- Set<K> keySet()
- V put(K key, V value)
- V remove(Object o)
- V replace(Object key, Objet value)
- int size()
- Collection<V> values()

TD06.03-Map

Exercice :

1. Dans votre IDE, créez un nouveau projet Java nommé par exemple “ChiffresEnLettres”
2. Définissez un nom de package
3. Créez une classe et y ajouter la méthode public static void main(String[ ] args) mais laissez-la vide
4. Créez une autre classe nommée DigitInLetter qui contiendra une Map pour y stocker les couples <chiffre, chiffre en toutes lettres>.
   a. Définir un attribut de type Map
   b. Coder un constructeur pour initialiser cette Map
   c. Coder une méthode pour afficher le nombre de chiffres dans la Map, chacun des chiffres et leur écriture en toutes lettres
5. Dans la méthode main, instanciez un objet DigitInLetter et affichez son contenu ainsi :
   1 : un
   2 : deux
   …

Collection : Queue

L'interface Queue est l'interface de base pour gérer des éléments ordonnés en mode FIFO (First In First Out).

Les principales classes qui implémentent Queue sont :

- BlockingQueue
  - ArrayBlockingQueue
  - LinkedBlockingQueue
  - PriorityBlockingQueue
- SynchronousQueue
- ConcurrentLinkedQueue
- LinkedTransferQueue
- PriorityQueue

Iterator

Iterator

L’interface java.util.Iterator permet de parcourir les données des collections dans un seul sens.

Elle remplace l’interface Enumeration

Voici les méthodes de Iterator :

- boolean hasNext()
- Object next()
- remove()

ListIterator

L'interface java.util.ListIterator permet de parcourir dans les deux sens les données des List.

Voici les méthodes de ListIterator :

- add(Object o)
- boolean hasNext() et boolean hasPrevious()
- Object next() et int nextIndex()
- Object previous() et int previousIndex()
- remove()
- set(Object o)

TD06.04-Iterator

Exercice : Reprenez le TD List

1. Remplacez la boucle d'affichage des éléments de la suite de Fibonacci en utilisant un ListIterator

Exercice : Reprenez le TD Set

1. Remplacez l'itération des éléments par un Iterator

Classe Collections

Classe Collections

La classe java.util.Collections propose de nombreuses méthodes pour manipuler des collections.

Voici quelques méthodes de Collections :

- copy(List, List)
- max(..) et min(..)
- reverse(List)
- shuffle(List)
- sort(..)

Questions/Réponses