TD API COLLECTION JAVA

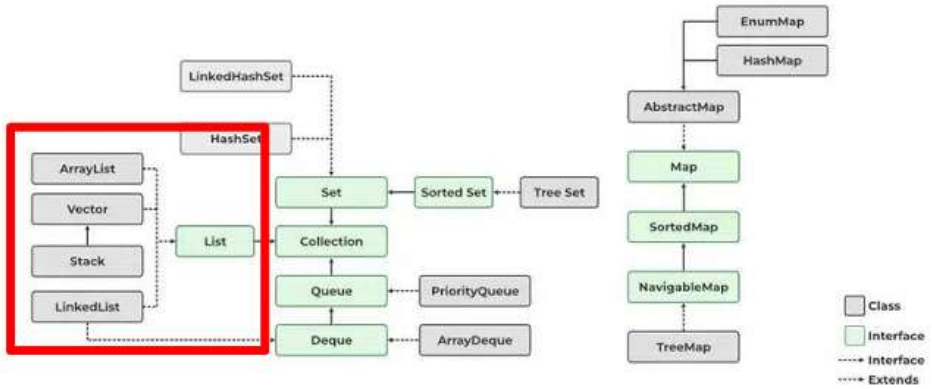
L'API Collections propose un ensemble d'interfaces et de classes dont le but est de stocker de multiples objets. Elle propose quatre grandes familles de collections, chacune définie par une interface de base :

- **List** : collection d'éléments ordonnés qui accepte les doublons
- **Set** : collection d'éléments non ordonnés par défaut qui n'accepte pas les doublons
- **Map** : collection sous la forme d'une association de paires clé/valeur
- **Queue et Deque** : collections qui stockent des éléments dans un certain ordre avant qu'ils ne soient extraits pour traitement.

Réponse aux Questions:

2.1.1 Les classes qui implémentent **l'interface « List »**

JDK JAVA : https://docs.oracle.com/en/java/javase/17/docs/api/index.html



Questions :

*1. Combien de méthodes abstraites sont déclarées dans l'interface « **List** » ?*

https://www.jmdoudoux.fr/java/dej/chap-collections.htm#collections-1

L'interface List définit plusieurs méthodes qui permettent un accès aux éléments de la liste à partir d'un index, de gérer les éléments, de rechercher la position d'un élément, d'obtenir une liste partielle (sublist) et d'obtenir des Iterator : 10.

Méthode	Rôle
void add(int index, E e)	Ajouter un élément à la position fournie en paramètre
boolean addAll(int index, Collection<? extends E> c)	Ajouter des éléments à la position fournie en paramètre
E get(int index)	Retourner l'élément à la position fournie en paramètre
int indexOf(Object o)	Retourner la première position dans la liste du premier élément fourni en paramètre. Elle renvoie -1 si l'élément n'est pas trouvé
int lastIndexOf(Object o)	Retourner la dernière position dans la liste du premier élément fourni en paramètre. Elle renvoie -1 si l'élément n'est pas trouvé
ListIterator<E> listIterator()	Renvoyer un Iterator positionné sur le premier élément de la liste
ListIterator<E> listIterator(int indx)	Renvoyer un Iterator positionné sur l'élément dont l'index est fourni en paramètre
E remove(int index)	Supprimer l'élément à la position fournie en paramètre
E set(int index, E e)	Remplacer l'élément à la position fournie en paramètre
List<E> subList(int fromIndex, int toIndex)	Obtenir une liste partielle de la collection contenant les éléments compris entre les index fromIndex inclus et toIndex exclus fournis en paramètres

*2. Quelle méthode de l'interface « **List** » est à réimplémenter pour remplacer un élément à une position donnée ?*

E set (int index, E e) -> Remplacer l'élément à la position fournie en paramètre

*3. Quelle est la différence entre la classe « **ArrayList** » et la classe « **Vector** » ?*

Vector:

- La classe Vector, présente depuis Java 1.0, est un tableau dont la taille peut varier selon le nombre d'éléments qu'il contient.
- Lors de la création d'une instance de type Vector, il est possible de lui préciser une capacité initiale et une taille d'incrémentation en utilisant la surcharge correspondante du constructeur.
- Toutes les méthodes de la classe Vector sont synchronized : elle est donc moins performante que la classe ArrayList car elle est **threadsafe**.
- **Multithread** (toutes ses méthodes sont synchronisées). Pour une utilisation dans un thread unique, la synchronisation des méthodes est inutile et coûteuse. Il est alors préférable d'utiliser un objet de la classe ArrayList.

ArrayList:

- **Nonthread-safe** : Les méthodes d'un ArrayList ne sont pas synchronisées. Cela signifie que si plusieurs threads accèdent à une instance de ArrayList et modifient cette instance sans une coordination appropriée (comme l'utilisation de verrous explicites), des problèmes tels que des incohérences de données ou des exceptions inattendues peuvent se produire.

*4. Quelle structure de données représente la classe « **Stack** » ?*

java.util.Stack<E>

Une implémentation d'une pile : elle hérite de la classe Vector et fournit des opérations pour un comportement de type LIFO (Last In First Out)

*5. Quel est l'objectif de la méthode « **pop** » pour un objet de la classe « **Stack** » ?*

E pop()

Obtenir le dernier élément de la liste et le retirer de la collection

*6. Quelle opération permet d'ajouter un élément à un objet de la classe « **Stack** » ?*

	Début de la collection		Fin de la collection	
	Lever une exception	Renvoyer une valeur	Lever une exception	Renvoyer une valeur
Ajout	addFirst()	offerFirst()	addLast()	offerLast()
Retirer	removeFirst()	poolFirst()	removeLast()	poolLast()
Consulter	getFirst()	peekFirst()	getLast()	peekLast()

- Il n'est pas possible d'accéder à un élément particulier hormis le premier et le dernier.
- Préférable d'utiliser offerFirst() ou offerLast(), car elles permettent de gérer le cas où l'ajout n'est pas possible.

void addFirst(E e)	Insérer un nouvel élément au début de la collection
void addLast(E e)	Insérer un nouvel élément à la fin de la collection
boolean offerFirst(E e)	Insérer un nouvel élément au début de la collection. Elle renvoie un booléen qui précise si l'opération a réussi
boolean offerLast(E e)	Insérer un nouvel élément à la fin de la collection. Elle renvoie un booléen qui précise si l'opération a réussi

*7. Qu'est la classe « **Stack** » par rapport à la classe « **Vector** » ?*

java.util.Vector<E>	Une implémentation thread-safe fournie depuis Java 1.0
java.util.Stack<E>	Une implémentation d'une pile : elle hérite de la classe Vector et fournit des opérations pour un comportement de type LIFO (Last In First Out)

-> Stack est une sous-classe de Vector.

*8. Quels sont les avantages d'objets de la classe « **LinkedList** » par rapport à la classe « **ArrayList** » ?*

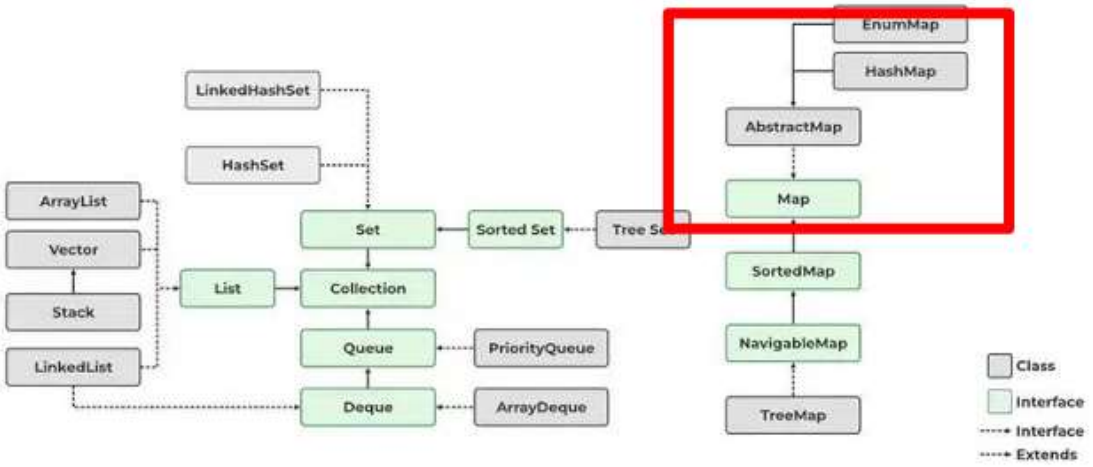
- LinkedList : liste **doublement chaînée** (parcours de la liste dans les deux sens) qui implémente l'interface List.

java.util.ArrayList<E>	Une implémentation qui n'est pas synchronized, donc à n'utiliser que dans un contexte monothread
java.util.LinkedList<E>	Une implémentation qui n'est pas synchronized d'une liste doublement chaînée. Les insertions de nouveaux éléments sont très rapides

****

2.1.2 Questions portant les classes qui implémentent **l'interface « Map »**

Vidéo Fonctionnement HashMap: https://www.youtube.com/watch?v=l9aBP0xm-lE



Questions :

1. Qu'est-ce qu'un tableau associatif (ou « table associative », ou « dictionnaire ») ?

Cette classe *Hashtable* met en œuvre la notion de table de hachage qui, nous l'avons vu, représente un couple déposé sous la forme **(clé, valeur)**.

2. En Java, quelle classe peut être utilisée pour créer un tableau associatif ?

L'API Collections propose un ensemble d'interfaces et de classes dont le but est de stocker de multiples objets. Elle propose quatre grandes familles de collections, chacune définie par une interface de base :

- List : collection d'éléments ordonnés qui accepte les doublons
- Set : collection d'éléments non ordonnés par défaut qui n'accepte pas les doublons
- **Map : collection sous la forme d'une association de paires clé/valeur => MAP**
- Queue et Deque : collections qui stockent des éléments dans un certain ordre avant qu'ils ne soient extraits pour traitement

3. Lors de l'instanciation d'un objet de la classe « HashMap », à quoi correspondent les classes indiquées entre les chevrons ? Par exemple, à quoi correspond « » dans l'extrait suivant :

```
HashMap<String, Integer> mapTest = new HashMap<>();
```

Définir les types attendus pour les clefs/valeurs:

- ici Clé -> String
- ici Valeur -> Integer

Ex: ("Toulouse",31)

4. Quelle méthode peut être utilisée pour ajouter une paire « clef-valeur » à un tableau associatif ?

.put()

```
Map<Character, String> morse = new HashMap<>();
morse.put('a', ".-");
morse.put('j', ".---");
morse.put('v', "...-");
// … idem pour les autres lettres et la ponctuation.
```

5. Dans quelle classe (ou interface) est déclarée la méthode que vous avez trouvé en répondant à la question précédente ?

Hashmap

*6. Quel est le type de retour de la méthode « **get(Object key)** » de la classe « **HashMap** » ?*

Du type de la valeur //key
Sinon -> null

****

<u>Implémentation d'Exemples:</u>

Créez/trouvez un exemple concret d'utilisation des classes suivantes :

- **« Stack »**
- **« Hashmap »**

A faire

Pour chaque exemple d'utilisation, implémentez une fonction que vous ajouterez à un projet nommé « api-collection-initiation » sur un dépôt Git.

Il vous est également demandé d'ajouter les réponses aux questions précédentes dans un fichier ajouté à ce projet.