Questions List:

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

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

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

E set(int index, E e)

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

Le framework Collections propose plusieurs implémentations possédant chacune un comportement et des fonctionnalités particulières.

Collection	Ordonné	Accès direct	Clé / valeur	Doublons	Null	Thread Safe
ArrayList	Oui	Oui	Non	Oui	Oui	Non
LinkedList	Oui	Non	Non	Oui	Oui	Non
HashSet	Non	Non	Non	Non	Oui	Non
TreeSet	Oui	Non	Non	Non	Non	Non
HashMap	Non	Oui	Oui	Non	Oui	Non
TreeMap	Oui	Oui	Oui	Non	Non	Non
Vector	Oui	Oui	Non	Oui	Oui	Oui
Hashtable	Non	Oui	Oui	Non	Non	Oui
Properties	Non	Oui	Oui	Non	Non	Oui
Stack	Oui	Non	Non	Oui	Oui	Oui
CopyOnWriteArrayList	Oui	Oui	Non	Oui	Oui	Oui
ConcurrentHashMap	Non	Oui	Oui	Non	Non	Oui
CopyOnWriteArraySet	Non	Non	Non	Non	Oui	Oui

ArrayList

- **Non thread-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.
- **Performance** : Étant donné que `ArrayList` n'a pas de mécanismes de synchronisation internes, ses opérations sont généralement plus rapides que celles d'un `Vector` dans des environnements à thread unique ou lorsque la synchronisation est gérée par l'utilisateur.

Vector

- **Thread-safe** : Les méthodes d'un `Vector` sont synchronisées. Cela signifie que toutes les opérations publiques de `Vector` sont sécurisées pour une utilisation simultanée par plusieurs threads. Par exemple, l'ajout d'un élément ou la lecture d'un élément à partir d'un `Vector` se fait de manière atomique, empêchant ainsi les interférences entre les threads.
- **Performance** : Cette synchronisation entraîne une surcharge supplémentaire. Par conséquent, les opérations sur un `Vector` peuvent être plus lentes que celles sur un `ArrayList` en raison du coût de la gestion des verrous internes.

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 de la collection hormis le premier et le dernier.

Il n'est pas recommandé d'insérer la valeur null dans une collection de type Deque essentiellement parce que la valeur null est retournée par plusieurs méthodes pour indiquer qu'elles ont échouées.

Il est préférable d'utiliser les méthodes offerFirst() et offerLast() pour ajouter un élément car elles permettent de gérer les cas ou l'ajout n'est pas possible.

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

What is the difference between stack and Vector in Java? Stack is a subclass of Vector with specific stack operations, while Vector is a general-purpose collection class in Java. 5 juin 2024

What is the difference between vector and stack in Java?

Stack is basically a special case of vector. Theoretically speaking vector can grow as you wish. You can remove elements at any index in a vector. However, in case of a stack you can remove elements and insert them only at its top (hence a special case of vector).

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

Collection	Ordonné	Accès direct	Clé / valeur	Doublons	Null	Thread Safe
ArrayList	Oui	Oui	Non	Oui	Oui	Non
LinkedList	Oui	Non	Non	Oui	Oui	Non

- ArrayList : tableau dynamique qui implémente l'interface List
- 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

Lors de l'ajout ou du retrait d'un élément, la collection doit réindexer ses éléments. Si la taille de la collection est importante et qu'il y a de nombreux ajouts et suppressions d'éléments alors il est préférable d'utiliser une collection de type LinkedList.

Question Map:

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

Une **table associative** (*map*) ou **dictionnaire** (*dictionary*) est une collection qui associe des **valeurs** (*values*) à des **clefs** (*keys*).

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

Les collections de type Map sont définies et implémentées comme des dictionnaires sous la forme d'associations de paires de type clés/valeurs. La clé doit être unique. En revanche, la même valeur peut être associée à plusieurs clés différentes.

Avant l'apparition du framework Collections, la classe dédiée à cette gestion était la classe Hashtable.

Un objet de type Map permet de lier un objet avec une clé qui peut être un type primitif ou un autre objet. Il est ainsi possible d'obtenir un objet à partir de sa clé.

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<>();

Ils permettent de définir les types des clés/valeurs attendues, ici la clé sera de type String et la valeur de type Integer. Exemple: <"Nombre de voiture", 57>

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

.put()

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 » ?

Le type de retour sera du type de la valeur correspondant à la key, si la clé n’existe pas celà renvoie null