|  |
| --- |
| **Collections Interview Questions** |
| **Q1) What is difference between ArrayList and vector?**  Ans: )  1) Synchronization - ArrayList is not thread-safe whereas Vector is thread-safe. In Vector class each method like add(), get(int i) is surrounded with a synchronized block and thus making Vector class thread-safe.  2) Data growth - Internally, both the ArrayList and Vector hold onto their contents using an Array. When an element is inserted into an ArrayList or a Vector, the object will need to expand its internal array if it runs out of room. A Vector defaults to doubling the size of its array, while the ArrayList increases its array size by 50 percent. |
| **Q2) How can Arraylist be synchronized without using Vector?**  Ans) Arraylist can be synchronized using:  Collection.synchronizedList(List list)  Other collections can be synchronized:  Collection.synchronizedMap(Map map)  Collection.synchronizedCollection(Collection c) |
| **Q3) If an Employee class is present and its objects are added in an arrayList. Now I want the list to be sorted on the basis of the employeeID of Employee class. What are the steps?**  Ans) 1) Implement Comparable interface for the Employee class and override the compareTo(Object obj) method in which compare the employeeID  2) Now call Collections.sort() method and pass list as an argument.  *Now consider that Employee class is a jar file.*  1) Since Comparable interface cannot be implemented, create Comparator and override the compare(Object obj, Object obj1) method .  2) Call Collections.sort() on the list and pass comparator as an argument. |
| **Q4)What is difference between HashMap and HashTable?**  Ans) Both collections implements Map. Both collections store value as key-value pairs. The key differences between the two are  1. Hashmap is not synchronized in nature but hshtable is.  2. Another difference is that iterator in the HashMap is fail-safe while the enumerator for the Hashtable isn't. **Fail-safe** - â€œif the Hashtable is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove method, the iterator will throw a ConcurrentModificationExceptionâ€?  3. HashMap permits null values and only one null key, while Hashtable doesn't allow key or value as null. |
| **Q5) What are the classes implementing List interface?**  Ans) There are three classes that implement List interface: 1) **ArrayList** : It is a resizable array implementation. The size of the ArrayList can be increased dynamically also operations like add,remove and get can be formed once the object is created. It also ensures that the data is retrieved in the manner it was stored. The ArrayList is not thread-safe.  2) **Vector**: It is thread-safe implementation of ArrayList. The methods are wrapped around a synchronized block.  3) **LinkedList**: the LinkedList also implements Queue interface and provide FIFO(First In First Out) operation for add operation. It is faster if than ArrayList if it performs insertion and deletion of elements from the middle of a list. |
| **Q6) Which all classes implement Set interface?**  Ans) A Set is a collection that contains no duplicate elements. More formally, sets contain no pair of elements e1 and e2 such that e1.equals(e2), and at most one null element. **HashSet,SortedSet and TreeSet** are the commnly used class which implements Set interface.  **SortedSet** - It is an interface which extends Set. A the name suggest , the interface allows the data to be iterated in the ascending order or sorted on the basis of Comparator or Comparable interface. All elements inserted into the interface must implement Comparable or Comparator interface.  **TreeSet** - It is the implementation of SortedSet interface.This implementation provides guaranteed log(n) time cost for the basic operations (add, remove and contains). The class is not synchronized.  **HashSet:** This class implements the Set interface, backed by a hash table (actually a HashMap instance). It makes no guarantees as to the iteration order of the set; in particular, it does not guarantee that the order will remain constant over time. This class permits the null element. This class offers constant time performance for the basic operations (add, remove, contains and size), assuming the hash function disperses the elements properly among the buckets |
| **Q7) What is difference between List and a Set?**  Ans) 1) List can contain duplicate values but Set doesnt allow. Set allows only to unique elements.  2) List allows retrieval of data to be in same order in the way it is inserted but Set doesnt ensures the sequence in which data can be retrieved.(Except HashSet) |
| **Q8) What is difference between Arrays and ArrayList ?**  Ans) Arrays are created of fix size whereas ArrayList is of not fix size. It means that once array is declared as :   * 1. int [] intArray= new int[6];   2. intArray[7]   // will give ArraysOutOfBoundException.   Also the size of array cannot be incremented or decremented. But with arrayList the size is variable.   1. Once the array is created elements cannot be added or deleted from it. But with ArrayList the elements can be added and deleted at runtime.   List list = new ArrayList(); list.add(1); list.add(3); list.remove(0) // will remove the element from the 1st location.   1. ArrayList is one dimensional but array can be multidimensional.               int[][][] intArray= new int[3][2][1];   // 3 dimensional array   1. To create an array the size should be known or initalized to some value. If not initialized carefully there could me memory wastage. But arrayList is all about dynamic creation and there is no wastage of memory. |
| **Q9) When to use ArrayList or LinkedList ?**  Ans)  Adding new elements is pretty fast for either type of list. For the ArrayList, doing  random lookup using "get" is fast, but for LinkedList, it's slow. It's slow because there's no efficient way to index into the middle of a linked list. When removing elements, using ArrayList is slow. This is because all remaining elements in the underlying array of Object instances must be shifted down for each remove operation. But here LinkedList is fast, because deletion can be done simply by changing a couple of links. So an ArrayList works best for cases where you're doing random access on the list, and a LinkedList works better if you're doing a lot of editing in the middle of the list.  Source : [Read More - from java.sun](http://java.sun.com/developer/TechTips/1999/tt0809.html) |
| **Q10) Consider a scenario. If an ArrayList has to be iterate to read data only, what are the possible ways and which is the fastest?**  Ans) It can be done in two ways, using for loop or using iterator of ArrayList. The first option is faster than using iterator. Because value stored in arraylist is indexed access. So while accessing the value is accessed directly as per the index. |
| **Q11) Now another question with respect to above question is if accessing through iterator is slow then why do we need it and when to use it.**  Ans) For loop does not allow the updation in the array(add or remove operation) inside the loop whereas Iterator does. Also Iterator can be used where there is no clue what type of collections will be used because all collections have iterator. |
| **Q12) Which design pattern Iterator follows?**  Ans) It follows Iterator design pattern. Iterator Pattern is a type of behavioral pattern. The Iterator pattern is one, which allows you to navigate through a collection of data using a common interface without knowing about the underlying implementation. Iterator should be implemented as an interface. This allows the user to implement it anyway its easier for him/her to return data. The benefits of Iterator are about their strength to provide a common interface for iterating through collections without bothering about underlying implementation.  Example of Iteration design pattern - Enumeration The class java.util.Enumeration is an example of the Iterator pattern. It represents and abstract means of iterating over a collection of elements in some sequential order without the client having to know the representation of the collection being iterated over. It can be used to provide a uniform interface for traversing collections of all kinds. |
| **Q12) Is it better to have a HashMap with large number of records or n number of small hashMaps?**  Ans) It depends on the different scenario one is working on: 1) If the objects in the hashMap are same then there is no point in having different hashmap as the traverse time in a hashmap is invariant to the size of the Map. 2) If the objects are of different type like one of Person class , other of Animal class etc then also one can have single hashmap but different hashmap would score over it as it would have better readability. | | |
| **Q13) Why is it preferred to declare: List<String> list = new ArrayList<String>(); instead of ArrayList<String> = new ArrayList<String>();**  Ans) It is preferred because:   1. If later on code needs to be changed from ArrayList to Vector then only at the declaration place we can do that. 2. The most important one – If a function is declared such that it takes list. E.g void showDetails(List list); When the parameter is declared as List to the function it can be called by passing any subclass of List like ArrayList,Vector,LinkedList making the function more flexible | | |
| **Q14) What is difference between iterator access and index access**?  Ans) Index based access allow access of the element directly on the basis of index. The cursor of the datastructure can directly goto the 'n' location and get the element. It doesnot traverse through n-1 elements.  In Iterator based access, the cursor has to traverse through each element to get the desired element.So to reach the 'n'th element it need to traverse through n-1 elements.  Insertion,updation or deletion will be faster for iterator based access if the operations are performed on elements present in between the datastructure.  Insertion,updation or deletion will be faster for index based access if the operations are performed on elements present at last of the datastructure.  Traversal or search in index based datastructure is faster.  ArrayList is index access and LinkedList is iterator access. | | |
| **Q15) How to sort list in reverse order?**  Ans) To sort the elements of the List in the reverse natural order of the strings, get a reverse Comparator from the Collections class with reverseOrder(). Then, pass the reverse Comparator to the sort() method.  List list = new ArrayList();  Comparator comp = Collections.reverseOrder();  Collections.sort(list, comp) | | |
| **Q16) Can a null element added to a Treeset or HashSet?**  Ans) A null element can be added only if the set contains one element because when a second element is added then as per set defination a check is made to check duplicate value and comparison with null element will throw NullPointerException. HashSet is based on hashMap and can contain null element. | | |
| **Q17) How to sort list of strings - case insensitive?**  Ans) using Collections.sort(list, String.CASE\_INSENSITIVE\_ORDER); | | |
| **Q18) How to make a List (ArrayList,Vector,LinkedList) read only?**  Ans) A list implemenation can be made read only using **Collections.unmodifiableList(list)**. This method returns a new list. If a user tries to perform add operation on the new list; UnSupportedOperationException is thrown. | | |
| **Q19) What is ConcurrentHashMap?**  Ans) A concurrentHashMap is thread-safe implementation of Map interface. In this class put and remove method are synchronized but not get method. This class is different from Hashtable in terms of locking; it means that hashtable use object level lock but this class uses bucket level lock thus having better performance. | | |
| **Q20) Which is faster to iterate LinkedHashSet or LinkedList?**  Ans) LinkedList. | | |
| **Q21) Which data structure HashSet implements**  Ans) HashSet implements hashmap internally to store the data. The data passed to hashset is stored as key in hashmap with null as value. | | |
| **Q22) Arrange in the order of speed - HashMap,HashTable, Collections.synchronizedMap,concurrentHashmap**  Ans) HashMap is fastest, ConcurrentHashMap,Collections.synchronizedMap,HashTable. | | |
| **Q23) What is identityHashMap?**  Ans) The IdentityHashMap uses == for equality checking instead of equals(). This can be used for both performance reasons, if you know that two different elements will never be equals and for preventing spoofing, where an object tries to imitate another. | | |
| **Q24) What is WeakHashMap?**  Ans) A hashtable-based Map implementation with weak keys. An entry in a WeakHashMap will automatically be removed when its key is no longer in ordinary use. More precisely, the presence of a mapping for a given key will not prevent the key from being discarded by the garbage collector, that is, made finalizable, finalized, and then reclaimed. When a key has been discarded its entry is effectively removed from the map, so this class behaves somewhat differently than other Map implementations. | | |