ITC 155
Anu Slorah
5/23/2017

Java Linked Data Structures Interview question:
What are the main concerns when choosing a collection?

Two main concerns are speed and consistency. In terms of speed there are

- speed of access
- speed of adding new element
- speed of removing an element
- speed of iteration

Some implementations have guaranteed speed of access, others have variable speed.

LinkedList vs ArrayList:
Adding or removing an item to Linkedlist is fast, only two links are affected, nothing needs to be copied, moved, or reordered. Accessing items, however, does not have a constant speed, to retrieve an item you need to traverse through all the nodes occurring before the requested item. LinkedList is great if there is a lot of adding and removing because it is dynamic in size.

In ArrayList every item after the addition or deleting needs to be copied one space over. ArrayList is static in size, so if the array capacity needs to be enlarged, the entire array has to be copied over. However, accessing an item in the array is fast, the access time is constant whether you are retrieving an item at position 0 or 1000. ArrayList is great for accessing data that does not need to be changed often and where fast access to items is needed.

Both LinkedList and ArrayList allow duplicate items.

Set vs Map

A Set contains only values while Map contains key value pairs. Both Set and Map contain only unique elements. A Set can be iterated but a Map needs to be converted to Set to be iterated. Sets can contain any type of elements, however a TreeSet can only contain similar items.

HashSet provides no guarantee about the order in which elements be retrieved. It allows storing only one NULL element. All subsequent calls to store NULL values are ignored. HashSet is implemented using HashMap, most of the functionality is defined in HashMap and its related classes.

HashMap allows NULL elements. It is very efficient for adding, removing, and locating items until there are no collisions, once collisions start happening the speed decreases. Also, once the HashMap exceeds default load of 0.75, it will rehash, that is double the size and reorder entries, a somewhat costly process. HashMap is not sortable.

TreeMap is sorted in ascending order. Because of this adding, and deleting items causes the nodes to re-sort themselves to maintain the order. To locate an item it might be necessary to traverse a large number of nodes. This is slower than HashMap. TreeMap does not allow null values.

If speed is the goal then HashMap or HashSet are a better choice than TreeMap. However, if ordering is important then TreeMap/TreeSet are better choices. Depending on the size of the collection, it may be faster to add elements to a HashMap, then convert the map to a TreeMap for sorted key traversal.

Java Linked Data Structures Interview question:
What are four ways to traverse Map?

- foreach loop
- KeySet Iterator
- EntrySet instead of KeySet. EntrySet is a collection of all Map Entries and contains both Key and Value.
- Combination of Iterator and EntrySet to display all keys and values of a Java Map.

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

public class MapsApp {

	public static void main(String[] args) {

		HashMap<String, String> numbers = new HashMap<String, String>();
		numbers.put("one", "üks");
		numbers.put("two", "kaks");
		numbers.put("three", "kolm");

		System.out.println("Iterating or looping HashMap using foreach loop");

		for (String key : numbers.keySet()) {
			System.out.println("key: " + key + " value: " + numbers.get(key));
		}

		System.out.println("-----------------------------------------------");
		System.out.println("Iterating HashMap using KeySet Iterator");

		Set<String> keySet = numbers.keySet();
		Iterator<String> keySetIterator = keySet.iterator();
		while (keySetIterator.hasNext()) {
			String key = keySetIterator.next();
			System.out.println("key: " + key + " value: " + numbers.get(key));
		}

		System.out.println("-----------------------------------------------");
		System.out.println("looping HashMap using EntrySet and for loop");


		Set<Map.Entry<String, String>> entrySet = numbers.entrySet();
		for (Entry entry : entrySet) {
			System.out.println("key: " + entry.getKey() + " value: " +
entry.getValue());
		}

		System.out.println("-----------------------------------------------");
		System.out.println("Iterating HashMap using EntrySet and Java iterator");


		Set<Map.Entry<String, String>> entrySet1 = numbers.entrySet();
		Iterator<Entry<String, String>> entrySetIterator = entrySet1.iterator();
		while (entrySetIterator.hasNext()) {
			Entry entry = entrySetIterator.next();
			System.out.println("key: " + entry.getKey() + " value: " +
entry.getValue());
		}
	}

}
```