EU8-JAVA-Collections

- ▼ What is Collection Framework?
  - interfaces and classes —→ organize our data
  - JAVA collections framework is a set of pre-written classes and interfaces that helps us to organize and manipulate groups of data
- ▼ Why we need collections?

  Data → store (group of data)

  Arrays (limited: fixed size, homogenious)

  Collections → more flexible and provide useful methods

  Benefits of collections: growable, different types of Data, supports OOP Concepts (polymorphism)
- ▼ What should we know about this topic?
  - ▼ Interview Perspective

    We need to know everything verbally

    What is the difference between .....and .....?

    How did you use ..... collection in your test framework?
  - ▼ Usage perspective

    We will use: List - ArrayList - Set - Map - Arrays

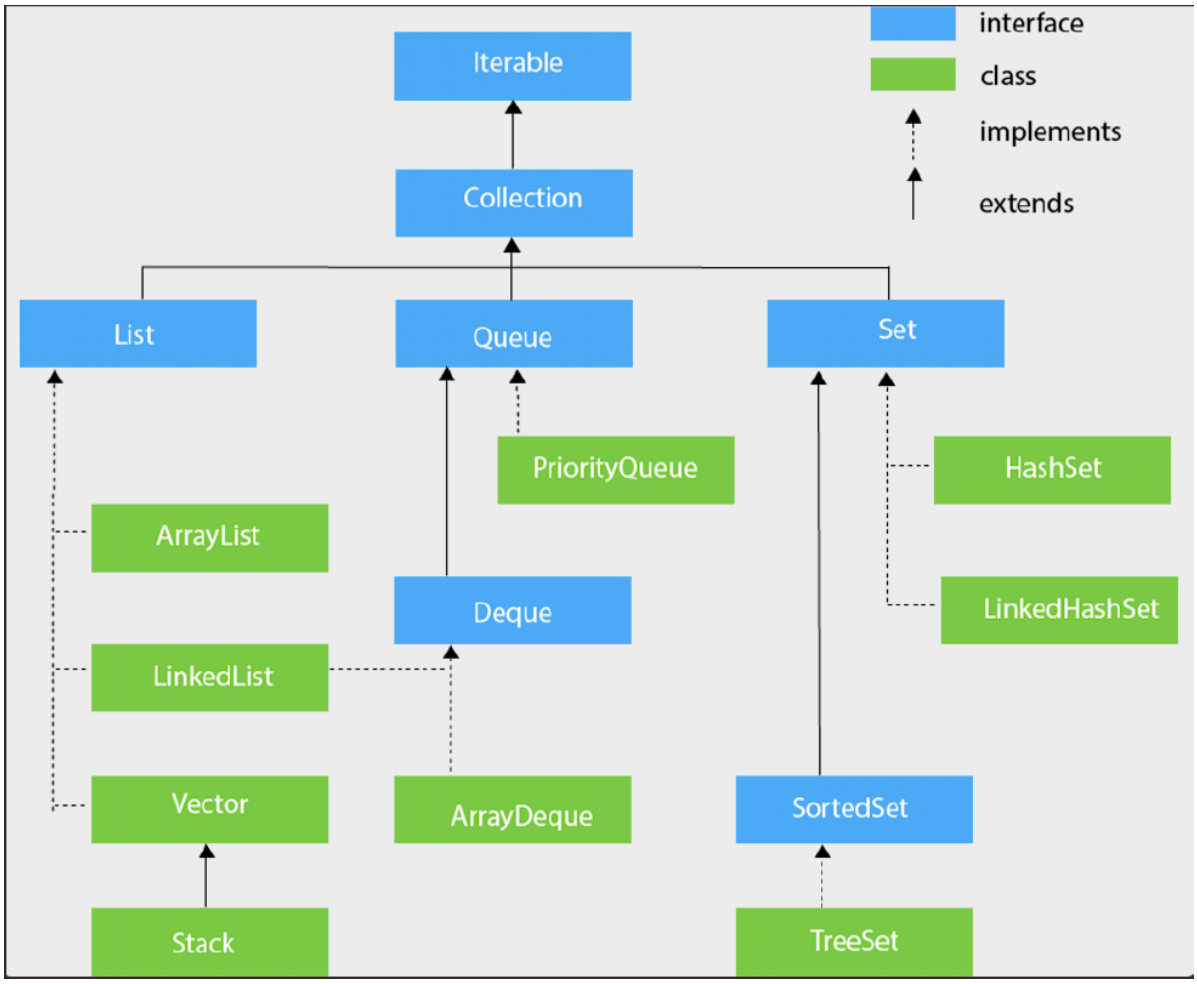
▼ All collections are iterable (Iterable is **an object, which one can iterate over**.)

Traverse: travel across or through / move back and forth or sideways.
"a probe is traversed along the tunnel"

1→2→3.....→10 int i;

- sync : running slow (Vector) sync also means Thread-Safe
- not sync : fast (ArrayList)

▼ Diagram



▼ List

```
interface, takes duplicate, index, keeps order
      allows multiple null values
```

▼ Set

an interface like List BUT: `unique, does not maintain order`

```
• Set is child interface of Collection.
• If we want to represent a group of individual objects as a single entity where dupl
icates are NOT allowed, and insertion order NOT preserved then we should go for Set.
HashSet: not sync, no order, allow null
LinkedHashSet: not sync, ordered (insertion), accepts null
TreeSet : sorted in asc. order, DOES NOT accept null
```

▼ LinkedList vs ArrayList

ArrayList	LinkedList
ArrayList internally uses a **dynamic array** to store the elements.	LinkedList internally uses a **doubly linked list** to store the elements.
Manipulation with ArrayList is **slow** because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory.	Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
An ArrayList class can **act as a list** only because it implements List only.	LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
ArrayList is **better for storing and accessing** data. **get()**	LinkedList is **better for manipulating** data. **add(), remove()**

▼ Vector

Vector implements a dynamic array. It is similar to ArrayList, but with two differences :
• Vector is synchronized(thread-safe)
• Vector contains many legacy methods that are not part of the collection framework.

▼ QUEUE

- It is child interface of Collection.
- A Queue is a ***First In First Out (FIFO)*** data structure.

▼ Loop Through Collection

1. *For each loop*

2. *Any other loop (for, while, do while) by using get(index) method*
3. ***forEach method*** that came with java 8 (lambda expression)
4. ***Iterator***

▼ How to use iterator

1. We need to ***create iterator object to able to use*** it***.***
2. ***We move the pointer using next() method***
3. ***hasNext() method return true, if there is still next value***
4. *We can remove values using remove() method*

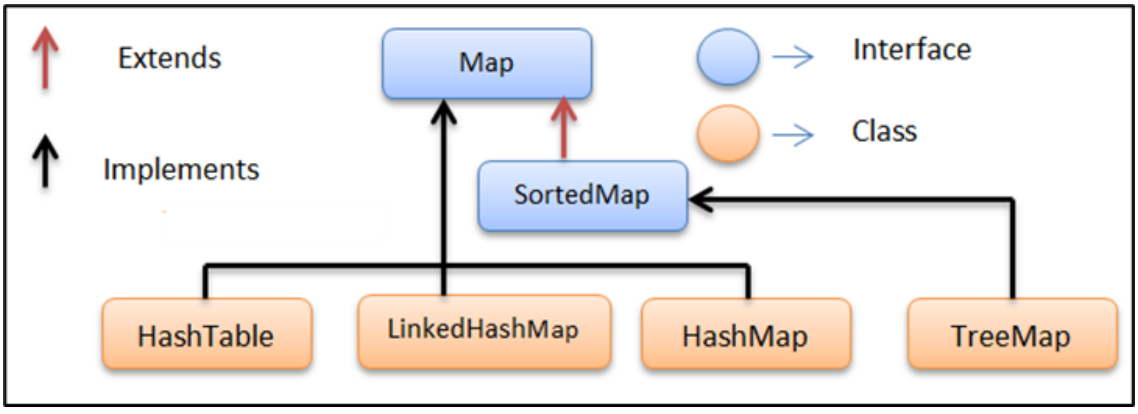
▼ PIQ: What is the difference between Iterator and For Each Loop?

- When using ***iterator*** object, we can ***remove values*** while looping
- When using ***for each*** loop, we ***cannot remove values*** from the collection
- We need to ***create iterator object to able to use*** it
- For ***each loop works with a temporary variable***

▼ Collection of Pairs : Map

▼ Data structure ***based on key + value pairs (example: dictionary)***

▼ Map interface does not extend Collection interface



▼ Map (I)

- pair of data, key & value format.

- Key must be unique (not duplicated)
- Does not support primitive

▼ HashMap (C)

accept null key, faster, order is random, using hash code

▼ LinkedHashMap (C)

accept null key, keeps the insertion order

▼ TreeMap (C)

does not accept null key, sorted order, using ASCII table

▼ HashTable (C)

does not accept null key, order is random, synchronized