Lesson:

Collection framework in Java

Pre Requisites:

- Basics of Java
- OOPS in Java
- Packages, Classes, Interfaces

List of concepts involved :

- Collections in Java
- What is a Framework
- Need for a separate Collection framework
- Collection framework hierarchy
- Collection interface hierarchy
- Methods of Collection interface
- Example
- Map interface
- Hierarchy of Map interface
- Methods of Map interface
- Example

Topic - Collections in Java

- A group of individual objects which are represented as a single unit is known as the collection of the objects.
- Java Collection means a single unit of objects.
- Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.
- The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.
- This Collection framework provides a set of interfaces and classes to implement various data structures and algorithms.

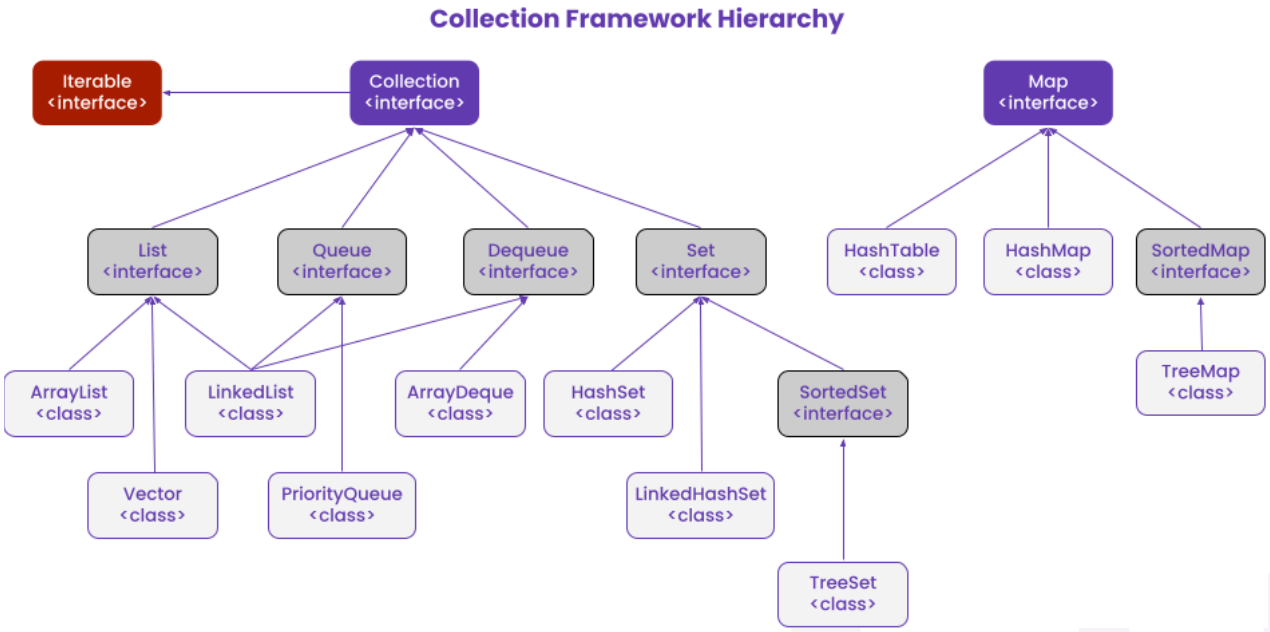
What is Framework ?

- A framework is a set of classes and interfaces which provide a ready-made architecture.
- Using frameworks saves time and reduces the risk of errors. You don't need to write everything from scratch.

Need for a separate Collection framework

- The Collection framework represents a unified architecture for storing and manipulating a group of objects. It has Interfaces and its implementations, i.e., classes and the algorithm.
- Before the Collection framework was introduced, the standard methods for grouping Java objects were Arrays or Vectors, or Hash Tables which had no common interface.
- Therefore, though the main aim of all the collections is the same, the implementation of all these collections was defined independently and had no correlation among them.

- And also, it is very difficult for the users to remember all the different methods, syntax, and constructors present in every collection class.



The Collection interface (java.util.Collection) and Map interface (java.util.Map) are the two main "root" interfaces of Java collection classes.

Topic: The Collection Interface Hierarchy

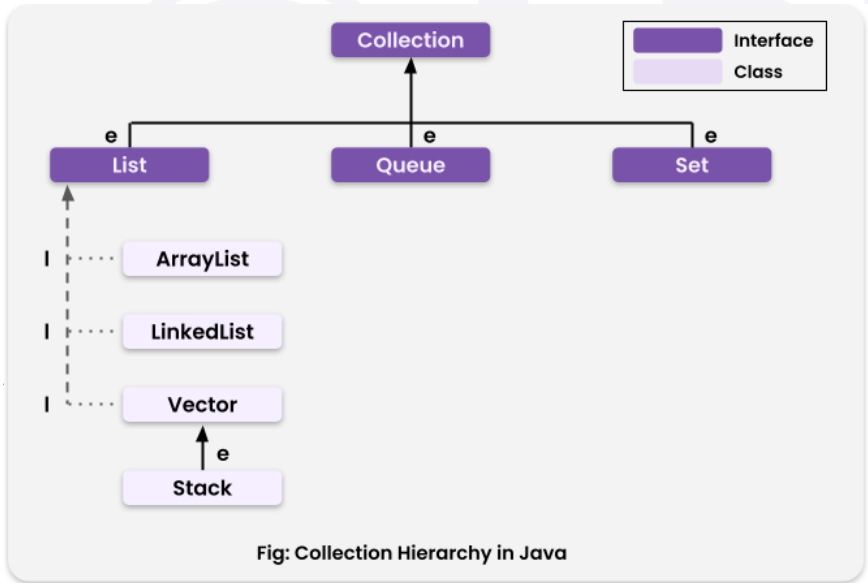


Fig: Collection Hierarchy in Java

e-> extends & i-> implements

Here, we are seeing the hierarchy of List Interface and a similar hierarchy is followed for all Interfaces.

- The **java.util** package contains all the classes and interfaces for the Collections framework.
- It has interfaces that can have methods, but the methods declared in an interface are by default abstract (only method signature, no body). Interfaces specify what a class must do and not how. It is the blueprint of the class. For example: List, Queue.
- These interfaces are extended by the collection framework.

- Classes like ArrayList class implements the List interface which is a subinterface of the Collection Interface.
- A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.
  For example: ArrayList, Linkedlist.

There are many implementations of each of the interfaces present inside Java Collection framework.
Few of the most commonly used classes of Collection framework are -

- HastSet
- TreeSet
- ArrayList
- LinkedList
- Stack
- Vector
- HashMap
- TreeMap
- Collections

Methods of Collection Interface

Method	Description
add(Object)	This method is used to add an object to the collection.
addAll(Collection c)	This method adds all the elements in the given collection to this collection.
clear()	This method removes all of the elements from this collection.
contains(Object o)	This method returns true if the collection contains the specified element.
containsAll(Collection c)	This method returns true if the collection contains all of the elements in the given collection.
equals(Object o)	This method compares the specified object with this collection for equality.
isEmpty()	This method returns true if this collection contains no elements.
max()	This method is used to return the maximum value present in the collection.
remove(Object o)	This method is used to remove the given object from the collection. If there are duplicate values, then this method removes the first occurrence of the object.

removeAll(Collection c)	This method is used to remove all the objects mentioned in the given collection from the collection.
size()	This method is used to return the number of elements in the collection.
toArray()	This method is used to return an array containing all of the elements in this collection.

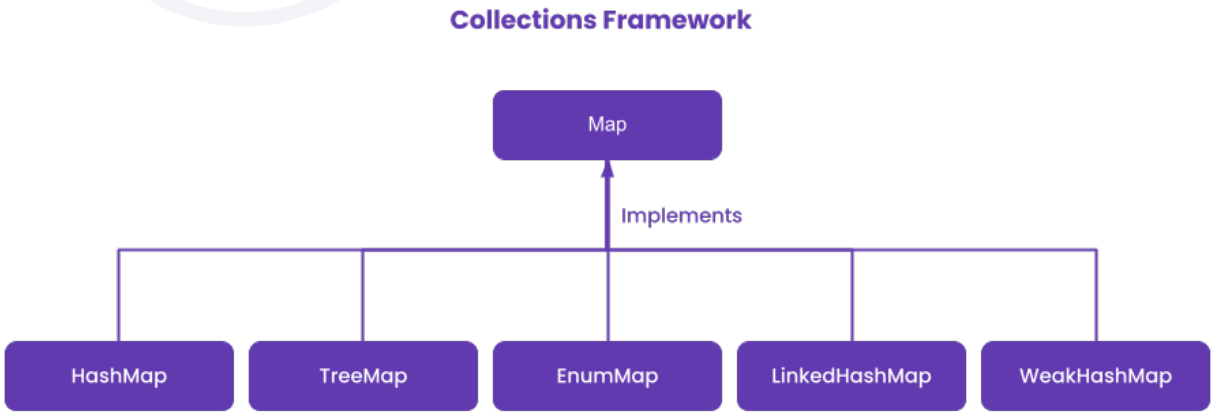
EXAMPLE:
Let's understand how to use collections framework to implement an ArrayList class.
We will declare an arraylist, add elements to it, display all the elements, remove the third element, display the final elements.
https://pastebin.com/4Z91cy57

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Content
[1, 2, 3, 4, 5]
[1, 2, 3, 5]
1 2 3 5
Process finished with exit code 0
```

MAP INTERFACE

- The Map interface is present in java.util package represents a mapping between a key and a value.
- The Map interface is not a subtype of the Collection interface.
- Therefore it behaves a bit differently from the rest of the collection types.
- A map contains unique keys.

Hierarchy of Map Interface



- There are two interfaces for implementing Map in java. They are Map and SortedMap, and three classes: HashMap, TreeMap, and LinkedHashMap.
- A Map cannot contain duplicate keys and each key can map to at most one value.
- The order of a map depends on the specific implementations. For example, TreeMap and LinkedHashMap have predictable orders, while HashMap does not.

- A HashMap stores key-value pairs in no particular order, a TreeMap stores key-value pairs in a sorted order by the keys and a LinkedHashMap stores key-value pairs in the order of insertion.

Method	Description
clear()	This method is used to clear and remove all of the elements or mappings from a specified Map collection.
containsKey(Object)	This method is used to check whether a particular key is being mapped into the Map or not. It takes the key element as a parameter and returns True if that element is mapped in the map.
containsValue(Object)	This method is used to check whether a particular value is being mapped by a single or more than one key in the Map. It takes the value as a parameter and returns True if that value is mapped by any of the key in the map.
entrySet()	This method is used to create a set out of the same elements contained in the map. It basically returns a set view of the map or we can create a new set and store the map elements into them.
equals(Object)	This method is used to check for equality between two maps. It verifies whether the elements of one map passed as a parameter is equal to the elements of this map or not.
get(Object)	This method is used to retrieve or fetch the value mapped by a particular key mentioned in the parameter. It returns NULL when the map contains no such mapping for the key.
isEmpty()	This method is used to check if a map is having any entry for key and value pairs. If no mapping exists, then this returns true.
keySet()	This method is used to return a Set view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa.
put(Object, Object)	This method is used to associate the specified value with the specified key in this map.
putAll(Map)	This method is used to copy all of the mappings from the specified map to this map.
remove(Object)	This method is used to remove the mapping for a key from this map if it is present in the map.

size()	This method is used to return the number of key/value pairs available in the map.
values()	This method is used to create a collection out of the values of the map. It basically returns a Collection view of the values in the HashMap.
getOrDefault(Object key, V defaultValue)	Returns the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key.

EXAMPLE:
Let's use a HashMap to create key value pairs and display their values using java Map classes' methods.
https://pastebin.com/TVHiDuMJ

```
/Library/Java/JavaVirtualMachines/jdk-
a:1
b:2
c:3
d:4

Process finished with exit code 0
```

Upcoming Class Teasers

- Linked Lists