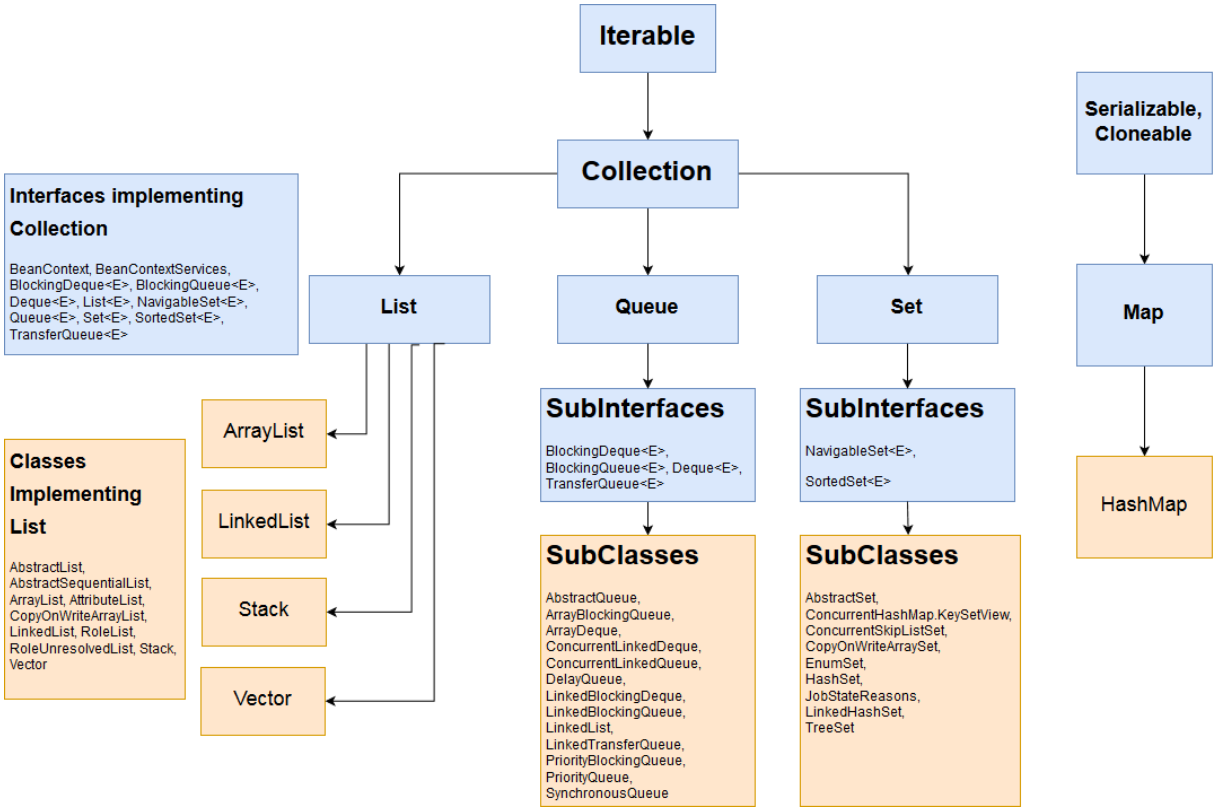
Table of Contents

Java Collections

Ways to access/organize data in the storage.



The Collections class is a utility class having static methods for doing operations on objects of classes which implement the Collection interface. For example, Collections has methods for finding the max element in a Collection.

The Collection interface defines methods common to structures which hold other objects. List and Set are subinterfaces of Collection, and ArrayList and HashSet are examples of concrete collections.

Classes that implements “Collection” (I) methods, can take advantage of the static Class “Collections” (C) methods.

List vs Set

List is an ordered sequence of elements whereas Set is a distinct list of elements which is unordered.

List<E>:

An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Set<E>:

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. As implied by its name, this interface models the mathematical set abstraction.

	List	Set
Duplicates	YES	NO
Order	ORDERED	DEPENDS ON IMPLEMENTATION
Positional Access	YES	NO

ArrayList vs HashSet

	ArrayList	HashSet
Implements	List (Interface)	Set (Interface)
Ordered	YES (Preserve the order of the elements)	NO (Unordered)
Synchronized	NO (not meant to be used in multi-threading and concurrent environment)	NO (not meant to be used in multi-threading and concurrent environment)
Iterable	YES (Iterator, ForEach)	YES (Iterator, ForEach)
Fail-Fast	YES (Will throw ConcurrentModificationException if is modified structurally once Iterator has been created)	YES (Will throw ConcurrentModificationException if is modified structurally once Iterator has been created)
Duplicates	YES	NO
Backed By	Array	HashMap
Access Method	Index Based: You can retrieve object by index.	Object Based: Elements is accessed directly from the Object Reference.

How HashSet Internally Works in Java

Not many programmer know that HashSet is internally implemented using HashMap in Java.
But, now a curious Java developer can question that, how come HashSet uses HashMap, because you need a key value pair to use with Map, while in HashSet we only store one object.

HashMap allows duplicate values and this property is exploited while implementing HashSet in Java. Since HashSet implements Set interface it needs to guarantee uniqueness and this is achieved by storing elements as keys with same value always.

HashMap vs HashSet

HashMap	**Hash Set**
HashMap is an implementation of Map interface	HashSet is an implementation of Set Interface
HashMap Stores data in form of key-value pair	HashSet Store only objects
Put method is used to add element in map	Add method is used to add element is Set
In hash map hashcode value is calculated using key object	Here member object is used for calculating hashcode value which can be same for two objects so equal () method is used to check for equality if it returns false that means two objects are different.
HashMap is faster than HashSet because unique key is used to access object	HashSet is slower than Hashmap