**Collections in java**

The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](https://www.tpointtech.com/java-arraylist), Vector, [LinkedList](https://www.tpointtech.com/java-linkedlist), [PriorityQueue](https://www.tpointtech.com/java-priorityqueue), HashSet, LinkedHashSet, TreeSet).

Any group of individual objects that are represented as a single unit is known as a Java Collection of Objects. In Java, a separate framework named the *“Collection Framework”* has been defined in JDK 1.2 which holds all the Java Collection Classes and Interface in it.

* In Java, the Collection interface (**java.util.Collection**) and Map interface (**java.util.Map**) are the two main “root” interfaces of Java collection classes.

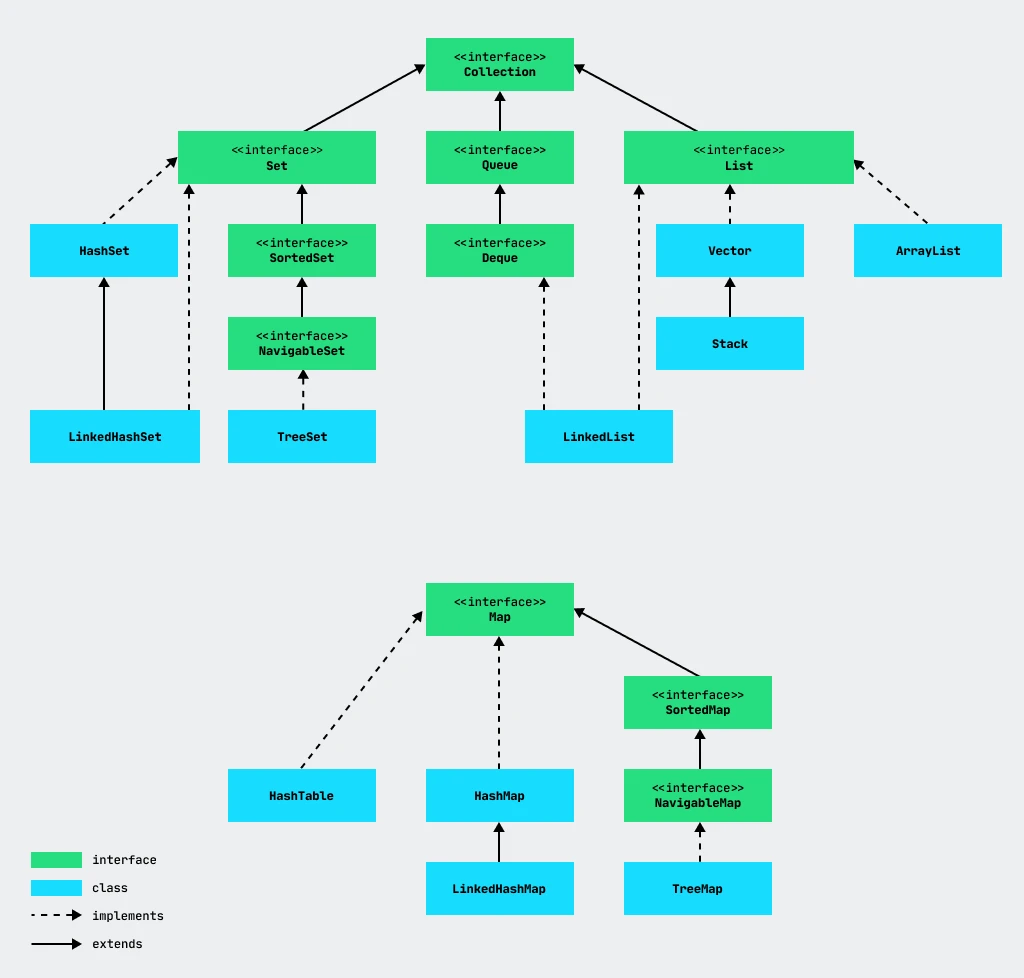
The **java.util** package contains all the [classes](https://www.tpointtech.com/object-and-class-in-java) and [interfaces](https://www.tpointtech.com/interface-in-java) that are required by the collection framework.

* **What is a framework?**

A framework provides a ready-made structure of classes and interfaces for building software applications efficiently. It simplifies adding new features by offering reusable components that perform similar tasks, eliminating the need to create a framework from scratch for each new project. This approach enhances object-oriented design, making development quicker, more consistent, and reliable.

* It provides readymade architecture.
* It represents a set of classes and interfaces.
* It is optional.

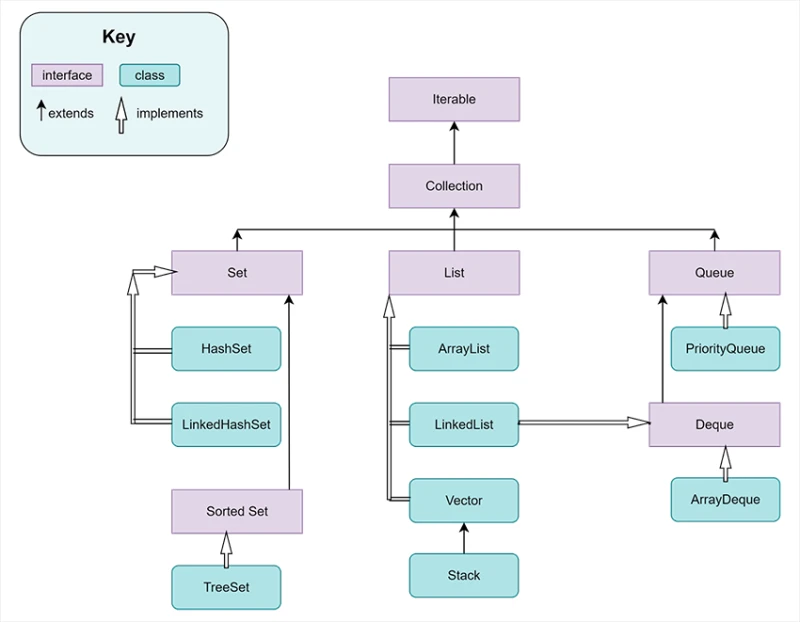
**Java Collections Framework**



The diagram shows that there are two basic interfaces that are implemented to form the rest of the classes and interfaces.

Let's take a look at these interfaces:

1. **Collection** — An ordinary collection that contains a set of elements (objects). This collection has basic methods for working with elements: insert (add, addAll), remove (remove, removeAll, clear), search (contains, containsAll), check whether collection is empty (isEmpty) and get size (size).
2. **Map** — A collection structured as key-value pairs. Moreover, each key in a **Map** is unique: no two keys have identical values. This collection is sometimes called a *dictionary*. **Map** is a separate interface. It does not implement the ***Collection*** interface, but is part of the **Java Collections Framework**.



**Collection interface**

The ***Collection*** interface extends the ***Iterable*** interface, which has a single method: iterator(). For us, this means that any collection that inherits ***Iterable*** will be able to return an iterator.

An **iterator** is a special object that you can use to access the elements of any collection, regardless of its specific implementation.

The figure shows that 3 interfaces inherit the ***Collection*** interface: ***List***, ***Queue*** and ***Set***. Now we'll look at each of them briefly.

***List*** is an ordered collection that allows duplicate values. A particular feature of a ***List*** is that its elements are numbered and can be accessed by number (index).

A ***Queue*** stores elements in the order they were added to the queue.

Unlike a list, a ***Set*** represents an unordered collection that does not allow repeated elements. The ***Set*** interface corresponds to the concept of a mathematical *set*.

|  |  |  |
| --- | --- | --- |
| **Collection** | **Class** | **Description** |
| **List** | ArrayList | List |
|  | LinkedList | Linked list |
|  | Vector | Vector |
|  | Stack | Stack |
|  | | |
| **Set** | HashSet | Set |
|  | TreeSet |  |
|  | LinkedHashSet |  |
|  | | |
| **Queue** | PriorityQueue | Queue |
|  | ArrayQueue |  |
|  | | |
| **Map** | HashMap | Map/Dictionary |
|  | TreeMap |  |
|  | HashTable |  |