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, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

# What is Collection in Java

* A Collection represents a single unit of objects, i.e., a group.

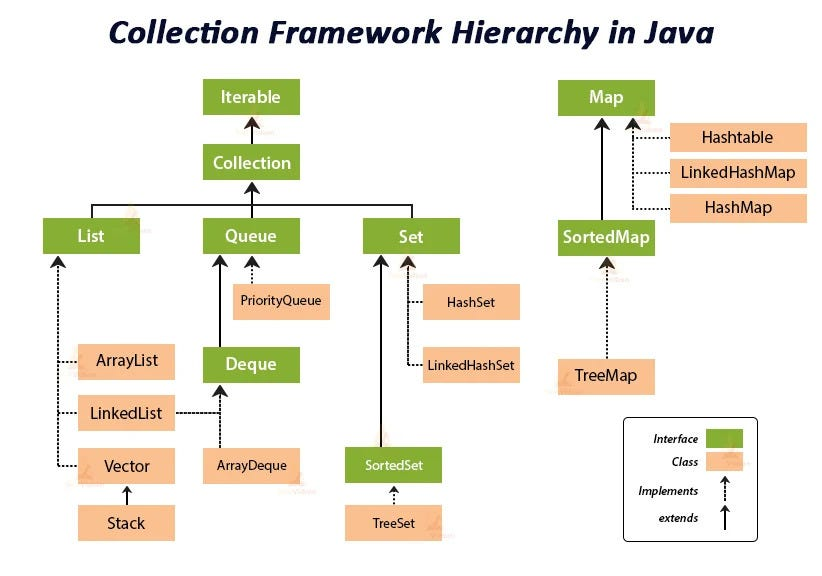
# What is a framework in Java

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

# What is 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

# Hierarchy of Collection Framework



## **Methods of Collection interface**

* **public boolean add(E e)** It is used to insert an element in this collection.
* **public boolean addAll(Collection<? extends E> c)** It is used to insert the specified collection elements in the invoking collection.
* **public boolean remove(Object element)** It is used to delete an element from the collection.
* **public boolean removeAll(Collection<?> c)** It is used to delete all the elements of the specified collection from the invoking collection.
* **default boolean removeIf(Predicate<? super E> filter)** It is used to delete all the elements of the collection that satisfy the specified predicate.
* **public boolean retainAll(Collection<?> c)** It is used to delete all the elements of invoking collection except the specified collection.
* **public int size()** It returns the total number of elements in the collection.
* **public void clear()** It removes the total number of elements from the collection.
* **public boolean contains(Object element)** It is used to search an element.
* public boolean containsAll(Collection<?> c) It is used to search the specified collection in the collection.
* **public Iterator iterator()** It returns an iterator.
* **public Object[] toArray()** It converts collection into array.
* **public <T> T[] toArray(T[] a)** It converts collection into array. Here, the runtime type of the returned array is that of the specified array.
* **public boolean isEmpty()** It checks if collection is empty.
* **default Stream<E> parallelStream()** It returns a possibly parallel Stream with the collection as its source.
* **default Stream<E> stream()** It returns a sequential Stream with the collection as its source.
* **default Spliterator<E> spliterator()** It generates a Spliterator over the specified elements in the collection.
* **public boolean equals(Object element)** It matches two collections.
* **public int hashCode()** It returns the hash code number of the collection.

## **Iterator interface**

Iterator interface provides the facility of iterating the elements in a forward direction only.

There are only three methods in the Iterator interface. They are:

Method Description

* public boolean hasNext() It returns true if the iterator has more elements otherwise it returns false.
* public Object next() It returns the element and moves the cursor pointer to the next element.
* public void remove() It removes the last elements returned by the iterator. It is less used.
* The Iterable interface is the root interface for all the collection classes. The Collection interface extends the Iterable interface and therefore all the subclasses of Collection interface also implement the Iterable interface.
* It contains only one abstract method. I.e.,
* Iterator<T> iterator()
* It returns the iterator over the elements of type T.

## **Iterable Interface**

* The Iterable interface is the root interface for all the collection classes. The Collection interface extends the Iterable interface and therefore all the subclasses of Collection interface also implement the Iterable interface.
* It contains only one abstract method. i.e.,
* Iterator<T> iterator()
* It returns the iterator over the elements of type T.

## 

## Collection Interface

* The Collection interface is the interface which is implemented by all the classes in the collection framework.
* It declares the methods that every collection will have.
* In other words, we can say that the Collection interface builds the foundation on which the collection framework depends.
* Some of the methods of Collection interface are Boolean add ( Object obj), Boolean addAll ( Collection c), void clear(), etc. which are implemented by all the subclasses of Collection interface.

# List Interface

* List interface is the child interface of Collection interface. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.
* List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.
  + To instantiate the List interface, we must use :
  + List <data-type> list1= new ArrayList();
  + List <data-type> list2 = new LinkedList();
  + List <data-type> list3 = new Vector();
  + List <data-type> list4 = new Stack();
* There are various methods in List interface that can be used to insert, delete, and access the elements from the list.

## ArrayList

* The ArrayList class implements the List interface.
* It uses a dynamic array to store the duplicate element of different data types.
* The ArrayList class maintains the insertion order and is non-synchronized.
* The elements stored in the ArrayList class can be randomly accessed.

import java.util.\*;

class TestJavaCollection1{

public static void main(String args[]){

ArrayList<String> list=new ArrayList<>();//Creating arraylist

list.add("Ravi");//Adding object in arraylist

list.add("Vijay");

list.add("Ravi");

list.add("Ajay");

//Traversing list through Iterator

Iterator itr=list.iterator();

while(itr.hasNext()){

System.out.println(itr.next());

}

}

}

## **LinkedList**

* LinkedList implements the Collection interface.
* It uses a doubly linked list internally to store the elements.
* It can store the duplicate elements.
* It maintains the insertion order and is not synchronized. In LinkedList, the manipulation is fast because no shifting is required.

