Collections

Abdul Haseeb

BS(AI)-II

Collections in java

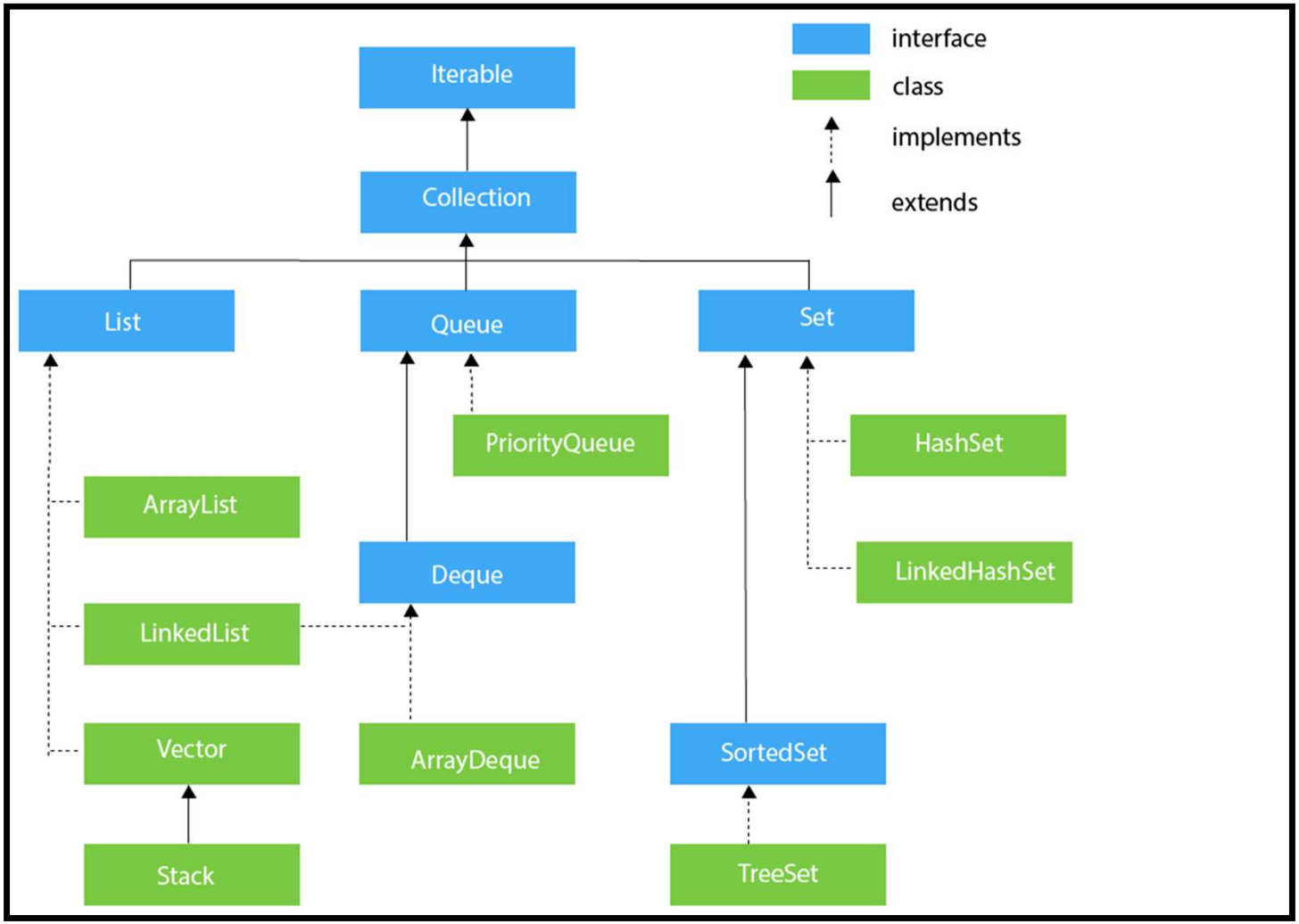
- Any group of individual objects that are represented as a single unit is known as a Java Collection of Objects.
- In Java, a separate framework named the *"Collection Framework"* has been defined in JDK 1.2 which holds all the Java Collection Classes and Interface in it.

What is framework?

- Set of classes and Interfaces
- Why?
  - Provides ready made architecture (set of methods)

Collection framework

- The Collection framework represents a unified architecture for storing and manipulating a group of objects.
- java.util.Collection
- It has:
  1. Interfaces and its implementations, i.e., classes
  2. Algorithm


interface
class
implements
extends
Iterable
Collection
List
Queue
Set
PriorityQueue
HashSet
ArrayList
Deque
LinkedHashSet
LinkedList
Vector
ArrayDeque
SortedSet
Stack
TreeSet

Iterator

- Iterator interface provides the facility to iterate the elements of a collection in a forward direction only

Collection Interface

- It builds the foundation of collection framework
- It contains methods, which are implemented by all the classes in Collection framework

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.

Generics vs Non-Generic

- Before JDK 5.0 Collection framework was Non-Generic (Non-Homogenous) which caused type safety issues.
- After JDK 5.0 Collection framework was made Generic, now every collection will be homogenous.

ArrayList

- Implements list interface
- Uses Dynamic Array

```
import java.util.*;
class TestJavaCollection1{
public static void main(String args[]){
ArrayList<String> list=new ArrayList<String>();//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());
}
}
}
```

Linked List

- LinkedList implements the Collection interface.
- It uses a doubly linked list internally to store the elements.
- It can store the duplicate elements.

```
import java.util.*;
public class TestJavaCollection2{
public static void main(String args[]){
LinkedList<String> al=new LinkedList<String>();
al.add("Ravi");
al.add("Vijay");
al.add("Ravi");
al.add("Ajay");
Iterator<String> itr=al.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

Vector

- Vector uses a dynamic array to store the data elements.
- It is similar to ArrayList, but a bit slower
- Contains many methods that are not the part of Collection framework.

```
import java.util.*;
public class TestJavaCollection3{
public static void main(String args[]){
Vector<String> v=new Vector<String>();
v.add("Ayush");
v.add("Amit");
v.add("Ashish");
v.add("Garima");
Iterator<String> itr=v.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

Stack

- The stack is the subclass of Vector.
- It implements the last-in-first-out data structure, i.e., Stack.
- The stack contains all of the methods of Vector class and also provides its methods like boolean push(), boolean peek(), boolean push(object o), which defines its properties.

```
import java.util.*;
public class TestJavaCollection4{
public static void main(String args[]){
Stack<String> stack = new Stack<String>();
stack.push("Ayush");
stack.push("Garvit");
stack.push("Amit");
stack.push("Ashish");
stack.push("Garima");
stack.pop();
Iterator<String> itr=stack.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

Queue Interface and its classes

- First in First Out
- Queue Interface has classes like: Priority queue, Deque, Array Deque
- javatpoint

Set Interface

- Set Interface extends collection interface
- Unordered
- Doesn’t allow duplicates

Hash Set

- HashSet class implements Set Interface.
- It represents the collection that uses a hash table for storage.
- Hashing is used to store the elements in the HashSet.

```
import java.util.*;
public class TestJavaCollection7{
public static void main(String args[]){
//Creating HashSet and adding elements
HashSet<String> set=new HashSet<String>();
set.add("Ravi");
set.add("Vijay");
set.add("Ravi");
set.add("Ajay");
//Traversing elements
Iterator<String> itr=set.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

- Explore LinkedHashSet, same but used LinkedList

Sorted Set

- SortedSet is the alternate of Set interface that provides a total ordering on its elements.
- The elements of the SortedSet are arranged in the increasing (ascending) order.

TreeSet

- Java TreeSet class implements the Set interface that uses a tree for storage.
- Access time is quite fast

```
import java.util.*;
public class TestJavaCollection9{
public static void main(String args[]){
//Creating and adding elements
TreeSet<String> set=new TreeSet<String>();
set.add("Ravi");
set.add("Vijay");
set.add("Ravi");
set.add("Ajay");
//traversing elements
Iterator<String> itr=set.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```