COJ :: Basics of Collections

TalentSprint

Licensed To Skill

Version 1.0.4

Collection Framework

The content in this presentation is aimed learners to learn:

- Define collections
- Understanding the importance of collections
- Identifying core collection interfaces and their implementation classes.
- Perform basic operations on all collections

Collection Framework

Java Collections Framework

- A Collection is a structured group of objects manipulate as a single object. Corresponds to a bag.

Limitations of Static Array

- Arrays are fixed size.
- An array can only hold one type of objects (including primitives).

Example: Employee[] emp = new Employee[10];

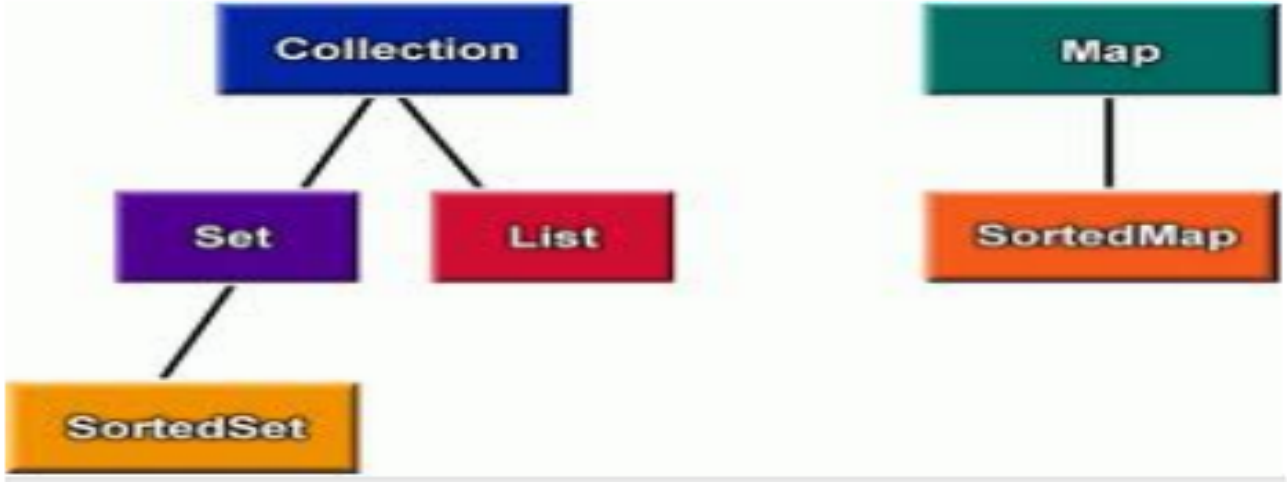
Note So we need Dynamic Arrays.

Collection Framework

- Collections are dynamic in nature and can grow as necessary.
- Collections are Heterogeneous, can store different objects as part of a collection.
- Collections can contain only Objects (reference types) and not primitives.
- Collections are defined in java.util package

Collection Framework

Collection interfaces



- Collections are primarily defined through a set of interfaces.
- As they are interfaces they do not provide any implementation
- They are supported by a set of classes that implement the interfaces

The Set Interface

Corresponds to the mathematical definition of a set

- No duplicates elements are allowed
- No ordering of elements.
- Indexing is not there

Set implementation classes are:

HashSet:

- Implemented using a hash table.
- No ordering of elements.

TreeSet:

- Implemented using a tree structure.
- Guarantees ordering of elements.

The Set Interface

add(Object) : Adds the specified element to this set if it is not already present

remove(Object) : Removes the specified element from this set if it is present

size() : Returns the number of elements in this set

The Set Interface

contains(Object) : Returns true if this set contains the specified element.

containsAll(Collection) : Returns true if this set contains all of the elements of the specified collection.

retainAll(Collection) : Retains only the elements in this set that are contained in the specified collection

Collection Framework

HashSet Example

```
import java.util.HashSet;
import java.util.Iterator;
public class SetDemo {
        public static void view(Iterator<String> it){
                while(it.hasNext())
                        System.out.println(it.next());
        }
        public static void main(String[] args) {
                HashSet<String> hs=new HashSet<String>();
                hs.add("Raju");
                hs.add("Kumar");
                hs.add("Vamsi");
                hs.add("Arun");
                hs.add("Vijay");
                hs.add("Rama");
                hs.add("Vamsi");
                hs.add("Kiran");
                hs.add("Kumar");
                hs.add("Raju");
                view(hs.iterator());
        }
}
```

This is generic type. We will see it in coming sessions

Will not allow the duplicates as hashcode is same but Order is unpredicted

Output:

```
Raju
Vijay
Arun
Kumar
Kiran
Vamsi
Rama
```

The List Interface

The List interface corresponds to an order group of elements.

- Duplicates elements are allowed
- Insertion ordering is maintained for elements.
- Access to elements via indexes, like arrays

The List Interface

List implementation classes are:

ArrayList :

- Its an array based implementation
- Elements can be accessed directly via the get and set methods using indexes.

LinkedList:

- Its a double linked list implementation.
- Gives better performance on add and remove operations when compared to ArrayList

Important methods of List interface:

add(Object) : adds element at the end of the list.

add(index, Object) : adds element at the specified index position.

remove(Object) : Removes the first occurrence of the specified element

Important methods of List interface:

indexOf(Object) : Returns the index of the first occurrence of the specified element

get(index) : Returns the element at the specified position in this list.

set(index, Object) : Replaces the element at the specified position in this list.

The Map Interface

- A Map is an object that maps keys to values
- Keys are unique, values can be duplicated
- A key is an object used to retrieve a value in Map
- Map does not extend Collection interface

The Map Interface

Map implementation classes are:

HashMap:

- The implementation is based on a hash table.
- No ordering on (key, value) pairs.

TreeMap:

- The implementation is based on tree structure.
- (key, value) pairs are ordered on the key.

Important methods of List interface:

put(Object key, Object value) : Associates the specified value with the specified key in this map.

get(Object key) : Returns the value to which the specified key is mapped.

remove(Object Key) : Removes the mapping for a key from this map if it is present.

Important methods of List interface:

keySet() : Returns a Set view of the keys contained in this map.

values() : Returns a Collection view of the values contained in this map.

Collection Framework