Collections

Why collections?

Collections are used to hold a collection of objects.

- List holds objects based on order of insertion and can hold non unique collection
- Set holds objects without any order and are unique
- Queue used to hold the elements about to be processed in FIFO(First In First Out) order.

Why collections?

- Arrays are not resizable.
- Java Collections Framework provides lots of different useful data types, such as linked lists (allows insertion anywhere in constant time), re-sizeable array lists (like Vector but cooler).

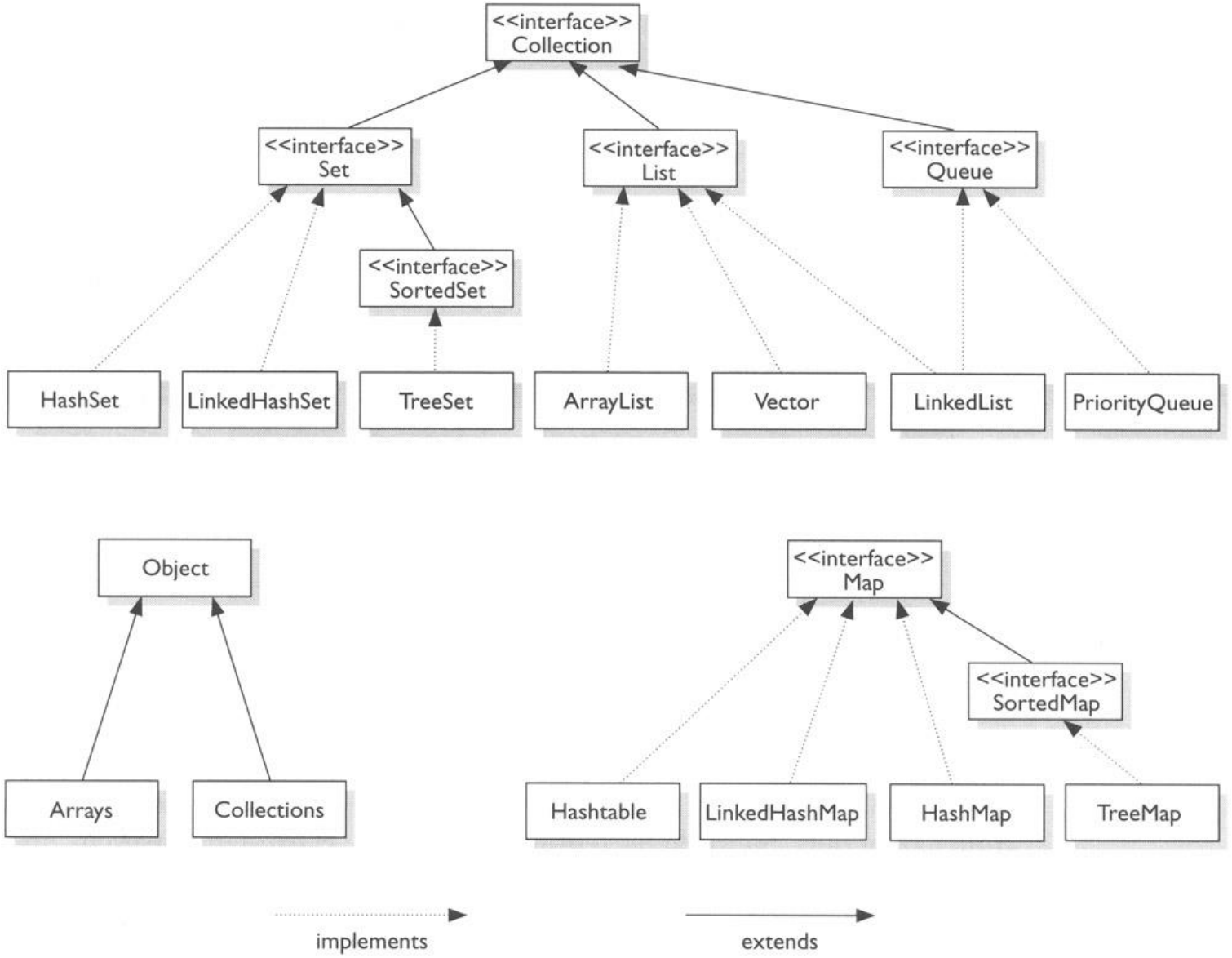
Why collections?

- Java Collections Framework provides abstractions, so you can refer to a list as a List, whether backed by an array list or a linked list; and you can refer to a map/dictionary as a Map, whether backed by a red-black tree or a hashtable.
- Java Collections Framework allows you to use the right data structure, because one size does not fit all.

Different Collections Interfaces

- List
- Set
- Queue
- Maps

Hierarchy


<<interface>>
Collection
<<interface>>
Set
<<interface>>
List
<<interface>>
Queue
<<interface>>
SortedSet
HashSet
LinkedHashSet
TreeSet
ArrayList
Vector
LinkedList
PriorityQueue
Object
Arrays
Collections
<<interface>>
Map
<<interface>>
SortedMap
Hashtable
LinkedHashMap
HashMap
TreeMap
implements
extends

List

- List is an interface with ArrayList, Vector and LinkedList as concrete implementations.
- List hold objects based on the insertion order and are non unique.
- Vector is deprecated
- Depending on the requirement ArrayList or LinkedList is used.
- Arraylist is used when more retrievals and less removals
- Linkedlist is used when more removals and less retrievals

Array List

- ArrayList is a part of collection framework and is present in java.util package.
- It provides us **dynamic** arrays in Java.
- Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed.
- ArrayList inherits AbstractList class and implements List interface.
- ArrayList is initialized by a size, however the size can increase if collection grows or shrunk if objects are removed from the collection.
- Java ArrayList allows us to randomly access the list.
- ArrayList can not be used for primitive types, like int, char, etc. We need a wrapper class for such cases.

Array List

- **Constructors in Java ArrayList**:
- ArrayList(): This constructor is used to build an empty array list
- ArrayList(Collection c): This constructor is used to build an array list initialized with the elements from collection c
- ArrayList(int capacity): This constructor is used to build an array list with initial capacity being specified

Array List

- Array is that it has a fixed length whereas ArrayList can dynamically grow and shrink after addition and removal of elements.
- **Arraylist** class implements List interface and it is based on an Array data structure.
- Array List declaration:

ArrayList<String> obj = new ArrayList<String>();

ArrayList<Integer> list=new ArrayList<Integer>();

List<String> obj = new ArrayList<String>();

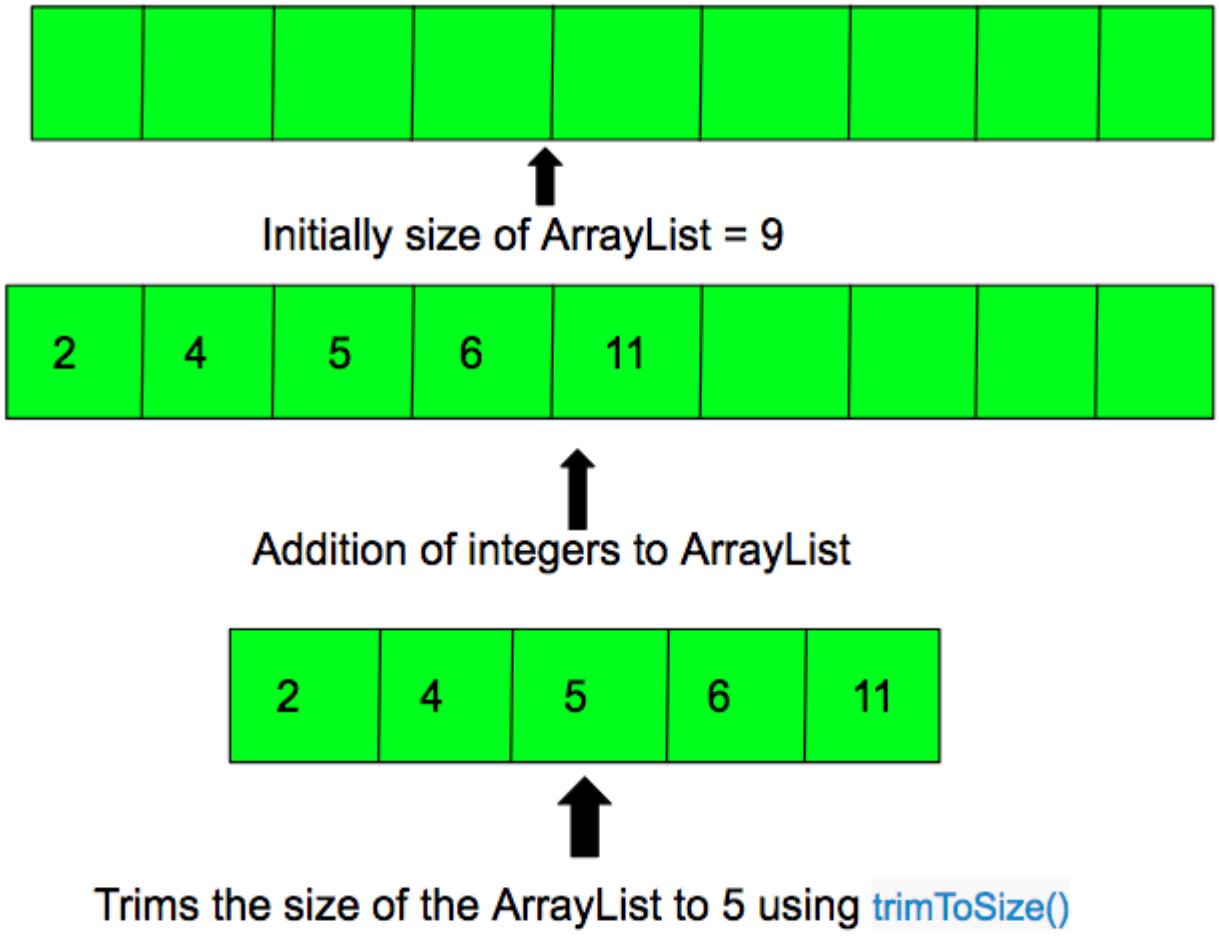
The type that a list can contain is given between the diamond brackets.

Array List

- **Methods in Java ArrayList:**
- void clear(): This method is used to remove all the elements from any list.
- void add(int index, Object element): This method is used to insert a specific element at a specific position index in a list.
- void trimToSize(): This method is used to trim the capacity of the instance of the ArrayLis to the list's current size.
- int indexOf(Object O): The index the first occurrence of a specific element is either returned, or -1 in case the element is not in the list.
- int lastIndexOf(Object O): The index the last occurrence of a specific element is either returned, or -1 in case the element is not in the list.
- **Object remove(int index):** This method removes an element from the specified index. It shifts subsequent elements(if any) to left and decreases their indexes by 1.
- Object[] toArray(): This method is used to return an array containing all of the elements in the list in correct order.
- **Object get(int index):** This method returns element at the specified index.
- boolean addAll(Collection C): This method is used to append all the elements from a specific collection to the end of the mentioned list, in such a order that the values are returned by the specified collection's iterator.
- boolean add(Object o): This method is used to append a specificd element to the end of a list.
- boolean addAll(int index, Collection C): Used to insert all of the elements starting at the specified position from a specific collection into the mentioned list.

Array List

- **Methods in Java ArrayList:**
- void trimToSize(): This method is used to trim the capacity of the instance of the ArrayLis to the list's current size.



Array	ArrayList
Array is a fixed length data structure that is you can not change length of Array once created in Java	ArrayList is a variable length Collection class that is ArrayList re-size itself when gets full depending upon capacity and load factor
We can not use Generics along with Array, as Array instance knows about what kind of type it can hold and throws ArrayStoreException, if you try to store type which is not convertible into type of Array	ArrayList allows you to use Generics to ensure type-safety
Array can contain both primitive data types as well as objects of a class depending on the definition of the array	ArrayList only supports object entries, not the primitive data types
Java provides add() method to insert element into ArrayList	We use assignment operator to store element into Array
Its mandatory to provide size of Array	We can create ArrayList with default size of 10
Each array object has the length variable which returns the length of the array	Length of the ArrayList is provided by the size() method
Array can be multi dimensional	ArrayList is always single dimensional
Use Array, when you know that the size is fixed	Use ArrayList, when you don't know about the size of elements

sort an ArrayList in **descending order**

- We are using **Collections.reverseOrder()** method along with Collections.sort() in order to sort the list in decreasing order. In the below example we have used the following statement for sorting in reverse order.
  Collections.sort(arraylist,Collections.reverseOrder());
- Collections.sort(list);
  Collections.reverse(list);