Language Advanced Concept : Collections

1. Java Collection framework
2. Collection Interface
3. Interface that extends Collection interface
4. Map Interface
5. Comparable
6. Comparators

Java collection framework

→ A framework is a set of classes and interfaces which provide a ready-made architecture

→ Any group of individual objects which are represented as a singe unit is known as collection of objects.

→ JCf is a set of classes & interfaces that implements commonly used data structures like set, list, map, queue, linkedlist etc

Need for separate collection framework in Java ?

→ Before collection framework (before JDK 1.2), the standard methods for grouping java objects were Arrays or Vectors or Hashtables.

→ There were no common interface
∴ The ways of each type was different

```
int arr[] = new int[] {1,2,3,4};
vector<Integer> v = new Vector();
Hashtable <Integer, String> h = new HashTable();
```

```
v.addElement(5);
h.put(1, "abc");
```
} diff. way to insert

```
System.out.println(arr[0]);
System.out.println(v.elementAt(0));
System.out.println(h.get(0));
```
} diff. way to access

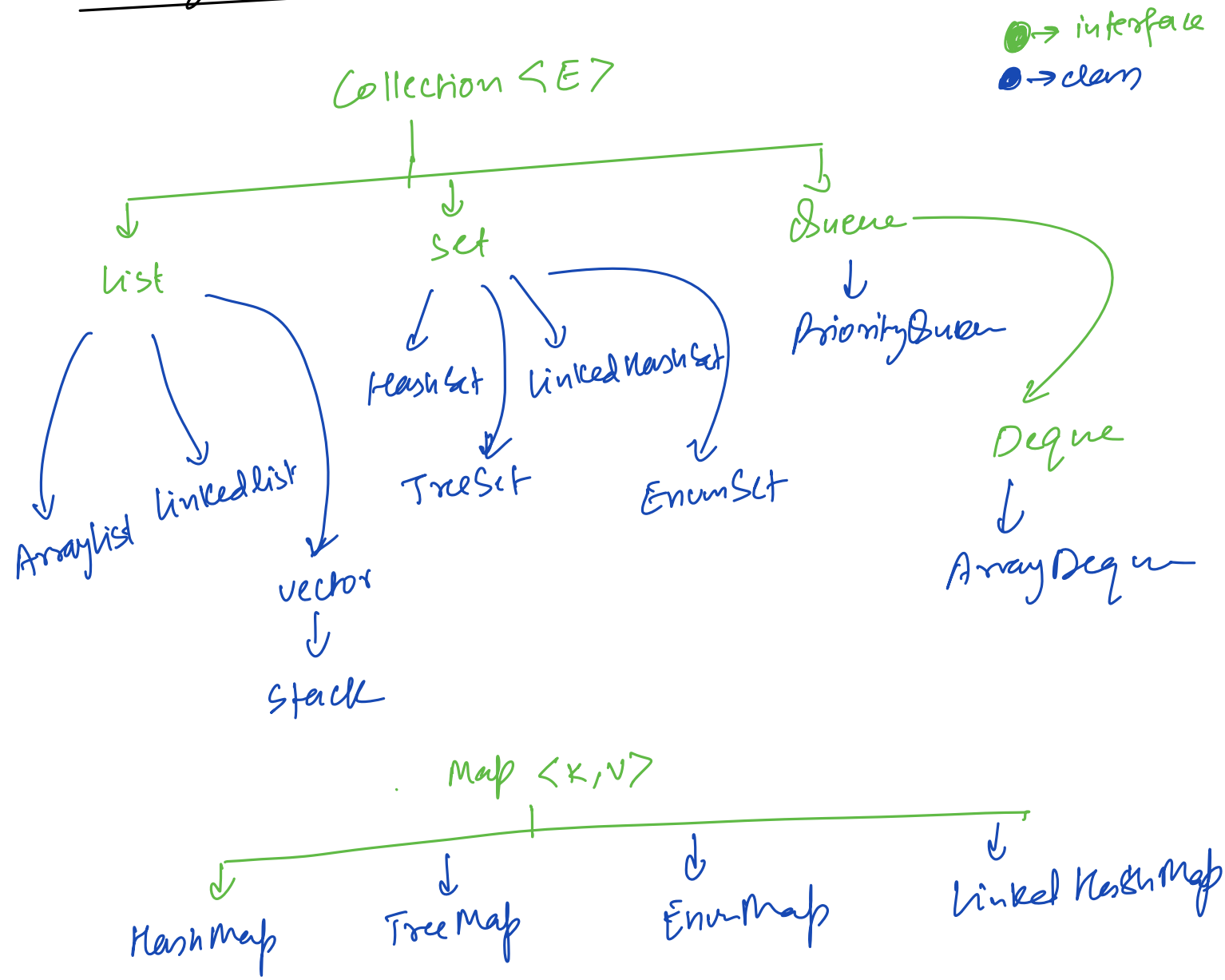
Advantages of JCF

→ consistent API

→ Reduces programming effort

→ Increases program speed & quality

Hierarchy of the Java Collection framework



Iterator <>

Collection Interface

→ Root interface of Java collection framework

→ It is part of java.util package

→ There is no direct implementation of this interface

→ The implementation classes which eventually implements it are
ArrayList, vector, stack, HashSet etc.

add()
size()
remove()
iterator()
addAll()
removeAll()
(Read)

Interfaces that extends the collection interface

1. List : ordered, allows duplicates
2. Set : unordered, doesn't allow duplicates
3. SortedSet : same as set + ordered
4. Queue : As name suggests, it will honor FIFO & similar type operation
5. Deque : More flexible queue, can have FIFO & LIFO both.

List Interface

→ ordered
→ allows duplicates

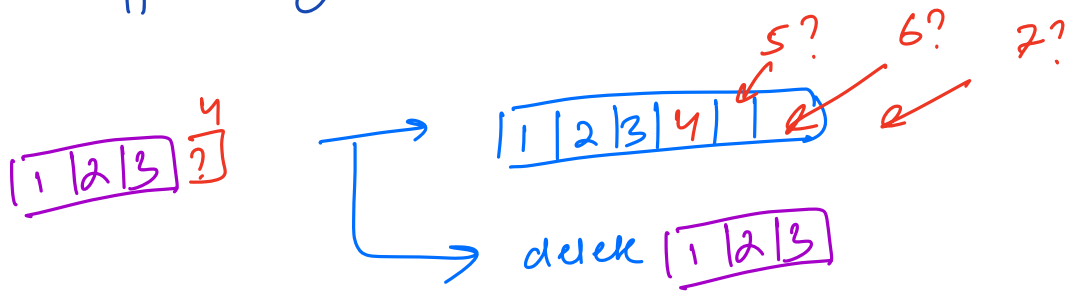
Classes which implements List:

1. ArrayList : resizable array
2. Vector : syncronized resizable array
3. Stack : subclass of vector which implements standard LIFO
4. LinkedList : doubly-linked list (doesn't use contigous memory)

Array list

↳ resizable array

1. Backing array
2. Resizing
3. Dynamic sizing (shrink & expand)
4. Efficiency : O(1)



Insert N elements

$1 \to 2 \to 4 \to 8 \to 16 \to 32 \to \cdots \to N$

total no. of operations $= 1 + 2 + 4 + 8 + \cdots + N$ (log terms)

$= O(N)$

N operation (insertion) → N iteration

1 operation → O(1) iteration

Vector

all same features as Arraylist + synchronized

Stack

→ It gives basic LIFO operation support

→ It also gives related methods to perform such operations.

→ push, pop, peek, empty, search

LinkedList

→ Double linked list

→ Uses non. contigous memory

→ Elements are linked using pointers & references.

Set interface

→ unordered

→ doesn't allow duplicates

HashSet is most common implementation of set interface.

LinkedHashSet

HashSet + maintain insertion order

SortedSet interface

→ don't contain duplicates

→ maintain natural order (sorted order)

TreeSet

→ Balanced Binary Search Tree

Map interface

→ Map
  → Sorted Map

<key, value>

Implemention classes

→ Hashmap

→ Linked Hashmap

→ TreeMap

Queue interface

→ support FIFO order

→ Allows deletion & insertion in O(1) for FIFO

LinkedList

→ Double linked list

→ Uses non. contigous memory

→ Elements are linked using pointers & references.

→ Implements queue interface too

Priority Queue

→ It's a queue which orders elements based on a priority

→ The priority will be natural order and can be provided as custom priority also.

→ It is a heap data structure implementation

Deque interface

→ More flexible queue

→ It can be used as LIFO, FIFO or anything.

Comparables

→ It is a Java interface which allows us to define natural order.

int compareTo(T other)

Comparator

→ It is an interface which allows to define custom order for a class.

int compare(T obj1, T obj2)