Object Oriented Programming with Java (OOPJ)

Session 5: Arrays

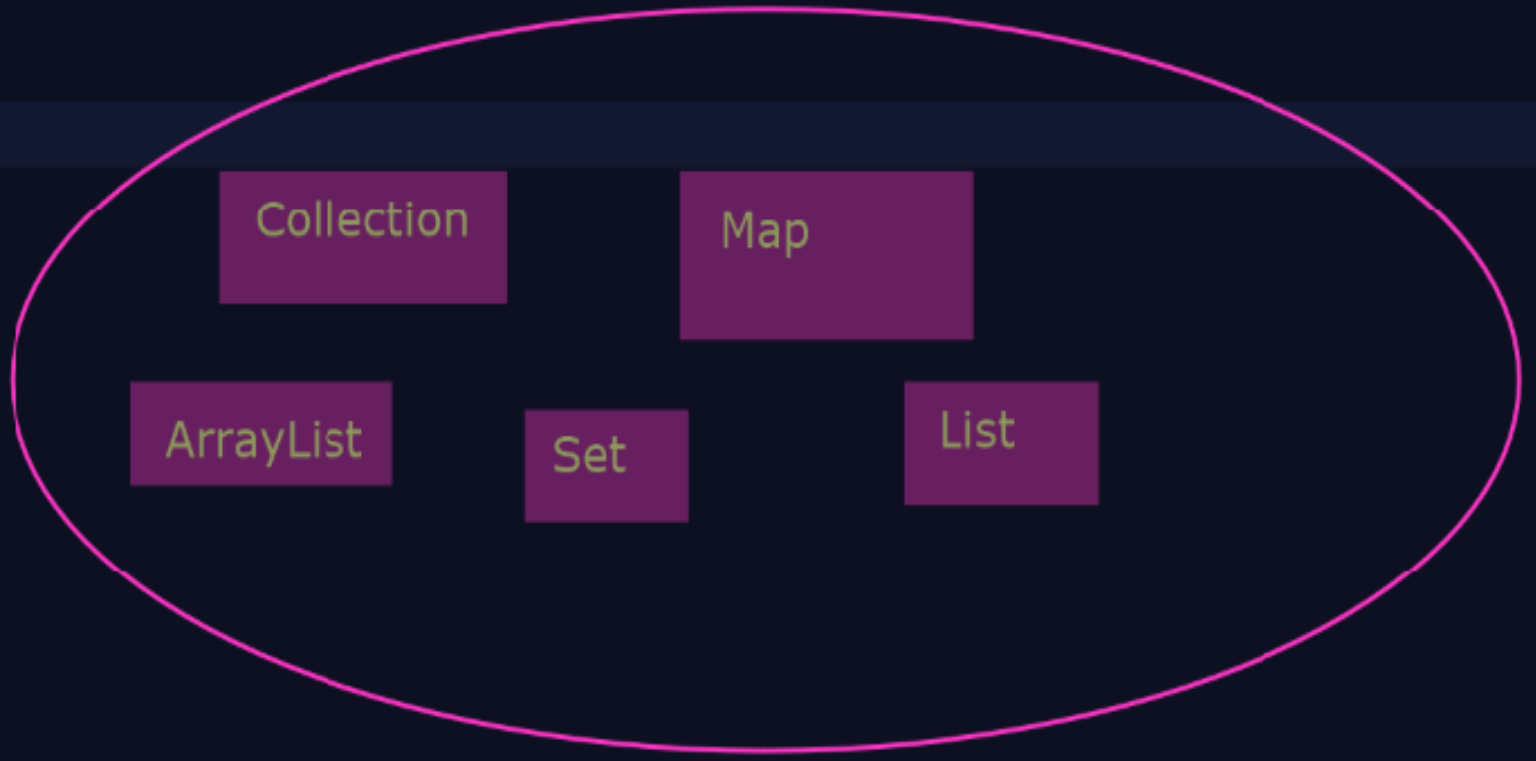
Kiran Waghmare

-A framework is a set of classes and interfaces which provide a readymade architecture.

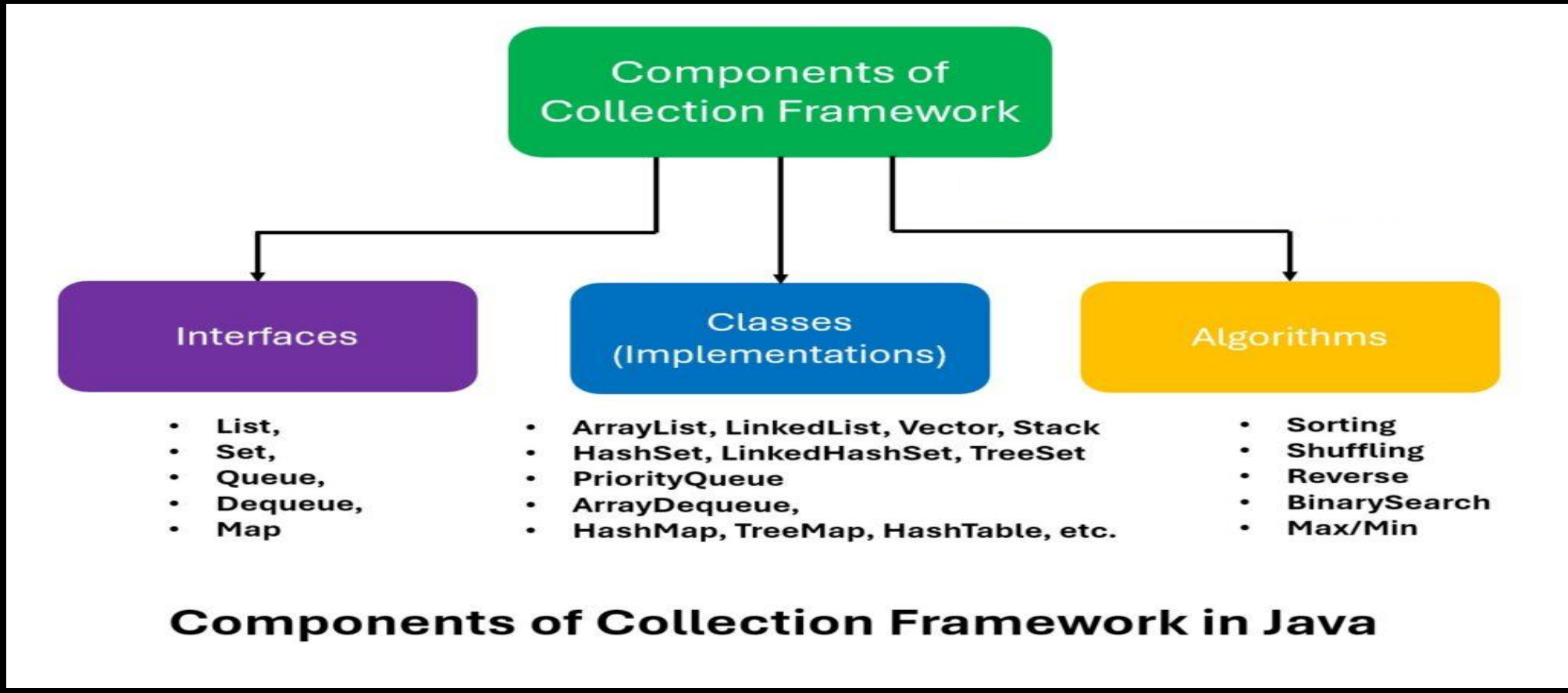
Collection Framework:

-Collection Framework is Java API(Application Programming Interface) which provides architecture to store and manipulate group of object.

Collections:



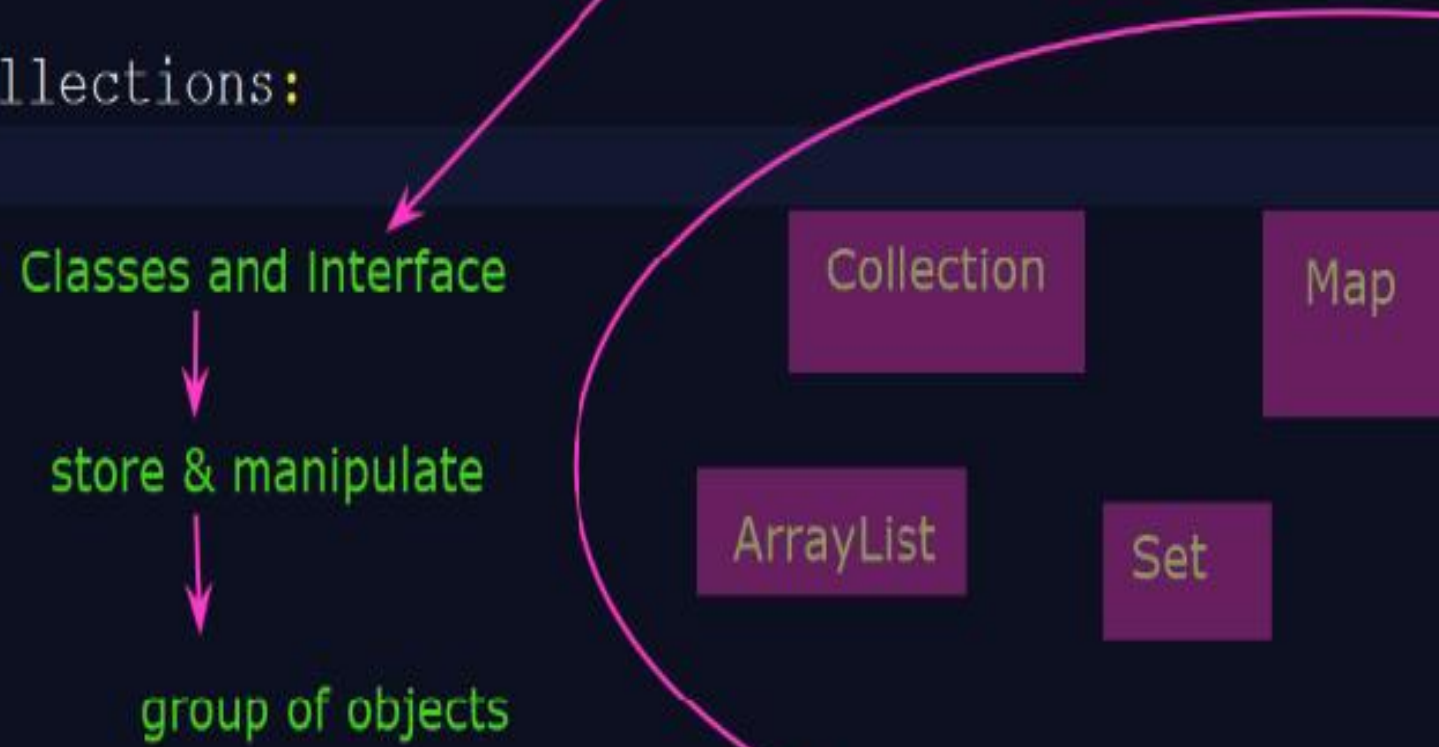
Collections Framework



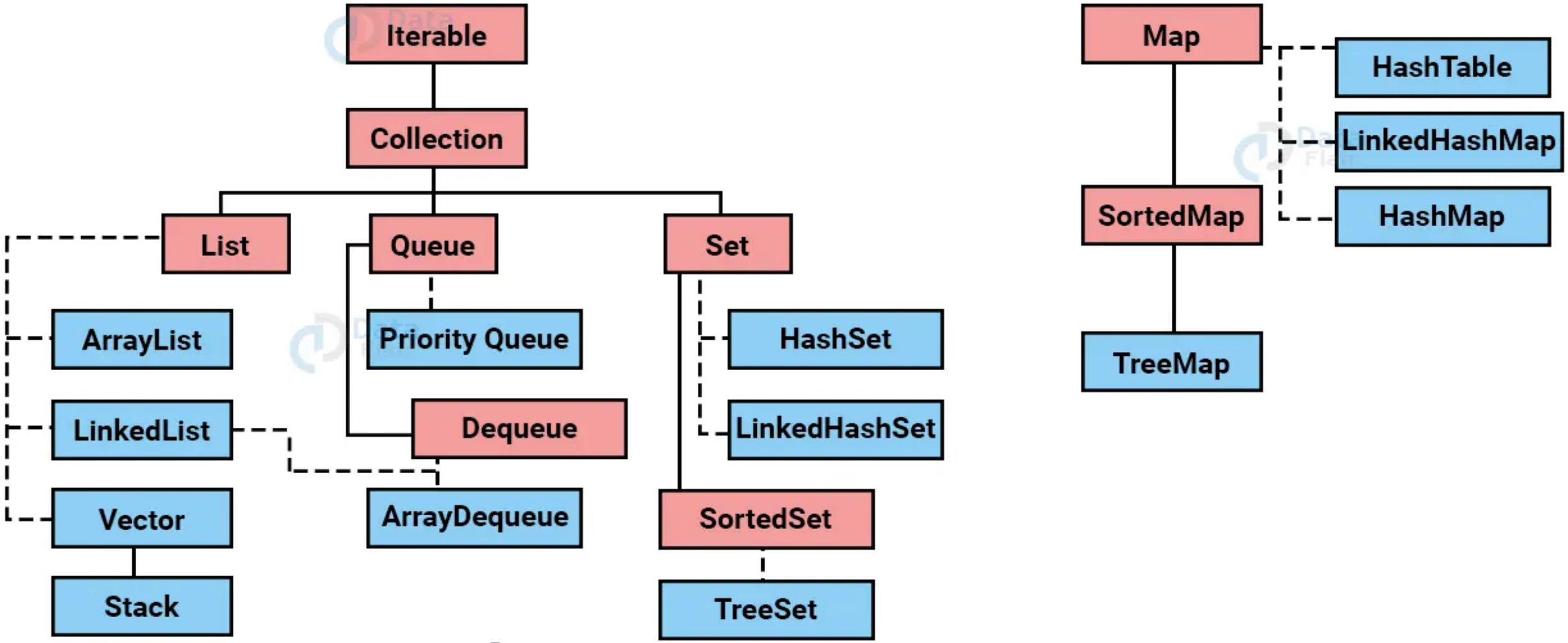
Collection Framework:

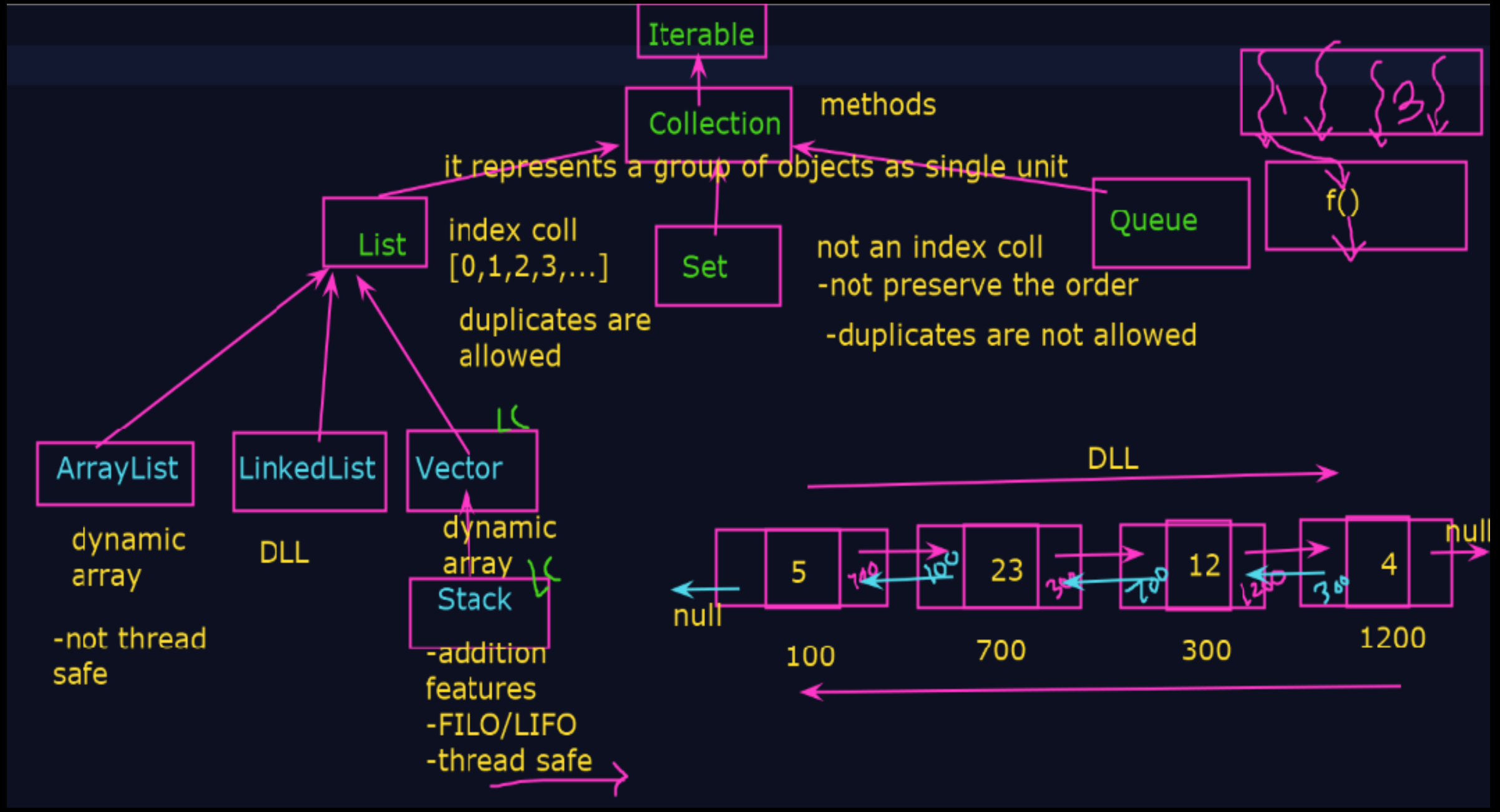
-Collection Framework is Java API(Application Programming Interface) which provides architecture to store and manipulate group of object.

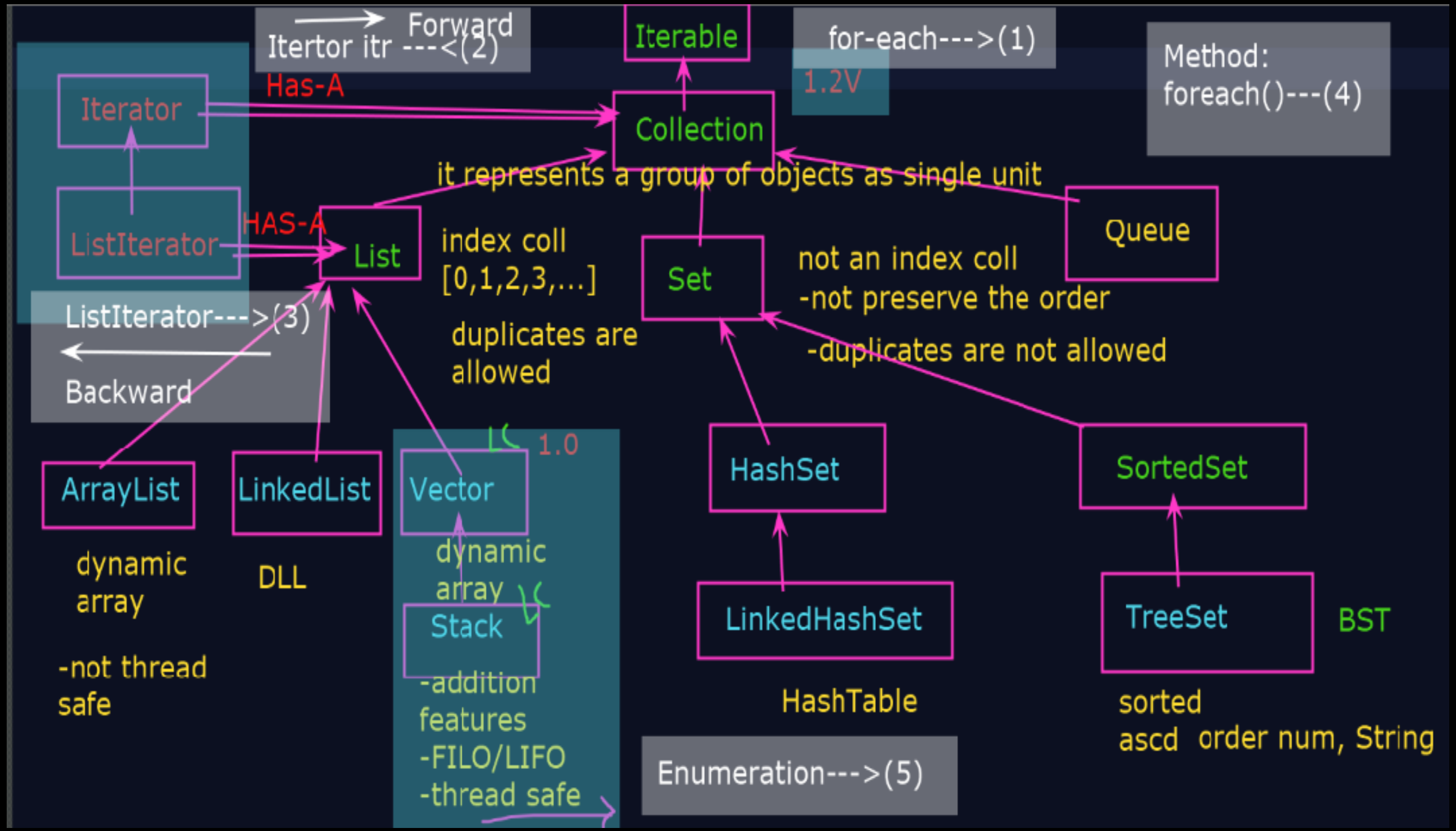
Collections:

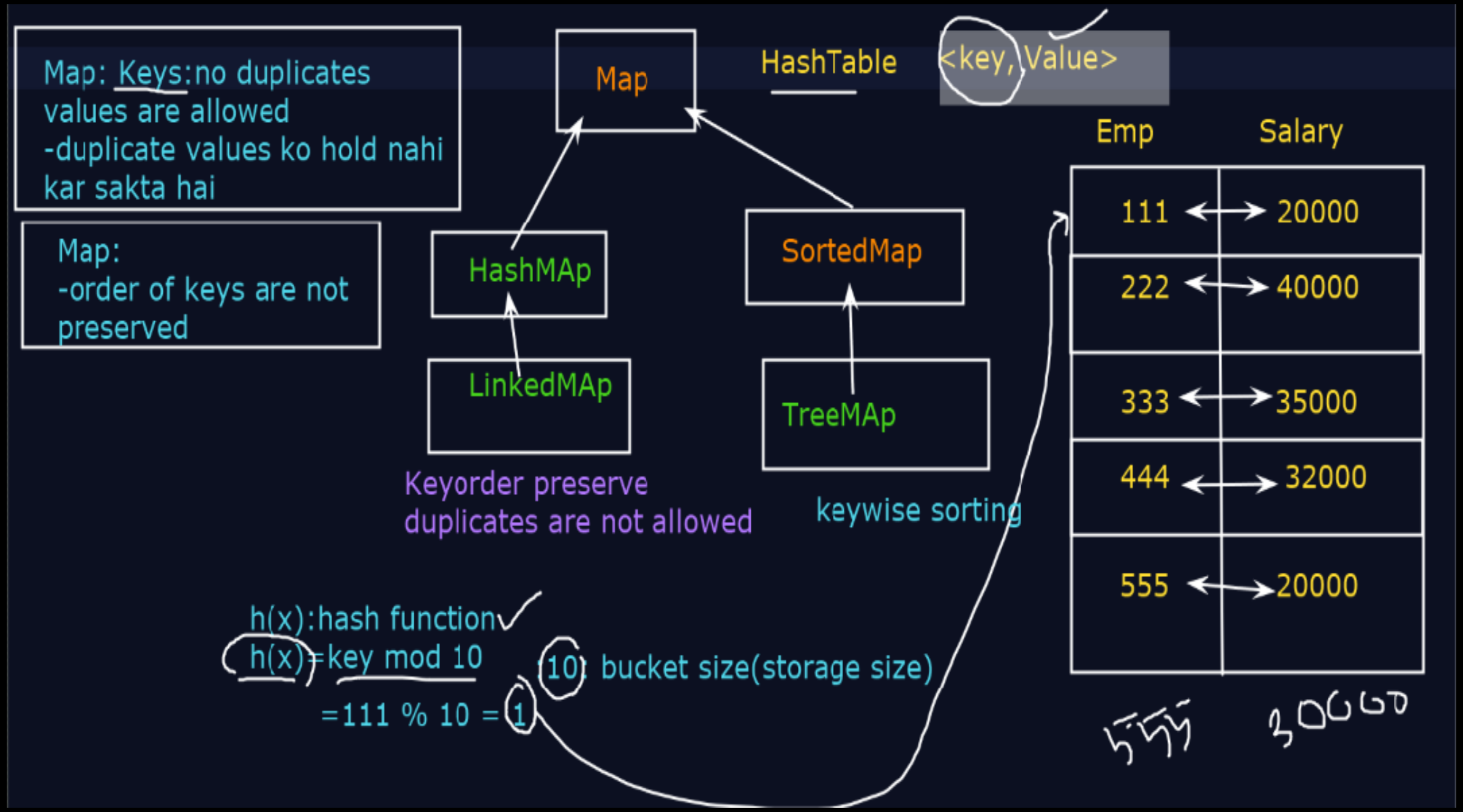


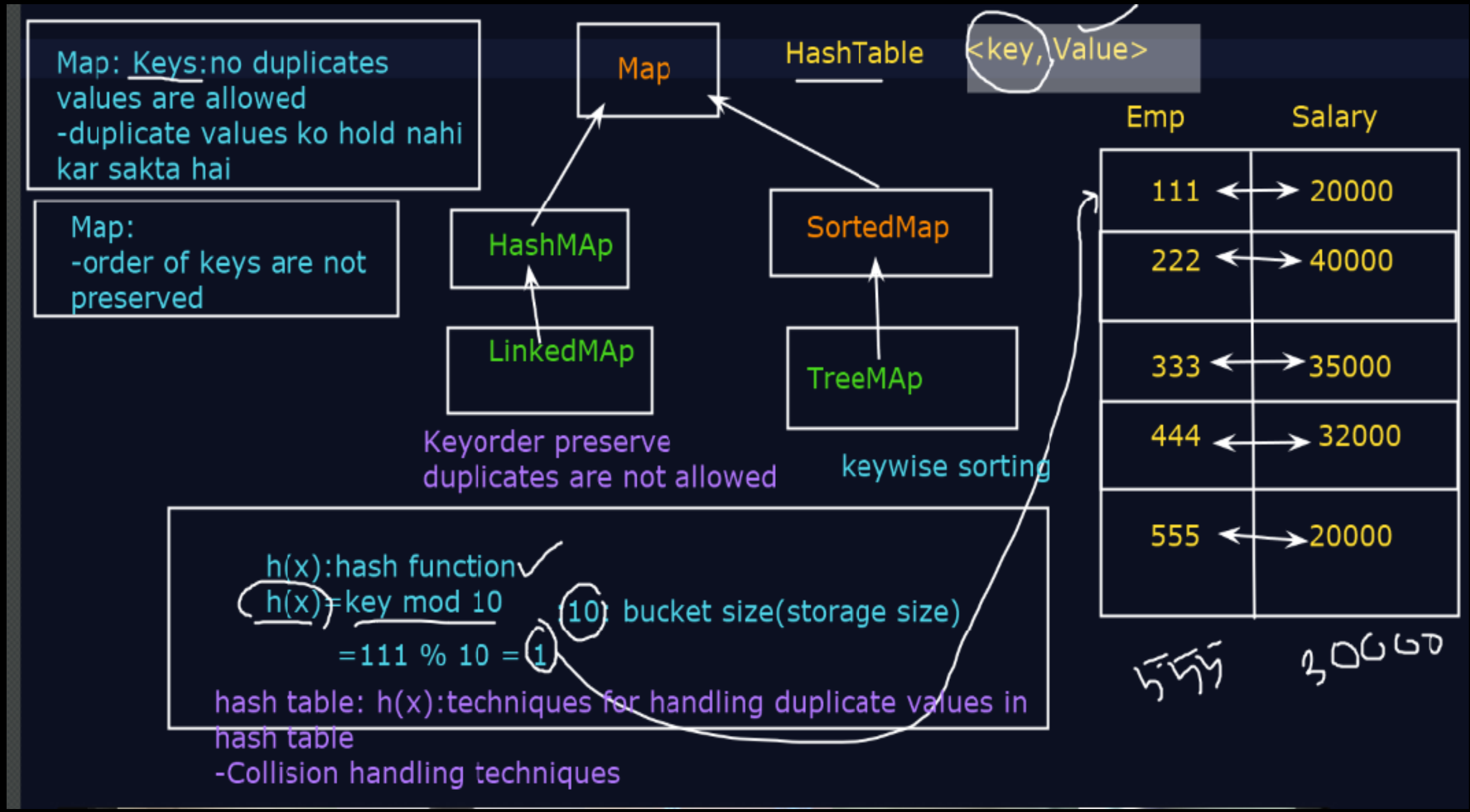
Hierarchy of Collection Framework in Java




Iterable
Collection
methods
it represents a group of objects as single unit
List
index coll
[0,1,2,3,...]
duplicates are allowed
Set
not an index coll
-not preserve the order
-duplicates are not allowed
Queue
f()
ArrayList
LinkedList
Vector
dynamic array
DLL
dynamic array
Stack
-not thread safe
-addition features
-FILO/LIFO
-thread safe
DLL
5
23
12
4
null
null
100
700
300
1200


Forward
Itertor itr ---<(2)
Iterable
for-each--->(1)
1.2V
Method:
foreach()---(4)
Has-A
Iterator
Collection
it represents a group of objects as single unit
HAS-A
ListIterator
List
index coll
[0,1,2,3,...]
Set
not an index coll
-not preserve the order
Queue
duplicates are
allowed
-duplicates are not allowed
ListIterator--->(3)
Backward
1.0
ArrayList
LinkedList
Vector
HashSet
SortedSet
dynamic
array
DLL
dynamic
array
Stack
-addition
features
-FILO/LIFO
-thread safe
LinkedHashSet
TreeSet
BST
-not thread
safe
HashTable
sorted
ascd order num, String
Enumeration--->(5)


Map: Keys:no duplicates
values are allowed
-duplicate values ko hold nahi
kar sakta hai
Map
HashTable
<key, Value>
Map:
-order of keys are not
preserved
HashMAp
SortedMap
LinkedMAp
TreeMAp
Keyorder preserve
duplicates are not allowed
keywise sorting
Emp
Salary
111
20000
222
40000
333
35000
444
32000
555
20000
h(x):hash function
h(x)=key mod 10
=111 % 10 = 1
10 bucket size(storage size)


Map: Keys:no duplicates
values are allowed
-duplicate values ko hold nahi kar sakta hai
Map
HashTable
<key, Value>
Map:
-order of keys are not preserved
HashMAp
SortedMap
LinkedMAp
TreeMAp
Keyorder preserve
duplicates are not allowed
keywise sorting
Emp
Salary
111
20000
222
40000
333
35000
444
32000
555
20000
h(x):hash function
h(x)=key mod 10
=111 % 10 = 1
10: bucket size(storage size)
hash table: h(x):techniques for handling duplicate values in hash table
-Collision handling techniques

Collection Interface methods

- add(Object)
- remove(Object)
- contains(Object)
- size()

ArrayList

- All methods of Collection interface +
- add(index,Object)
- remove(index)
- get(index)
- indexOf(Object)
- lastIndexOf(Object)

LinkedList

- All methods of Collection interface +
- peek()
- poll()
- offer(Object)
- pollFirst(), pollLast()
- peekFirst(), peekLast()
- addFirst(Object), addLast(Object)
- removeFirst(), removeLast()
- getFirst(), getLast()

Vector

- All methods of Collection interface +
- capacity()

HashSet, LinkedHashSet

- All methods of Collection interface +

TreeSet

- All methods of Collection interface +
- Subset Related Methods
  - headSet(Object)
  - tailSet(Object)
  - subSet(Object,Object)
- first(), last()
- lower(Object), higher(Object)
- ceiling(Object), floor(Object)
- pollFirst(), pollLast()

Map

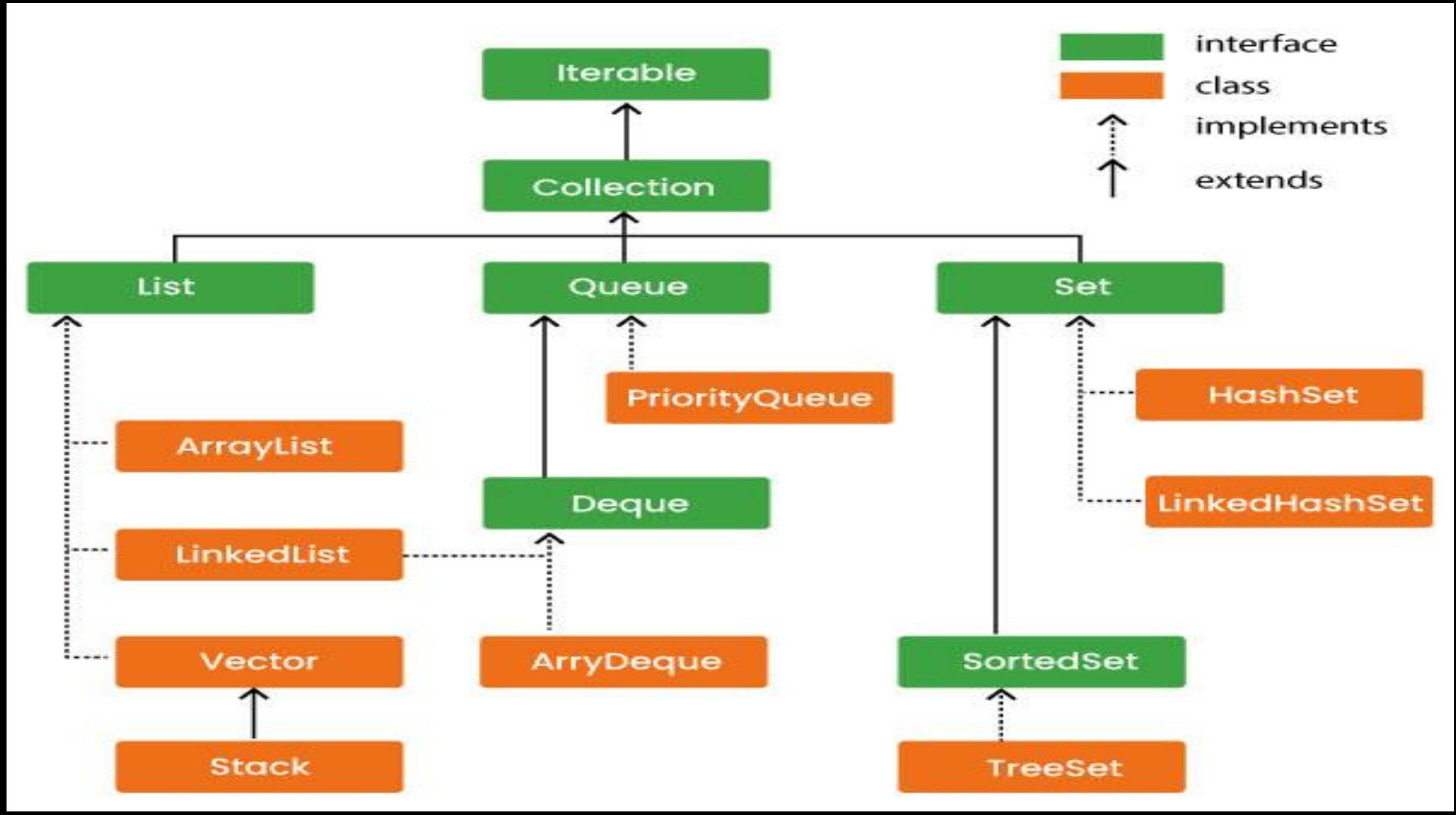
- put(K,V)
- get(KeyObject)
- remove(KeyObject)
- containsKey()
- containsValue()
- size()
- clear()

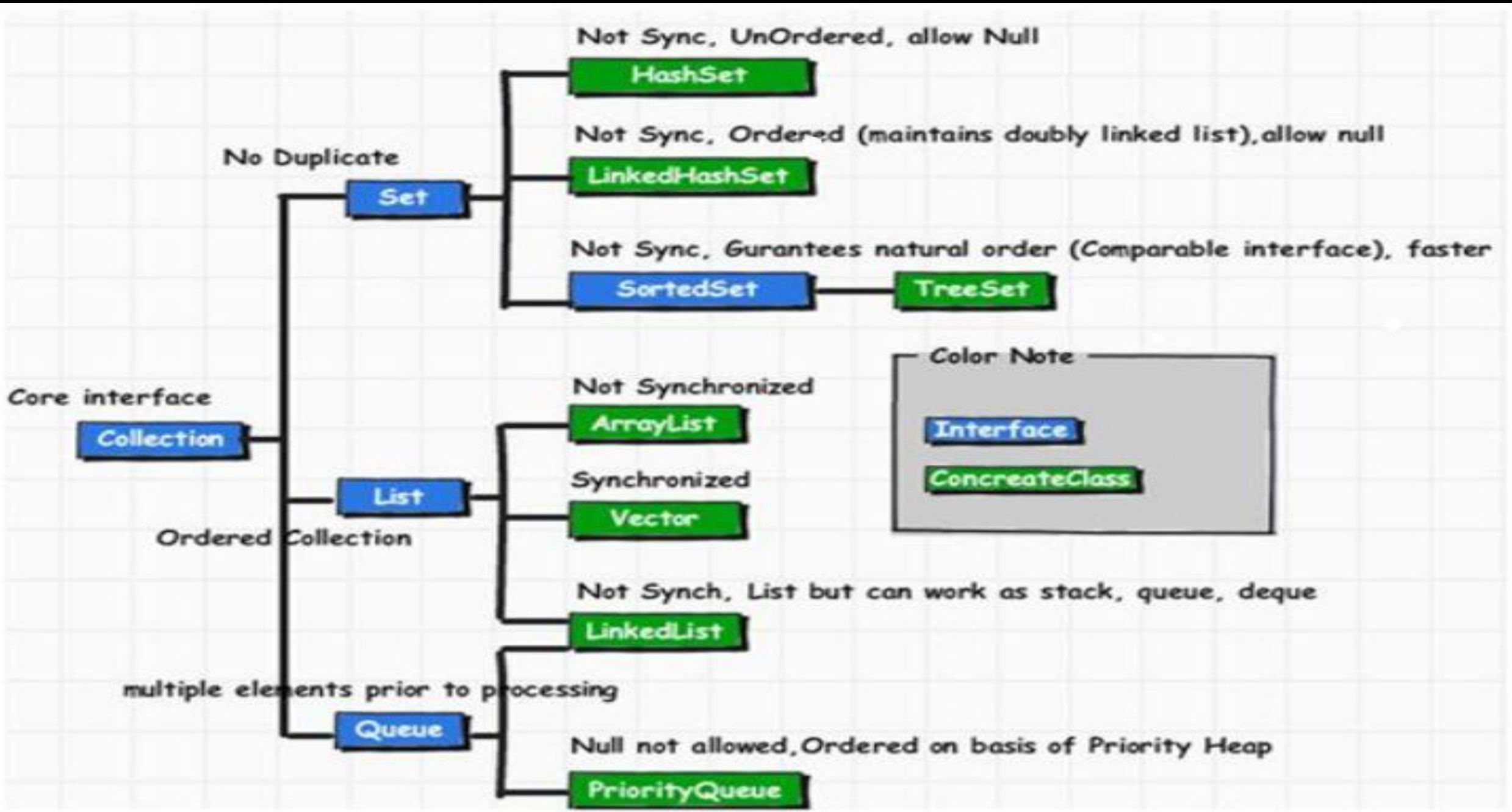
Hashtable & HashMap & LinkedHashMap

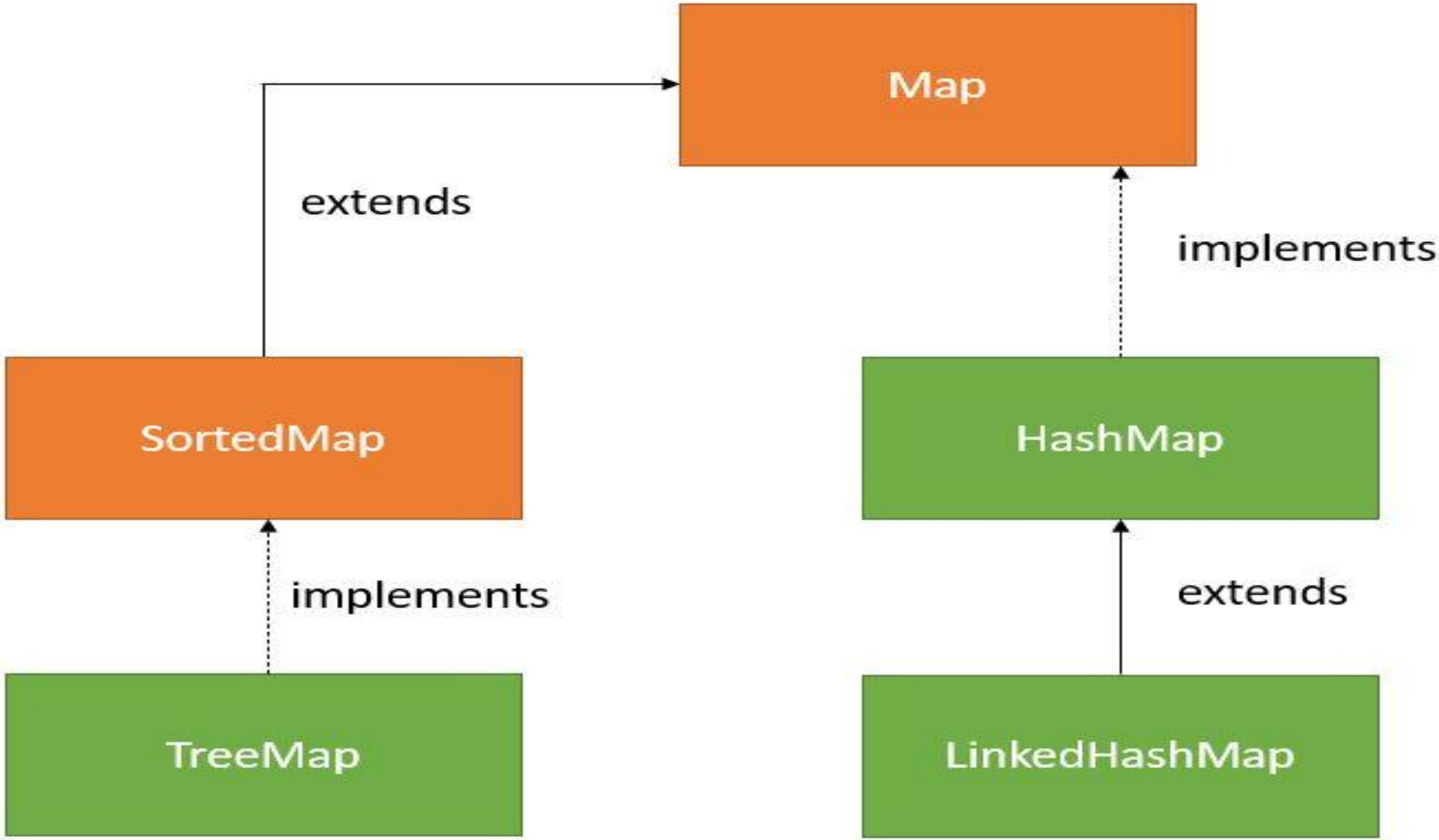
- All methods of Map +

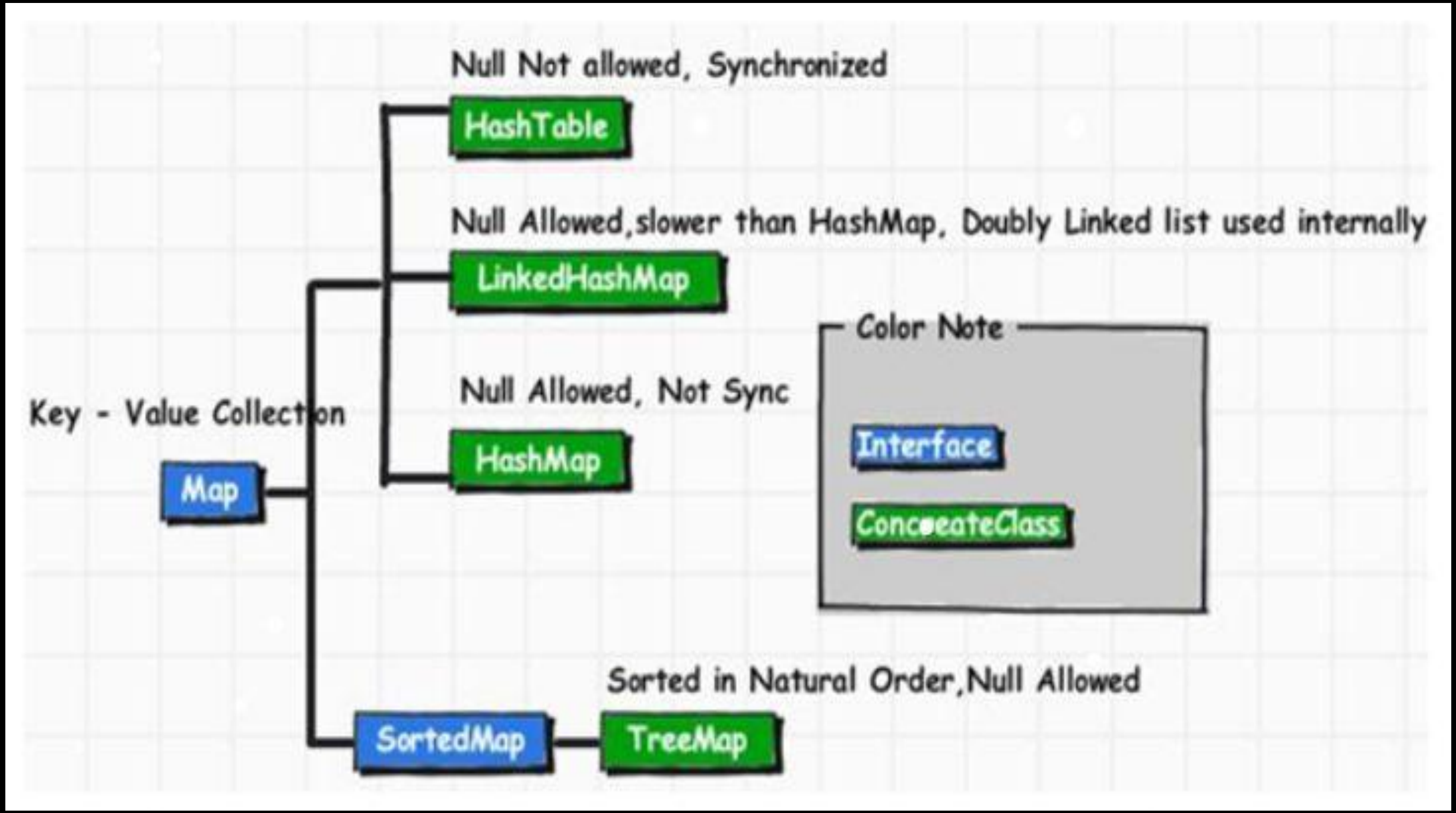
interface +
Subset Related Methods

- All methods of Map interface +
- Subset Related Methods
  - headMap(KeyObject)
  - tailMap(KeyObject)
  - subMap(KeyObject, KeyObject)
- first(), last()
- lowerKey(KeyObject), higherKey(KeyObject)
- ceilingKey(KeyObject), floorKey(KeyObject)
- pollFirstEntry(), pollLastEntry()


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


Not Sync, UnOrdered, allow Null
HashSet
Not Sync, Ordered (maintains doubly linked list),allow null
LinkedHashSet
No Duplicate
Set
Not Sync, Gurantees natural order (Comparable interface), faster
SortedSet
TreeSet
Color Note
Interface
ConcreateClass
Core interface
Collection
Not Synchronized
ArrayList
Synchronized
List
Vector
Ordered Collection
Not Synch, List but can work as stack, queue, deque
LinkedList
multiple elements prior to processing
Queue
Null not allowed,Ordered on basis of Priority Heap
PriorityQueue


Map
extends
implements
SortedMap
HashMap
implements
extends
TreeMap
LinkedHashMap


Null Not allowed, Synchronized
HashTable
Null Allowed,slower than HashMap, Doubly Linked list used internally
LinkedHashMap
Color Note
Interface
ConcreateClass
Null Allowed, Not Sync
Key - Value Collection
HashMap
Map
Sorted in Natural Order,Null Allowed
SortedMap
TreeMap

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        list.add("A");
        list.add("B");
        list.add(1, "C");
        System.out.println(list);
    }
}
```

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Set<String> set = new HashSet<>();
        set.add("Hello");
        set.add("World");
        set.add("Hello");
        System.out.println(set.size());
    }
}
```

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Queue<Integer> queue = new
LinkedList<>();
        queue.offer(10);
        queue.offer(20);
        queue.offer(30);
        System.out.println(queue.poll());
    }
}
```

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Set<String> set = new HashSet<>();
        set.add("Hello");
        set.add("World");
        set.add("Hello");
        System.out.println(set.size());
    }
}
```

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Queue<Integer> queue = new LinkedList<>();
        queue.offer(10);
        queue.offer(20);
        queue.offer(30);
        System.out.println(queue.poll());
    }
}
```

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);

        for (Integer num : list) {
            if (num == 2) {
                list.remove(num);
            }
        }
        System.out.println(list);
    }
}
```