Object Oriented Programming with Java (OOPJ)

Session 5: Arrays

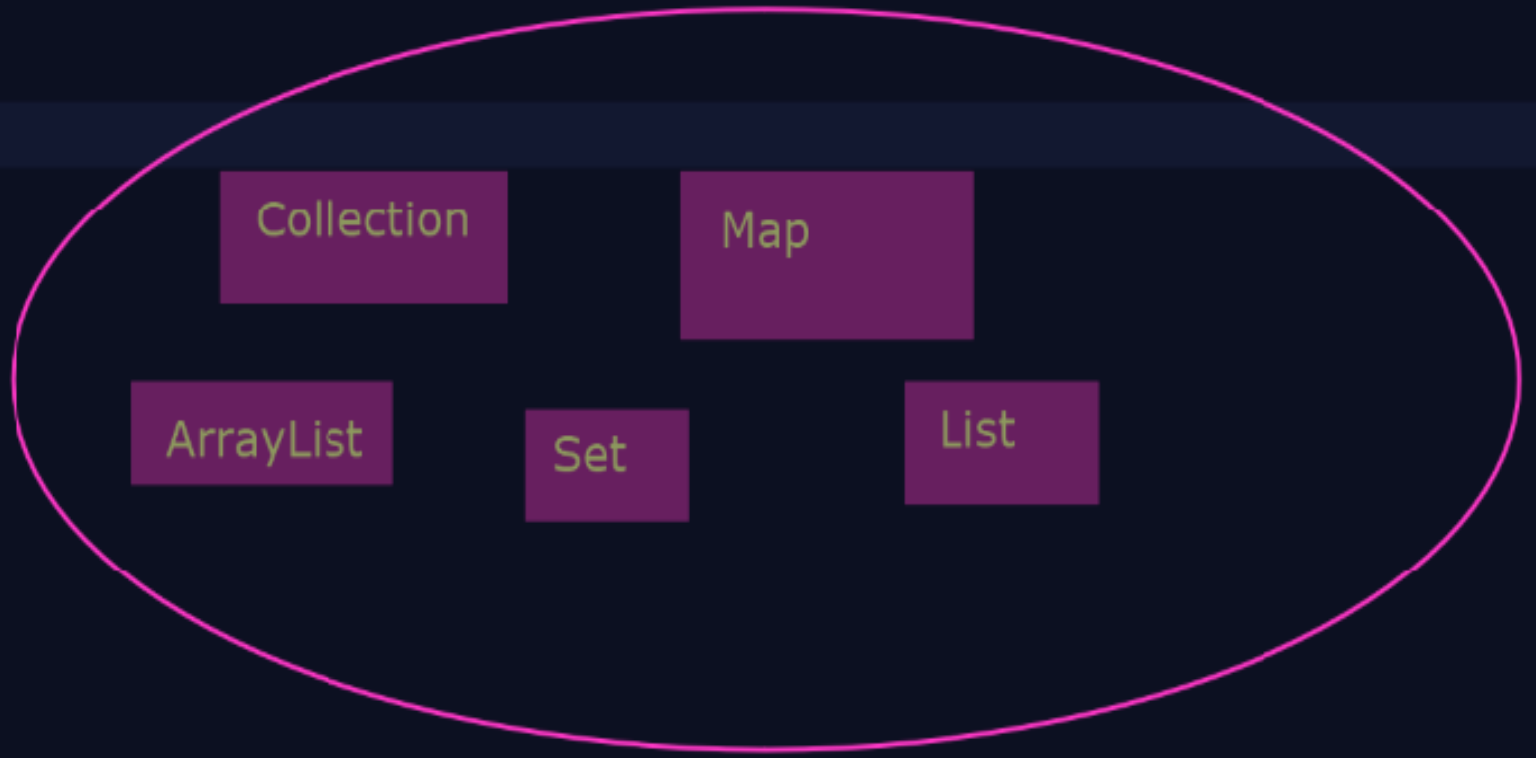
Kiran Waghmare

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

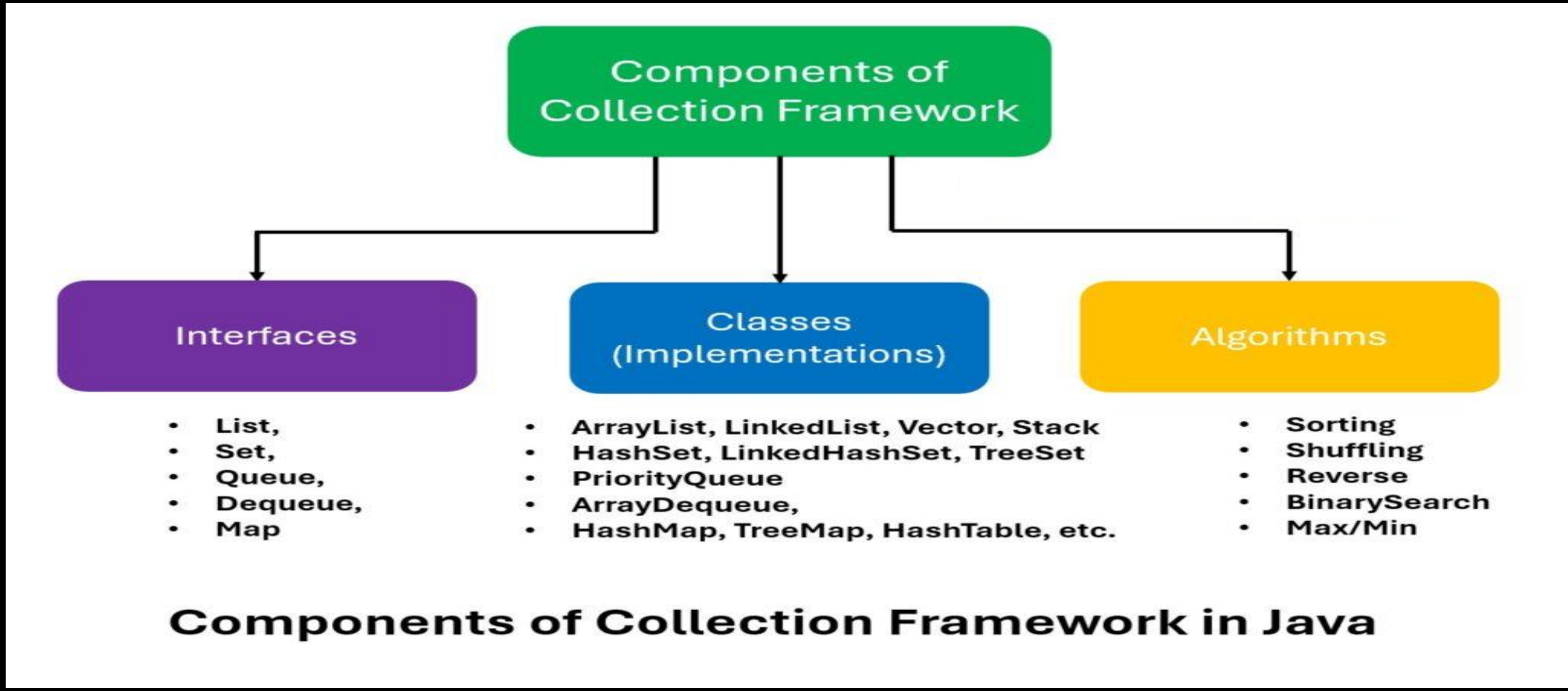
Collection Framework:

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

Collections:



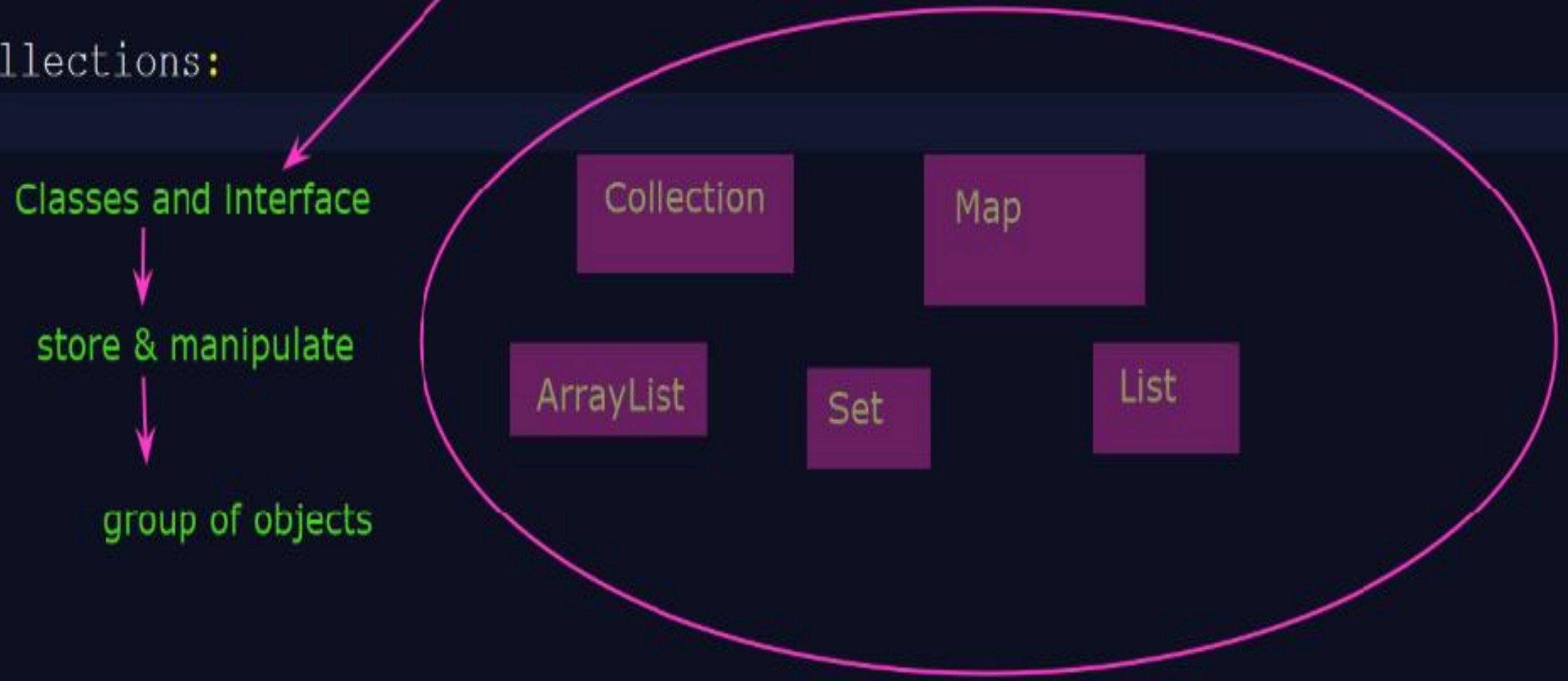
Collections Framework



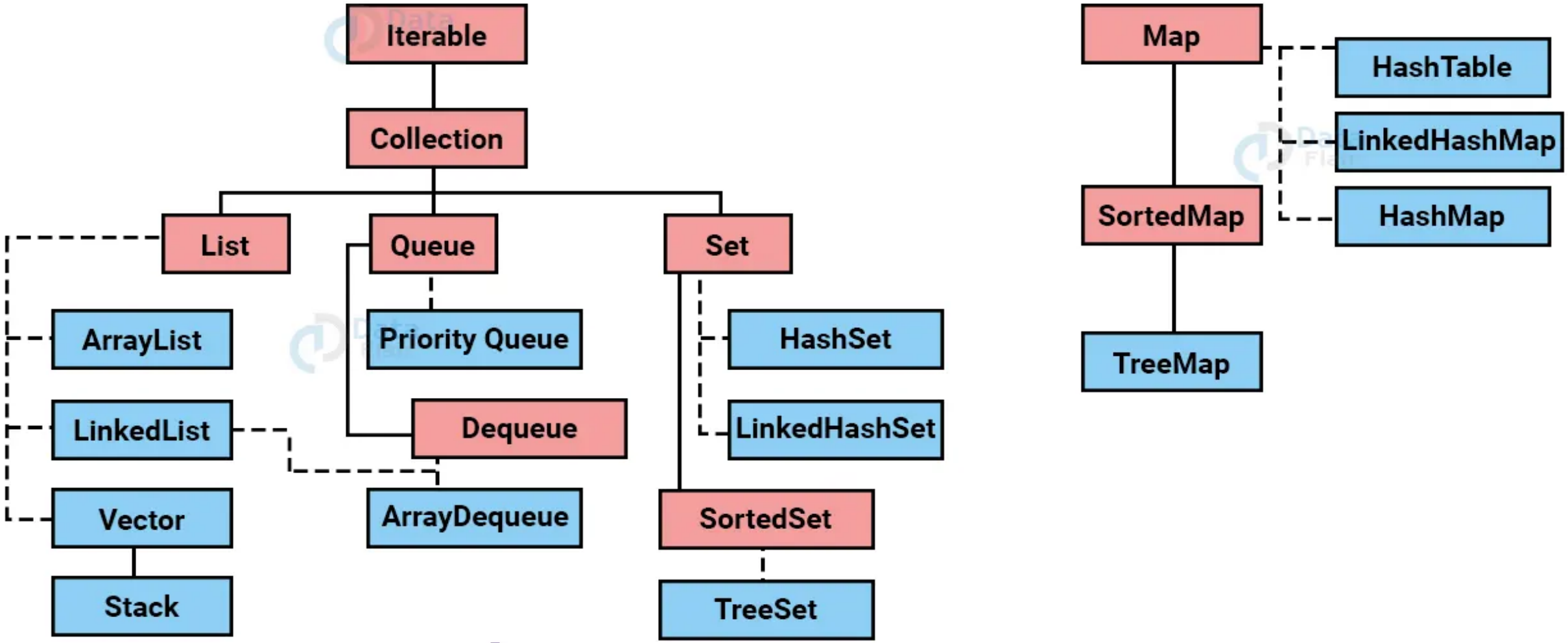
Collection Framework:

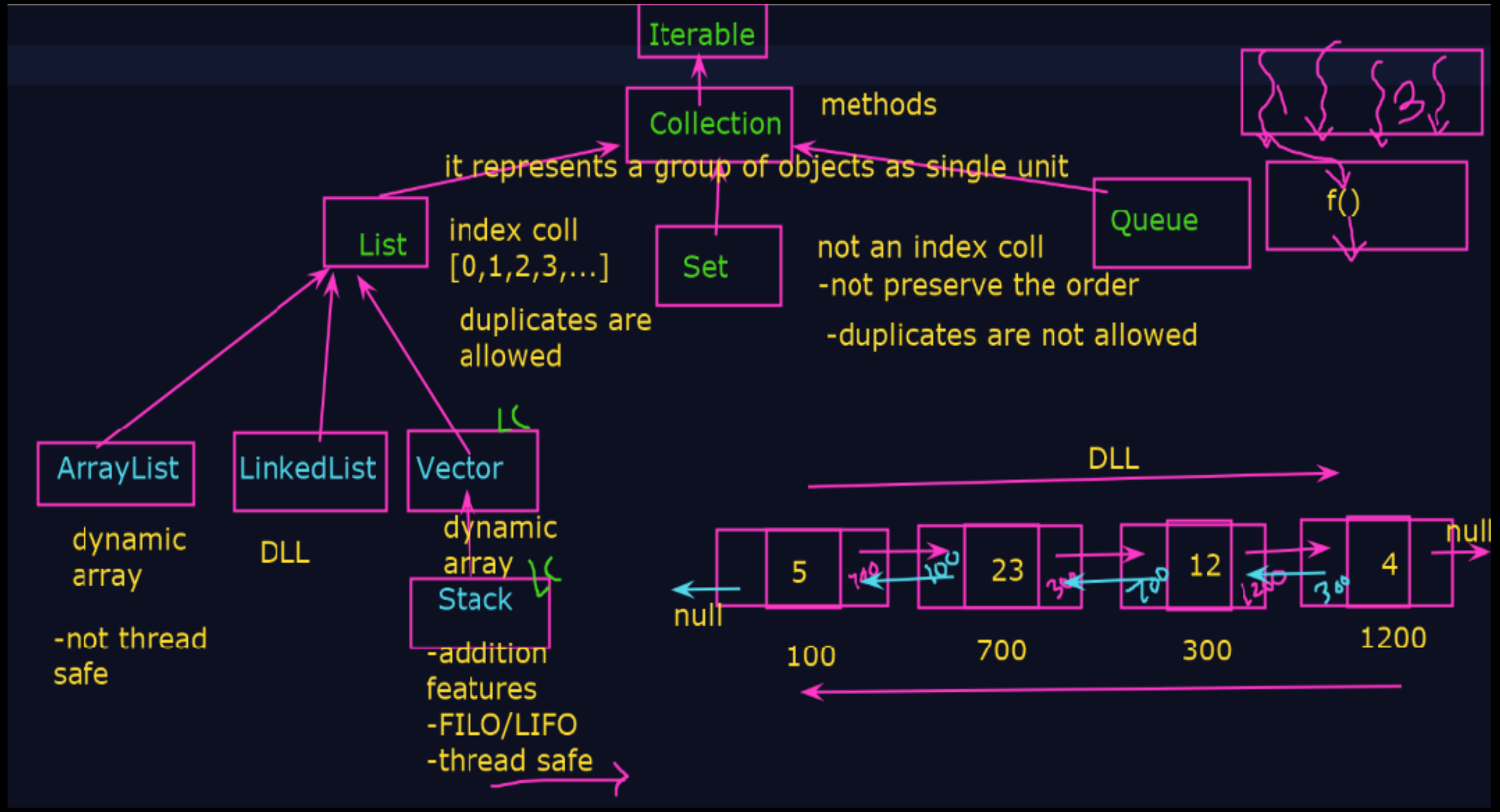
-Collection Framework is Java API(Application Programming Interface) which provides architecture to store and manipulate group of object.

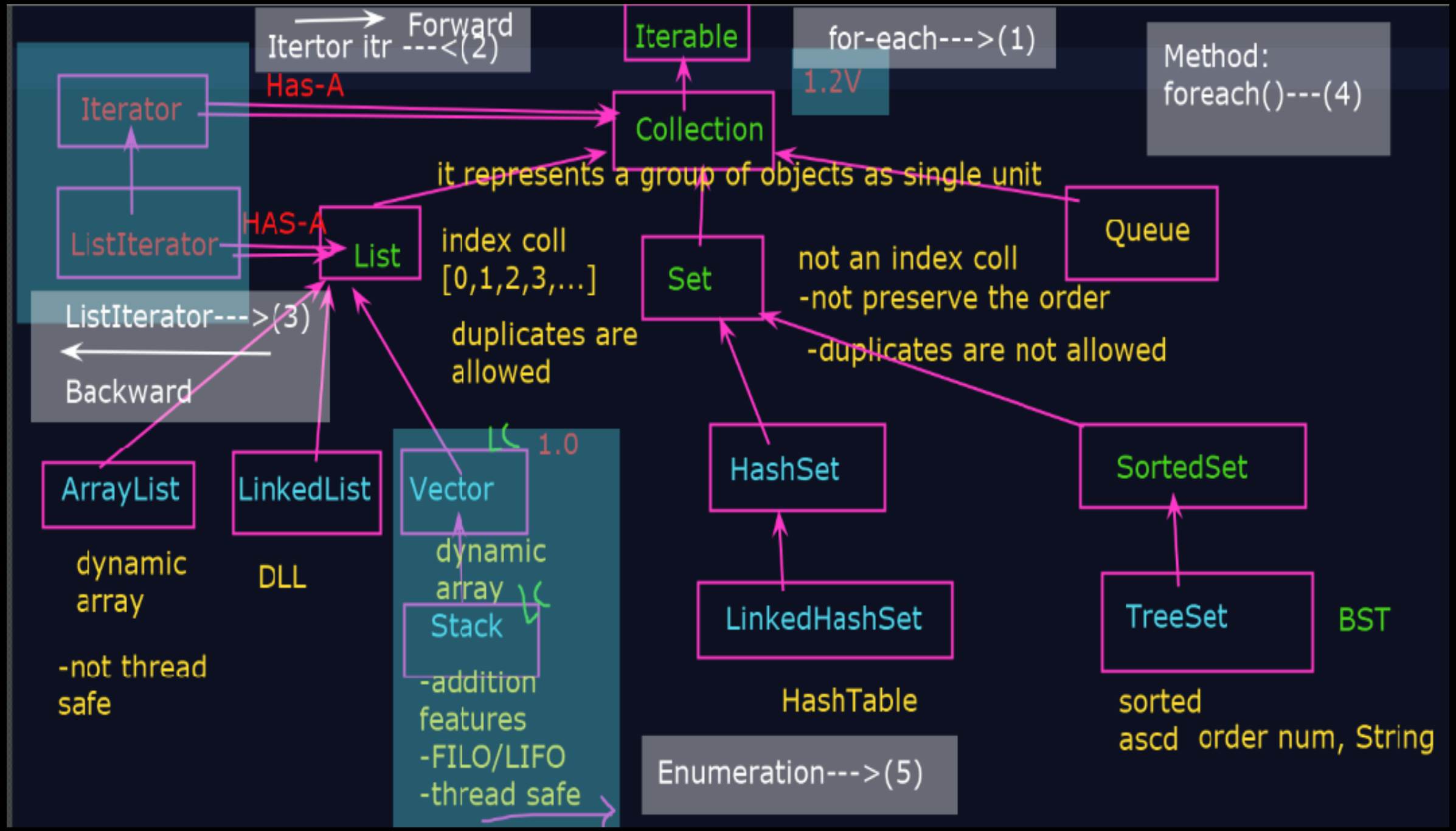
Collections:

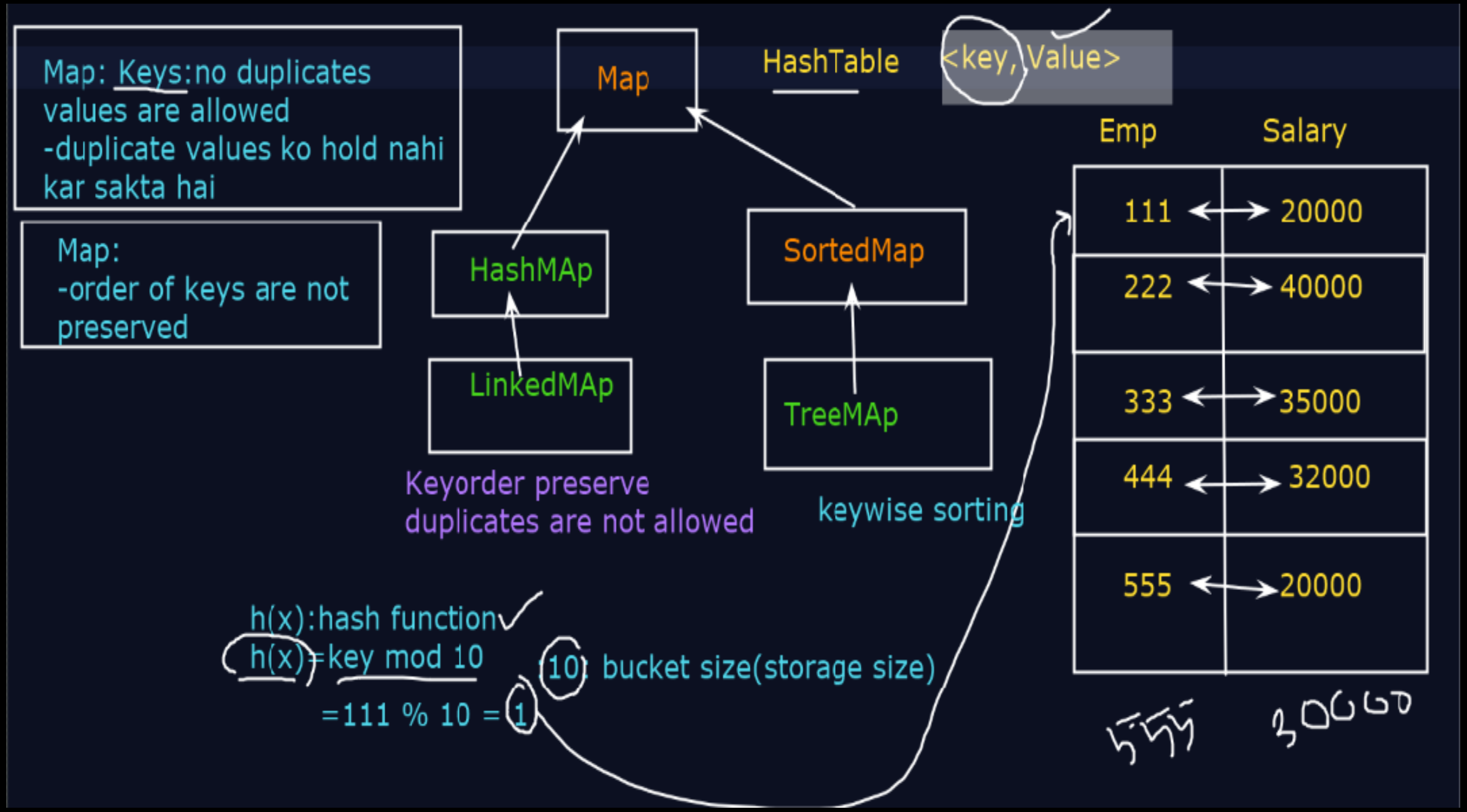


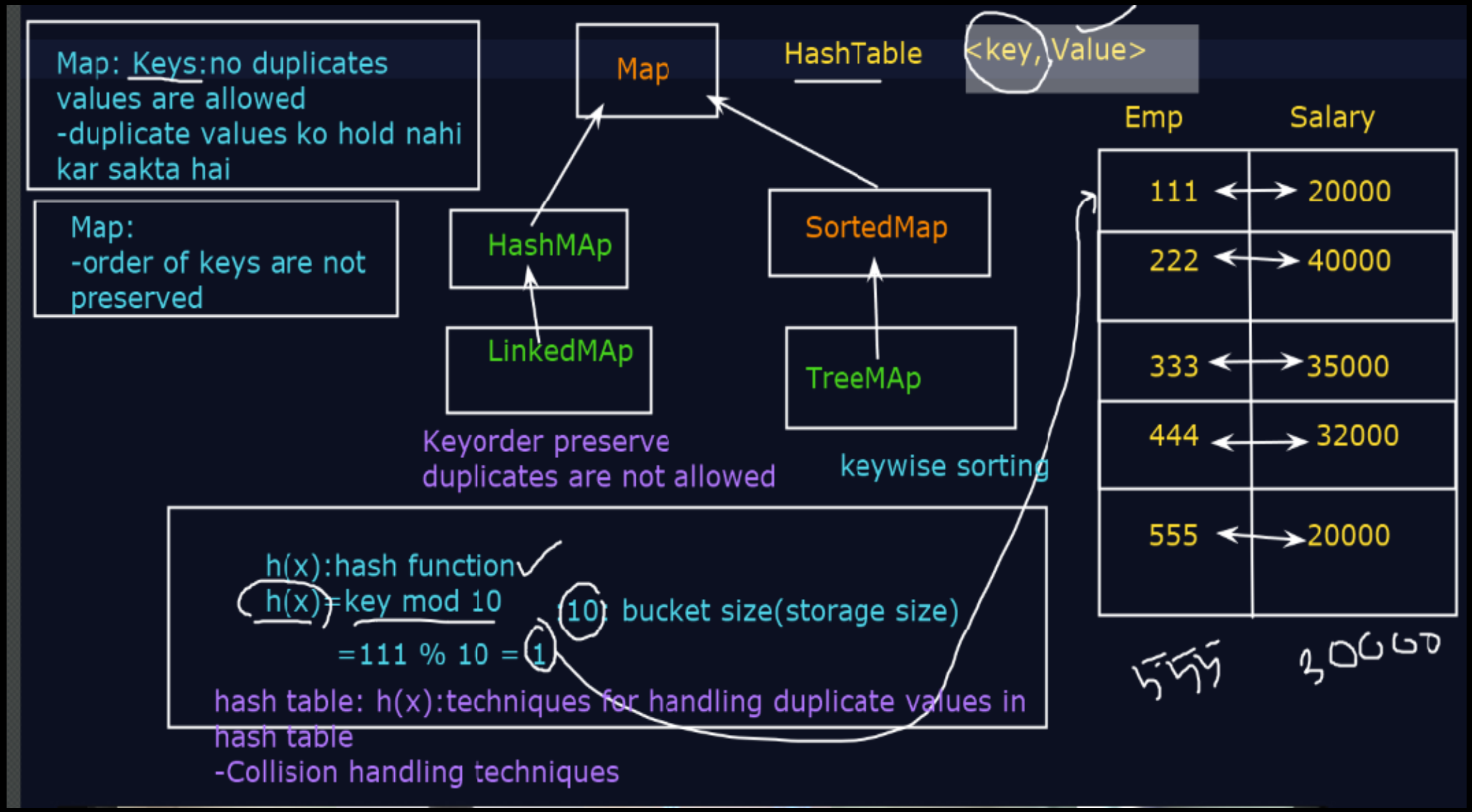
Hierarchy of Collection Framework in Java




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
-not thread
safe
DLL
dynamic
array
Stack
-addition
features
-FILO/LIFO
-thread safe
DLL
null
5
23
12
4
null
100
700
300
1200


Forward
Itertor itr ---<(2)
Has-A
Iterator
ListIterator
HAS-A
ListIterator--->(3)
Backward
Iterable
for-each--->(1)
1.2V
Method:
foreach()---(4)
Collection
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
ArrayList
dynamic
array
-not thread
safe
LinkedList
DLL
Vector
1.0
dynamic
array
Stack
-addition
features
-FILO/LIFO
-thread safe
HashSet
LinkedHashSet
HashTable
SortedSet
TreeSet
BST
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

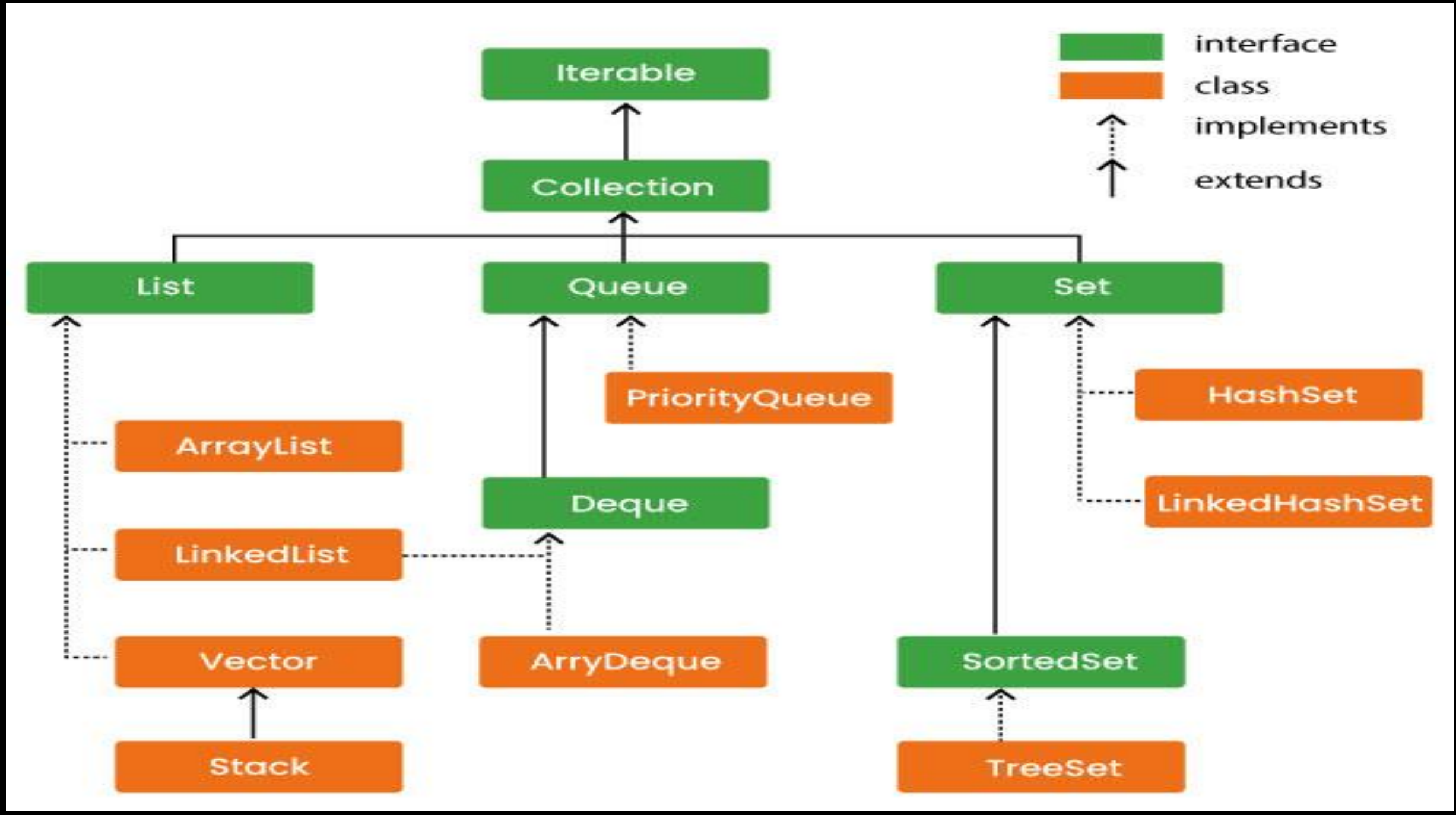
- put(K,V)
- get(KeyObject)
- remove(KeyObject)
- containsKey()
- containsValue()
- size()
- clear()

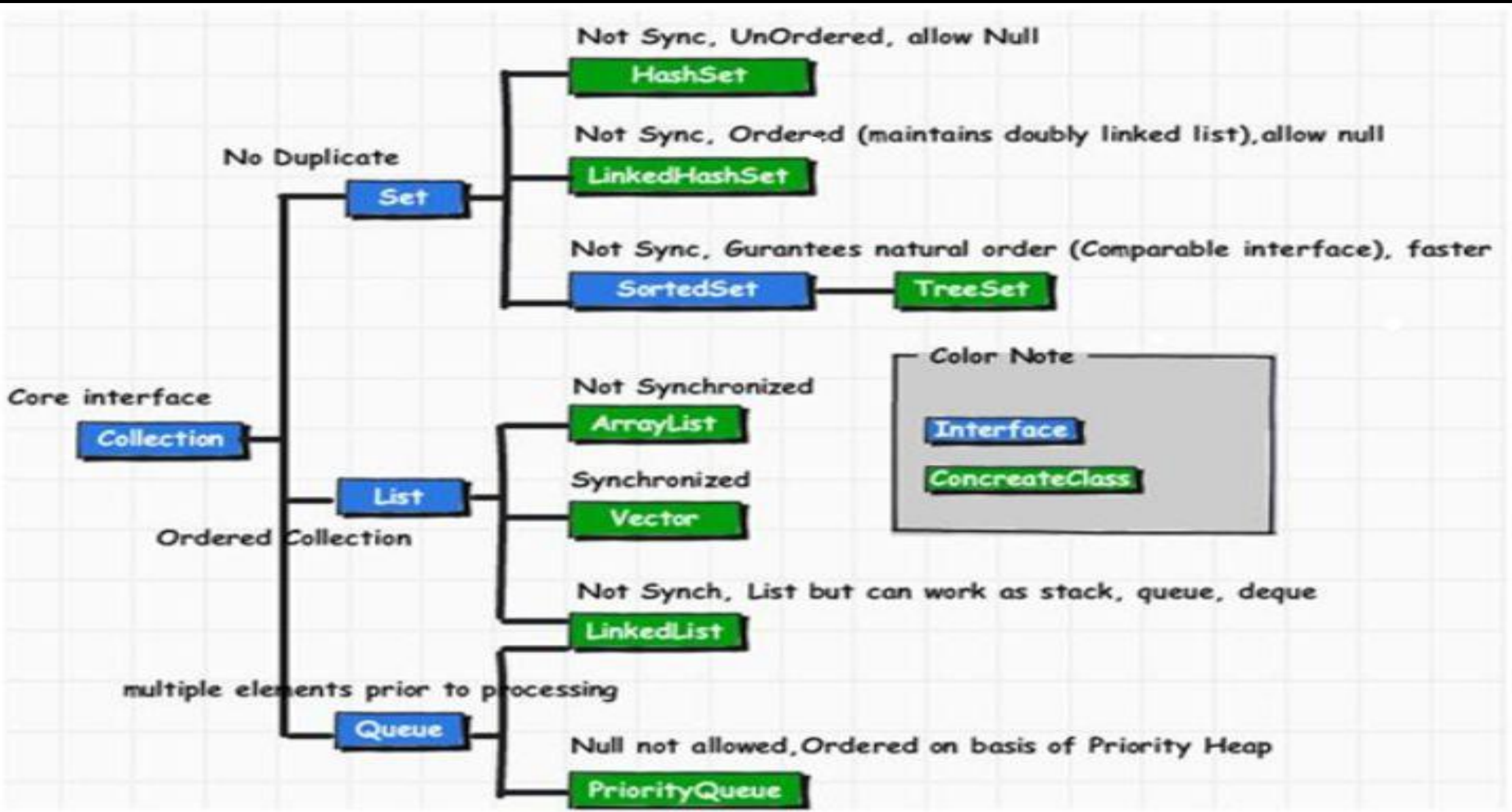
Hashtable & HashMap & LinkedHashMap

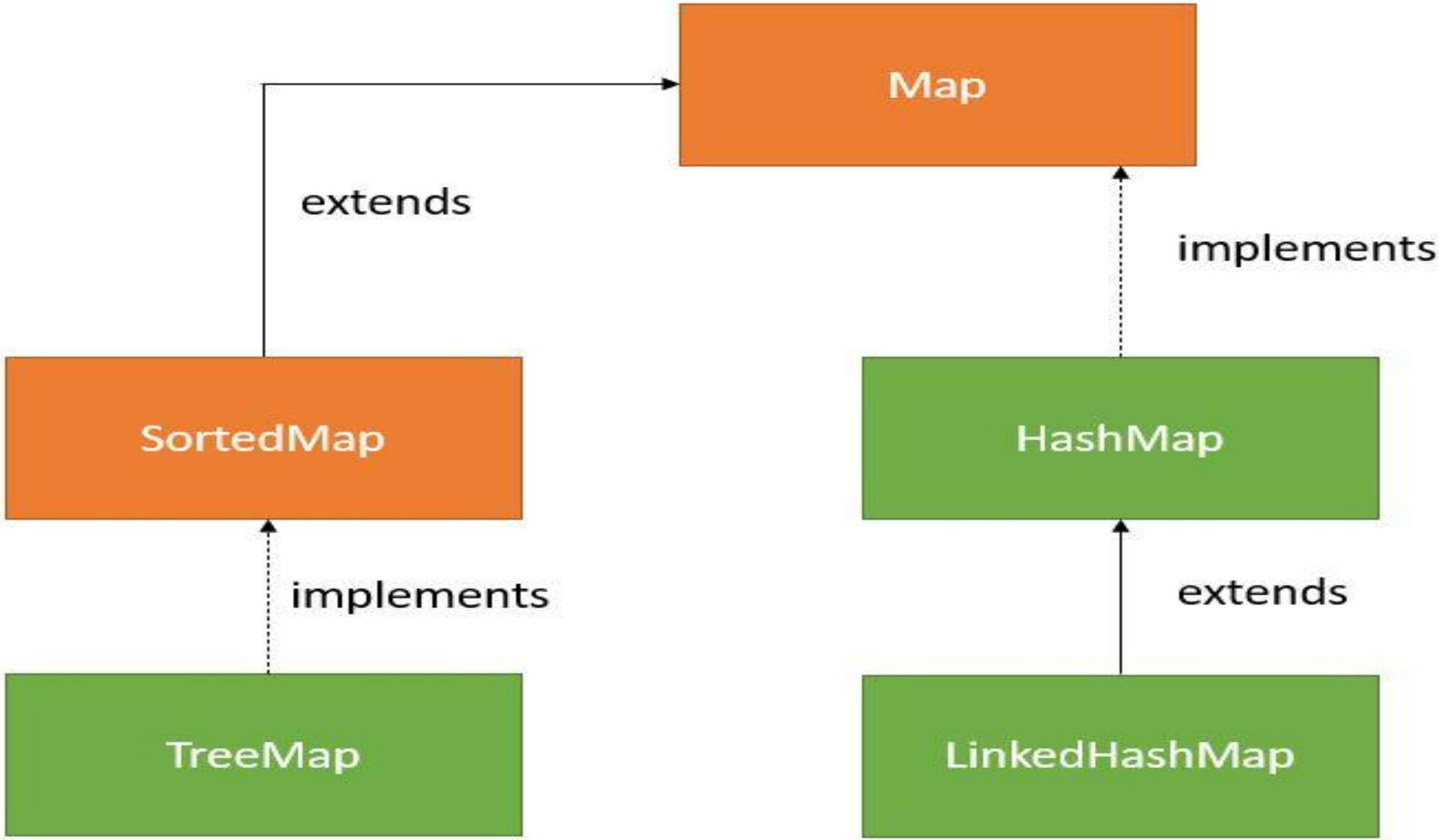
- All methods of Map +

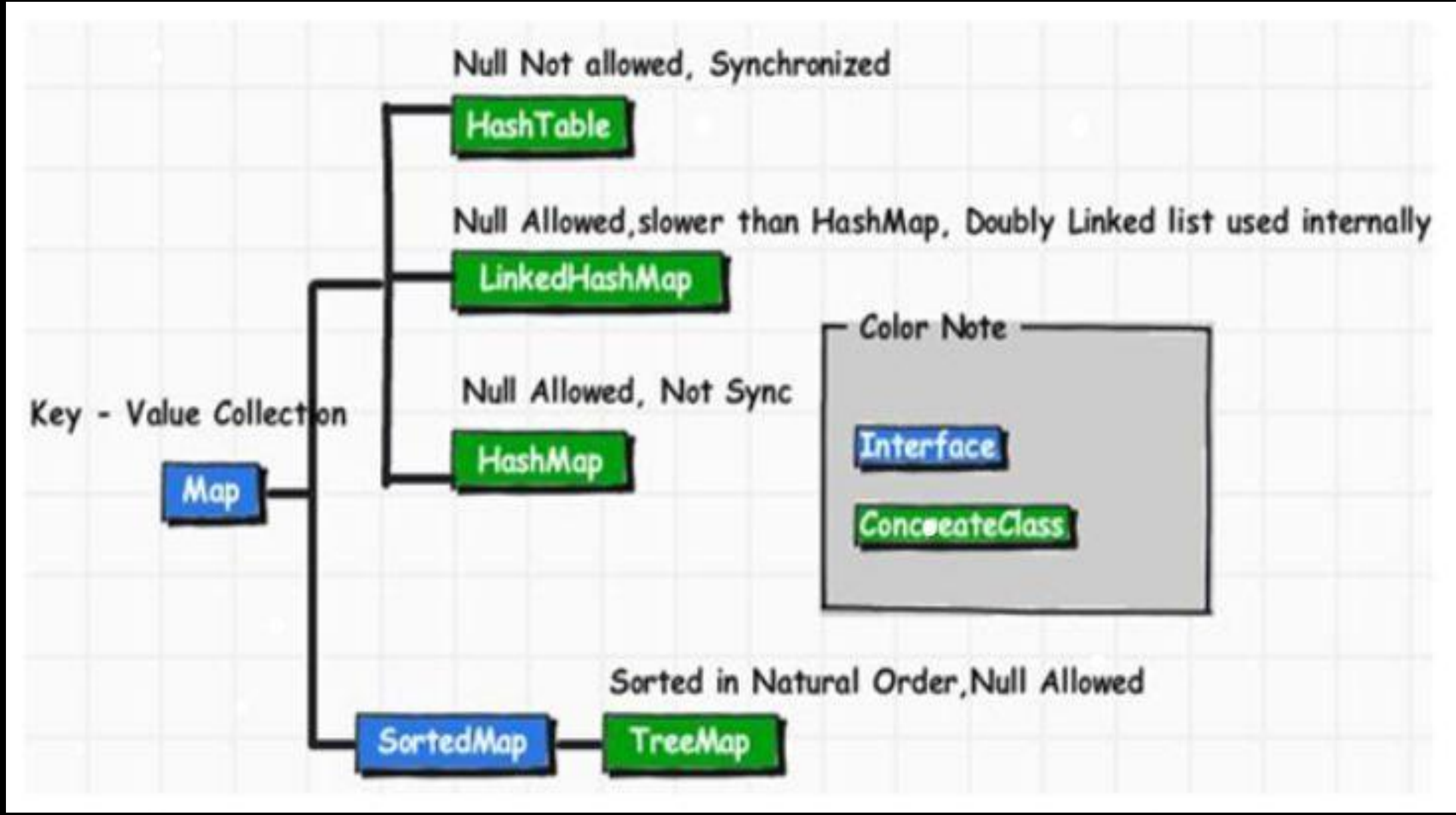
interface +
Subset Related Methods

- 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);
    }
}
```

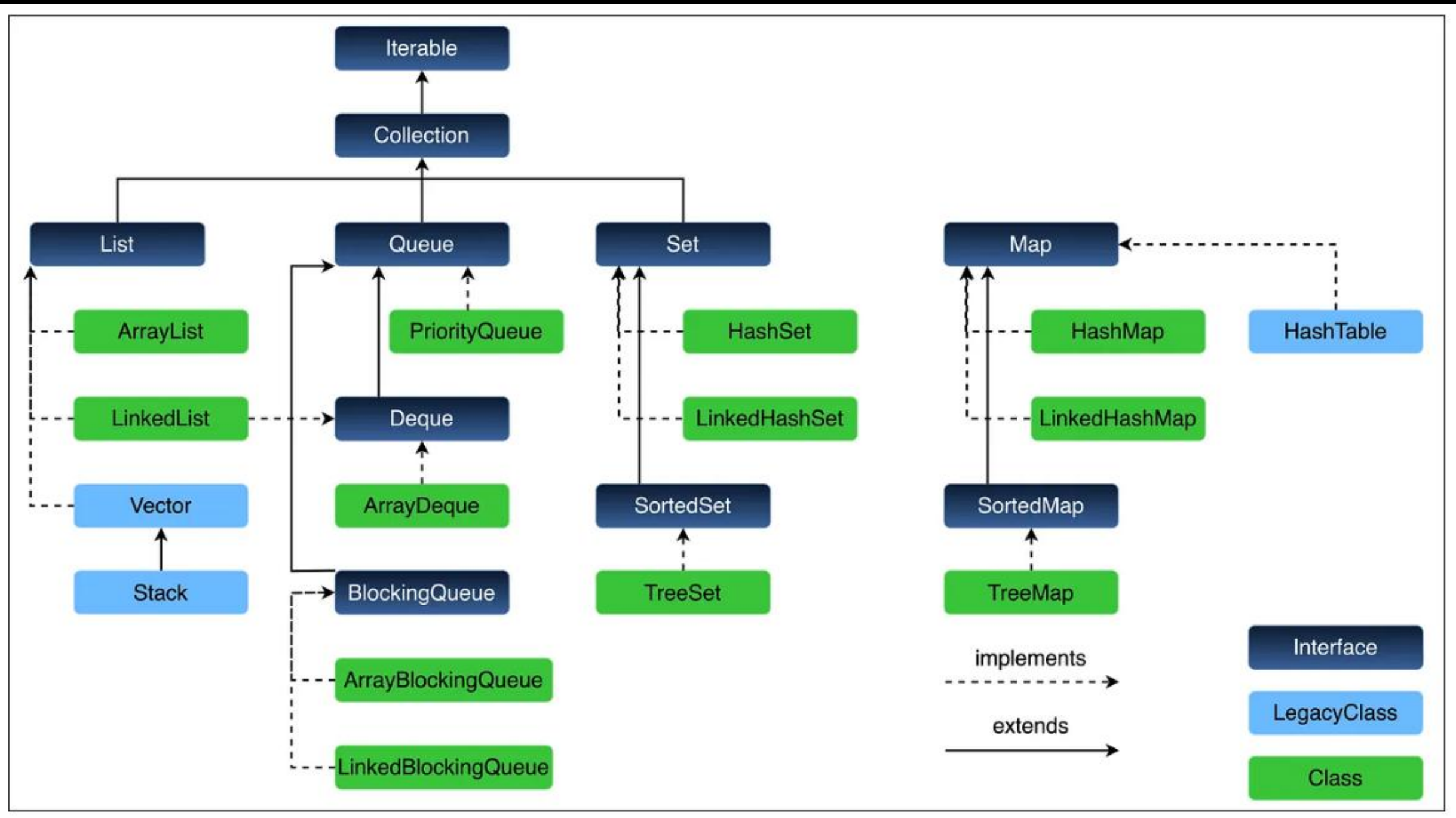
```
import java.util.*;

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());
    }
}
```

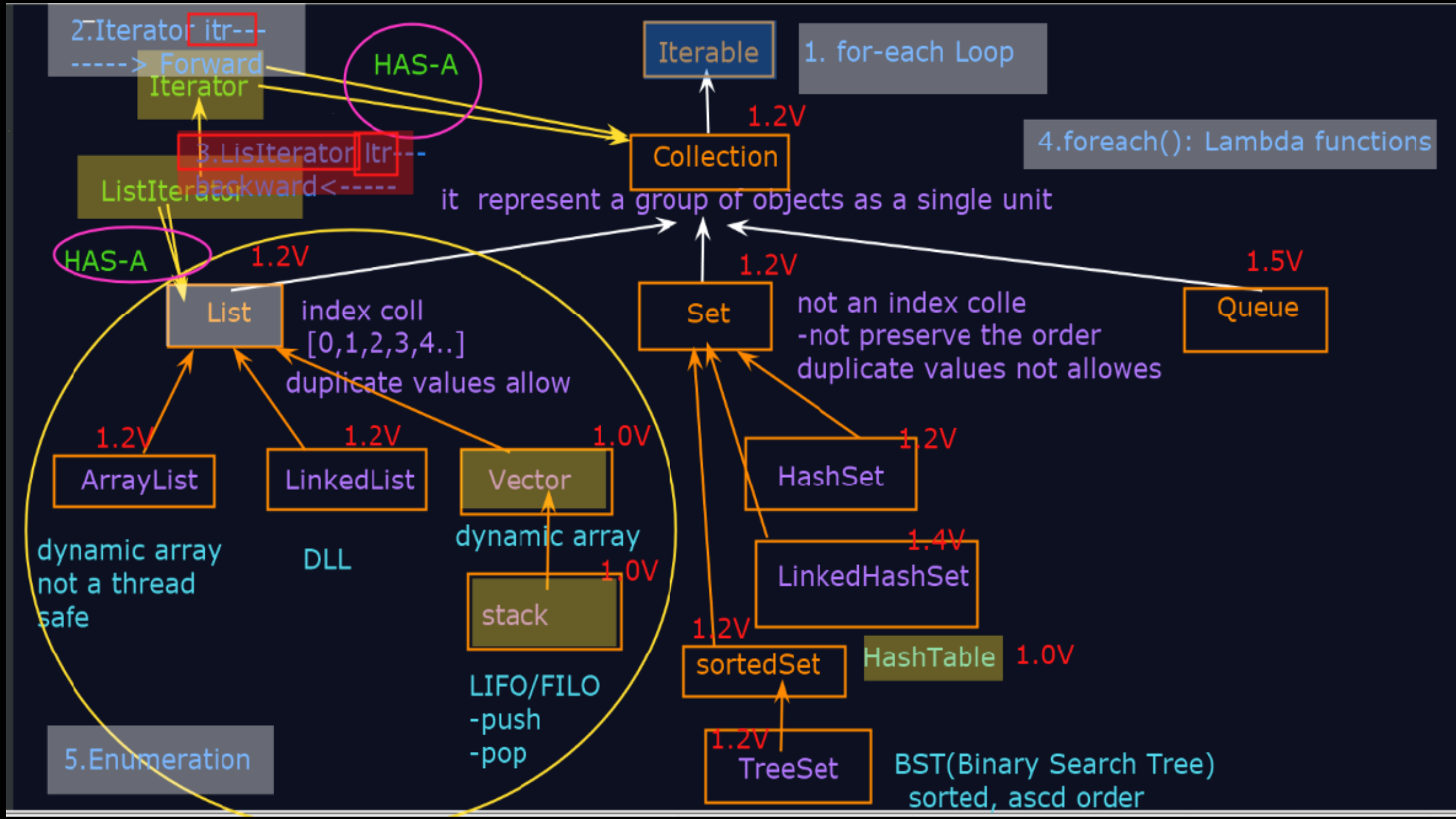
```
import java.util.*;

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);
    }
}
```


Iterable
Collection
List
Queue
Set
Map
ArrayList
PriorityQueue
HashSet
HashMap
HashTable
LinkedList
Deque
LinkedHashSet
LinkedHashMap
Vector
ArrayDeque
SortedSet
SortedMap
Stack
BlockingQueue
TreeSet
TreeMap
ArrayBlockingQueue
LinkedBlockingQueue
implements
extends
Interface
LegacyClass
Class


2.Iterator itr---
-----> Forward
Iterator
HAS-A
Iterable
1. for-each Loop
1.2V
Collection
4.foreach(): Lambda functions
3.LisIterator ltr---
backward<-----
ListIterator
it represent a group of objects as a single unit
HAS-A
1.2V
1.2V
1.5V
List
index coll
[0,1,2,3,4..]
duplicate values allow
Set
not an index colle
-not preserve the order
duplicate values not allowes
Queue
1.2V
1.2V
1.0V
ArrayList
LinkedList
Vector
dynamic array
not a thread
safe
DLL
dynamic array
1.0V
stack
LIFO/FILO
-push
-pop
1.2V
HashSet
1.4V
LinkedHashSet
1.2V
sortedSet
HashTable
1.0V
1.2V
TreeSet
BST(Binary Search Tree)
sorted, ascd order
5.Enumeration

```java
        System.out.println(a);

        Collection a1 = new ArrayLis
        a1.add(1234);
        a1.add("Raviyadav");
        a1.add(123.45456);
        a1.add("Raviyadav");

        System.out.println(a1);
        System.out.println("--------
        a.addAll(a1);
        System.out.println(a);
        System.out.println(a1);
        System.out.println("--------
        a.removeAll(a1);
        System.out.println(a);
        System.out.println(a1);


}
```

```
C:\WINDOWS\syster
C:\Test>javac CollectionDemo2.java
Note: CollectionDemo2.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Test>java CollectionDemo2
[123, Ravi, 123.45, 123.45]
[1234, Raviyadav, 123.45456, Raviyadav]
------------
[123, Ravi, 123.45, 123.45, 1234, Raviyadav, 123.45456, Raviyada
v]
[1234, Raviyadav, 123.45456, Raviyadav]
------------
[123, Ravi, 123.45, 123.45]
[1234, Raviyadav, 123.45456, Raviyadav]

C:\Test>
```

```java
import java.util.*;

class CollectionDemo5{

    public static void main(String[] args){

        Collection a = new ArrayList();
        a.add(123);
        a.add("Ravi");
        a.add(123.45);
        a.add(123.45);

        System.out.println(a);


        //Travers the collections :
        Iterator it = a.iterator();
        while(it.hasNext())
        {
            System.out.println(it.next());
        }

        System.out.println(a);
```