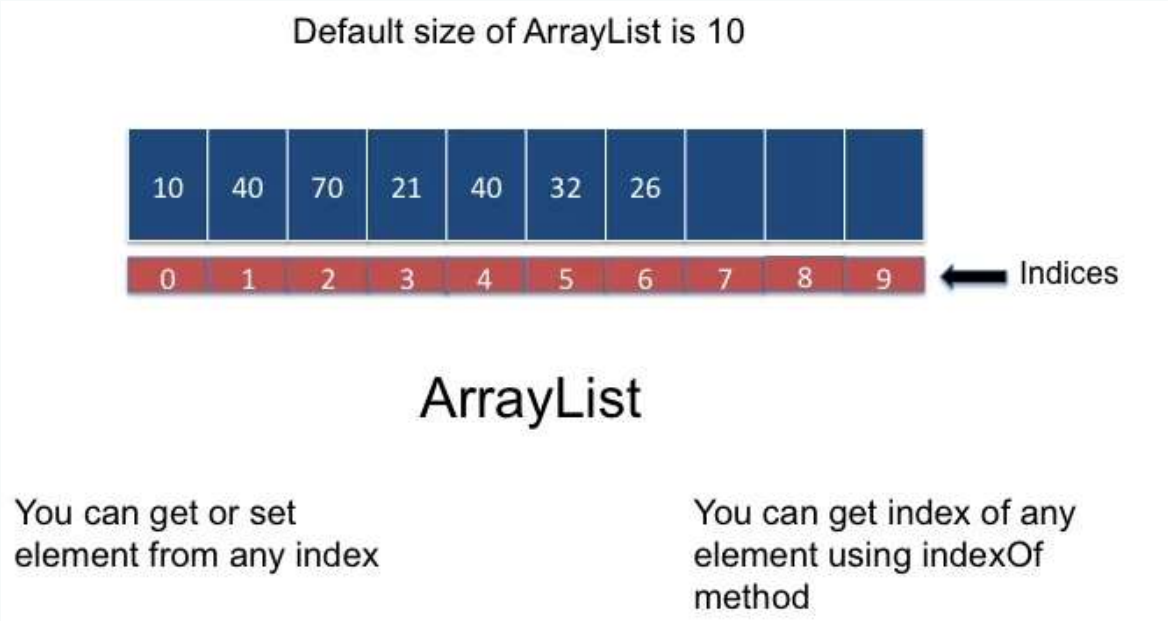
cloudthat
move up.

Java Collections

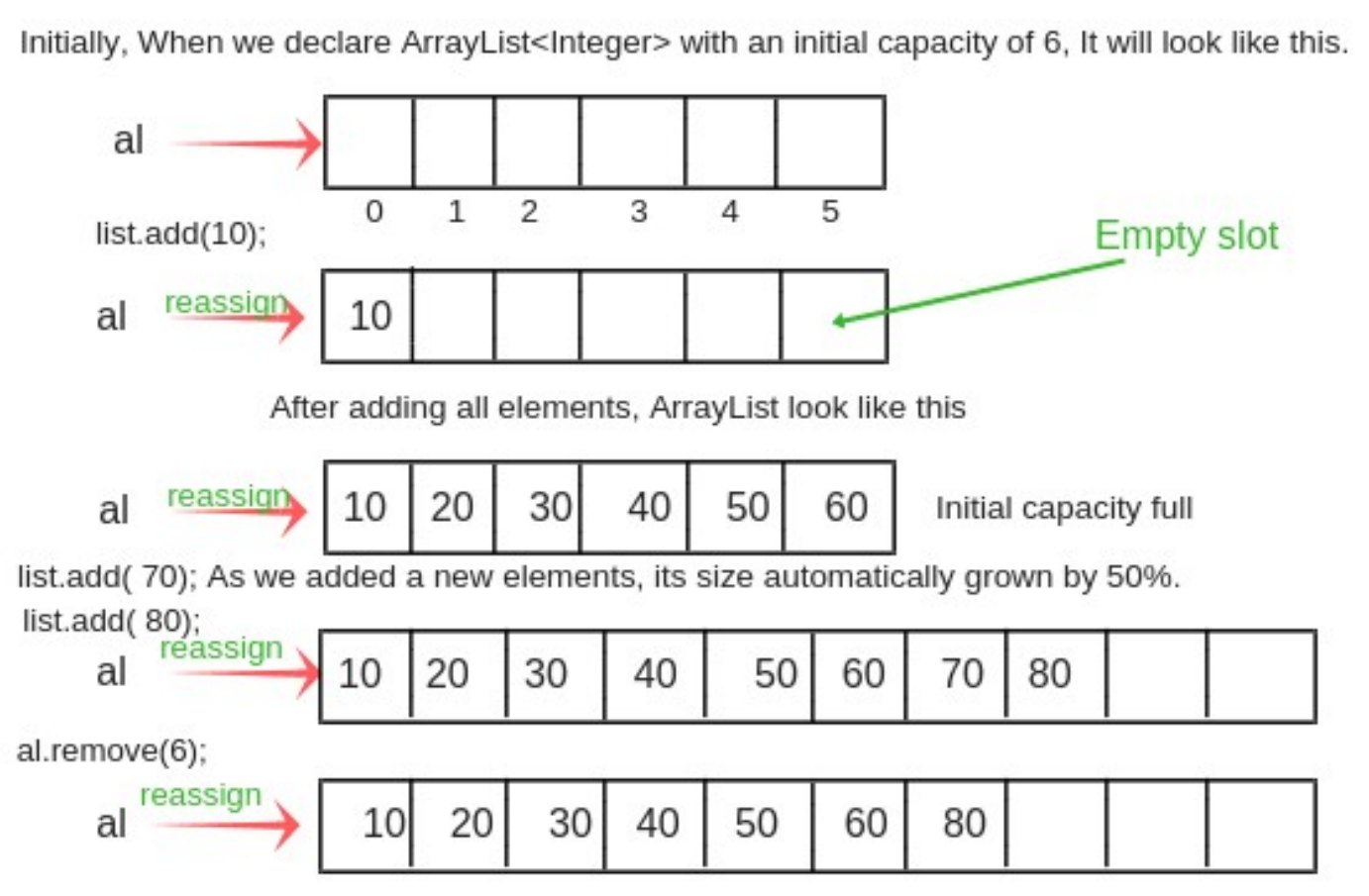
Collections in Java

- A framework that provides an architecture to store and manipulate the group of objects.

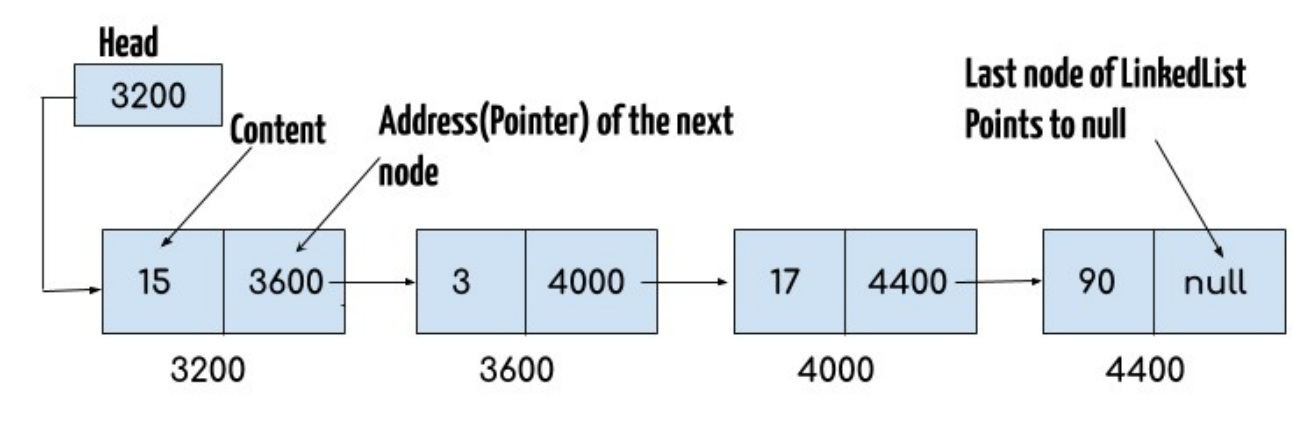
Array List


Default size of ArrayList is 10
10
40
70
21
40
32
26
0
1
2
3
4
5
6
7
8
9
Indices
ArrayList
You can get or set element from any index
You can get index of any element using indexOf method

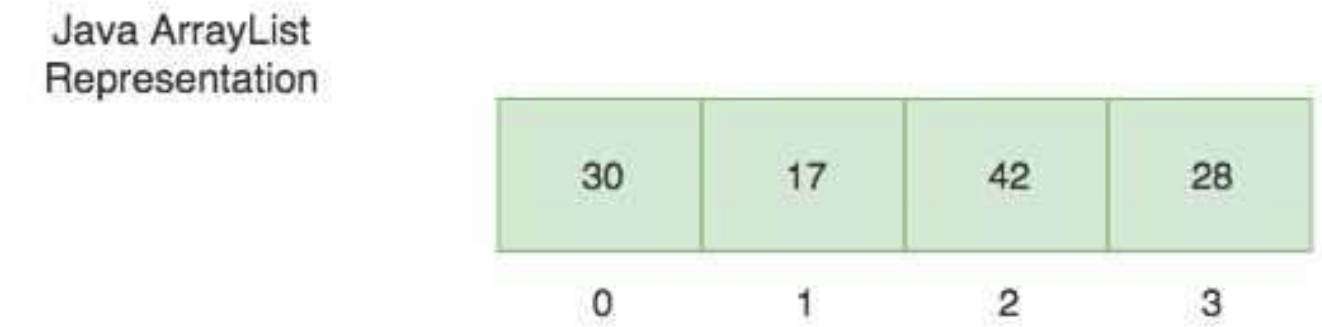
Array List

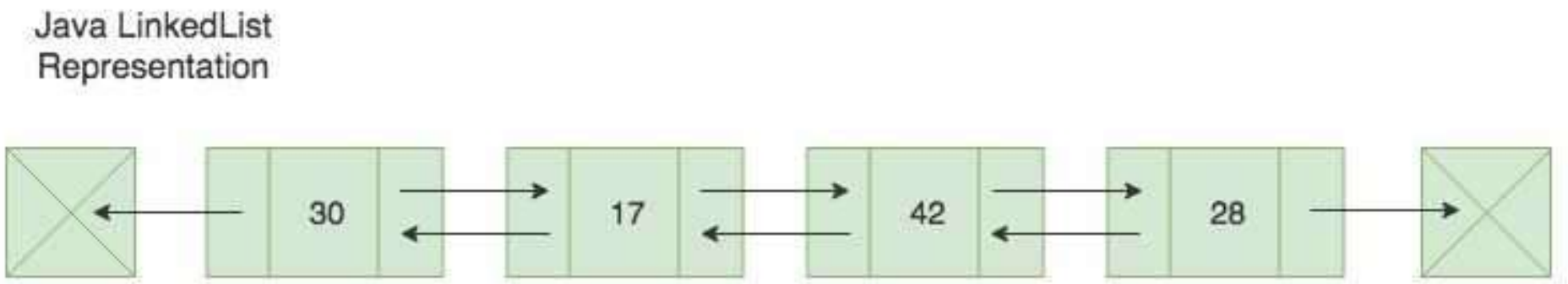

Initially, When we declare ArrayList<Integer> with an initial capacity of 6, It will look like this.
al
0 1 2 3 4 5
list.add(10);
Empty slot
al
reassign
10
After adding all elements, ArrayList look like this
al
reassign
10 20 30 40 50 60
Initial capacity full
list.add(70); As we added a new elements, its size automatically grown by 50%.
list.add(80);
al
reassign
10 20 30 40 50 60 70 80
al.remove(6);
al
reassign
10 20 30 40 50 60 80

Linked List

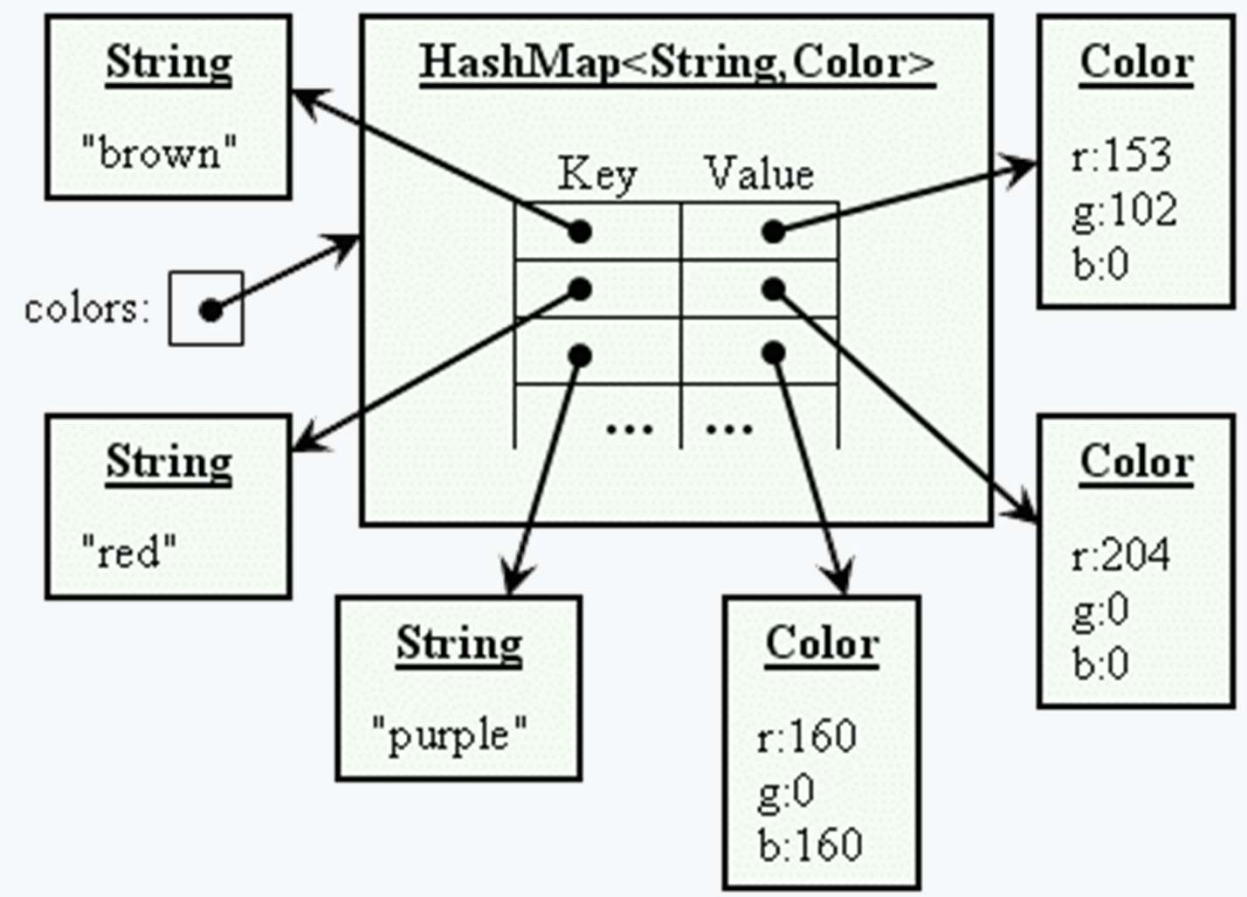

Head
3200
Content
Address(Pointer) of the next node
Last node of LinkedList Points to null
15
3600
3
4000
17
4400
90
null
3200
3600
4000
4400

ArrayList vs LinkedList

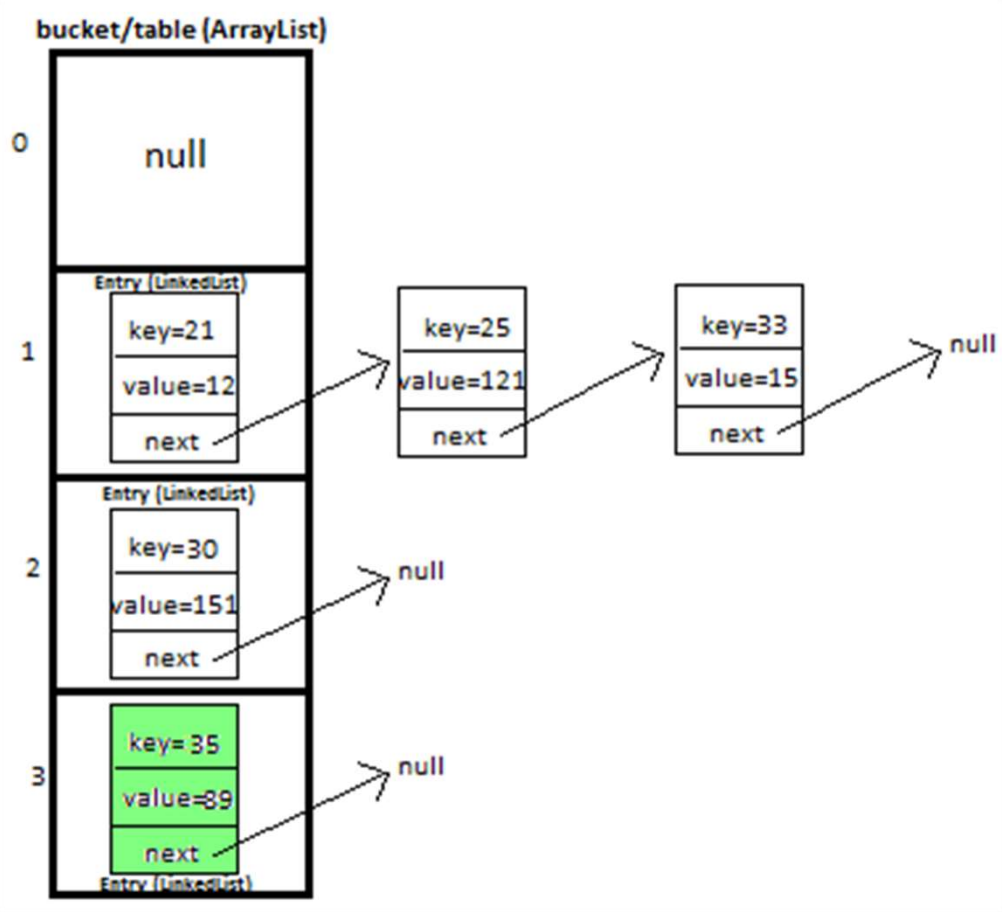




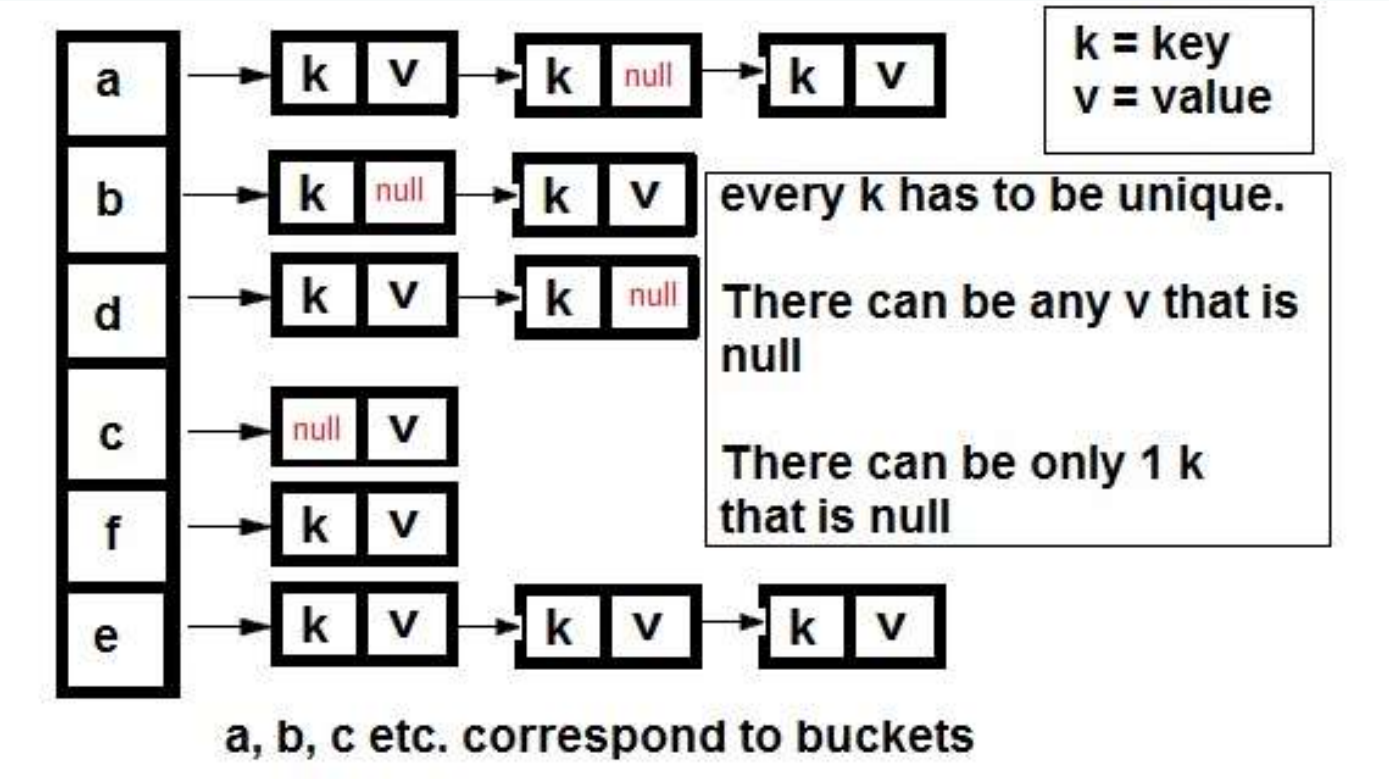
HashMap


String
"brown"
HashMap<String, Color>
Key
Value
Color
r:153
g:102
b:0
colors:
String
"red"
Color
r:204
g:0
b:0
String
"purple"
Color
r:160
g:0
b:160

HashMap

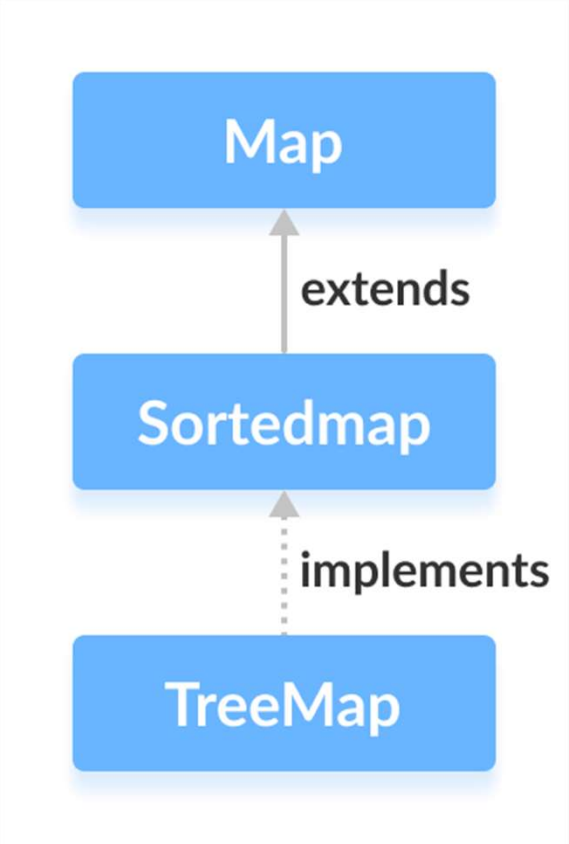

bucket/table (ArrayList)
0
null
1
Entry (LinkedList)
key=21
value=12
next
key=25
value=121
next
key=33
value=15
next
null
2
Entry (LinkedList)
key=30
value=151
next
null
3
key= 35
value=89
next
null
Entry (LinkedList)

HashMap


a
b
d
c
f
e
k v
k null
null v
k = key
v = value
every k has to be unique.
There can be any v that is null
There can be only 1 k that is null
a, b, c etc. correspond to buckets

SortedMaps

- The SortedMap interface of the Java collections framework provides sorting of keys stored in a map.
- Since SortedMap is an interface, we cannot create objects from it.
- In order to use the functionalities of the SortedMap interface, we need to use the class TreeMap that implements it.

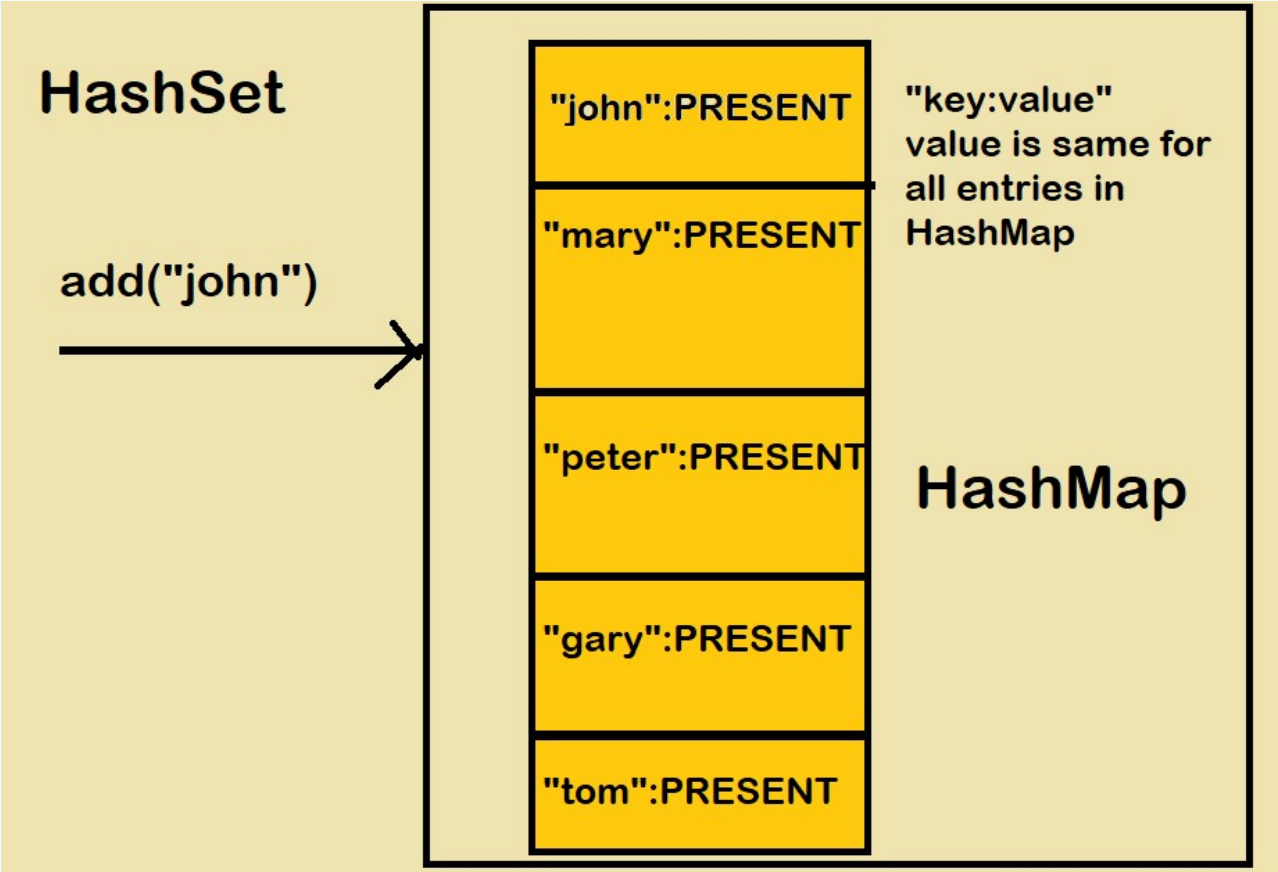


HashSet

- A collection of items where every item is unique

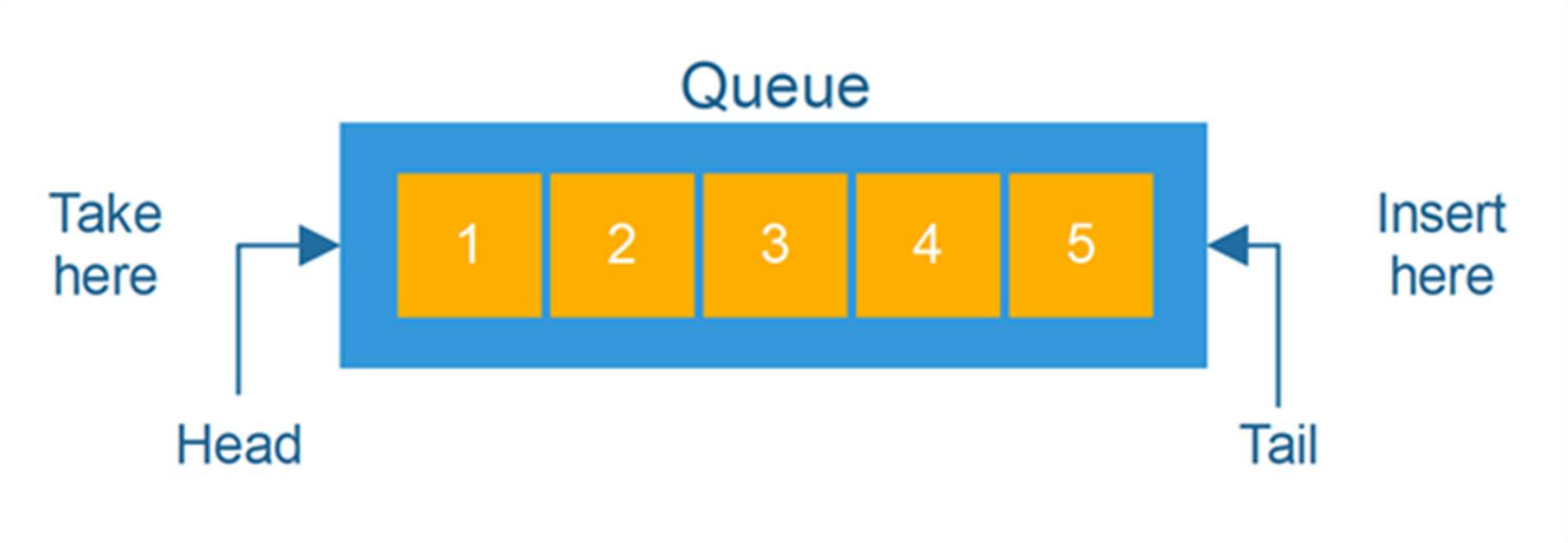
```
import java.util.HashSet; // Import the HashSet class
public class Main {
  public static void main(String[] args) {
    HashSet<String> cars = new HashSet<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("BMW");
    cars.add("Mazda");
    System.out.println(cars);
  }
}
```

HashSet vs HashMap



Queue

- Follows FIFO (First In, First Out) ordering of elements.
- Element inserted first in the queue will be the first element to be removed

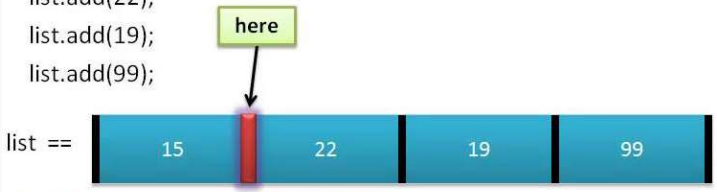


Iterators

```
import java.util.ArrayList;
import java.util.Iterator;
ArrayList <Integer> list = new ArrayList<Integer>();
list.add(15);
list.add(22);
list.add(19);
list.add(99);
```


```
Iterator here = list.iterator();
```

```
import java.util.ArrayList;
import java.util.ListIterator;
ArrayList <Integer> list = new ArrayList<Integer>();
list.add(15);
list.add(22);
list.add(19);
list.add(99);
```




```
ListIterator here = list.listIterator();
System.out.println(here.next());
```

Output

15

Iterators

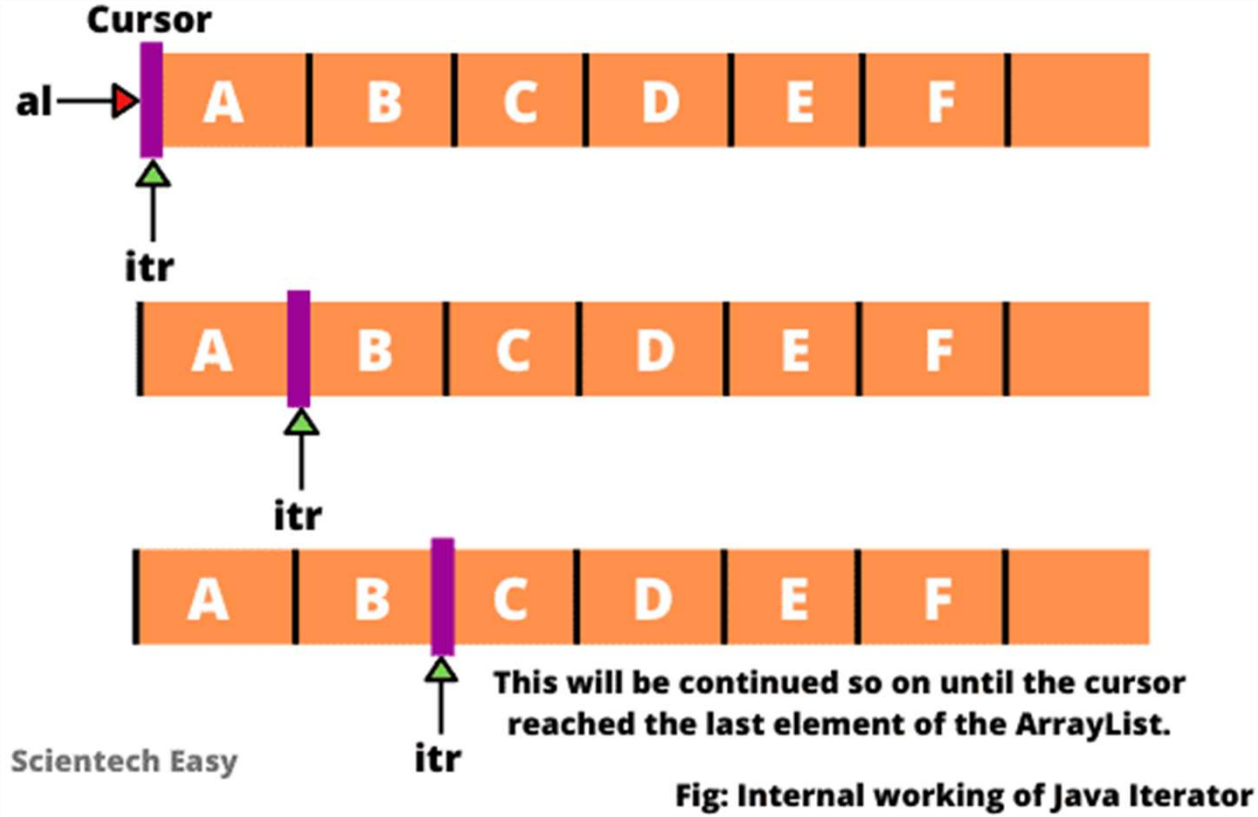
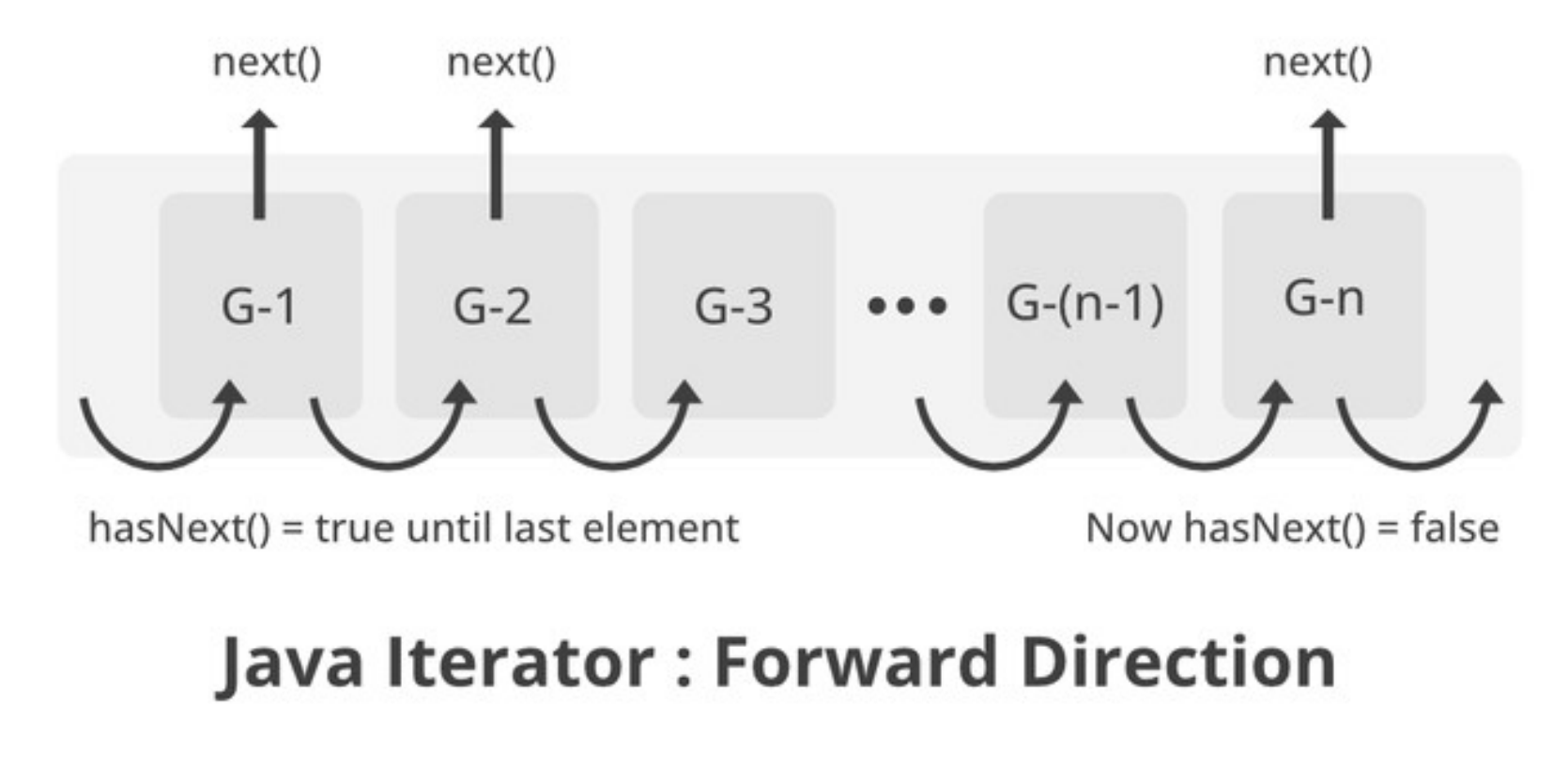


Fig: Internal working of Java Iterator

Iterators


next()
next()
next()
G-1
G-2
G-3
G-(n-1)
G-n
hasNext() = true until last element
Now hasNext() = false
Java Iterator : Forward Direction

Thank You