**Collection**

* The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects
* Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

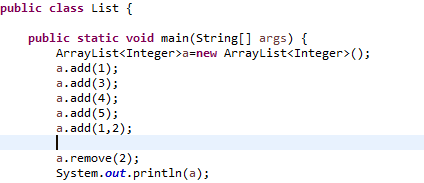


**List:**

* List is an [interface](http://en.wikipedia.org/wiki/Interface_%28Java%29) in Java, which means that it may have multiple implementations. One of these implementations are ArrayList,LinkedList,
* List interface is an ordered collection of objects in which duplicate values can be stored.
* List Maintains Insertion order

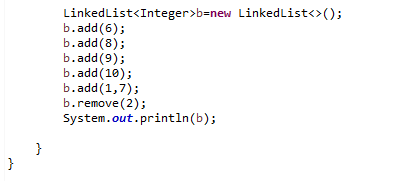
**Array List:**

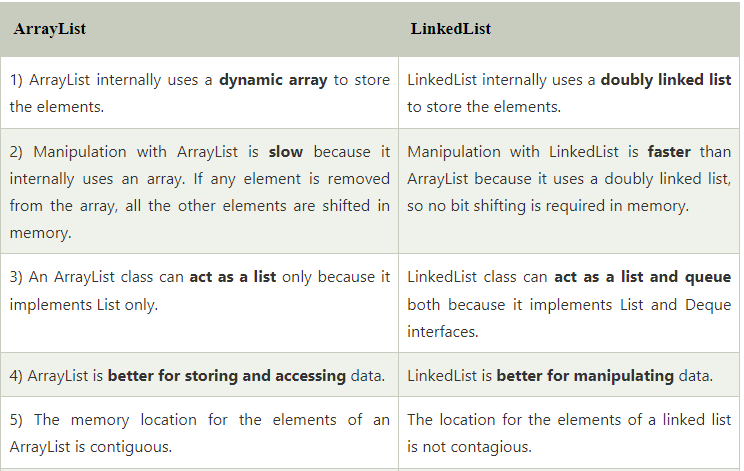
* Array List class can contain duplicate elements.
* Array List class maintains insertion order.
* Array List class is non [synchronized](https://www.javatpoint.com/synchronization-in-java).



**Linked List:**

* Java Linked List class uses a doubly linked list to store the elements. It provides a linked-list data structure
* Linked List class is non synchronized.
* Linked List class maintains insertion order.

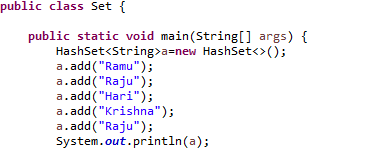




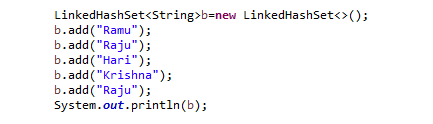
**Set:**

* The **set** is an interface available in the **java.util** package.
* The **set** interface extends the Collection interface
* Set does not maintain the insertion order
* Set does not allow duplicate values

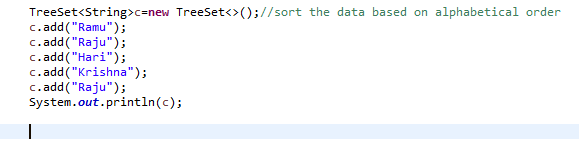
**Hashset:**

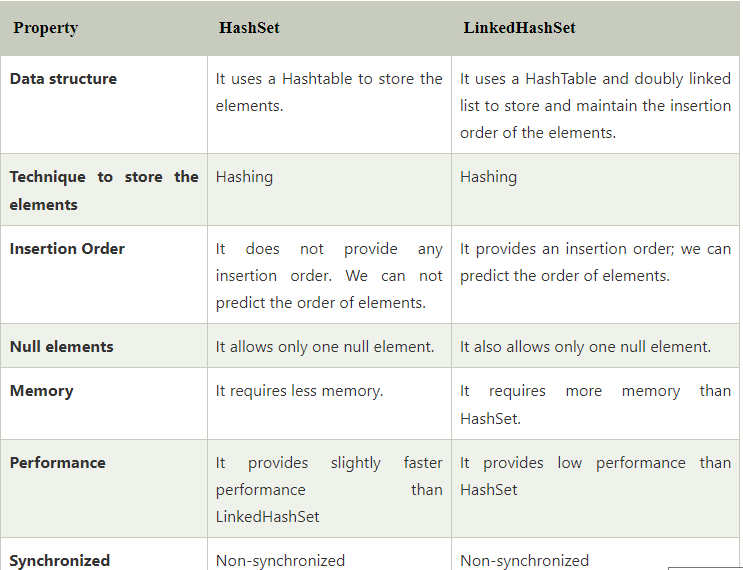


**LinkedHashset:**



**TreeSet:**



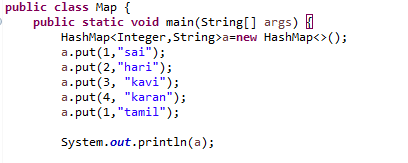


**Map:**

* A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry. A Map contains unique keys.
* A Map is useful if you have to search, update or delete elements on the basis of a key.
* Map uses put() and Get() method

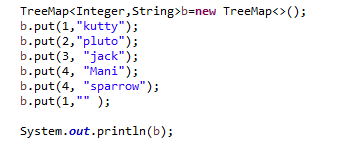
Java Map Hierarchy

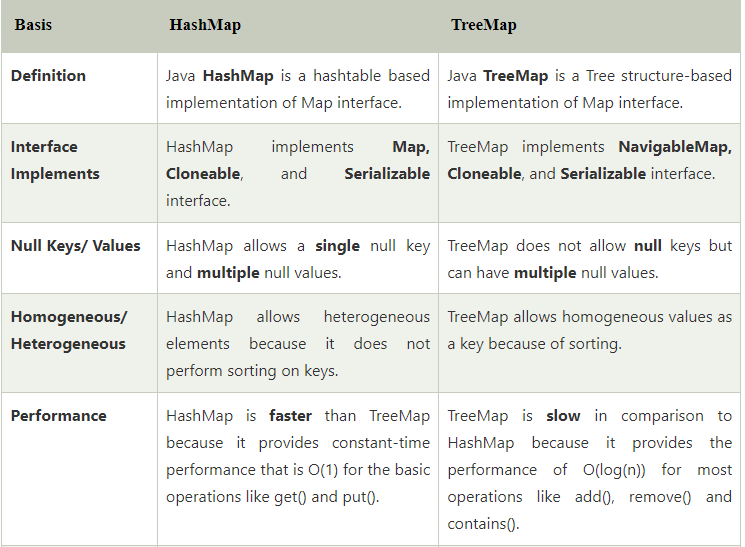
**HashMap:**



**Tree Map:**

* A Tree Map is always sorted based on keys



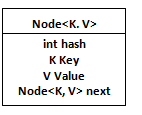


**Hashing:**

* It is the process of converting an object into an integer value. The integer value helps in indexing and faster searches.

**Hash Map:**

* HashMap is a part of the Java collection framework
* It uses a technique called Hashing.
* It stores the data in the pair of Key and Value.
* Hash Map contains an array of the nodes, and the node is represented as a class. It uses an array and Linked List data structure internally for storing Key and Value. There are four fields in Hash Map



Representation of Node

Before understanding the internal working of Hash Map, you must be aware of hash Code () and equals () method...

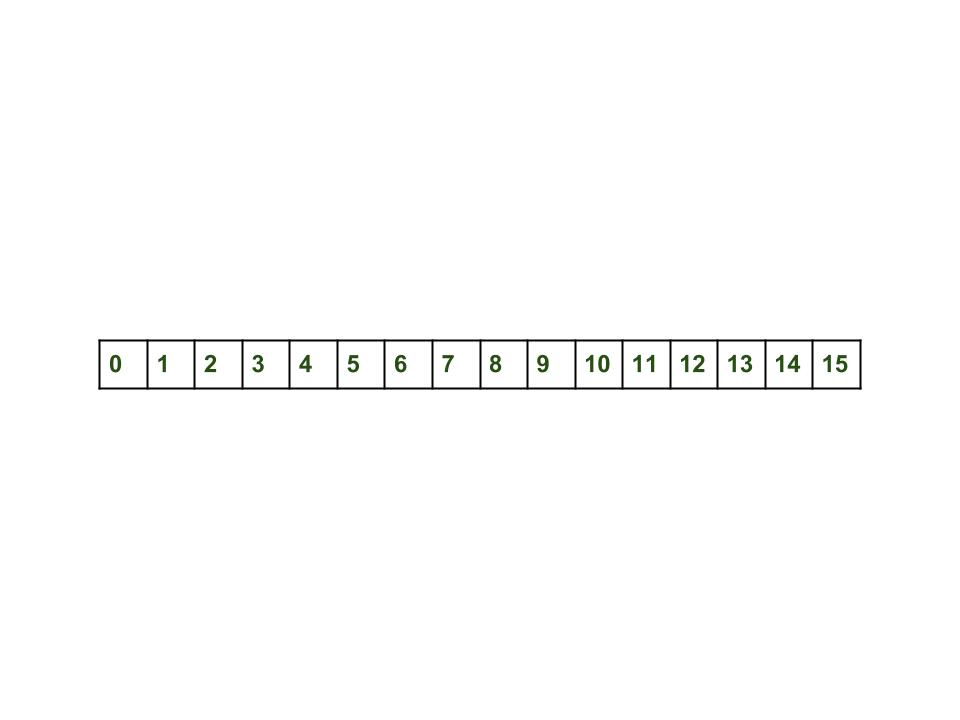
**Equals ():** It checks the equality of two objects. It compares the Key, whether they are equal or not

**Hash Code ():** This is the method of the object class. It returns the memory reference of the object in integer form. The value received from the method is used as the bucket number.

**Buckets:** Array of the node is called buckets. Each node has a data structure like a Linked List.



The default size of Hash Map is 16 (0 to 15).



Hash Map<String, Integer> map = **new** Hash Map<> ();

map.put("Aman", 19);

map.put("Sunny", 29);

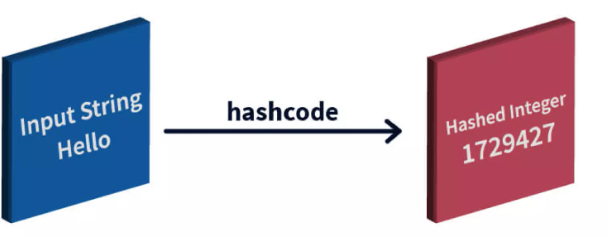
map.put("Ritesh", 39);

### **Calculating Index:**

 Formula for calculating the index is

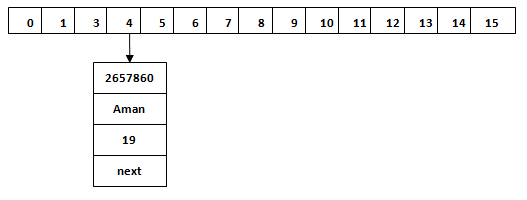
**Index = hashcode(Key) & (n-1)**

Where n is the size of the array. Hence the index value for "Aman" is



**Index = 2657860 & (16-1) = 4**

The value 4 is the computed index value where the Key and value will store in Hash Map.



## Hash Collision:

## This is the case when the calculated index value is the same for two or more Keys. Let's calculate the hash code for another Key "Sunny." Suppose the hash code for "Sunny" is 63281940

## Index=63281940 & (16-1) = 4

## 

## Working of HashMap in Java