Collections Framework

– ***present in java.util package***

- List
    - Maintains insertion order
    - Allows duplicate values
    - Null values allowed
- Set
    - Doesn’t allow duplicate values
    - Doesn’t maintain insertion order.
    - One Null Value allowed.(Except for treeSet)
- Map
    - Stores data in key value format
    - Key is unique
    - One null key and multiple null values allowed

– ***Tree type data structure doesn’t allow null values because it internally users compareTo() for comparing values which returns NPE on comparing will null whereas other data structures uses equals() for comparison which doesn’t throw NPE.***

ListInterface

- **ArrayList**
    - Best for storing and accessing data.
    - Uses dynamic array.
    - Initial capacity = 10.
    - Size increases by n+(n/2)+1 on exceeding the size.
    - Non synchronised.

- **LinkedList**
    - Best for insert, update, delete.
    - Uses doubly linked list.
    - Doesn’t have initial capacity.
    - Non synchronised.

- **Vector**
    - Legacy class.
    - Initial capacity = 10.
    - Size increases by 100% on exceeding the size.
    - Synchronised.
    - Slow

SetInterface

- **HashSet**
    - Best for searching operations
    - Contains unique elements
    - Allows null
    - Non synchronised
    - Default size = 16, load factor = 0.75
    - Doesn’t maintain insertion order

- **LinkedHashSet**
    - Same as HashSet but maintains insertion order

- **TreeSet**
    - Same as HashSet but stores elements in ascending order(default) and doesn’t allow null values

MapInterface

- **HashMap**
    - Unique keys.
    - One null key and multiple null values allowed.
    - Non synchronised.
    - Doesn’t maintain insertion order.
    - Default size = 16, Load factor = 0.75.
    - Size is doubled as soon as it reaches threshold.

- **LinkedHashMap**
    - Same as HashMap but maintains insertion order.

- **TreeMap**
    - TreeMap class is a red-black tree based implementation.
    - Same as HashMap but stores values in ascending order of keys and doesn’t allow null keys (multiple null values are allowed).

– ***HashSet internally uses HashMap. The values are stored as key in HashMap. This is done as the functionality of HashSet and HashMap is same in terms of uniqueness and hence the maintenance will be less.***

Hashing Mechanism

- Process of converting object into integer form using hashCode(). This hash value is a memory reference in integer form.
-

Internal Working of HashMap

- **put()**
    - Calculate hashCode of key. A bucket is selected based on the generated hashCode. If key is null bucket-0 as hashCode of null is zero.
    - Place the Entry object in that bucket.
    - If an element is already present in that bucket the next element is appended to the linkedList.
    - If an element is having same key, the bucket is found using hashCode(). Then equals() method is used to check the equality of keys. When a match is found, the value is replaced with the new one.
- **get()**
    - Bucket is found using hashCode() on key. If element is present it returns the object else returns null.
    - If more than one elements are present in that bucket, equals() is used to match the keys.