**Collection Framework – Part\_02**

* **9 key interfaces of collection framework:**

1. Collection
2. List
3. Set
4. SortedSet
5. NavigableSet
6. Queue
7. Map
8. SortedMap
9. NavigableMap

* **Collection(I):**

If we want to represent a group of individual objects as a single entity, then we should go for collection.

Collection interface defines the most common methods which are applicable for any collection object.

In general Collection interface is considered as root interface of collection framework.

There is no concrete class which implements Collection interface directly.

* **Difference between Collection and Collections:**

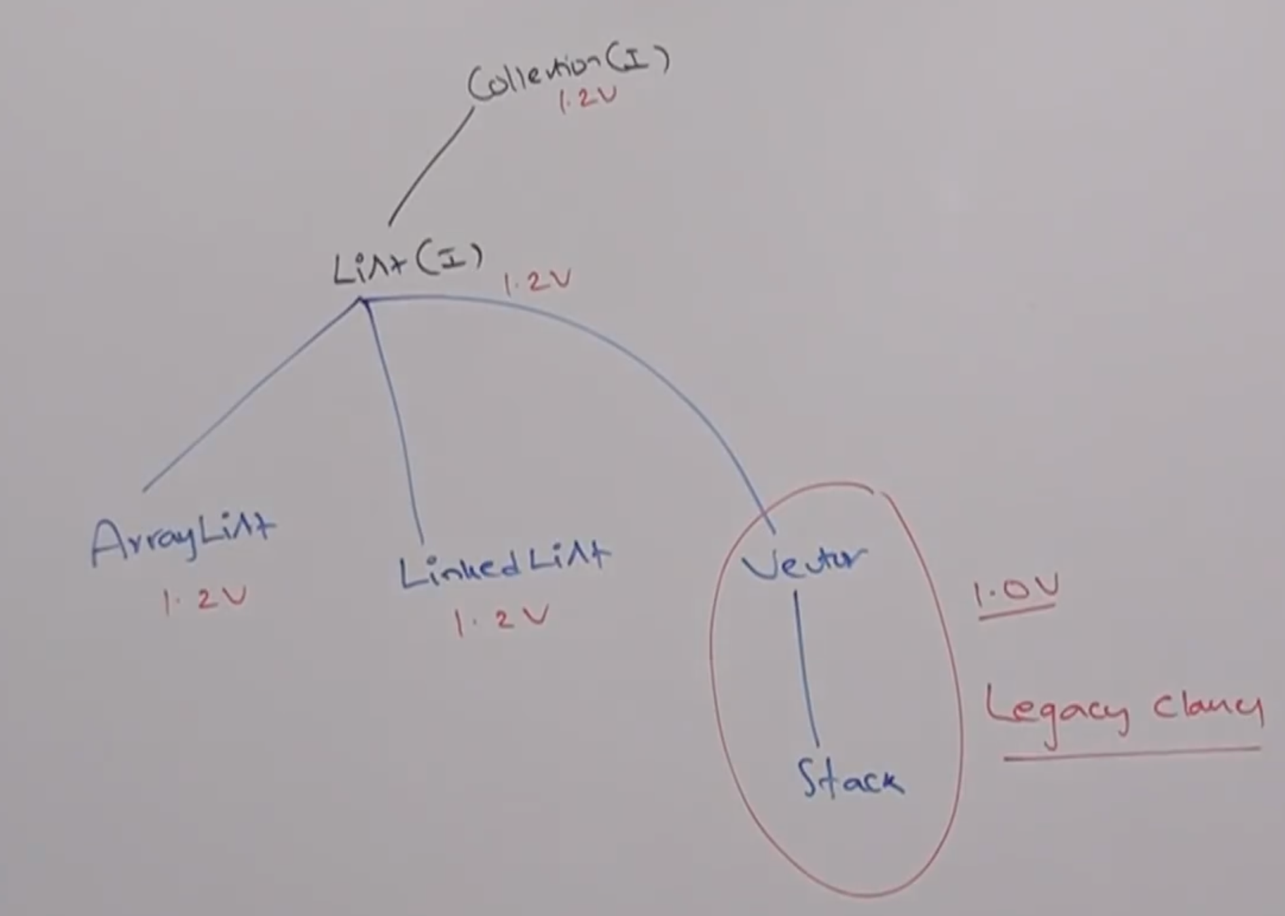
Collection is an interface. If we want to represent a group of individual objects as a single entity then we should go for collection.

Collections is an utility class present in java.util package to define several utility methods for collection objects (like sorting, searching etc).

* **List(I):**

It is the child interface of collection.

If we want to represent a group of individual objects as a single entity where duplicates are allowed and insertion order must be preserved. Then we should go for List.



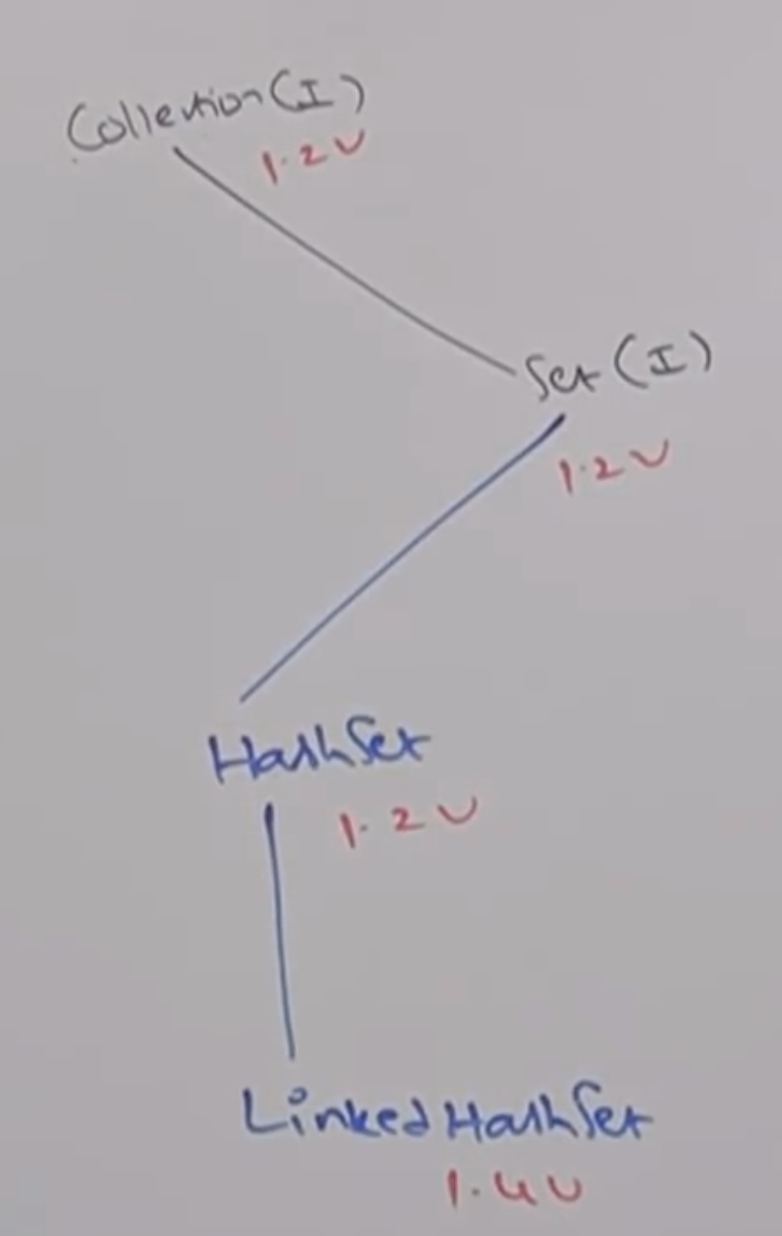
Note:

In 1.2 version Vector and Stack classes are re-engineered/modified/updated to implement List interface.

* **Set(I):**

It is the child interface of Collection.

If we want to represent a group of individual objects as a single entity where duplicates are not allowed and insertion order not required. Then we should go for Set interface.



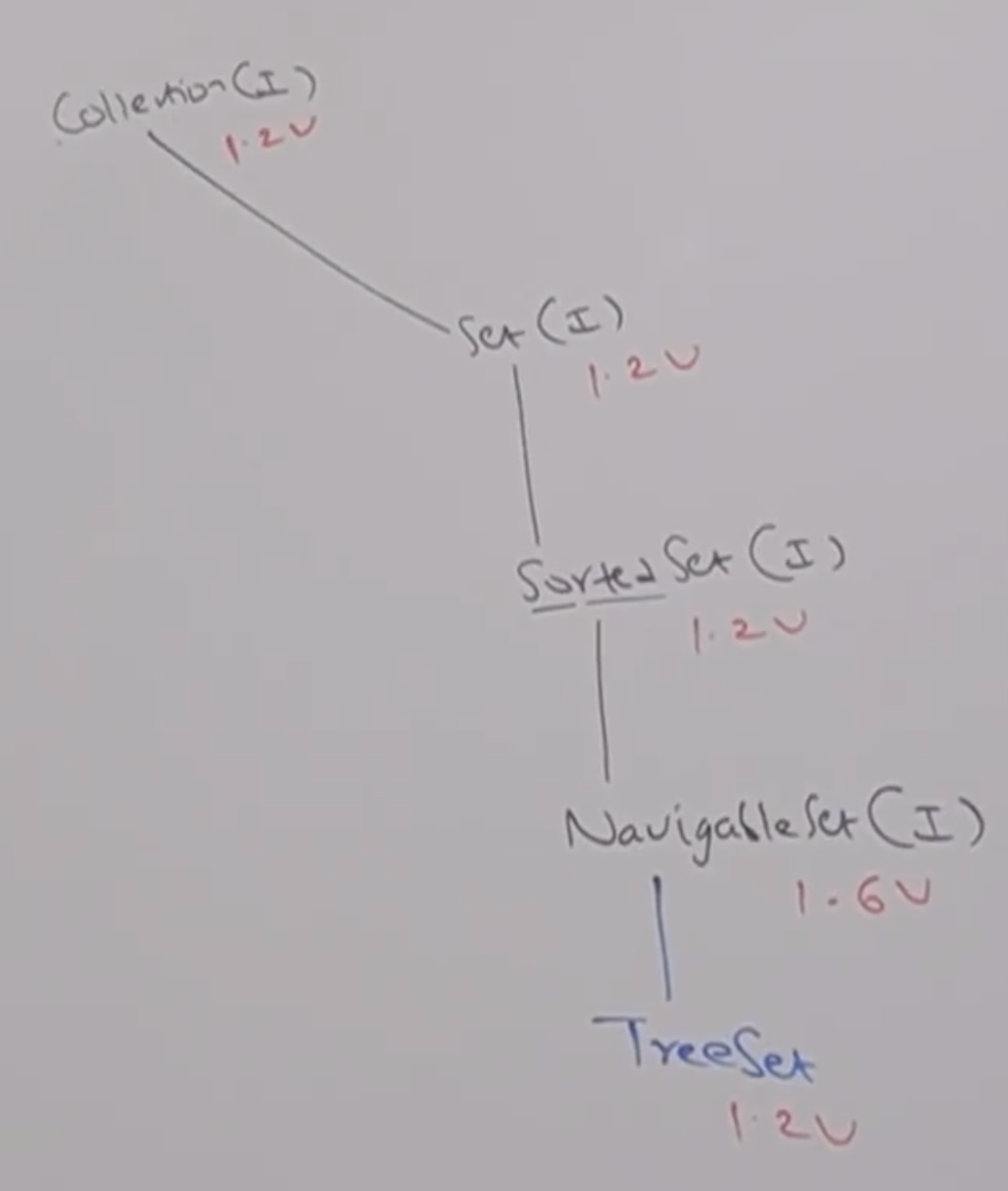
* **SortedSet(I):**

It is the child interface of Set. If we want to represent a group of individual objects as a single entity where duplicates are not allowed and all objects should be inserted according to some sorting order. Then we should go for SortedSet.

* **NavigableSet(I):**

It is the child interface of SortedSet.

It contains several methods for navigation purposes.



* **Differences between List(I) and Set(I)**

|  |  |  |
| --- | --- | --- |
| S.No | List | Set |
| 1 | Duplicates are allowed | Duplicates are not allowed |
| 2 | Insertion order preserved | Insertion order not preserved. |

* **Queue(I):**

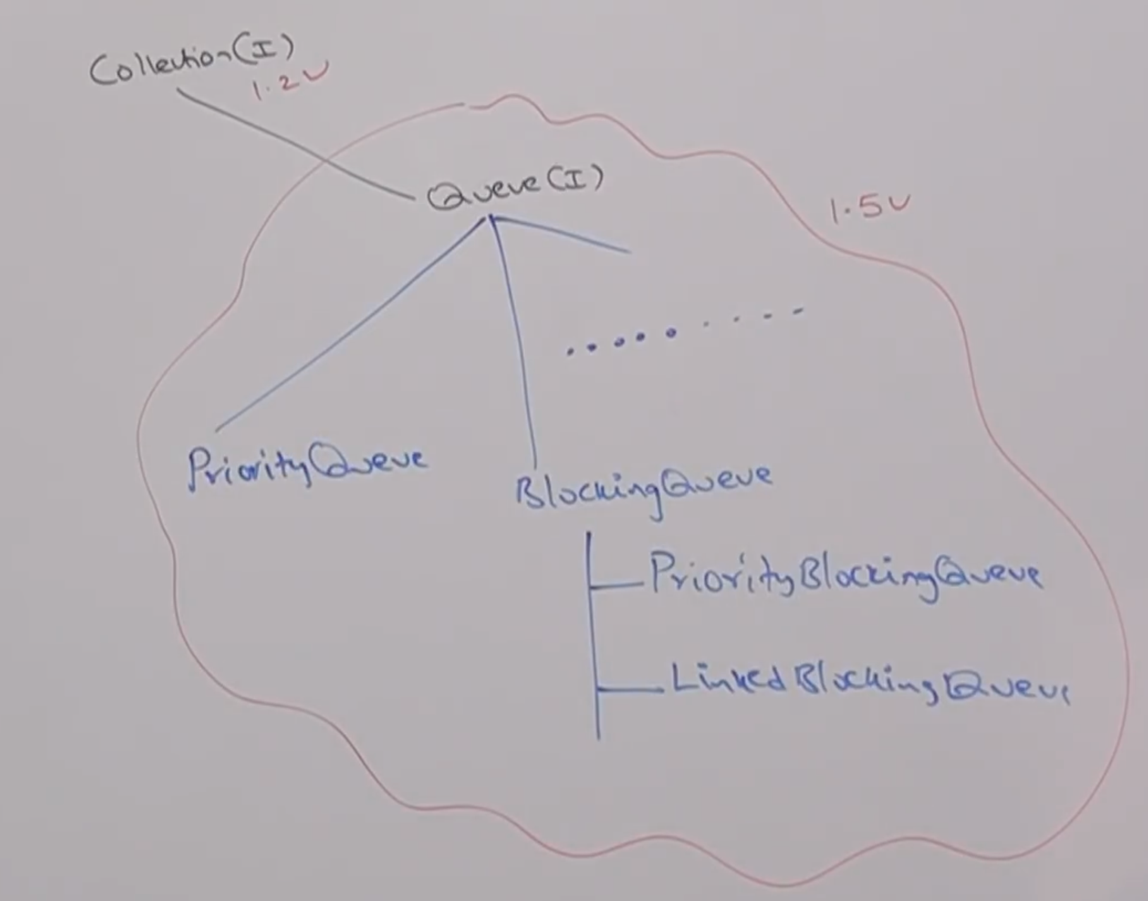
It is the child interface of Collection.

If we want to represent a group of individual objects “prior to processing” then we should go for queue.

Usually queue follows FIFO (First In First Out) order, but based on our requirement we can implement our own priority order also.

Example:

Before sending a mail we have to save all the email id we have to store in some data structure. In which order we added mail id’s in the same order only mail should be delivered. For this requirement Queue is best choice.



**Note:**

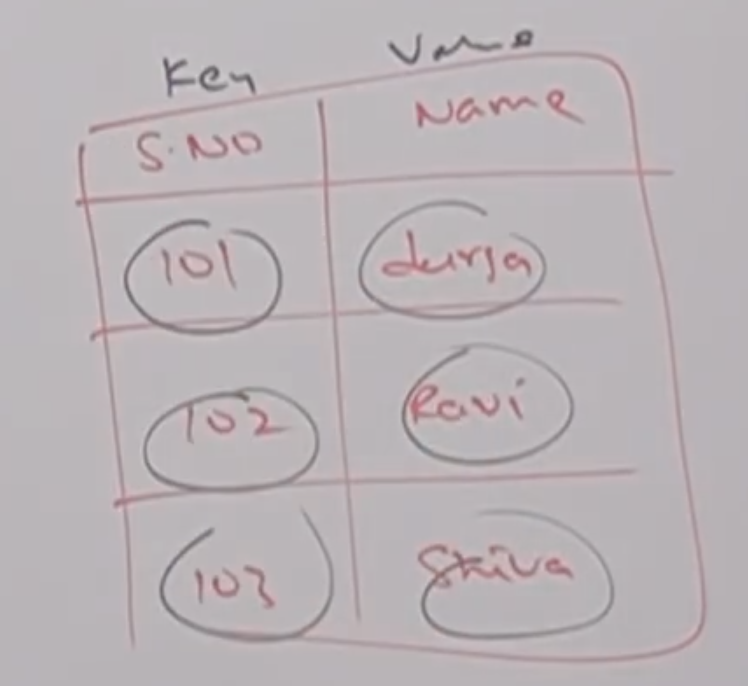
All the above interfaces (Collection, List, Set, SortedSet, NavigableSet & Queue) meant for representing a group of individual objects.

If we want to represent a group of objects as key value pairs, then we should go for Map.

* **Map(I):**

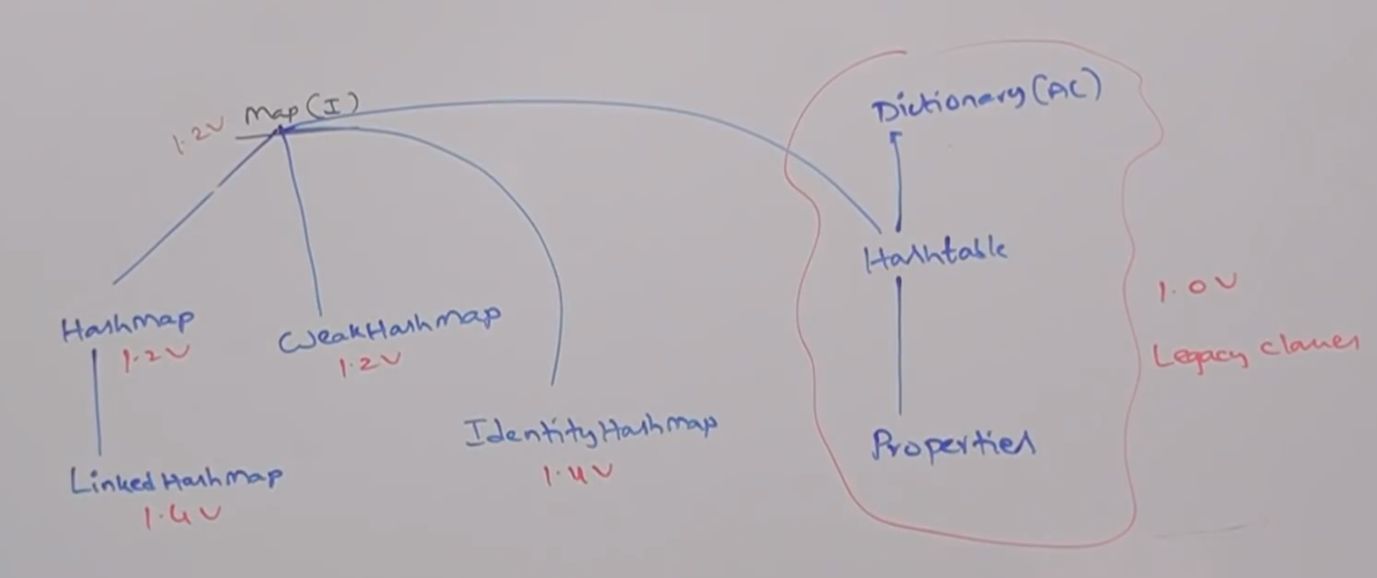
Map is not child interface of collection.

If we want to represent a group of objects as key-value pairs then we should go for Map.



Both key and value are objects only.

Duplicate keys are not allowed, but values can be duplicated.

****

* **SortedMap(I):**

It is the child interface of Map.

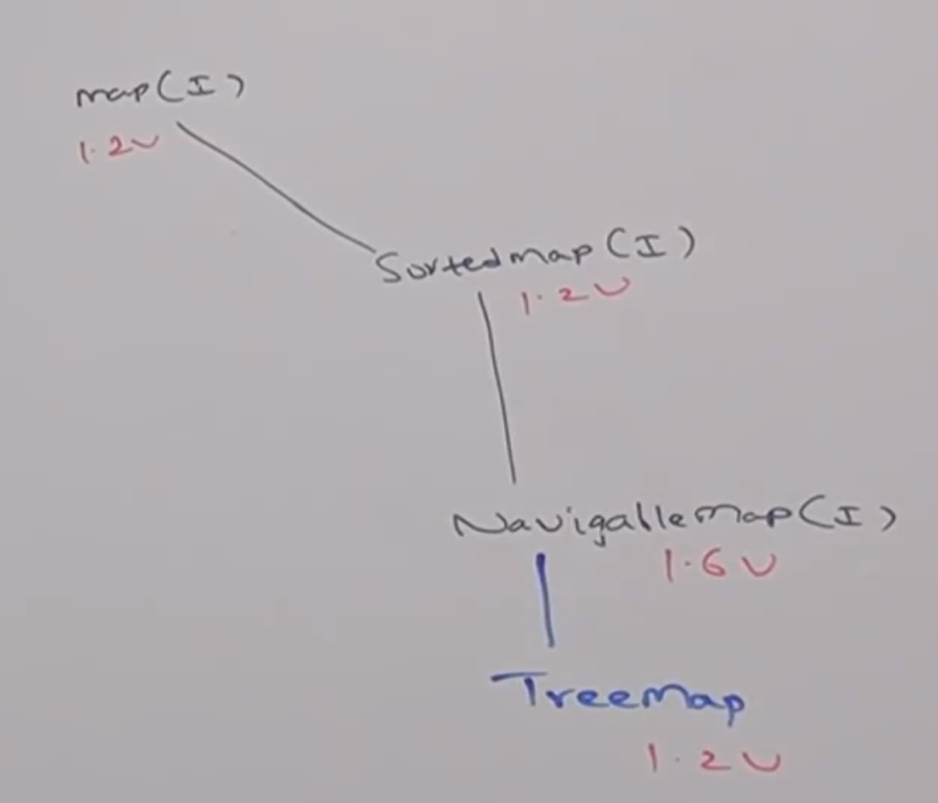
If we want to represent a group of key-value pairs according to some sorting order of keys then we should go for sorted map.

In SortedMap, the sorting should be based on Key, but not based on Value.

* **NavigableMap(I):**

It is the child interface of SortedMap.

It defines several methods for navigation purposes.

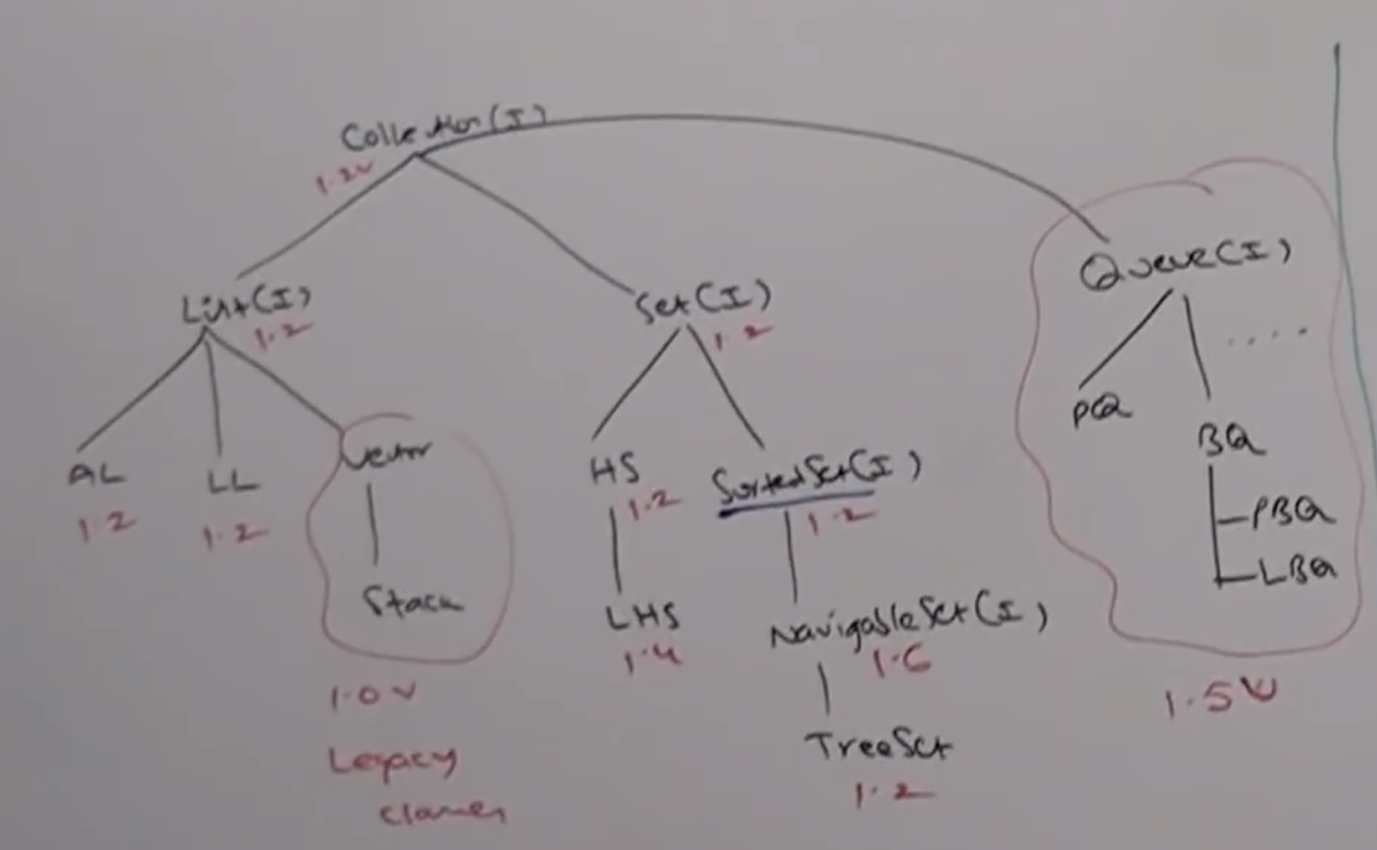


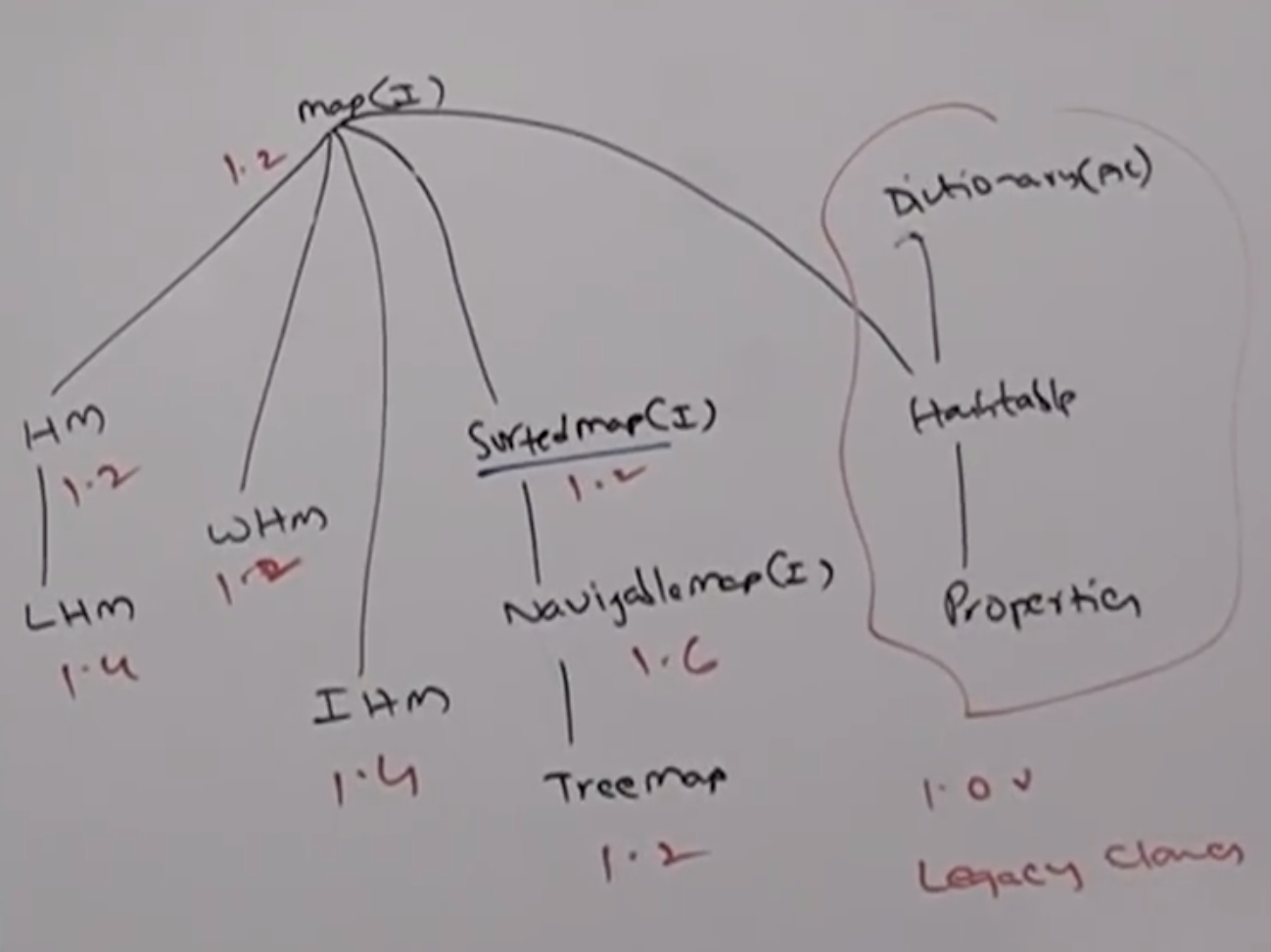
* **Note:**

The following are legacy characters present in collection framework.

1. Enumeration(I)
2. Dictionary(AC)
3. Vector(c)
4. Stack(c)
5. Hashtable(C)
6. Properties(C)

* **Collection Diagrams:**

****

****

**Sorting:**

Comparable(I)

Comparator(I)

**Cursors:**

Enumeration(I)

Iterator(I)

ListIterator(I)

**Utility Classes:**

Collections(C)

Arrays(C)