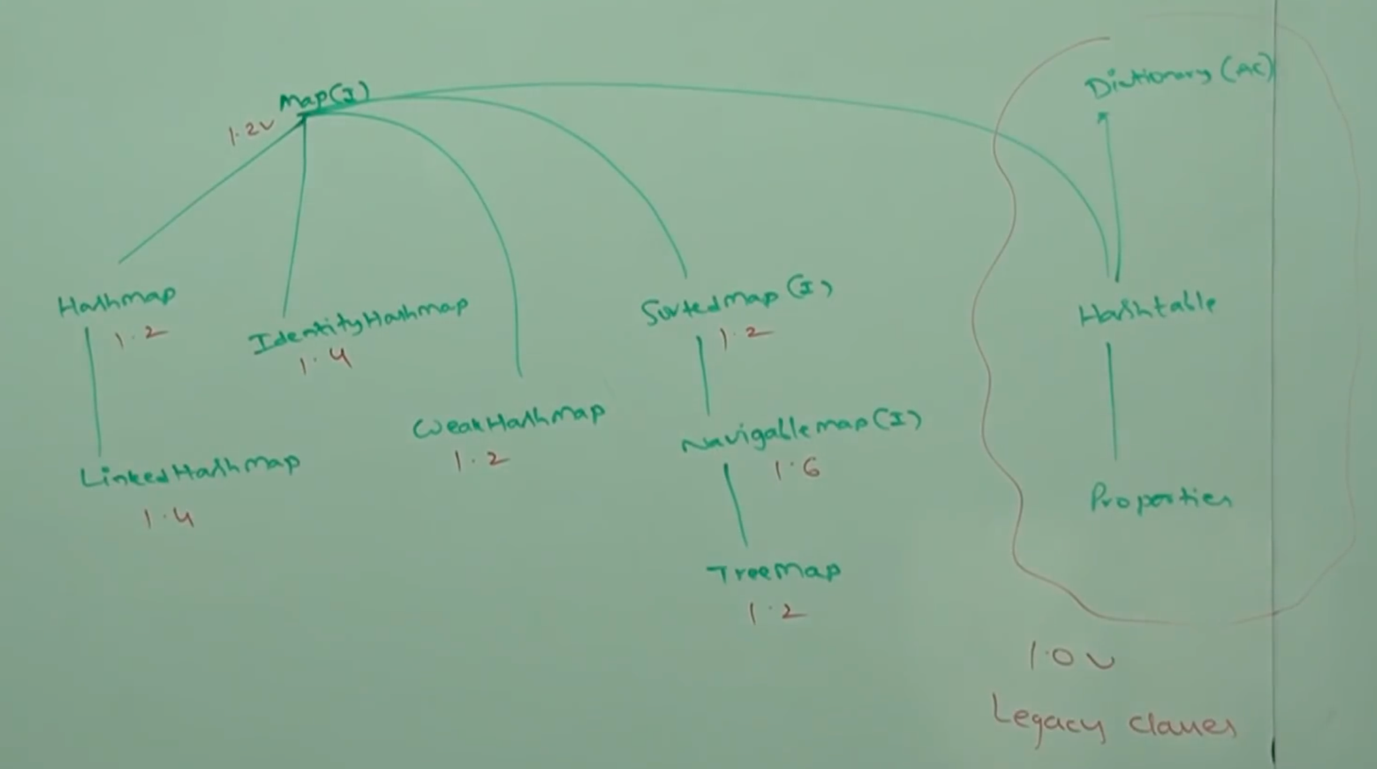
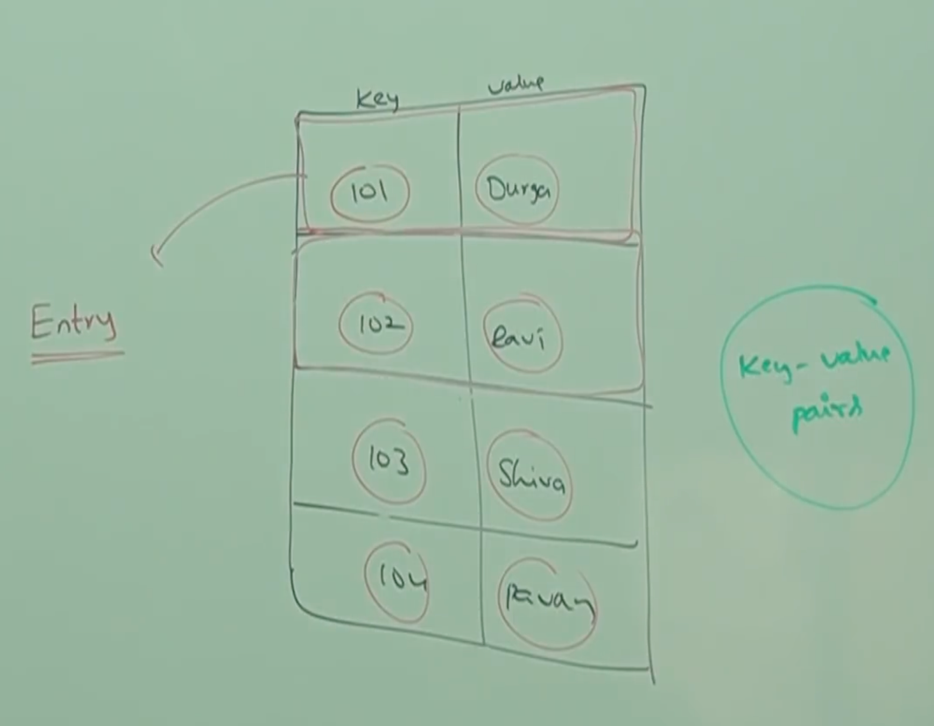
**Collection Framework – Part\_11**

* **Map Agenda:**

****

* **Map:**

1. Map is not child interface of Collection.
2. If we want to represent a group of objects as key-value pairs, then we should go for Map.



1. Both keys and values are objects only.
2. Duplicates key are not allowed. But values can be duplicated.
3. Each key-value pair is called Entry. Hence Map is considered as a collection of Entry objects.

* **Map interface methods:**

Object put(Object key, Object value);

Note: Object 🡪 old value

To add one key-value pair to the Map.

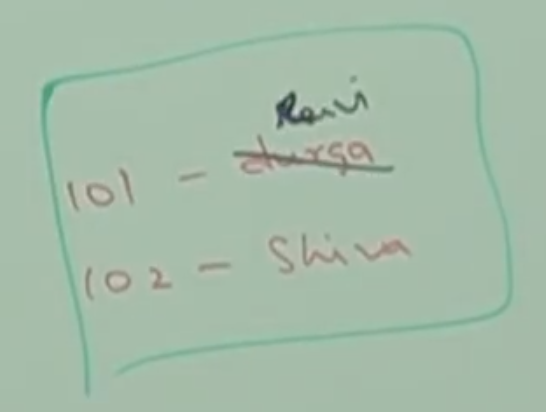
If the key is already present then old value will be replaced with new value and returns old value.

Example:

m.put(101, “durga”); // null because no replacement

m.put(102, “shiva”); // null because no replacement

m.put(101, “ravi”); //durga



void putAll(Map m)

Object get(Object key);

returns the value associated with specified key.

Object remove(Object key)

Removes the entry associated with specified key.

boolean containsKey();

boolean containsValue();

boolean isEmpty();

int size();

void clear();

Set keySet();

Collection values();

Set entrySet();

The last three methods are called Collection views of Map.

* **Entry(I):**

A Map is a group of key-value pairs and each key-value pair is called an Entry, hence Map is considered as a collection of Entry objects. Without existing Map object there is no chance of existing Entry object, hence Entry interface is defined inside Map interface.

interface Map{

interface Entry{

Object getKey();

Object getValue();

Object setValue();

}

}

The methods defined above inside the Entry are specific to Entry and we can apply only on Entry object.

* **HashMap:**

1. The underlying data structure is Hashtable.
2. Insertion order is not preserved and it is based on hashcode of keys.
3. Duplicates keys are not allowed but values can be duplicated.
4. Heterogeneous are allowed for both key and value.
5. Null is allowed for key (only once). Null is allowed for values (any number of times).
6. HashMap implements Serializable and Cloneable interfaces, but not RandomAccess.
7. HashMap is the best choice, if our frequent operation is search operation.

Constructors:

HashMap m = new HashMap();

Creates an empty HashMap object with default initial capactiy 16 and default fill ration 0.75

HashMap m = new HashMap(int initialCapacity);

Creates an empty HashMap object with specified initial capacity and default fill ratio.

HashMap m = new HashMap(int initialCapacity, float fillRatio);

HashMap m = new HashMap(Map m)

* **HashMap Example:**

import java.util.\*;

class HashMapDemo{

public static void main(String[] args){

HashMap m = new HashMap();

m.put(“chiranjeevi”, 700 );

m.put(“balaiah”, 800);

m.put(“Venkatesh”, 200);

m.put(“Nagarjuna”, 500);

System.out.println(m); // {K=V, K=V}

System.out.println(m.put(“chiranjeevi”, 1000)); //700

Set s = m.keySet();

System.out.println(s); //[K K, K]

Collection c = m.values();

System.out.println(c);

Set s1 = m.entrySet();

System.out.println(s1); // [K=V, K=V]

Iterator itr = s1.interator();

while(itr.hasNext()){

Map.Entry m1 = (Map.Entry)itr.next();

System.out.println(m1.getKey()  
+”…”+m1.getValue());

if(m1.getKey().equals(“nagarjuna”)){

m1.setValue(10000);

}

}

System.out.println(m);

}

}

* **Differences between HashMap and Hashtable:**

|  |  |  |
| --- | --- | --- |
| S.No | HashMap | Hashtable |
| 1 | Every method present in HashMap is non-synchronized. | Every method present in HashTable is synchronized. |
| 2 | At a time multiple threads are allowed to operate on HashMap object and hence it is not thread-safe. | At a time only one thread is allowed to operate on Hashtable and hence it is thread-safe. |
| 3 | Relatively performance is high because threads are not required to wait to operate on HashMap object. | Relatively performance is low because threads are required to wait to operate on Hashtable object. |
| 4 | Null is allowed for both Key and Value. | Null is not allowed for keys and values otherwise we will get NullPointerException |
| 5 | Introduced in 1.2 version and it is not legacy. | Introduced in 1.0 version and it is legacy. |

* **How to get synchronized version of HashMap object?**

By default HashMap is non-synchornized but we can get

synchorinized version of HashMap by using synchronizedMap method of Collections class.

HashMap m = new HashMap();

Map m1 = Collections.synchronizedMap(m);

