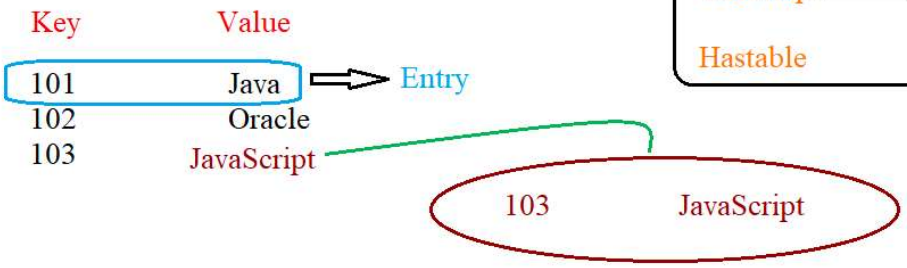
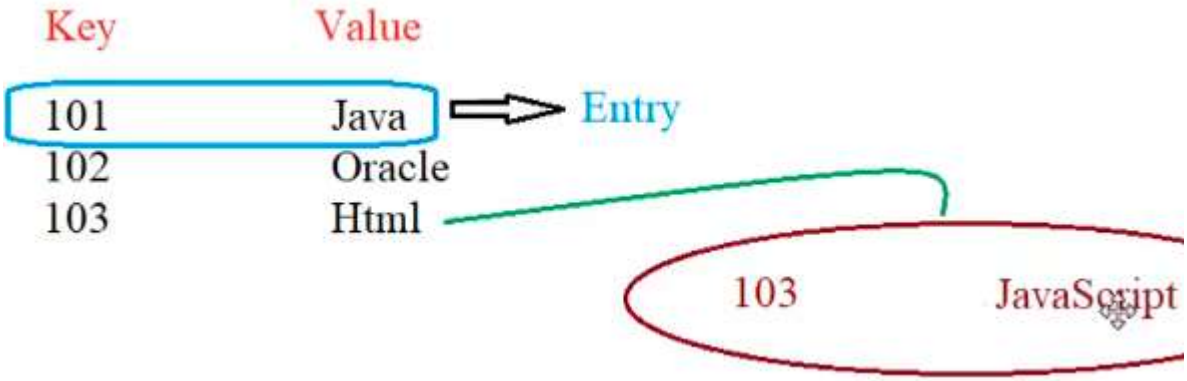
Map Interface:

1) Map is NOT a child Interface for Collection Interface.
2) It is one of the root interfaces for Collection framework.
3) In Map Implementation classes elements will be stored in the form of Key-Value pairs.
4) **Keys should be unique.**
5) Values can be duplicate.

HashMap ⇒ HashSet
LinkedHashMap ⇒ LinkedHashSet
TreeMap ⇒ TreeSet

Hastable





Understanding Map Interface

- In Map elements are stored in the form of Key-Value pairs.
- Keys are unique in Map interface, duplicate keys are not allowed but duplicate values are allowed.
- Each key can map to at most one value.
- Each key-value pair is called "one entry".
- Duplicates will be replaced but they are not rejected.
- Map interface is not child interface of Collection interface.
- Some map implementations have restrictions on the keys and values they may contain.

Map Interface Methods

Method	Description
int size();	Returns the number of key-value mappings in this map
boolean isEmpty();	Returns true if this map contains no key-value mappings
boolean containsKey (Object key);	Returns true if this map contains a mapping for the specified key.
boolean containsValue (Object value);	Returns true if this map maps one or more keys to the specified value
V get(Object key)	Returns the value to which the specified key is mapped
V put(K key, V value)	Associates the specified value with the specified key in this map
V remove(Object key)	Removes the mapping for a key from this map if it is present
Set<K> keySet()	Returns a Set view of the keys contained in this map

HashMap:

- HashMap is available since jdk1.2V.
- It is available in java.util package.
- It allows duplicate values with unique keys & insertion order is not maintained. (Duplicate keys are replaced)
- Default capacity is 16 & load factor is 0.75.
- HashMap is not synchronized by default.

HashMap m=new HashMap();

HashMap m=new HashMap(int initialcapacity);

LinkedHashMap:

- The only difference between HashMap & LinkedHashMap is HashMap doesn't maintain the insertion order where as LinkedHashMap maintains it.
- LinkedHashMap is available since jdk 1.4

```
1  package com.pack1;
2
3  import java.util.HashMap;
4
5  public class ClassA
6  {
7      void meth1()
8      {
9          System.out.println("Implementing Map Implementation Classes");
10
11         HashMap<Object, Object> map=new HashMap<Object, Object>();
12
13         map.add(101,"Java");
14     }
15 }
```

Error because of add() is presented in collection interface

Map is not the child interface for collection.

We need to use put() to add elements.

```
    void meth1()
    {
        System.out.println("Implementing Map Implementation Classes\n");

        HashMap<Object, Object> map=new HashMap<Object, Object>();

        map.put(101,"Java"); // Insertion order is NOT maintained
        map.put("Java",1000); // Heterogeneous keys & Heterogeneous values are allowed
        map.put(null,null); // null keys & null values are allowed
        map.put(105,"Java");// Duplicate VALUES are allowed
        map.put(102,'A'); // It is available from Java 1.2v
        map.put(true,2000); // Its default capacity is 16
        map.put(104,"Oracle");// Its size increases by DOUBLE
        map.put("Kishan","Java is awesome"); // It is NOT Synchronized

        System.out.println("map : "+map);

    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.Map.Entry;

public class ClassA
{
    void meth1()
    {
        System.out.println("Implementing Map Implementation Classes\n");

        //HashMap<Object, Object> map=new HashMap<Object, Object>();// Insertion order is NOT maintained (Java 1.2v)
        LinkedHashMap<Object, Object> map=new LinkedHashMap<Object, Object>(); // Insertion order is maintained (Java 1.4v)

        map.put(101,"Java");
        map.put("Java",1000); // Heterogeneous keys & Heterogeneous values are allowed
        map.put(null,null); // null keys & null values are allowed
        map.put(105,"Java");// Duplicate VALUES are allowed
        map.put(102,'A'); // It is available from Java 1.2v
        map.put(true,2000); // Its default capacity is 16
        map.put(104,"Oracle");// Its size increases by DOUBLE
```

```
28          map.put("Kishan","Java is awesome"); // It is NOT Synchronized
29          
30          System.out.println("map : "+map);
31          
32          System.out.println("\nmap.get(Kishan) : "+map.get("Kishan"));
33          
34          map.put("Kishan","Java Trainer");
35          
36          System.out.println("map.get(Kishan) : "+map.get("Kishan"));
37          System.out.println("size() : "+map.size());
38          
39          
40          System.out.println("\nReteriving all the Keys from Map");
41          ArrayList<Object> al1=new ArrayList<Object>(map.keySet());
42          LinkedHashSet<Object> hs1=new LinkedHashSet<Object>(map.keySet());
```

```
43          
44          System.out.println("Keys in ArrayList : "+al1);
45          System.out.println("Keys in LinkedHashSet : "+hs1);
46          
47          System.out.println("\nReteriving all the Key-values from Map");
48          for(Object o:al1)
49          {
50              System.out.println(o+" "+map.get(o));
51          }
52          
53          System.out.println("\nReteriving all the Key-values from Map by using Entry Interface");
54          LinkedList<Object> ll=new LinkedList<Object>(map.entrySet());
55          Iterator<Object> i=ll.iterator();
56          while(i.hasNext())
57          {
58              System.out.println(i.next());
59              //Entry e=(Entry)i.next();
60              //System.out.println(e.getKey()+" "+e.getValue());
61          }
62      }
```

```
63      public static void main(String[] args)
64      {
65          ClassA aobj=new ClassA();
66          aobj.meth1();
67      }
68  }
69  
70  /*
71   1) In all map implementation classes keys should be unique
72   2) Whenever we are passing a duplicate key-value pair we will
73      not be getting any error (or) an exception OLD value which
74      is associated with the key will be replaced with the new value.
75   */
```

Title: Product Inventory Management System

Description:
You are tasked with creating a simple inventory management system for a small store using Java. The system should use a combination of HashMap and ArrayList to manage the products in the inventory.

Requirements:

Create a **Product** class with the following attributes:

productId (int)
productName (String)
price (double)
quantity (int)
Create a **ProductInventory** class which will manage the inventory using a HashMap and an ArrayList. This class should have methods to:

Add a product to the inventory.
Remove a product from the inventory.
Update the quantity of a product.
Display details of a particular product.
Display details of all products in the inventory.
Use a HashMap to store the products, where the key is the productId and the value is the Product object.

Use an ArrayList to maintain the order of the products as they are added to the inventory.

Implement a menu-driven interface for interacting with the inventory system. The menu should include options to:

Add a new product.
Remove a product.
Update the quantity of a product.
Display details of a product.
Display details of all products.
Exit the program.