

```
HashMap m = new HashMap();
Map m1 = Collections.synchronizedMap(m);
```

### → LinkedHashMap :-

→ It is the child class of HashMap.

→ It is exactly same as HashMap (including methods & Constructors) except the following differences:-

HashMap	LinkedHashMap
① <del>HashMap</del> underlying data structure is HashTable	① <del>ec</del> LinkedList + HashTable (Hybrid data structure)
② Insertion order is not preserved. and it is based on hash code of keys.	② Insertion order is preserved.
③ Introduced in 1.2v	③ Introduced in 1.4v.

⇒ In the above prog (Hash Map prog), if we replace HashMap with LinkedHashMap then o/p is  
 { chiranjivi = 700, balaiah = 800, Venkatesh = 200, nagarjuna = 500 }  
 i.e. Insertion order is preserved.

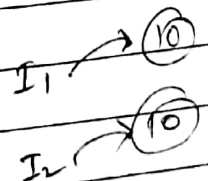
NOTE:- LinkedHashMap and LinkedHashSet are commonly used for developing caching based applications.

⇒ Diff b/w "==" operator and "equals" method.

→ In general "==" operator maint for reference comparison (address comparison) whereas "equals()" method for content comparison.

Ex.

```
Integer I1 = new Integer(10);
"      I2 = "      "      (10);
```



S.o.p ( I1 == I2 ) ; ⇒ false

S.o.p ( I1.equals(I2) ; ⇒ True.

⇒ IdentityHashMap :-

→ It is exactly same as HashMap (including methods and constructors except the following differences.

In the case of normal HashMap, JVM will use `equals()` method to identify the duplicate keys, which is maint for content comparison.

But, in the case of IdentityHashMap, JVM will use `==` operator to identify duplicate keys. which is maint for reference comparison (address comparison).

Ex:-

```
HashMap m = new HashMap();
Integer i1 = new Integer(10);
      i2 = new Integer(10);
m.put(i1, "pawan");           i1 → 10
m.put(i2, "kalyan");          i2 → 10
So, op(m) → {10 = kalyan}
```

$i1$  and  $i2$  are duplicate keys because

`i1.equals(i2)` returns true.

If we replace HashMap with IdentityHashMap, then  $i1$  and  $i2$  are not duplicate keys. bcoz, `(i1 == i2)` returns false.

In this case op is :-

{10 = Pawan, 10 = kalyan}.

⇒ WeakHashMap :-

→ It is exactly same as HashMap except the following differences.

In the case of HashMap, even though object doesn't have any reference, it is not eligible for GC. If it is associated with HashMap i.e. HashMap dominates Garbage collector.