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 ~~HashTable~~ Underlying data structure is HashTable	① ie 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 (HashMap prog), If we replace HashMap with LinkedHashMap then o/p is
{ chiranjivi=700, balaiah=800, venkatesh=300, nagarjuna=500 }.
i.e. Insertion order is preserved.

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

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

→ In general "==" operator meant for reference comparision (address comparison) whereas ".equal()" method for content comparision.

Ex.

Integer I1 = new Integer(10);
" I2 = " " (10);
S.O.p(I1 == I2); ⇒ false
S.O.p(I1.equals(I2); ⇒ True.

I1 → 10
I2 → 10

Page No.:

⇒ **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 comparision.

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

Ex:-

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

I1 → (10)
I2 → (10)

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. because (I1 == I2) returns false.

In this case o/p 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.