The internal workings of a **LinkedHashMap** are similar to the internal workings of **HashMap** with one major difference. In **LinkedHashMap**, each **Entry** maintains the record of the **Entry** that was inserted before it and after it.

If we look at the **Entry** class of **LinkedHashMap**, then we can see that it has two extra fields in comparison to the **Entry** class of **HashMap**. These extra fields are *before* and *after*. For a given **Entry**, the before field points to the **Entry** that was inserted prior to this **Entry**. The *after* field points to the **Entry** that was inserted after this **Entry**.

```
static class Entry<K,V> extends HashMap.Node<K,V> {
    Entry<K,V> before, after;
    Entry(int hash, K key, V value, Node<K,V> next) {
        super(hash, key, value, next);
        }
    }
```

There are two additional fields in the **LinkedHashMap** namely *head* and *tail*. The *head* points to the first node that was inserted in the Map and *tail* points to the last node that was inserted in the Map.

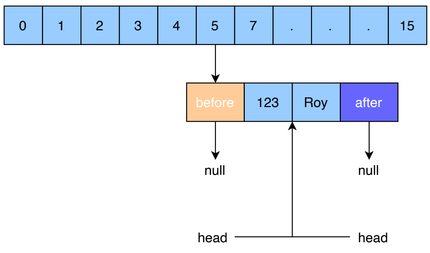
Let's take a step by step look at how elements are inserted in a **LinkedHashMap**.

## Inserting the first element

Let's consider creating a **LinkedHashMap** that stores the student information. The key is the id of the student and the value is the name of the student.

We are inserting our first record, 123; "Roy". The following process will occur:

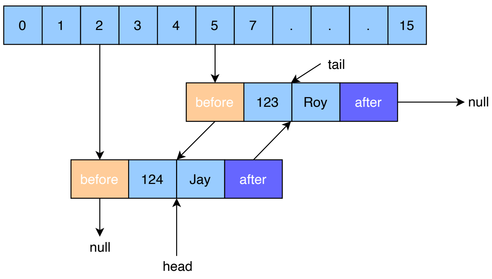
1. The hash of 123 will be calculated and based on the hash value, a bucket will be decided. Let's say the bucket is 5.
2. An Entry object is created with the key as **123** and the value as **Roy**. The before and after fields are set to null as this is the first record.
3. Since there is no element in the LinkedHashMap, both the head and tail variables are null. Now both these variables will point towards the newly created Entry.




## Inserting the second element

Now we will insert the second record, 124; "Jay", in the **LinkedHashMap**. The following process will happen:

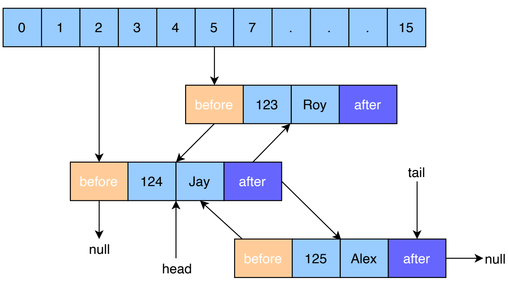
1. The hash of 124 will be calculated and based on the hash value, a bucket will be decided. Let's say the bucket is 2.
2. An Entry object is created with the key as **124** and the value as **Jay**. The *before* field is set to the previous Entry, and the *after* field is set to null.
3. The tail will now point to this entry, and the head will remain unchanged.




## Inserting the third element

Now we will insert the second record, 125; "Alex", in the **LinkedHashMap**. The following process will happen:

1. The hash of 125 will be calculated and based on the hash value, a bucket will be decided. Let's say the bucket is 2 again.
2. An Entry object is created with the key as **125** and the value as **Alex**. The *before* field is set to the previous Entry, and the *after* field is set to null.
3. The tail will now point to this entry, and the head will remain unchanged.




Now it should be clear how a **LinkedHashMap** works. Basically, a doubly **LinkedList** is maintained that keeps track of the insertion order of the elements.