# 自定义哈希表

**Entry**

```java
public class Entry<K, V> {
    final K key;
    V value;
    Entry<K, V> next;

    public Entry(K key, V value, Entry<K, V> next) {
        this.key = key;
        this.value = value;
        this.next = next;
    }

    public final K getKey() {
        return key;
    }

    public final V getValue() {
        return value;
    }

    public final void setValue(V value) {
        this.value = value;
    }

    @Override
    public boolean equals(Object obj) {
        if (!(obj instanceof Entry)) {
            return false;
        }
        Entry e = (Entry) obj;
        Object k1 = getKey();
        Object k2 = e.getKey();
        if (k1 == k2 || (k1 != null && k1.equals(k2))) {
            Object v1 = getValue();
            Object v2 = e.getValue();
            if (v1 == v2 || (v1 != null && v1.equals(v2))) {
                return true;
            }
        }
        return false;
    }

    public final int hashCode() {
        return (key == null ? 0 : key.hashCode()) ^ (value == null ? 0 :
value.hashCode());
    }

    public final String toString() {
        return getKey() + "=" + getValue();
    }
}
```

**MyHashMap**

```
//保证key与value不为空
public class MyHashMap<K, V> {
  private Entry[] table;//Entry数组表
  static final int DEFAULT_INITIAL_CAPACITY = 16;//默认数组长度
  private int size;

  // 构造函数
  public MyHashMap() {
      table = new Entry[DEFAULT_INITIAL_CAPACITY];
      size = DEFAULT_INITIAL_CAPACITY;
  }

  //获取数组长度
  public int getSize() {
      return size;
  }

  // 求index
  static int indexFor(int h, int length) {
      return h % (length - 1);
  }

  //获取元素
  public V get(Object key) {
      if (key == null)
          return null;
      int hash = key.hashCode();// key的哈希值
      int index = indexFor(hash, table.length);// 求key在数组中的下标
      for (Entry<K, V> e = table[index]; e != null; e = e.next) {
          Object k = e.key;
          if (e.key.hashCode() == hash && (k == key || key.equals(k)))
              return e.value;
      }
      return null;
  }

  // 添加元素
  public V put(K key, V value) {
      if (key == null)
          return null;
      int hash = key.hashCode();
      int index = indexFor(hash, table.length);

      // 如果添加的key已经存在，那么只需要修改value值即可
      for (Entry<K, V> e = table[index]; e != null; e = e.next) {
          Object k = e.key;
          if (e.key.hashCode() == hash && (k == key || key.equals(k))) {
              V oldValue = e.value;
              e.value = value;
              return oldValue;// 原来的value值
          }
      }
```

```
        // 如果key值不存在，那么需要添加
        Entry<K, V> e = table[index];// 获取当前数组中的e
        table[index] = new Entry<K, V>(key, value, e);// 新建一个Entry，并将其指向原先的e
        return null;
    }

}
```