**Hashtable**

Hashtable is basically a datastructure to retain values of key-value pair.

* It didn’t allow null for both key and value. You will get NullPointerException if you add null value.
* It is synchronized. So it comes with its cost. Only one thread can access in one time

?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8 | Hashtable<Integer,String>; cityTable = new Hashtable<Integer,String>();  cityTable.put(1, "Lahore");  cityTable.put(2, "Karachi");  cityTable.put(3, null); /\* NullPointerEcxeption at runtime\*/    System.out.println(cityTable.get(1));  System.out.println(cityTable.get(2));  System.out.println(cityTable.get(3));  **HashMap**  Like Hashtable it also accepts key value pair.   * It allows null for both key and value * It is unsynchronized. So come up with better performance   [?](http://www.pakzilla.com/2009/08/24/hashmap-vs-hashtable-vs-hashset/)   |  |  |  |  | | --- | --- | --- | --- | | 2  3 | HashMap<Integer,String> productMap = new HashMap<Integer,String>();  productMap.put(1, "Keys");  productMap.put(2, null);  **HashSet**  HashSet does not allow duplicate values. It provides add method rather put method. You also use its contain method to check whether the object is already available in HashSet. HashSet can be used where you want to maintain a unique list.[?](http://www.pakzilla.com/2009/08/24/hashmap-vs-hashtable-vs-hashset/)   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9 | HashSet<String> stateSet = new HashSet<String>();  stateSet.add ("CA");  stateSet.add ("WI");  stateSet.add ("NY");    if (stateSet.contains("PB")) /\* if CA, it will not add but shows following message\*/       System.out.println("Already found");  else      stateSet.add("PB"); | | |