# [How HashSet Works Internally In Java?](https://javaconceptoftheday.com/how-hashset-works-internally-in-java/)

**HashSet** uses HashMap internally to store it’s objects. Whenever you create a HashSet object, one **HashMap** object associated with it is also created. This HashMap object is used to store the elements you enter in the HashSet. The elements you add into HashSet are stored as **keys** of this HashMap object. The value associated with those keys will be a **constant**.

Every constructor of HashSet class internally creates one HashMap object. You can check this in the source code of HashSet class in JDK installation directory. Below is the some sample code of the constructors of HashSet class.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | private transient HashMap<E,Object> map;    //Constructor - 1    public HashSet()  {          map = new HashMap<>();          //Creating internally backing HashMap object  }    //Constructor - 2    public HashSet(Collection<? extends E> c)  {          map = new HashMap<>(Math.max((int) (c.size()/.75f) + 1, 16));     //Creating internally backing HashMap object          addAll(c);  }    //Constructor - 3    public HashSet(int initialCapacity, float loadFactor)  {          map = new HashMap<>(initialCapacity, loadFactor); //Creating internally backing HashMap object  }    //Constructor - 4    public HashSet(int initialCapacity)  {          map = new HashMap<>(initialCapacity);   //Creating internally backing HashMap object  } |

You can notice that each and every constructor internally creates one new HashMap object.

**How HashSet Works Internally In Java?**

Whenever you insert an element into HashSet using **add()** method, it actually creates an entry in the internally backing HashMap object with element you have specified as it’s key and constant called “**PRESENT**” as it’s value. This “PRESENT” is defined in the HashSet class as below.

|  |  |
| --- | --- |
| 1  2 | // Dummy value to associate with an Object in the backing Map  private static final Object PRESENT = new Object(); |

Let’s have a look at add() method of HashSet class.

|  |  |
| --- | --- |
| 1  2  3  4 | public boolean add(E e)  {          return map.put(e, PRESENT)==null;  } |

You can notice that, add() method of HashSet class internally calls put() method of backing HashMap object by passing the element you have specified as a key and constant “PRESENT” as it’s value.

**remove()** method also works in the same manner.

|  |  |
| --- | --- |
| 1  2  3  4 | public boolean remove(Object o)  {          return map.remove(o)==PRESENT;  } |

Let’s see one example of HashSet and how it maintains HashMap internally.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | public class HashSetExample  {      public static void main(String[] args)      {          //Creating One HashSet object            HashSet<String> set = new HashSet<String>();            //Adding elements to HashSet            set.add("RED");            set.add("GREEN");            set.add("BLUE");            set.add("PINK");            //Removing "RED" from HashSet            set.remove("RED");      }  } |

See the below picture how above program works internally. You can observe that internal HashMap object contains elements of HashSet as keys and constant “PRESENT” as their value.

