The HashMap class uses a hashtable to implement the Map interface. This allows the execution time of basic operations, such as get and put, to remain constant even for large sets.

Below given is the list of constructors supported by the HashMap class.

<table>
<thead>
<tr>
<th>SN</th>
<th>Constructors and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>HashMap</strong></td>
</tr>
<tr>
<td></td>
<td>This constructor constructs a default HashMap.</td>
</tr>
<tr>
<td>2</td>
<td><strong>HashMap</strong> <em>Map</em> <strong>m</strong></td>
</tr>
<tr>
<td></td>
<td>This constructor initializes the hash map by using the elements of the given Map object m.</td>
</tr>
<tr>
<td>3</td>
<td><strong>HashMap</strong> <em>int</em> <strong>capacity</strong></td>
</tr>
<tr>
<td></td>
<td>This constructor initializes the capacity of the hash map to the given integer value, capacity.</td>
</tr>
<tr>
<td>4</td>
<td><strong>HashMap</strong> <em>int</em> <strong>capacity</strong>, <em>float</em> <strong>fillRatio</strong></td>
</tr>
<tr>
<td></td>
<td>This constructor initializes both the capacity and fill ratio of the hash map by using its arguments.</td>
</tr>
</tbody>
</table>

Apart from the methods inherited from its parent classes, HashMap defines the following methods:

<table>
<thead>
<tr>
<th>SN</th>
<th>Methods with Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>void clear</strong></td>
</tr>
<tr>
<td></td>
<td>Removes all mappings from this map.</td>
</tr>
<tr>
<td>2</td>
<td><strong>Object clone</strong></td>
</tr>
<tr>
<td></td>
<td>Returns a shallow copy of this HashMap instance: the keys and values themselves are not cloned.</td>
</tr>
<tr>
<td>3</td>
<td><strong>boolean containsKey</strong> <em>Object</em> <strong>key</strong></td>
</tr>
<tr>
<td></td>
<td>Returns true if this map contains a mapping for the specified key.</td>
</tr>
<tr>
<td>4</td>
<td><strong>boolean containsValue</strong> <em>Object</em> <strong>value</strong></td>
</tr>
<tr>
<td></td>
<td>Returns true if this map maps one or more keys to the specified value.</td>
</tr>
<tr>
<td>5</td>
<td><strong>Set entrySet</strong></td>
</tr>
<tr>
<td></td>
<td>Returns a collection view of the mappings contained in this map.</td>
</tr>
</tbody>
</table>
| 6  | **Object get** _Object_ **key**
Returns the value to which the specified key is mapped in this identity hash map, or null if
the map contains no mapping for this key.

7 boolean isEmpty
Returns true if this map contains no key-value mappings.

8 Set keySet
Returns a set view of the keys contained in this map.

9 Object putObjectkey, Objectvalue
Associates the specified value with the specified key in this map.

10 putAllMapm
Copies all of the mappings from the specified map to this map These mappings will
replace any mappings that this map had for any of the keys currently in the specified map.

11 Object removeObjectkey
Removes the mapping for this key from this map if present.

12 int size
Returns the number of key-value mappings in this map.

13 Collection values
Returns a collection view of the values contained in this map.

Example:
The following program illustrates several of the methods supported by this collection:

```java
import java.util.*;

public class HashMapDemo {
    public static void main(String args[]) {
        // Create a hash map
        HashMap hm = new HashMap();
        // Put elements to the map
        hm.put("Zara", new Double(3434.34));
        hm.put("Mahnaz", new Double(123.22));
        hm.put("Ayan", new Double(1378.00));
        hm.put("Daisy", new Double(99.22));
        hm.put("Qadir", new Double(-19.08));

        // Get a set of the entries
        Set set = hm.entrySet();
        // Get an iterator
        Iterator i = set.iterator();
        // Display elements
        while(i.hasNext()) {
            Map.Entry me = (Map.Entry)i.next();
            System.out.print(me.getKey() + " : ");
```
System.out.println(me.getValue());
}
System.out.println();
// Deposit 1000 into Zara's account
double balance = ((Double)hm.get("Zara")).doubleValue();
hm.put("Zara", new Double(balance + 1000));
System.out.println("Zara's new balance: " + hm.get("Zara"));
}

This would produce the following result:

Zara: 3434.34
ahnaz: 123.22
Daisy: 99.22
Ayan: 1378.0
Qadir: -19.08
Zara's new balance: 4434.34