Java Collection Framework Hashmap

Java HashMap class is an implementation of Map interface based on hash table. It stores elements in key & value pairs which is denoted as HashMap<Key, Value> or HashMap<K, V>.

It extends AbstractMap class, and implements Map interface and can be accessed by importing **java.util** package. Declaration of this class is given below.

HashMap Declaration

```
public class HashMap<K,V>extends AbstractMap<K,V>implements
Map<K,V>,Cloneable, Serializable
```

Copy

Important Points:

- It is member of the Java Collection Framework.
- It uses a hashtable to store the map. This allows the execution time of `get()` and `put()` to remain same.
- HashMap does not maintain order of its element.
- It contains values based on the key.
- It allows only unique keys.
- It is unsynchronized.
- Its initial default capacity is 16.
- It permits null values and the null key
- It is unsynchronized.

HashMap Constructors

HashMap class provides following four constructors.

```
HashMap()

HashMap(Map< ? extends k, ? extends V> m)

HashMap(int capacity)

HashMap(int capacity, float loadfactor)
```

Copy

Example: Creating a HashMap

Lets take an example to create a hashmap that can store integer type key and string values. Initially it is empty because we did not add elements to it. Its elements are enclosed into curly braces.

```
import java.util.*;

class Demo

{

  public static void main(String args[])

  {

  // Creating HashMap

    HashMap<Integer,String> hashMap = new HashMap<Integer,String>();


    // Displaying HashMap

    System.out.println(hashMap);

  }

}
```

Copy

```
{}
```

Adding Elements To HashMap

After creating a hashmap, now lets add elements to it. HashMap provides `put()` method that takes two arguments first is key and second is value. See the below example.

```
import java.util.*;

class Demo

{

  public static void main(String args[])

  {

  // Creating HashMap

    HashMap<Integer,String> hashMap = new HashMap<Integer,String>();

    // Adding elements

    hashMap.put(1, "One");

    hashMap.put(2, "Two");
    hashMap.put(3, "Three");

    hashMap.put(4, "Four");

    // Displaying HashMap

    System.out.println(hashMap);

  }

}
```

Copy

```
{1=One, 2=Two, 3=Three, 4=Four}
```

Removing Elements From HashMap

In case, we need to remove any element from the hashmap. We can use remove() method that takes key as an argument. it has one overloaded remove() method that takes two arguments first is key and second is value.

```
import java.util.*;

class Demo

{

  public static void main(String args[])

  {

  // Creating HashMap

    HashMap<Integer,String> hashMap = new HashMap<Integer,String>();

    // Adding elements

    hashMap.put(1, "One");

    hashMap.put(2, "Two");

    hashMap.put(3, "Three");

    hashMap.put(4, "Four");

    // Displaying HashMap

    System.out.println(hashMap);

    // Remove element by key

    hashMap.remove(2);

    System.out.println("After Removing 2 :\n"+hashMap);
    // Remove by key and value

    hashMap.remove(3, "Three");

    System.out.println("After Removing 3 :\n"+hashMap);


  }

}
```

Copy

```
{1=One, 2=Two, 3=Three, 4=Four}

After Removing 2 :

{1=One, 3=Three, 4=Four}

After Removing 3 :

{1=One, 4=Four}
```

Traversing Elements

To access elements of the hashmap, we can traverse them using the loop. In this example, we are using for loop to iterate the elements.

```
  import java.util.*;

class Demo

{

  public static void main(String args[])

  {

  // Creating HashMap

    HashMap<Integer,String> hashMap = new HashMap<Integer,String>();

    // Adding elements

    hashMap.put(1, "One");

    hashMap.put(2, "Two");

    hashMap.put(3, "Three");

    hashMap.put(4, "Four");

    // Traversing HashMap

    for(Map.Entry<Integer, String> entry : hashMap.entrySet()) {

      System.out.println(entry.getKey()+" : "+entry.getValue());
    }

  }

}
```

Copy

```
1 : One

2 : Two

3 : Three

4 : Four
```

Replace HashMap Elements

HashMap provides built-in methods to replace elements. There are two overloaded replace methods: first takes two arguments one for key and second for the value we want to replace with. Second method takes three arguments first is key and second is value associated with the key and third is value that we want to replace with the key-value.

```
import java.util.*;

class Demo

{

  public static void main(String args[])

  {

  // Creating HashMap

    HashMap<Integer,String> hashMap = new HashMap<Integer,String>();

    // Adding Elements

    hashMap.put(1, "One");

    hashMap.put(2, "Two");

    hashMap.put(3, "Three");

    hashMap.put(4, "Four");

    // Traversing HashMap

    for(Map.Entry<Integer, String> entry : hashMap.entrySet()) {

      System.out.println(entry.getKey()+" : "+entry.getValue());

    }

    // Replacing Elements of HashMap
    hashMap.replace(1, "One-1");

    System.out.println(hashMap);

    hashMap.replace(1, "One-1", "First");

    System.out.println(hashMap);

  }

}
```

Copy

```
OUTPUT

1 : One

2 : Two

3 : Three

4 : Four

{1=One-1, 2=Two, 3=Three, 4=Four}

{1=First, 2=Two, 3=Three, 4=Four}
```