Java HashMap class



Java HashMap class implements the map interface by using a hash table. It inherits AbstractMap class and implements Map interface.

Points to remember

* Java HashMap class contains values based on the key.
* Java HashMap class contains only unique keys.
* Java HashMap class may have one null key and multiple null values.
* Java HashMap class is non synchronized.
* Java HashMap class maintains no order.
* The initial default capacity of Java HashMap class is 16 with a load factor of 0.75.

Hierarchy of HashMap class

As shown in the above figure, HashMap class extends AbstractMap class and implements Map interface.

HashMap class declaration

Let's see the declaration for java.util.HashMap class.

1. **public** **class** HashMap<K,V> **extends** AbstractMap<K,V> **implements** Map<K,V>, Cloneable, Serializable

HashMap class Parameters

Let's see the Parameters for java.util.HashMap class.

* **K**: It is the type of keys maintained by this map.
* **V**: It is the type of mapped values.

Constructors of Java HashMap class

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| HashMap() | It is used to  construct a  default HashMap. |
| HashMap(Map<? extends K,? extends V> m) | It is used to  initialize the  hash map by  using  the elements  of the given  Map object m. |
| HashMap(int capacity) | It is used to  initializes  the capacity  of the hash map  to the given integer value,  capacity. |
| HashMap(int capacity, float loadFactor) | It is used to initialize  both the capacity and load  factor of the hash map by using  its arguments. |

Methods of Java HashMap class

|  |  |
| --- | --- |
| **Method** | **Description** |
| void clear() | It is used to remove all of the mappings from this map. |
| boolean isEmpty() | It is used to return true if this map contains no key-value mappings. |
| Object clone() | It is used to return a shallow copy of this HashMap instance: the keys and values themselves are not cloned. |
| Set entrySet() | It is used to return a collection view of the  mappings contained in this map. |
| Set keySet() | It is used to return a set view of the keys  contained in this map. |
| V put(Object key, Object value) | It is used to insert an entry in the map. |
| void putAll(Map map) | It is used to insert the specified map in the  map. |
| V putIfAbsent(K key, V value) | It inserts the specified value with the  specified key in the map only if it is not  already specified. |
| V remove(Object key) | It is used to delete an entry for the  specified key. |
| boolean remove(Object key, Object value) | It removes the specified values with the  associated specified keys from the map. |
| V compute(K key, BiFunction<? super K,? super V,? extends V> remappingFunction) | It is used to compute a mapping for the  specified key and its current mapped value  (or null if there is no current mapping). |
| V computeIfAbsent(K key, Function<? super K,? extends V> mappingFunction) | It is used to compute its value using the  given mapping function, if the specified  key is not already associated with a  value (or is mapped to null), and enters  it into this map unless null. |
| V computeIfPresent(K key, BiFunction<? super K,? super V,? extends V> remappingFunction) | It is used to compute a new mapping  given the key and its current mapped  value if the value for the specified key is  present and non-null. |
| boolean containsValue(Object value) | This method returns true if some value  equal to the value exists within the  map, else return false. |
| boolean containsKey(Object key) | This method returns true if some key  equal to the key exists within the map,  else return false. |
| boolean equals(Object o) | It is used to compare the specified  Object with the Map. |
| void forEach(BiConsumer<? super K,? super V> action) | It performs the given action for each  entry in the map until all entries have  been processed or the action throws an  exception. |
| V get(Object key) | This method returns the object that  contains the value associated with the  key. |
| V getOrDefault(Object key, V defaultValue) | It returns the value to which the  specified key is mapped, or defaultValue  if the map contains no mapping for the  key. |
| boolean isEmpty() | This method returns true if the map is  empty; returns false if it contains at  least one key. |
| V merge(K key, V value, BiFunction<? super V,? super V,? extends V> remappingFunction) | If the specified key is not already  associated with a value or is associated  with null, associates it with the given  non-null value. |
| V replace(K key, V value) | It replaces the specified value for a  specified key. |
| boolean replace(K key, V oldValue, V newValue) | It replaces the old value with the new  value for a specified key. |
| void replaceAll(BiFunction<? super K,? super V,? extends V> function) | It replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception. |
| Collection<V> values() | It returns a collection view  of the values contained in the map. |
| int size() | This method returns the number  of entries in the map. |

Java HashMap example to add() elements

Here, we see different ways to insert elements.

1. **import** java.util.\*;
2. **class** HashMap1{
3. **public** **static** **void** main(String args[]){
4. HashMap<Integer,String> hm=**new** HashMap<Integer,String>();
5. System.out.println("Initial list of elements: "+hm);
6. hm.put(100,"Amit");
7. hm.put(101,"Vijay");
8. hm.put(102,"Rahul");
10. System.out.println("After invoking put() method ");
11. **for**(Map.Entry m:hm.entrySet()){
12. System.out.println(m.getKey()+" "+m.getValue());
13. }
15. hm.putIfAbsent(103, "Gaurav");
16. System.out.println("After invoking putIfAbsent() method ");
17. **for**(Map.Entry m:hm.entrySet()){
18. System.out.println(m.getKey()+" "+m.getValue());
19. }
20. HashMap<Integer,String> map=**new** HashMap<Integer,String>();
21. map.put(104,"Ravi");
22. map.putAll(hm);
23. System.out.println("After invoking putAll() method ");
24. **for**(Map.Entry m:map.entrySet()){
25. System.out.println(m.getKey()+" "+m.getValue());
26. }
27. }
28. }

Initial list of elements: {}

After invoking put() method

100 Amit

101 Vijay

102 Rahul

After invoking putIfAbsent() method

100 Amit

101 Vijay

102 Rahul

103 Gaurav

After invoking putAll() method

100 Amit

101 Vijay

102 Rahul

103 Gaurav

104 Ravi

Java HashMap example to remove() elements

Here, we see different ways to remove elements.

1. **import** java.util.\*;
2. **public** **class** HashMap2 {
3. **public** **static** **void** main(String args[]) {
4. HashMap<Integer,String> map=**new** HashMap<Integer,String>();
5. map.put(100,"Amit");
6. map.put(101,"Vijay");
7. map.put(102,"Rahul");
8. map.put(103, "Gaurav");
9. System.out.println("Initial list of elements: "+map);
10. //key-based removal
11. map.remove(100);
12. System.out.println("Updated list of elements: "+map);
13. //value-based removal
14. map.remove(101);
15. System.out.println("Updated list of elements: "+map);
16. //key-value pair based removal
17. map.remove(102, "Rahul");
18. System.out.println("Updated list of elements: "+map);
19. }
20. }

Output:

Initial list of elements: {100=Amit, 101=Vijay, 102=Rahul, 103=Gaurav}

Updated list of elements: {101=Vijay, 102=Rahul, 103=Gaurav}

Updated list of elements: {102=Rahul, 103=Gaurav}

Updated list of elements: {103=Gaurav}

Java HashMap example to replace() elements

Here, we see different ways to replace elements.

1. **import** java.util.\*;
2. **class** HashMap3{
3. **public** **static** **void** main(String args[]){
4. HashMap<Integer,String> hm=**new** HashMap<Integer,String>();
5. hm.put(100,"Amit");
6. hm.put(101,"Vijay");
7. hm.put(102,"Rahul");
8. System.out.println("Initial list of elements:");
9. **for**(Map.Entry m:hm.entrySet())
10. {
11. System.out.println(m.getKey()+" "+m.getValue());
12. }
13. System.out.println("Updated list of elements:");
14. hm.replace(102, "Gaurav");
15. **for**(Map.Entry m:hm.entrySet())
16. {
17. System.out.println(m.getKey()+" "+m.getValue());
18. }
19. System.out.println("Updated list of elements:");
20. hm.replace(101, "Vijay", "Ravi");
21. **for**(Map.Entry m:hm.entrySet())
22. {
23. System.out.println(m.getKey()+" "+m.getValue());
24. }
25. System.out.println("Updated list of elements:");
26. hm.replaceAll((k,v) -> "Ajay");
27. **for**(Map.Entry m:hm.entrySet())
28. {
29. System.out.println(m.getKey()+" "+m.getValue());
30. }
31. }
32. }

Initial list of elements:

100 Amit

101 Vijay

102 Rahul

Updated list of elements:

100 Amit

101 Vijay

102 Gaurav

Updated list of elements:

100 Amit

101 Ravi

102 Gaurav

Updated list of elements:

100 Ajay

101 Ajay

102 Ajay

Difference between HashSet and HashMap

HashSet contains only values whereas HashMap contains an entry(key and value).

Java HashMap Example: Book

1. **import** java.util.\*;
2. **class** Book {
3. **int** id;
4. String name,author,publisher;
5. **int** quantity;
6. **public** Book(**int** id, String name, String author, String publisher, **int** quantity) {
7. **this**.id = id;
8. **this**.name = name;
9. **this**.author = author;
10. **this**.publisher = publisher;
11. **this**.quantity = quantity;
12. }
13. }
14. **public** **class** MapExample {
15. **public** **static** **void** main(String[] args) {
16. //Creating map of Books
17. Map<Integer,Book> map=**new** HashMap<Integer,Book>();
18. //Creating Books
19. Book b1=**new** Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
20. Book b2=**new** Book(102,"Data Communications & Networking","Forouzan","Mc Graw Hill",4);
21. Book b3=**new** Book(103,"Operating System","Galvin","Wiley",6);
22. //Adding Books to map
23. map.put(1,b1);
24. map.put(2,b2);
25. map.put(3,b3);
27. //Traversing map
28. **for**(Map.Entry<Integer, Book> entry:map.entrySet()){
29. **int** key=entry.getKey();
30. Book b=entry.getValue();
31. System.out.println(key+" Details:");
32. System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
33. }
34. }
35. }

Output:

1 Details:

101 Let us C Yashwant Kanetkar BPB 8

2 Details:

102 Data Communications & Networking Forouzan Mc Graw Hill 4

3 Details:

103 Operating System Galvin Wiley 6

Next To