

Set Custom implementation in java - How HashSet works internally with diagrams and full program

You are here : [Home](#) / [Core Java Tutorials](#) / [Data structures](#) / [Collection framework](#)

Contents of page :

- 1) [Methods used in custom HashMap >](#)
- 2) [Let's find out answer of few very important questions before proceeding >](#)
- 3) [Full Program/SourceCode for implementing custom HashSet >](#)

HashSet

In this post i will be explaining [HashSet custom implementation](#).

1) Methods used in custom HashMap >

public void add (E value)	Add objects in setCustom
public boolean contains (E obj)	Method returns true if setCustom contains the object.
public boolean remove (E obj)	Method removes object from setCustom .
public void display ()	-Method displays all objects in setCustom. - Insertion order is not guaranteed , for maintaining insertion order refer LinkedHashSet

Must read: Find single LinkedList is circular or not

Reverse words in sentence.

2) Let's find out answer of few very important questions before proceeding >

Q1. How HashSet implements **hashing**?

A. Method internally uses [HashMap's](#) hash method for hasihng.

Q2. How **add method** works internally?

```
A. public void add(E value){
    hashMapCustom.put(value, null);
}
```

Method internally uses HashMap's put method for storing object.

Q3. How **contains method** works internally?

```
A. public boolean contains(E obj){
    return hashMapCustom.contains(obj) !=null ? true :false;
}
```

Method internally uses [HashMap's](#) contains method for storing object.

Q4. How **remove method** works internally?

```
A. public boolean remove(E obj){
    return hashMapCustom.remove(obj);
}
```

Method internally uses HashMap's put remove for storing object.

REFER: [Set Custom implementation - add, contains, remove Employee object](#).

3) Full Program/SourceCode for implementing custom HashSet >

```
package com.ankit;

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
/**
 * @author AnkitMittal
 * Copyright (c), AnkitMittal . All Contents are copyrighted and must not be reproduced in any form.
 * This class provides custom implementation of HashSet(without using java api's- we will be using HashMapCustom)- which allows does not allow you to store duplicate values.
 * Note- implementation does not allow you to store null values.
 * does not maintain insertion order.
 * @param <K>
 * @param <V>
 */
class HashSetCustom<E>{

    private HashMapCustom<E, Object> hashMapCustom;

    public HashSetCustom(){
        hashMapCustom=new HashMapCustom<>();
    }

    /**
     * add objects in SetCustom.
     */
    public void add(E value){
        hashMapCustom.put(value, null);
    }

    /**
     * Method returns true if set contains the object.
     * @param key
     */
    public boolean contains(E obj){
        return hashMapCustom.contains(obj) !=null ? true :false;
    }

    /**
     * Method displays all objects in setCustom.
     * insertion order is not guaranteed, for maintaining insertion order refer LinkedHashSet.
     */
    public void display(){
        hashMapCustom.displaySet();
    }

    /**
     * Method removes object from setCustom.
     * @param obj
     */
    public boolean remove(E obj){
        return hashMapCustom.remove(obj);
    }
}

/**
 * @author AnkitMittal
 * Copyright (c), AnkitMittal . All Contents are copyrighted and must not be reproduced in any form.
 * This class provides custom implementation of HashMap(without using java api's)- which allows us to store data in key-value pair form..
 * @param <K>
 * @param <V>
 */
class HashMapCustom<K, V> {

    private Entry<K,V>[] table; //Array of Entry.
    private int capacity= 4; //Initial capacity of HashMap

    static class Entry<K, V> {
        K key;
        V value;
        Entry<K,V> next;

        public Entry(K key, V value, Entry<K,V> next){
            this.key = key;
            this.value = value;
            this.next = next;
        }
    }

    @SuppressWarnings("unchecked")
    public HashMapCustom(){
        table = new Entry[capacity];
    }

    /**
     * Method allows you put key-value pair in HashMapCustom.
     * If the map already contains a mapping for the key, the old value is replaced.
     * Note: method does not allows you to put null key thought it allows null values.
     * Implementation allows you to put custom objects as a key as well.
     * Key Features: implementation provides you with following features:-
     * >provide complete functionality how to override equals method.
     * >provide complete functionality how to override hashCode method.
     * @param newKey
     * @param data
     */
    public void put(K newKey, V data){
        if(newKey==null)
            return; //does not allow to store null.

        int hash=hash(newKey);
        Entry<K,V> newEntry = new Entry<K,V>(newKey, data, null);

        if(table[hash] == null){
            table[hash] = newEntry;
        }else{
            Entry<K,V> previous = null;
            Entry<K,V> current = table[hash];

            while(current != null){ //we have reached last entry of bucket.
                if(current.key.equals(newKey)){
                    if(previous==null){ //node has to be insert on first of bucket.
                        newEntry.next=current.next;
                        table[hash]=newEntry;
                        return;
                    }
                    else{
                        newEntry.next=current.next;
                        previous.next=newEntry;
                        return;
                    }
                }
                previous=current;
                current = current.next;
            }
            previous.next = newEntry;
        }
    }

    /**
     * Method returns value corresponding to key.
     * @param key
     */
    public V get(K key){
        int hash = hash(key);
        if(table[hash] == null){
            return null;
        }else{
            Entry<K,V> temp = table[hash];
            while(temp!= null){
                if(temp.key.equals(key))
                    return temp.value;
                temp = temp.next; //return value corresponding to key.
            }
            return null; //returns null if key is not found.
        }
    }

    /**
     * Method removes key-value pair from HashMapCustom.
     * @param key
     */
    public boolean remove(K deleteKey){

        int hash=hash(deleteKey);

        if(table[hash] == null){
            return false;
        }else{
            Entry<K,V> previous = null;
            Entry<K,V> current = table[hash];

            while(current != null){ //we have reached last entry node of bucket.
                if(current.key.equals(deleteKey)){
                    if(previous==null){ //delete first entry node.
                        table[hash]=table[hash].next;
                        return true;
                    }
                    else{
                        previous.next=current.next;
                        return true;
                    }
                }
                previous=current;
                current = current.next;
            }
            return false;
        }
    }

    /**
     * Method displays all key-value pairs present in HashMapCustom.,
     * insertion order is not guaranteed, for maintaining insertion order refer LinkedHashMapCustom.
     * @param key
     */
    public void display(){

        for(int i=0;i<capacity;i++){
            if(table[i]!=null){
                Entry<K, V> entry=table[i];
                while(entry!=null){
                    System.out.print("{ "+entry.key+"="+entry.value+"} " + " ");
                    entry=entry.next;
                }
            }
        }
    }

    /**
     * Method returns null if set does not contain object.
     * @param key
     */
    public K contains(K key){
        int hash = hash(key);
        if(table[hash] == null){
            return null;
        }else{
            Entry<K,V> temp = table[hash];
            while(temp!= null){
                if(temp.key.equals(key))
                    return key;
                temp = temp.next; //return value corresponding to key.
            }
            return null; //returns null if key is not found.
        }
    }

    /**
     * Method displays all objects in setCustom.
     * insertion order is not guaranteed, for maintaining insertion order refer LinkedHashSet.
     */
    public void displaySet(){

        for(int i=0;i<capacity;i++){
            if(table[i]!=null){
                Entry<K, V> entry=table[i];
                while(entry!=null){
                    System.out.print(entry.key+" ");
                    entry=entry.next;
                }
            }
        }
    }

    /**
     * Method implements hashing functionality, which helps in finding the appropriate bucket location to store our data.
     * This is very important method, as performance of HashMapCustom is very much dependent on this method's implementation.
     * @param key
     */
    private int hash(K key){
        return Math.abs(key.hashCode()) % capacity;
    }
}

/**
 * Main class- to test HashMap functionality.
 */
public class HashSetCustomApp {

    public static void main(String[] args) {
        HashSetCustom<Integer> hashSetCustom = new HashSetCustom<Integer>();
        hashSetCustom.add(21);
        hashSetCustom.add(25);
        hashSetCustom.add(30);
        hashSetCustom.add(33);
        hashSetCustom.add(35);

        System.out.println("HashSetCustom contains 21 =" +hashSetCustom.contains(21));
        System.out.println("HashSetCustom contains 51 =" +hashSetCustom.contains(51));

        System.out.print("Displaying HashSetCustom: ");
        hashSetCustom.display();

        System.out.println("\n\n21 removed: " +hashSetCustom.remove(21));
        System.out.println("22 removed: " +hashSetCustom.remove(22));

        System.out.print("Displaying HashSetCustom: ");
        hashSetCustom.display();
    }
}

/*Output

HashSetCustom contains 21 =true
HashSetCustom contains 51 =false
Displaying HashSetCustom: 21 25 33 30 35

21 removed: true
22 removed: false
Displaying HashSetCustom: 25 33 30 35

*/
```

Summary of article >

In this tutorial we learned how to create and implement own/custom [Set in java](#) with full program, diagram and examples to insert and retrieve values in it.