JavaMadeSoEasy.com (JMSE)

Home Core Java Tutorials Interview Questions Interview Programs Custom Quiz Database More About

- Algorithm
- Apache Tomcat Server
- Collection Framework
- Collection programs
- Collections implementation
- Core Java
- Core Java Differences
- Core Java Tutorials
- Custom implementation of Data Structures
- Data Structure
- Date tutorials
- eclipse
- Exceptions
- File IO/File handling
- Garbage collection in java

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 :

Search This Blog

Search

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.

*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.

- OutOfMemoryError
- Overriding equals and hashCode
- PostgreSQL
- Producer Consumer problem/pattern
- Pyramid generation
- RegEx
- Serialization
- Thread Concurrency
- Threads
- TUTORIALS
- Windows

Join our Groups>

- FACEBOOK
- LINKED IN

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;
```

Subscribe-Our exclusive posts for FREE. Stay on top! Join 2693+ subscribers

Email address.. Submit

All Labels

Algorithm(34) Apache Tomcat Server(4) Arrays Java(30) Autoboxing(5) Basic java programs for beginners(15) Binary Trees(6) Collection Framework(68) Collection programs(105) Collections implementation(16) Comparable and Comparator program(22) Core Java(1032) core java Basics(38) Core java Conversions(21) Core Java Differences(11) Core Java

```
/** 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.
   */
```

Tutorials(12) CRUD MongoDB java(5) CRUD operations MongoDB(3) Custom implementation of Data Structures(11) Data Structure(26) Database(1) Database Tutorials(3) Date tutorials(11) DeSerialization(19) eclipse(27) Exceptions(71) File IO/File handling(71) Garbage collection in java(43) Generics(5) Hashing(8) Hibernate(1) Interview questions(14) iText Pdf tutorial(45) Java 4(1) java 5(3) Java 7(32) Java 8(80) Java 9(12) Java keywords(11) Java Mcq(Multiple choice questions)(5) Java Programs(8) Java QUIZ(7) JavaScript(2) JDBC(74) JUNIT(7) JVM(22) Level1 programs (beginners)(34) Level2 programs (intermediate)(23) Level3 programs (advanced)(13) Linux(6) Matrix programs(10) MongoDB(87) MongoDB Java(20) MsSql(1) MultiThreading(97) MySql(7) Notepad++(6) Oracle(18) OutOfMemoryError(13) Overriding equals and hashCode(11) Pattern generating(14) PostgreSQL(1) Producer Consumer problem/pattern(10) Pyramid generation(14) Recursion(10) RegEx(7) Serialization(20) Serialization top interview questions and answers in java(18) Thread Concurrency(35) Threads(73) TUTORIALS(22) Windows(8)

Popular Posts of JavaMadeSoEasy

```
    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;
        }
```

- HashMap Custom implementation in java - How HashMap works internally with diagrams and full program
- Collection Quiz in Java - MCQ - Multiple choice questions
- CORE JAVA - Top 120 most interesting and important interview questions and answers in core java
- THREADS - Top 80 interview questions and answers in java for freshers and experienced(detailed explanation with programs) Set-1 > Q1- Q60
- Core Java Tutorial in detail with diagram and programs - BEST EXPLANATION EVER
- COLLECTION - Top 100 interview questions and answers in java for fresher and experienced in detail - Set-1 > Q1- Q50
- Find sum of both diagonals in matrix - program in java
- LinkedHashMap Custom implementation in java - How LinkedHashMap works internally with diagrams and full program
- Thread/multi threading Quiz in Java - MCQ - Multiple choice questions
- EXCEPTIONS - Top 60 interview questions and answers in java for fresher and experienced - detailed explanation with diagrams Set-1 > Q1- Q25

```
}


@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;
```

JavaMadeSoeasy Archive

- ► 2018 (17)
- ► 2017 (211)
- ► 2016 (240)

```
                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
     */
```

- ▶ 2015 (722)
  - ▶ December (61)
  - ▶ November (96)
  - ▶ October (14)
  - ▶ September (13)
  - ▶ August (73)
  - ▶ July (54)
  - ▶ June (29)
  - ▶ May (53)
  - ▶ April (109)
  - ▶ March (93)
  - ▶ February (99)
  - ▶ January (28)

```
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

*/
```

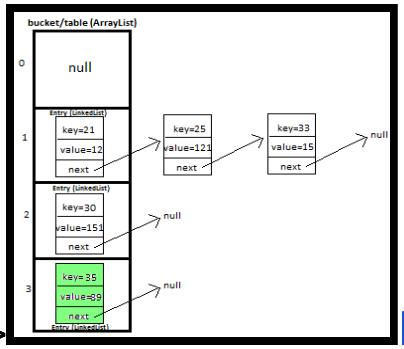
Summary of article >

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.

Having any doubt? or you you liked the tutorial! Please comment in below section.

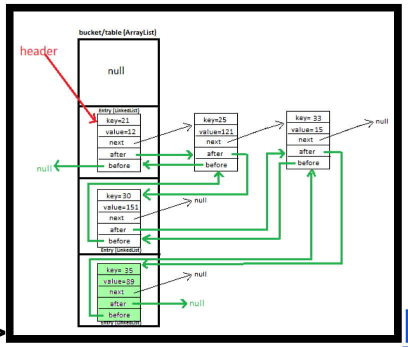
*Please express your love by liking **JavaMadeSoEasy.com** (**JMSE**) on* ***facebook****, following on* ***google+*** *or* *Twitter*.

RELATED LINKS>

1 > 

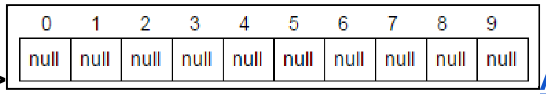
HashMap Custom implementation in java/ Develop own HashMap with full java code

>HashMap Custom implementation in java - put, get, remove Employee object

2 > 

LinkedHashMap own/Custom implementation/ with full code in java

>LinkedHashMap Custom implementation - put, get, remove Employee object

3 > 

ArrayList custom implementation in java/ Create own ArrayList with full code

>ArrayList custom implementation - add, get, remove Employee object

4 >

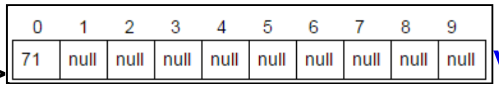
Set Custom implementation in java

>Set Custom implementation - add, contains, remove Employee object

5 >

LinkedHashSet Custom implementation in java

>LinkedHashSet Custom implementation - add, contains, remove Employee object

6 > 

Vector custom implementation in java

Labels: Algorithm Collections implementation Core Java Custom implementation of Data Structures Hashing How set works internally

Map Overriding equals and hashCode Set Single Linked List

Must read for you :

1 Comment www.javamadesoeasy.com 1 Login

Recommend Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Elfreda Huffer • 2 years ago

It was great to stumble upon this post. Thanks for the info, super helpful. So here is my saving grace: PDFfiller helped me to fill out the a form and and esign them. Just try it `http://goo.gl/5a7hi8`, you'll love it.

• Reply • Share ›

ALSO ON WWW.JAVAMADESOEASY.COM

JavaMadeSoEasy.com: what is the default initial capacity of ARRAYLIST, how it is …
2 comments • 3 years ago
Ankit Mittal — Hi @sudhir,Thanks for comment,When, new ArrayList<integer>() is executed, Size of ArrayList is 0.Internally, …

JavaMadeSoEasy.com (JMSE): Exception in thread java.lang.OutOfMemoryError: Java …
1 comment • a year ago
Rahul Dhawan — What all can be taken as prevention measures for OutOfMemoryException?

JavaMadeSoEasy.com: How to convert String to long and long to String in java
1 comment • 2 years ago
dhiyamaddy — Those programming was very easily understand and more important coding are provided on this post and this is very valuable …

JavaMadeSoEasy.com: Connection Pooling in java with example
1 comment • 3 years ago
Kevin Chen — I need some help with your code. I noticed that once all the available connections are used, the code get stuck on the "wait()" in …

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

Newer Post **Home** **Older Post**