Java HashSet



Java HashSet class is used to create a collection that uses a hash table for storage. It inherits the AbstractSet class and implements Set interface.

The important points about Java HashSet class are:

* HashSet stores the elements by using a mechanism called **hashing.**
* HashSet contains unique elements only.
* HashSet allows null value.
* HashSet is the best approach for search operations.
* The initial default capacity of HashSet is 16, and the load factor is 0.75.

Difference between List and Set

A list can contain duplicate elements whereas Set contains unique elements only.

Hierarchy of HashSet class

The HashSet class extends AbstractSet class which implements Set interface. The Set interface inherits Collection and Iterable interfaces in hierarchical order.

Play Video

HashSet class declaration

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

1. **public** **class** HashSet<E> **extends** AbstractSet<E> **implements** Set<E>, Cloneable, Serializable

Constructors of Java HashSet class

|  |  |  |
| --- | --- | --- |
| **SN** | **Constructor** | **Description** |
| 1) | HashSet() | It is used to construct a default HashSet. |
| 2) | HashSet(int capacity) | It is used to initialize the capacity of the hash set to the given integer value capacity. The capacity grows automatically as elements are added to the HashSet. |
| 3) | HashSet(int capacity, float loadFactor) | It is used to initialize the capacity of the hash set to the given integer value capacity and the specified load factor. |
| 4) | HashSet(Collection<? extends E> c) | It is used to initialize the hash set by using the elements of the collection c. |

Methods of Java HashSet class

Various methods of Java HashSet class are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Modifier & Type** | **Method** | **Description** |
| 1) | boolean | [add(E e)](https://www.javatpoint.com/java-hashset-add-method) | It is used to add the specified element to this set if it is not already present. |
| 2) | void | [clear()](https://www.javatpoint.com/java-hashset-clear-method) | It is used to remove all of the elements from the set. |
| 3) | object | [clone()](https://www.javatpoint.com/java-hashset-clone-method) | It is used to return a shallow copy of this HashSet instance: the elements themselves are not cloned. |
| 4) | boolean | [contains(Object o)](https://www.javatpoint.com/java-hashset-contains-method) | It is used to return true if this set contains the specified element. |
| 5) | boolean | [isEmpty()](https://www.javatpoint.com/java-hashset-isempty-method) | It is used to return true if this set contains no elements. |
| 6) | Iterator<E> | [iterator()](https://www.javatpoint.com/java-hashset-iterator-method) | It is used to return an iterator over the elements in this set. |
| 7) | boolean | [remove(Object o)](https://www.javatpoint.com/java-hashset-remove-method) | It is used to remove the specified element from this set if it is present. |
| 8) | int | [size()](https://www.javatpoint.com/java-hashset-size-method) | It is used to return the number of elements in the set. |
| 9) | Spliterator<E> | [spliterator()](https://www.javatpoint.com/java-hashset-spliterator-method) | It is used to create a late-binding and fail-fast Spliterator over the elements in the set. |

Java HashSet Example

Let's see a simple example of HashSet. Notice, the elements iterate in an unordered collection.

1. **import** java.util.\*;
2. **class** HashSet1{
3. **public** **static** **void** main(String args[]){
4. //Creating HashSet and adding elements
5. HashSet<String> set=**new** HashSet();
6. set.add("One");
7. set.add("Two");
8. set.add("Three");
9. set.add("Four");
10. set.add("Five");
11. Iterator<String> i=set.iterator();
12. **while**(i.hasNext())
13. {
14. System.out.println(i.next());
15. }
16. }
17. }

Five

One

Four

Two

Three

Java HashSet example ignoring duplicate elements

In this example, we see that HashSet doesn't allow duplicate elements.

1. **import** java.util.\*;
2. **class** HashSet2{
3. **public** **static** **void** main(String args[]){
4. //Creating HashSet and adding elements
5. HashSet<String> set=**new** HashSet<String>();
6. set.add("Ravi");
7. set.add("Vijay");
8. set.add("Ravi");
9. set.add("Ajay");
10. //Traversing elements
11. Iterator<String> itr=set.iterator();
12. **while**(itr.hasNext()){
13. System.out.println(itr.next());
14. }
15. }
16. }

Ajay

Vijay

Ravi

Java HashSet example to remove elements

Here, we see different ways to remove an element.

1. **import** java.util.\*;
2. **class** HashSet3{
3. **public** **static** **void** main(String args[]){
4. HashSet<String> set=**new** HashSet<String>();
5. set.add("Ravi");
6. set.add("Vijay");
7. set.add("Arun");
8. set.add("Sumit");
9. System.out.println("An initial list of elements: "+set);
10. //Removing specific element from HashSet
11. set.remove("Ravi");
12. System.out.println("After invoking remove(object) method: "+set);
13. HashSet<String> set1=**new** HashSet<String>();
14. set1.add("Ajay");
15. set1.add("Gaurav");
16. set.addAll(set1);
17. System.out.println("Updated List: "+set);
18. //Removing all the new elements from HashSet
19. set.removeAll(set1);
20. System.out.println("After invoking removeAll() method: "+set);
21. //Removing elements on the basis of specified condition
22. set.removeIf(str->str.contains("Vijay"));
23. System.out.println("After invoking removeIf() method: "+set);
24. //Removing all the elements available in the set
25. set.clear();
26. System.out.println("After invoking clear() method: "+set);
27. }
28. }

An initial list of elements: [Vijay, Ravi, Arun, Sumit]

After invoking remove(object) method: [Vijay, Arun, Sumit]

Updated List: [Vijay, Arun, Gaurav, Sumit, Ajay]

After invoking removeAll() method: [Vijay, Arun, Sumit]

After invoking removeIf() method: [Arun, Sumit]

After invoking clear() method: []

Java HashSet from another Collection

1. **import** java.util.\*;
2. **class** HashSet4{
3. **public** **static** **void** main(String args[]){
4. ArrayList<String> list=**new** ArrayList<String>();
5. list.add("Ravi");
6. list.add("Vijay");
7. list.add("Ajay");
9. HashSet<String> set=**new** HashSet(list);
10. set.add("Gaurav");
11. Iterator<String> i=set.iterator();
12. **while**(i.hasNext())
13. {
14. System.out.println(i.next());
15. }
16. }
17. }

Vijay

Ravi

Gaurav

Ajay

Java HashSet Example: Book

Let's see a HashSet example where we are adding books to set and printing all the books.

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** HashSetExample {
15. **public** **static** **void** main(String[] args) {
16. HashSet<Book> set=**new** HashSet<Book>();
17. //Creating Books
18. Book b1=**new** Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
19. Book b2=**new** Book(102,"Data Communications & Networking","Forouzan","Mc Graw Hill",4);
20. Book b3=**new** Book(103,"Operating System","Galvin","Wiley",6);
21. //Adding Books to HashSet
22. set.add(b1);
23. set.add(b2);
24. set.add(b3);
25. //Traversing HashSet
26. **for**(Book b:set){
27. System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
28. }
29. }
30. }

Output:

101 Let us C Yashwant Kanetkar BPB 8

102 Data Communications & Networking Forouzan Mc Graw Hill 4

103 Operating System Galvin Wiley 6