Some of the important points about HashSet in java are;

1. HashSet doesn’t allow duplicate entries.
2. HashSet allows null as a value.
3. HashSet doesn’t guarantee the insertion order of elements.
4. HashSet is not thread-safe. You can get thread-safe HashSet using Collections.synchronizedSetmethod at the cost of performance. You can also use CopyOnWriteArraySet [concurrency](https://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers) class for thread safety.
5. HashSet iterator methods are fail-fast. So any structural modification to the set after creation of iterator will throw ConcurrentModificationException.
6. HashSet supports Generics, this is the recommended approach to avoid ClassCastException at runtime.
7. HashSet uses [HashMap](https://www.journaldev.com/11560/java-hashmap) for storing elements, so the objects should provide good implementation of hashCode() and equals() method to avoid unwanted results.

**Java HashSet Constructors**

Java HashSet provides four constructors.

1. **public HashSet()**: Creates a new empty HashSet, the backing HashMap is initialized with default initial capacity as 16 and load factor 0.75.
2. **public HashSet(int initialCapacity)**: Creates a empty HashSet with backing HashMap initialized with specified capacity and load factor 0.75.
3. **public HashSet(int initialCapacity, float loadFactor)**: Creates a empty HashSet with backing HashMap initialized with specified capacity and specified load factor.
4. **public HashSet(Collection<? extends E> c)**: Creates a new Set containing the elements in the specified collection. The backing HashMap is created with default load factor (0.75) and an initial capacity sufficient enough to contain all the elements in the specified collection.

Below code snippet is showing all these HashSet constructors example usage.

**Copy**

**Set<String> set = new HashSet<>();**

**//initial capacity should be power of 2**

**set = new HashSet<>(32);**

**//setting backing HashMap initial capacity and load factor**

**set = new HashSet<>(32, 0.80f);**

**//creating HashSet from another Collection**

**Set<String> set1 = new HashSet<>(set);**

**Set<String> set2 = new HashSet<>(new ArrayList<>());**

**Java HashSet Methods**

Some of the useful HashSet methods are;

1. **public boolean add(E e)**: Adds the given element to the Set if not already present. This method internally uses equals() method to check for duplicates, so make sure your object defines equals() method properly.
2. **public void clear()**: Removes all the elements from the Set.
3. **public Object clone()**: Returns a shallow copy of the Set instance.
4. **public boolean contains(Object o)**: Returns true if the Set contains the given element, othrweise false.
5. **public boolean isEmpty()**: Returns true if Set contains no elements, otherwise false.
6. **public Iterator<E> iterator()**: Returns an iterator over the elements in this set. The elements are returned in no particular order.
7. **public boolean remove(Object o)**: Removes the given element from this set if it is present and return true. If the element is not present in the set, just returns false.
8. **public int size()**: Returns the number of elements in the set.
9. **public Spliterator<E> spliterator()**: Creates a late-binding and fail-fast Spliterator over the elements in this set. This is introduced in [Java 8](https://www.journaldev.com/2389/java-8-features-with-examples), however I have not used it till now.
10. **public boolean removeAll(Collection<?> c)**: HashSet inherits this method from AbstractSet. This method will remove all the elements in the set that are part of the specified collection.
11. **public class HashSetExample {**
12. public static void main(String[] args) {
13. Set<String> fruits = new HashSet<>();
15. //add example
16. fruits.add("Apple");
17. fruits.add("Banana");
19. //isEmpty example
20. System.out.println("fruits set is empty = "+fruits.isEmpty());
21. //contains example
22. System.out.println("fruits contains Apple = "+fruits.contains("Apple"));
23. System.out.println("fruits contains Mango = "+fruits.contains("Mango"));
25. //remove example
26. System.out.println("Apple removed from fruits set = "+fruits.remove("Apple"));
27. System.out.println("Mango removed from fruits set = "+fruits.remove("Mango"));
29. //size example
30. System.out.println("fruits set size = "+fruits.size());
32. //addAll example
33. List<String> list = new ArrayList<>();
34. list.add("Apple"); list.add("Apple");
35. list.add("Banana"); list.add("Mango");
37. System.out.println("fruits set before addAll = "+fruits);
38. System.out.println("list = "+list);
39. fruits.addAll(list);
40. System.out.println("fruits set after addAll = "+fruits);
41. //iterator example
42. Iterator<String> iterator = fruits.iterator();
43. while(iterator.hasNext()){
44. System.out.println("Consuming fruit "+iterator.next());
45. }
47. //removeAll example
48. fruits.add("Orange");
49. System.out.println("fruits set before removeAll = "+fruits);
50. System.out.println("list = "+list);
51. fruits.removeAll(list);
52. System.out.println("fruits set after removeAll = "+fruits);
54. //clear example
55. fruits.clear();
56. System.out.println("fruits set is empty = "+fruits.isEmpty());
57. }
58. }
59. Output of above HashSet example

public class HashSetToArrayExample {

public static void main(String[] args) {

Set<Integer> ints = new HashSet<>();

for(int i=0; i<10; i++){

ints.add(i);

}

System.out.println("ints set = "+ints);

// set to array example

Integer[] intArray = new Integer[ints.size()];

intArray = ints.toArray(intArray);

System.out.println("intArray = "+Arrays.toString(intArray));

ints.remove(0);ints.remove(1);

System.out.println("intArray = "+Arrays.toString(intArray));

//array to set example

ints = new HashSet<>(Arrays.asList(intArray));

System.out.println("ints from array = "+ints);

ints.remove(0);ints.remove(1);

System.out.println("ints from array after remove = "+ints);

System.out.println("intArray = "+Arrays.toString(intArray));

}

}

public class HashSetToArrayExample {

public static void main(String[] args) {

Set<Integer> ints = new HashSet<>();

for(int i=0; i<10; i++){

ints.add(i);

}

System.out.println("ints set = "+ints);

// set to array example

Integer[] intArray = new Integer[ints.size()];

intArray = ints.toArray(intArray);

System.out.println("intArray = "+Arrays.toString(intArray));

ints.remove(0);ints.remove(1);

System.out.println("intArray = "+Arrays.toString(intArray));

//array to set example

ints = new HashSet<>(Arrays.asList(intArray));

System.out.println("ints from array = "+ints);

ints.remove(0);ints.remove(1);

System.out.println("ints from array after remove = "+ints);

System.out.println("intArray = "+Arrays.toString(intArray));

}

}

ints set = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

intArray = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

intArray = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

ints from array = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

ints from array after remove = [2, 3, 4, 5, 6, 7, 8, 9]

intArray = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

### Java HashSet to List Example

package com.journaldev.examples;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class HashSetToListExample {

public static void main(String[] args) {

Set<String> vowels = new HashSet<>();

vowels.add("a"); vowels.add("e"); vowels.add("i");

//set to list example

List<String> vowelsList = new ArrayList<>(vowels);

System.out.println("vowels set = "+vowels);

System.out.println("vowelsList = "+vowelsList);

vowels.add("o");

vowelsList.add("a");vowelsList.add("u");

System.out.println("vowels set = "+vowels);

System.out.println("vowelsList = "+vowelsList);

//list to set example

vowels = new HashSet<>(vowelsList);

System.out.println("vowels set = "+vowels);

}

}

### Java HashSet equals() and hashCode() methods

package com.journaldev.examples;

import java.util.HashSet;

import java.util.Set;

public class HashSetEqualsMethodImportance {

public static void main(String[] args) {

Set<Emp> emps = new HashSet<>();

emps.add(new Emp(1,"Pankaj"));

emps.add(new Emp(2, "David"));

emps.add(new Emp(1, "Pankaj"));

System.out.println(emps);

Emp e = new Emp(3, "Lisa");

emps.add(e);

System.out.println(emps);

//set values to make it duplicate

e.setId(1);

System.out.println(emps);

e.setName("Pankaj");

System.out.println(emps);

}

}

class Emp {

private String name;

private int id;

public Emp(int i, String n) {

this.setId(i);

this.setName(n);

}

@Override

public boolean equals(Object obj){

if(obj == null || !(obj instanceof Emp)) return false;

Emp e = (Emp) obj;

if(e.getId() == this.getId() && this.getName().equals(e.getName())) return true;

return false;

}

@Override

public int hashCode(){

return getId();

}

@Override

public String toString(){

return "{"+getId()+","+getName()+"}";

}

public String getName() {

return name;

}

public int getId() {

return id;

}

public void setName(String name) {

this.name = name;

}

public void setId(int id) {

this.id = id;

}

}

[{1,Pankaj}, {2,David}]

[{1,Pankaj}, {2,David}, {3,Lisa}]

[{1,Pankaj}, {2,David}, {1,Lisa}]

[{1,Pankaj}, {2,David}, {1,Pankaj}]