Set

LeSuccess

What you'll Learn

Set

HashSet

LinkedHashSet

TreeSet

Examples

Set

A collection that contains no duplicate elements.

Sets contain no pair of elements e1 and e2 such that e1.equals(e2), and at most one null element.

As implied by its name, this interface models the mathematical *set* abstraction.

The Set interface places additional stipulations, beyond those inherited from the Collection interface, on the contracts of all constructors and on the contracts of the add, equals and hashCode methods.

HashSet

HashSet stores the elements by using a mechanism called **hashing.**

HashSet contains unique elements only.

HashSet allows null value.

HashSet class is non synchronized.

HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode.

HashSet is the best approach for search operations.

The initial default capacity of HashSet is 16, and the load factor is 0.75.

Constructor

Constructor	Description
HashSet()	It is used to construct a default HashSet.
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.
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.
HashSet(Collection<? extends E> c)	It is used to initialize the hash set by using the elements of the collection c.

HashSet Class Declaration

Hashset class which is implemented in the collection framework is an inherent implementation of the hash table datastructure. The objects that we insert into the hashset does not guarantee to be inserted in the same order. The objects are inserted based on their hashcode. This class also allows the insertion of NULL elements.

Methods

Modifier & Type	Method	Description
boolean	add(E e)	It is used to add the specified element to this set if it is not already present.
void	clear()	It is used to remove all of the elements from the set.
object	clone()	It is used to return a shallow copy of this HashSet instance: the elements themselves are not cloned.
boolean	contains(Object o)	It is used to return true if this set contains the specified element.

LeSuccess

Methods

Modifier & Type	Method	Description
boolean	isEmpty()	It is used to return true if this set contains no elements.
Iterator<E>	iterator()	It is used to return an iterator over the elements in this set.
boolean	remove(Object o)	It is used to remove the specified element from this set if it is present.
int	size()	It is used to return the number of elements in the set.
Spliterator<E>	spliterator()	It is used to create a late-binding and fail-fast Spliterator over the elements in the set.

Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<String> set=new HashSet();
    set.add("1");
    set.add("2");
    set.add("3");
    set.add("4");
    set.add("5");
    Iterator<String> i=set.iterator();
    while(i.hasNext()){
      System.out.println(i.next());
    }
  }
}
```

Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<String> set=new HashSet();
    set.add("1");
    set.add("1");
    set.add("1");
    set.add("2");
    set.add("2");
    Iterator<String> i=set.iterator();
    while(i.hasNext()){
      System.out.println(i.next());
    }
  }
}
```

Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<Integer> set=new HashSet();
    set.add(1);
    set.add(1);
    set.add(2);
    set.add(3);
    set.add(3);
    Iterator<Integer> i=set.iterator();
    while(i.hasNext()){
      System.out.println(i.next());
    }
  }
}
```

addAll() - Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<String> set=new HashSet<String>();
    set.add("aa");
    set.add("bb");
    set.add("cc");
    System.out.println("An initial Set of elements: "+set);
    HashSet<String> set1=new HashSet<String>();
    set1.add("DD");
    set1.add("EE");
    set.addAll(set1);
    System.out.println("Updated Set: "+set);
  }
}
```

remove() - Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<String> set=new HashSet<String>();
    set.add("aa");
    set.add("bb");
    set.add("cc");
    set.add("dd");
    System.out.println("An initial list of elements: "+set);
    set.remove("aa");
    System.out.println("After invoking remove(object) method:
"+set);
  }
}
```

remove() - Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<String> set=new HashSet<String>();
    set.add("aa");
    set.add("bb");
    set.add("cc");
    set.add("dd");
    System.out.println("An initial list of elements: "+set);
    System.out.println(set.remove("ee"));
  }
}
```

Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<String> set=new HashSet<String>();
    set.add("ee");
    set.add("bb");
    set.add("aa");
    set.add("cc");
    System.out.println("An initialf elements: "+set);
    set.removeAll(set);
    System.out.println("After invoking removeAll() method:
"+set);
  }
}
```

removeIf() - Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<String> set=new HashSet<String>();
    set.add("ff");
    set.add("gg");
    set.add("jj");
    set.add("hh");
    System.out.println("An initial list of elements: "+set);
    set.removeIf(str->str.contains("jj"));
    System.out.println("After invoking removeIf() method: "+set);
  }
}
```

clear() - Method

```
import java.util.*;
class Main{
  public static void main(String args[]){
    HashSet<String> set=new HashSet<String>();
    set.add("ff");
    set.add("gg");
    set.add("jj");
    set.add("hh");
    System.out.println("An initial list of elements: "+set);
    set.clear();
    System.out.println("After invoking clear() method: "+set);
  }
}
```

Linkedhashset

Java LinkedHashSet class contains unique elements only like HashSet.

Java LinkedHashSet class provides all optional set operation and permits null elements.

Java LinkedHashSet class maintains insertion order.

Constructor

Constructor	Description
HashSet()	It is used to construct a default HashSet.
HashSet(Collection c)	It is used to initialize the hash set by using the elements of the collection c.
LinkedHashSet(int capacity)	It is used initialize the capacity of the linked hash set to the given integer value capacity.
LinkedHashSet(int capacity, float fillRatio)	It is used to initialize both the capacity and the fill ratio (also called load capacity) of the hash set from its argument.

add() - Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    LinkedHashSet<String> set=new LinkedHashSet();
    set.add("aa");
    set.add("ff");
    set.add("ee");
    set.add("dd");
    set.add("bb");
    Iterator<String> i=set.iterator();
    while(i.hasNext()){
      System.out.println(i.next());
    }
  }
}
```

Example

```
import java.util.*;
class Main{
  public static void main(String args[]){
    LinkedHashSet<String> al=new LinkedHashSet<String>();
    al.add("Ravi");
    al.add("Vijay");
    al.add("Ravi");
    al.add("Ajay");
    Iterator<String> itr=al.iterator();
    while(itr.hasNext()){
      System.out.println(itr.next());
    }
  }
}
```

TreeSet

SortedSet interface

Duplicate values are not allowed

Ascending order

Elements are sorted by keys

Constructor

Constructor	Description
TreeSet()	It is used to construct an empty tree set that will be sorted in ascending order according to the natural order of the tree set.
TreeSet(Collection<? extends E> c)	It is used to build a new tree set that contains the elements of the collection c.
TreeSet(Comparator<? super E> comparator)	It is used to construct an empty tree set that will be sorted according to given comparator.
TreeSet(SortedSet<E> s)	It is used to build a TreeSet that contains the elements of the given SortedSet.

Methods

Methods	Return
descendingIterator() - Returns an iterator over the elements in this set in descending order	Iterator<E>
descendingSet() - Returns a reverse order view of the elements contained in this set	NavigableSet <E>
first() - Returns the first (lowest) element currently in this set	E
tailSet(E fromElement) - Returns a view of the portion of this set whose elements are greater than or equal to fromElement	SortedSet<E>
remove(Object o) - Removes the specified element from this set if it is present	boolean

add() - Example

```
import java.util.TreeSet;
public class Main {
    public static void main(String args[]){
        TreeSet<String> Ethnus = new TreeSet<String>();
        Ethnus.add("codemithra.com");
        Ethnus.add("eguru.ooo");
        Ethnus.add("aptimithra.com");
        System.out.println("Our product: " + Ethnus);
    }
}
```

Iterator() - Example

```
import java.util.*;
public class Main {
  public static void main(String args[]){
    TreeSet<String> Ethnus = new TreeSet<String>();
    Ethnus.add("codemithra.com");
    Ethnus.add("eguru.ooo");
    Ethnus.add("aptimithra.com");
    NavigableSet<String> treereverse = Ethnus.descendingSet();
    Iterator<String> iterator = treereverse.iterator();
    while (iterator.hasNext()) {
      System.out.println("Our product: "+ iterator.next());
    }
  }
}
```

pollLast() - Example

```
import java.util.*;
public class Main {
 public static void main(String args[]) {
   TreeSet<String> Ethnus = new TreeSet<String>();
   Ethnus.add("codemithra.com");
   Ethnus.add("eguru.ooo");
   Ethnus.add("aptimithra.com");
   System.out.println("Our product: "+ Ethnus.pollLast());
   System.out.println("Our product: "+ Ethnus);
 }
}
```

pollFirst() - Example

```
import java.util.*;
public class Main {
 public static void main(String args[]) {
   TreeSet<String> Ethnus = new TreeSet<String>();
   Ethnus.add("codemithra.com");
   Ethnus.add("eguru.ooo");
   Ethnus.add("aptimithra.com");
   System.out.println("Our product: "+ Ethnus.pollFirst());
   System.out.println("Our product: "+ Ethnus);
 }
}
```

SortedSet - Example

```
import java.util.*;
public class Main {
 public static void main(String args[]) {
   SortedSet<String> Ethnus = new TreeSet<String>();
   Ethnus.add("codemithra.com");
   Ethnus.add("eguru.ooo");
   Ethnus.add("aptimithra.com");
   System.out.println("Our product: "+ Ethnus);
 }
}
```

THANK YOU