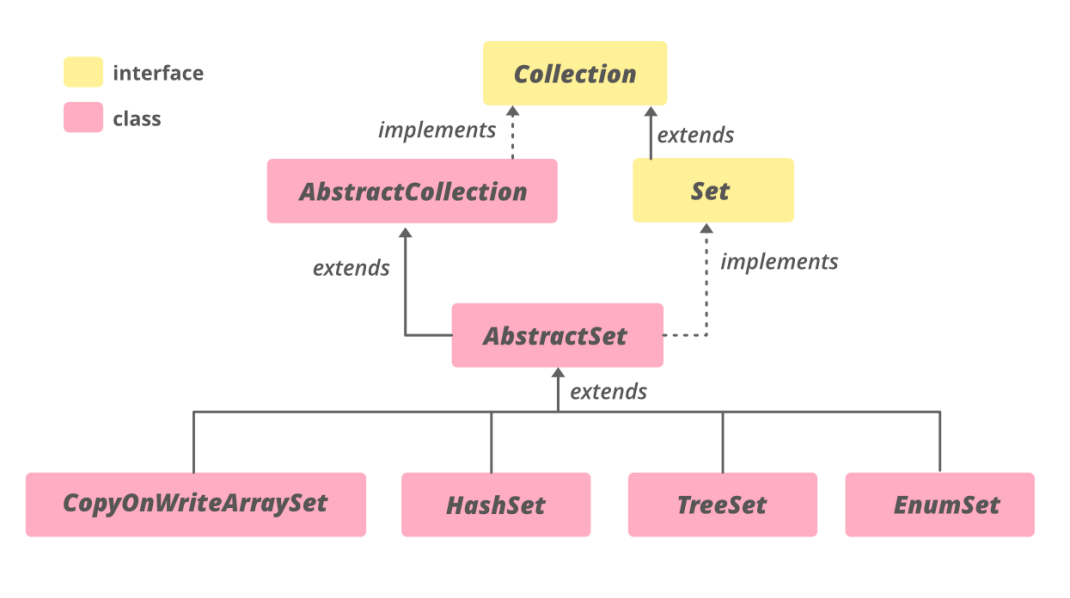
AbstractSet Class in Java with Examples

Difficulty Level : Easy Last Updated : 17 Jan, 2022

AbstractSet class in Java is a part of the Java Collection Framework which implements the *Collection interface* and extends the *AbstractCollection class*. It provides a skeletal implementation of the Set interface. This class does not override any of the implementations from the AbstractCollection class, but merely adds implementations for the equals() and hashCode() method.

The process of implementing a set by extending this class is identical to that of implementing a Collection by extending AbstractCollection, except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the Set interface (for instance, the add method must not permit the addition of multiple instances of an object to a set).



From the class hierarchy diagram, it can be concluded that it implements **Iterable<E>**, **Collection<E>**, Set<E> interfaces. The direct subclasses are ConcurrentSkipListSet, CopyOnWriteArraySet, EnumSet, HashSet, TreeSet. As we already discussed above, AbstractSet is an abstract class, so it should be assigned an instance of its subclasses such as TreeSet, HashSet, or EnumSet.

Syntax: Declaration

```
public abstract class AbstractSet<E>
extends AbstractCollection<E>
implements Set<E>
```

Where **E** is the type of elements maintained by this Set.

Constructor of AbstractSet Class

1. protected AbstractSet(): The default constructor, but being protected, it doesn't allow the creation of an AbstractSet object.

```
AbstractSet<E> as = new TreeSet<E>();
```

Methods of AbstractSet

METHOD	DESCRIPTION
equals(Object o)	Compares the specified object with this set for equality.
hashCode()	Returns the hash code value for this set.
removeAll (Collection<?> c)	Removes from this set all of its elements that are contained in the specified collection (optional operation).

Example 1:

```
// Java Program to Illustrate AbstractSet Class

// Importing required classes
import java.util.*;

// Main class
public class GFG {

    // Main driver method
    public static void main(String[] args) throws Exception
    {
        // Try block to check for exceptions
        try {

            // Creating an empty TreeSet of integer type by
            // creating object of AbstractSet
```

```
            AbstractSet<Integer> abs_set
                = new TreeSet<Integer>();

            // Populating TreeSet
            // using add() method
            abs_set.add(1);
            abs_set.add(2);
            abs_set.add(3);
            abs_set.add(4);
            abs_set.add(5);

            // Printing the elements inside above TreeSet
            System.out.println("AbstractSet: " + abs_set);
        }

        // Catch block to handle the exceptions
        catch (Exception e) {

            // Display exception on console
            System.out.println(e);
        }
    }
}
```

Output

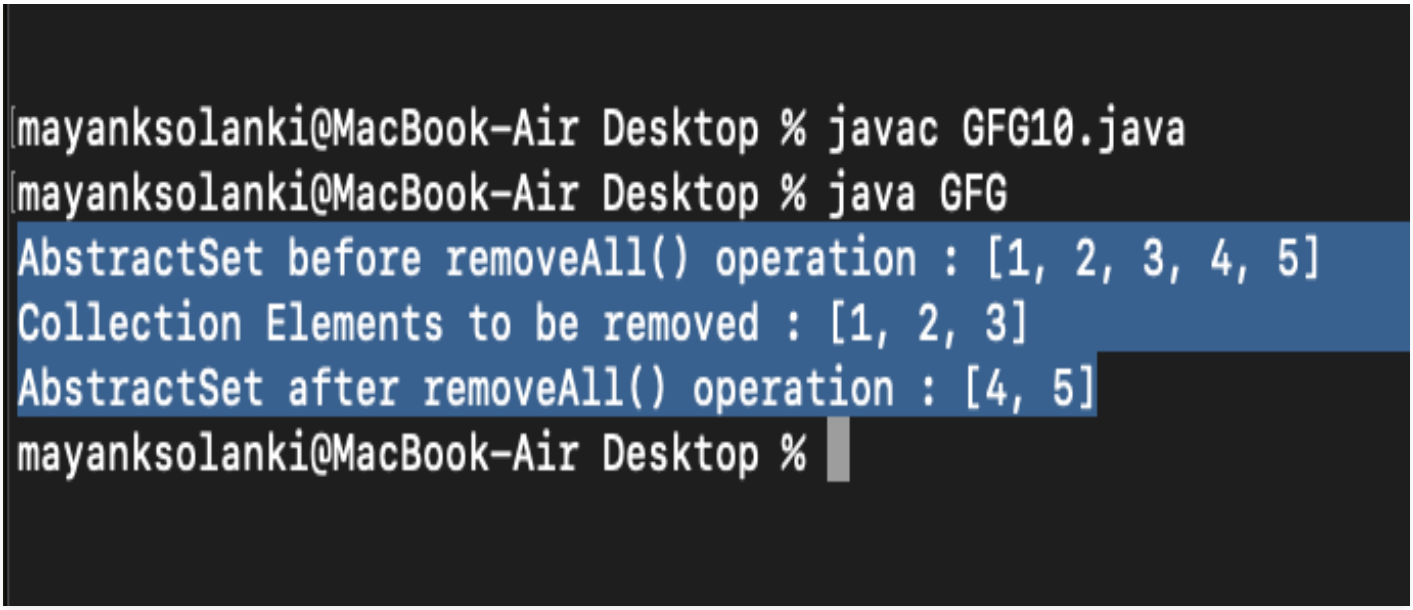
```
AbstractSet: [1, 2, 3, 4, 5]
```

Example 2:

```
// Java Program to Illustrate Methods
// of AbstractSet class

// Importing required classes
import java.util.*;

// Main class
public class GFG {

    // Main driver method
    public static void main(String[] args) throws Exception
    {

        // Try block to check for exceptions
        try {
```

```
            // Creating an empty TreeSet of integer type
            AbstractSet<Integer> abs_set
                = new TreeSet<Integer>();

            // Populating above TreeSet
            // using add() method
            abs_set.add(1);
            abs_set.add(2);
            abs_set.add(3);
            abs_set.add(4);
            abs_set.add(5);

            // Printing the elements inside TreeSet
            System.out.println("AbstractSet before "
                               + "removeAll() operation : "
                               + abs_set);

            // Creating an ArrayList of integer type
            Collection<Integer> arrlist2
                = new ArrayList<Integer>();

            // Adding elements to above ArrayList
            arrlist2.add(1);
            arrlist2.add(2);
            arrlist2.add(3);

            // Printing the ArrayList elements
            System.out.println("Collection Elements"
                               + " to be removed : "
                               + arrlist2);

            // Removing elements from AbstractSet specified
            // using removeAll() method
            abs_set.removeAll(arrlist2);

            // Printing the elements of ArrayList
            System.out.println("AbstractSet after "
                               + "removeAll() operation : "
                               + abs_set);
        }

        // Catch block to handle the exceptions
        catch (NullPointerException e) {

            // Display exception on console
            System.out.println("Exception thrown : " + e);
        }
    }
}
```

Output:



Some other methods of classes or interfaces are defined below that somehow helps us to get a better understanding of AbstractSet Class as follows:

Methods Declared in Class java.util.AbstractCollection

METHOD	DESCRIPTION
add(E e)	Ensures that this collection contains the specified element (optional operation).
addAll (Collection<? extends E> c)	Adds all of the elements in the specified collection to this collection (optional operation).
clear()	Removes all of the elements from this collection (optional operation).
contains(Object o)	Returns true if this collection contains the specified element.
containsAll (Collection<?> c)	Returns true if this collection contains all of the elements in the specified collection.
isEmpty()	Returns true if this collection contains no elements.
iterator()	Returns an iterator over the elements contained in this collection.
remove(Object o)	Removes a single instance of the specified element from this collection, if it is present (optional operation).

METHOD	DESCRIPTION
retainAll (Collection<?> c)	Retains only the elements in this collection that are contained in the specified collection (optional operation).
toArray()	Returns an array containing all of the elements in this collection.
toArray(T[] a)	Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array.
toString()	Returns a string representation of this collection.

Methods Declared in Interface java.util.Collection

METHOD	DESCRIPTION
parallelStream()	Returns a possibly parallel Stream with this collection as its source.
removeIf (Predicate<? super E> filter)	Removes all of the elements of this collection that satisfy the given predicate.
stream()	Returns a sequential Stream with this collection as its source.
toArray (IntFunction<T[]> generator)	Returns an array containing all of the elements in this collection, using the provided generator function to allocate the returned array.

Methods Declared in interface java.lang.Iterable

METHOD	DESCRIPTION
forEach (Consumer<? super T> action)	Performs the given action for each element of the Iterable until all elements have been processed or the action throws an exception.

Methods Declared in interface java.util.Set

METHOD	DESCRIPTION
add(E e)	Adds the specified element to this set if it is not already present (optional operation).
addAll (Collection<? extends E> c)	Adds all of the elements in the specified collection to this set if they're not already present (optional operation).
clear()	Removes all of the elements from this set (optional operation).
contains(Object o)	Returns true if this set contains the specified element.
containsAll (Collection<?> c)	Returns true if this set contains all of the elements of the specified collection.
isEmpty()	Returns true if this set contains no elements.
iterator()	Returns an iterator over the elements in this set.
remove(Object o)	Removes the specified element from this set if it is present (optional operation).
retainAll (Collection<?> c)	Retains only the elements in this set that are contained in the specified collection (optional operation).
size()	Returns the number of elements in this set (its cardinality).
spliterator()	Creates a Spliterator over the elements in this set.
toArray()	Returns an array containing all of the elements in this set.
toArray(T[] a)	Returns an array containing all of the elements in this set; the runtime type of the returned array is that of the specified array.