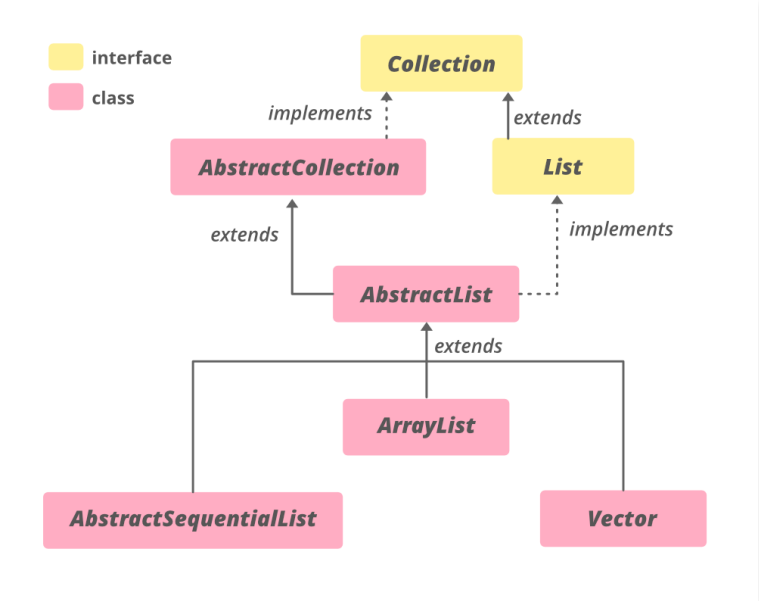
AbstractList in Java with Examples

Difficulty Level : Easy Last Updated : 11 Nov, 2020

The **AbstractList** class in Java is a part of the Java Collection Framework and implements the *Collection interface* and the *AbstractCollection class*. This class provides a skeletal implementation of the List interface to minimize the effort required to implement this interface backed by a **Random Access** data store (such as an array). For sequential access data (such as a linked list), AbstractSequentialList should be used in preference to this class.

To implement an unmodifiable list, for which one needs to only extend this AbstractList Class and implement the get(int) and the size() methods. To implement a modifiable list, for which one additionally override the set(int index, E element) method (which otherwise throws an UnsupportedOperationException). If the list is variable-size, for which one should override the add(int, E) and remove(int) methods.

Class Hierarchy:



Declaration:

```
public abstract class AbstractList<E> extends AbstractCollection<E> implements

where E is the type of elements maintained by this collection.
```

Constructor: protected AbstractList() – The default constructor, but being protected, it doesn't allow to create an AbstractList object.

AbstractList<E> al = new ArrayList<E>();

Example 1: AbstractList is an abstract class, so it should be assigned an instance of its subclasses such as ArrayList, LinkedList, or Vector.

```
// Java code to illustrate AbstractList
import java.util.*;

public class AbstractListDemo {
    public static void main(String args[])
    {

        // Creating an empty AbstractList
        AbstractList<String> list = new ArrayList<String>();

        // Use add() method to add elements in the list
        list.add("Geeks");
        list.add("for");
        list.add("Geeks");
        list.add("10");
        list.add("20");

        // Displaying the AbstractList
        System.out.println("AbstractList:" + list);
    }
}
```

Output:

```
AbstractList:[Geeks, for, Geeks, 10, 20]
```

Example 2:

```java
// Java code to illustrate
// methods of AbstractCollection

import java.util.*;

public class AbstractListDemo {
    public static void main(String args[])
    {

        // Creating an empty AbstractList
        AbstractList<String>
            list = new LinkedList<String>();

        // Using add() method to add elements in the list
        list.add("Geeks");
        list.add("for");
        list.add("Geeks");
        list.add("10");
        list.add("20");

        // Output the list
        System.out.println("AbstractList: " + list);

        // Remove the head using remove()
        list.remove(3);

        // Print the final list
        System.out.println("Final AbstractList: " + list);

        // getting the index of last occurence
        // using lastIndexOf() method
        int lastindex = list.lastIndexOf("A");

        // printing the Index
        System.out.println("Last index of A : "
                           + lastindex);
    }
}
```

Output:

```
AbstractList: [Geeks, for, Geeks, 10, 20]
Final AbstractList: [Geeks, for, Geeks, 20]
Last index of A : -1
```

Methods in AbstractList

METHOD	DESCRIPTION
add(int index, E element)	Inserts the specified element at the specified position in this list (optional operation).
add(E e)	Appends the specified element to the end of this list (optional operation).
addAll(int index, Collection<? extends E> c)	Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
clear()	Removes all of the elements from this list (optional operation).
equals(Object o)	Compares the specified object with this list for equality.
get(int index)	Returns the element at the specified position in this list.
hashCode()	Returns the hash code value for this list.
indexOf(Object o)	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
iterator()	Returns an iterator over the elements in this list in proper sequence.
lastIndexOf(Object o)	Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
listIterator()	Returns a list iterator over the elements in this list (in proper sequence).
listIterator(int index)	Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list.
remove(int index)	Removes the element at the specified position in this list (optional operation).
removeRange(int fromIndex, int toIndex)	Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive.

METHOD	DESCRIPTION
set(int index, E element)	Replaces the element at the specified position in this list with the specified element (optional operation).
subList(int fromIndex, int toIndex)	Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.

Methods declared in class java.util.AbstractCollection

METHOD	DESCRIPTION
addAll (Collection<? extends E> c)	Adds all of the elements in the specified collection to 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.
remove(Object o)	Removes a single instance of the specified element from this collection, if it is present (optional operation).
removeAll (Collection<?> c)	Removes all of this collection’s elements that are also contained in the specified collection (optional operation).
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.util.List

METHOD	DESCRIPTION
addAll(Collection<? extends E> c)	Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator (optional operation).
contains(Object o)	Returns true if this list contains the specified element.
containsAll (Collection<?> c)	Returns true if this list contains all of the elements of the specified collection.
isEmpty()	Returns true if this list contains no elements.
remove(int index)	Removes the element at the specified position in this list (optional operation).
removeAll(Collection<?> c)	Removes from this list all of its elements that are contained in the specified collection (optional operation).

METHOD	DESCRIPTION
replaceAll (UnaryOperator<E> operator)	Replaces each element of this list with the result of applying the operator to that element.
retainAll(Collection<?> c)	Retains only the elements in this list that are contained in the specified collection (optional operation).
size()	Returns the number of elements in this list.
sort(Comparator<? super E> c)	Sorts this list according to the order induced by the specified Comparator.
spliterator()	Creates a Spliterator over the elements in this list.
toArray()	Returns an array containing all of the elements in this list in proper sequence (from first to last element).
toArray(T[] a)	Returns an array containing all of the elements in this list in proper sequence (from first to last element) ; the runtime type of the returned array is that of the specified array.

Reference: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/AbstractList.html