# Java - The Collection Interface

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **boolean add(Object obj)**  Adds obj to the invoking collection. Returns true if obj was added to the collection. Returns false if obj is already a member of the collection, or if the collection does not allow duplicates. |
| 2 | **boolean addAll(Collection c)**  Adds all the elements of c to the invoking collection. Returns true if the operation succeeds (i.e., the elements were added). Otherwise, returns false. |
| 3 | **void clear( )**  Removes all elements from the invoking collection. |
| 4 | **boolean contains(Object obj)**  Returns true if obj is an element of the invoking collection. Otherwise, returns false. |
| 5 | **boolean containsAll(Collection c)**  Returns true if the invoking collection contains all elements of **c**. Otherwise, returns false. |
| 6 | **boolean equals(Object obj)**  Returns true if the invoking collection and obj are equal. Otherwise, returns false. |
| 7 | **int hashCode( )**  Returns the hash code for the invoking collection. |
| 8 | **boolean isEmpty( )**  Returns true if the invoking collection is empty. Otherwise, returns false. |
| 9 | **Iterator iterator( )**  Returns an iterator for the invoking collection. |
| 10 | **boolean remove(Object obj)**  Removes one instance of obj from the invoking collection. Returns true if the element was removed. Otherwise, returns false. |
| 11 | **boolean removeAll(Collection c)**  Removes all elements of c from the invoking collection. Returns true if the collection changed (i.e., elements were removed). Otherwise, returns false. |
| 12 | **boolean retainAll(Collection c)**  Removes all elements from the invoking collection except those in c. Returns true if the collection changed (i.e., elements were removed). Otherwise, returns false. |
| 13 | **int size( )**  Returns the number of elements held in the invoking collection. |
| 14 | **Object[ ] toArray( )**  Returns an array that contains all the elements stored in the invoking collection. The array elements are copies of the collection elements. |
| 15 | **Object[ ] toArray(Object array[ ])**  Returns an array containing only those collection elements whose type matches that of array. |

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// ArrayList* List a1 = **new** ArrayList();  a1.add(**"Zara"**);  a1.add(**"Mahnaz"**);  a1.add(**"Ayan"**);  System.***out***.println(**" ArrayList Elements"**);  System.***out***.print(**"\t"** + a1);   *// LinkedList* List l1 = **new** LinkedList();  l1.add(**"Zara"**);  l1.add(**"Mahnaz"**);  l1.add(**"Ayan"**);  System.***out***.println();  System.***out***.println(**" LinkedList Elements"**);  System.***out***.print(**"\t"** + l1);   *// HashSet* Set s1 = **new** HashSet();  s1.add(**"Zara"**);  s1.add(**"Mahnaz"**);  s1.add(**"Ayan"**);  System.***out***.println();  System.***out***.println(**" Set Elements"**);  System.***out***.print(**"\t"** + s1);   *// HashMap* Map m1 = **new** HashMap();  m1.put(**"Zara"**, **"8"**);  m1.put(**"Mahnaz"**, **"31"**);  m1.put(**"Ayan"**, **"12"**);  m1.put(**"Daisy"**, **"14"**);  System.***out***.println();  System.***out***.println(**" Map Elements"**);  System.***out***.print(**"\t"** + m1);  } } | ArrayList Elements  [Zara, Mahnaz, Ayan]  LinkedList Elements  [Zara, Mahnaz, Ayan]  Set Elements  [Ayan, Zara, Mahnaz]  Map Elements  {Daisy=14, Ayan=12, Zara=8, Mahnaz=31} |

java.util.Collections.addAll() Method

The **addAll(Collection<? super T>, T..)** method is used to add all of the specified elements to the specified collection.

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// create array list object* List arrlist = **new** ArrayList();   *// populate the list* arrlist.add(**"A"**);  arrlist.add(**"B"**);  arrlist.add(**"C"**);   System.***out***.println(**"Initial collection value: "**+arrlist);   *// add values to this collection* **boolean** b = Collections.*addAll*(arrlist, **"1"**,**"2"**,**"3"**);   System.***out***.println(**"Final collection value: "**+arrlist);  } } | Initial collection value: [A, B, C]  Final collection value: [A, B, C, 1, 2, 3] |

.clear method

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// create array list object* List arrlist = **new** ArrayList();   *// populate the list* arrlist.add(**"A"**);  arrlist.add(**"B"**);  arrlist.add(**"C"**);   System.***out***.println(**"Initial collection value: "**+arrlist);  arrlist.clear();  System.***out***.println(**".clear method collection value: "**+arrlist);   *// add values to this collection  // boolean b = Collections.addAll(arrlist, "1","2","3");   // System.out.println("Final collection value: "+arrlist);* } } | Initial collection value: [A, B, C]  .clear method collection value: [] |

**boolean contains(Object obj)**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// create array list object* List arrlist = **new** ArrayList();   *// populate the list* arrlist.add(**"A"**);  arrlist.add(**"B"**);  arrlist.add(**"C"**);   System.***out***.println(**"Initial collection value: "**+arrlist);  **if** (arrlist.contains(**"B"**))  System.***out***.println(**"found"**);  **else** System.***out***.println(**"not found"**);   **if** (arrlist.contains(**"X"**))  System.***out***.println(**"found"**);  **else** System.***out***.println(**"not found"**);   } } | Initial collection value: [A, B, C]  found  not found |

**boolean containsAll(Collection c)**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// create three empty Vectors vec and vectest with an initial capacity of 4* ArrayList<Integer> vec = **new** ArrayList<Integer>(4);  ArrayList<Integer> vectest = **new** ArrayList<Integer>(4);  ArrayList<Integer> vecdiff = **new** ArrayList<Integer>(4);   *// use add() method to add elements in the vector vec* vec.add(4);  vec.add(3);  vec.add(2);   *// use add() method to add elements in the vector vectest* vectest.add(4);  vectest.add(3);  vectest.add(2);   *// use add() method to add elements in the vector vecdiff* vecdiff.add(4);  vecdiff.add(3);  vecdiff.add(12);   *// let us check vec and vectest* System.***out***.println(**"Checking contents in vec and vectest :- "**+vectest.containsAll(vec));   *// let us check vec and vecdiff* System.***out***.println(**"Checking contents in vec and vecdiff :- "**+vecdiff.containsAll(vec));  } } | Checking contents in vec and vectest :- true  Checking contents in vec and vecdiff :- false |

**boolean equals(Object obj)**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// create three empty Vectors vec and vectest with an initial capacity of 4* ArrayList<Integer> vec = **new** ArrayList<Integer>(4);  ArrayList<Integer> vectest = **new** ArrayList<Integer>(4);  ArrayList<Integer> vecdiff = **new** ArrayList<Integer>(4);   *// use add() method to add elements in the vector vec* vec.add(4);  vec.add(3);  vec.add(2);   *// use add() method to add elements in the vector vectest* vectest.add(4);  vectest.add(3);  vectest.add(2);   *// use add() method to add elements in the vector vecdiff* vecdiff.add(4);  vecdiff.add(3);  vecdiff.add(12);   *// let us check vec and vectest* System.***out***.println(**"Checking contents in vec and vectest :- "**+vectest.equals(vec));   *// let us check vec and vecdiff* System.***out***.println(**"Checking contents in vec and vecdiff :- "**+vecdiff.equals(vec));  } } | Checking contents in vec and vectest :- true  Checking contents in vec and vecdiff :- false |

**Int hashcode()**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// create three empty Vectors vec and vectest with an initial capacity of 4* ArrayList<Integer> arr = **new** ArrayList<Integer>();  arr.add(4);  arr.add(5);  arr.add(6);  arr.add(7);   Integer inti = **new** Integer(10);  **int** hashVal = inti.hashCode();  System.***out***.println(hashVal);   **int** hasn = arr.hashCode();  **for** (**int** i = 0; i <= arr.size(); i++) {  System.***out***.println(hasn);  }  } } | 10  1047683  1047683  1047683  1047683  1047683 |

**boolean isEmpty( )**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// create three empty Vectors vec and vectest with an initial capacity of 4* ArrayList<Integer> arr = **new** ArrayList<Integer>();  arr.add(4);  arr.add(5);  arr.add(6);  arr.add(7);   ArrayList<Integer> arr2 = **new** ArrayList<Integer>();   **if** (arr.isEmpty())  System.***out***.println(**"Empty"**);  **else** System.***out***.println(**"Not empty"**);   **if** (arr2.isEmpty())  System.***out***.println(**"Empty"**);  **else** System.***out***.println(**"Not empty"**);  } } | Not empty  Empty |

**Iterator iterator( )**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Sr.No.** | **Method & Description** | | 1 | **boolean hasNext( )**  Returns true if there are more elements. Otherwise, returns false. | | 2 | **Object next( )**  Returns the next element. Throws NoSuchElementException if there is not a next element. | | 3 | **void remove( )**  Removes the current element. Throws IllegalStateException if an attempt is made to call remove( ) that is not preceded by a call to next( ). | | |  |  | | --- | --- | | **Sr.No.** | **Method & Description** | | 1 | **void add(Object obj)**  Inserts obj into the list in front of the element that will be returned by the next call to next( ). | | 2 | **boolean hasNext( )**  Returns true if there is a next element. Otherwise, returns false. | | 3 | **boolean hasPrevious( )**  Returns true if there is a previous element. Otherwise, returns false. | | 4 | **Object next( )**  Returns the next element. A NoSuchElementException is thrown if there is not a next element. | | 5 | **int nextIndex( )**  Returns the index of the next element. If there is not a next element, returns the size of the list. | | 6 | **Object previous( )**  Returns the previous element. A NoSuchElementException is thrown if there is not a previous element. | | 7 | **int previousIndex( )**  Returns the index of the previous element. If there is not a previous element, returns -1. | | 8 | **void remove( )**  Removes the current element from the list. An IllegalStateException is thrown if remove( ) is called before next( ) or previous( ) is invoked. | | 9 | **void set(Object obj)**  Assigns obj to the current element. This is the element last returned by a call to either next( ) or previous( ). | |

EXAMPLES

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// Create an array list* ArrayList al = **new** ArrayList();   *// add elements to the array list* al.add(**"C"**);  al.add(**"A"**);  al.add(**"E"**);  al.add(**"B"**);  al.add(**"D"**);  al.add(**"F"**);   *// -------------- The Methods Declared by Iterator -------------------  // Use iterator to display contents of al* System.***out***.print(**"Original contents of al: "**);  Iterator itr = al.iterator();   **while**(itr.hasNext()) { *//.hasNext()* Object element = itr.next(); *//.next()* System.***out***.print(element + **" "**);  }  System.***out***.println();   itr.remove(); *//remove* **if** (itr.hasNext())  System.***out***.println(itr.next());  **else** System.***out***.println(**"No elements found"**);   *// -------------- The Methods Declared by List Iterator -------------------  // Modify objects being iterated* ListIterator litr = al.listIterator();   **while**(litr.hasNext()) { *//hasnext* Object element = litr.next(); *//next* litr.set(element + **"+"**); *//set* }  System.***out***.print(**"Modified contents of al: "**);  itr = al.iterator();   **while**(itr.hasNext()) {  Object element = itr.next();  System.***out***.print(element + **" "**);  }  System.***out***.println();   *// Now, display the list backwards* System.***out***.print(**"Modified list backwards: "**);   **while**(litr.hasPrevious()) { *//hasprevious* Object element = litr.previous(); *//previous (skips the last element)* System.***out***.print(element + **" "**);  }  System.***out***.println();  *// while (litr.hasNext()){ // litr.add("Z"); // Object element = litr.next(); // System.out.println(element + "+"); // }  // System.out.println("adding at beginning"); // litr.add("Z"); // while (litr.hasPrevious()){ // Object element = litr.previous(); // System.out.println(element + "+"); // }* } } | Original contents of al: C A E B D F  No elements found  Modified contents of al: C+ A+ E+ B+ D+  Modified list backwards: D+ B+ E+ A+ C+ |

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// Create an array list* ArrayList al = **new** ArrayList();   *// add elements to the array list* al.add(**"C"**);  al.add(**"A"**);  al.add(**"E"**);  al.add(**"B"**);  al.add(**"D"**);  al.add(**"F"**);   *// -------------- The Methods Declared by Iterator -------------------  // Use iterator to display contents of al* System.***out***.print(**"Original contents of al: "**);  Iterator itr = al.iterator();   **while**(itr.hasNext()) { *//.hasNext()* Object element = itr.next(); *//.next()* System.***out***.print(element + **" "**);  }  System.***out***.println();   itr.remove(); *//remove* **if** (itr.hasNext())  System.***out***.println(itr.next());  **else** System.***out***.println(**"No elements found"**);   *// -------------- The Methods Declared by List Iterator -------------------  // Modify objects being iterated* ListIterator litr = al.listIterator();   **while**(litr.hasNext()) { *//hasnext* Object element = litr.next(); *//next* litr.set(element + **"+"**); *//set* }  System.***out***.print(**"Modified contents of al: "**);  itr = al.iterator();  *// while(itr.hasNext()) { // Object element = itr.next(); // System.out.print(element + " "); // } // System.out.println(); // // // Now, display the list backwards // System.out.print("Modified list backwards: "); // // while(litr.hasPrevious()) { //hasprevious // Object element = litr.previous(); //previous (skips the last element) // System.out.print(element + " "); // } // System.out.println();* **while** (litr.hasNext()){  litr.add(**"Z"**);  Object element = litr.next();  System.***out***.println(element + **"+"**);  }  *// System.out.println("adding at beginning"); // litr.add("Z"); // while (litr.hasPrevious()){ // Object element = litr.previous(); // System.out.println(element + "+"); // }* } } | Original contents of al: C A E B D F  No elements found  Modified contents of al: |

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// Create an array list* ArrayList al = **new** ArrayList();   *// add elements to the array list* al.add(**"C"**);  al.add(**"A"**);  al.add(**"E"**);  al.add(**"B"**);  al.add(**"D"**);  al.add(**"F"**);   *// -------------- The Methods Declared by Iterator -------------------  // Use iterator to display contents of al* System.***out***.print(**"Original contents of al: "**);  Iterator itr = al.iterator();   **while**(itr.hasNext()) { *//.hasNext()* Object element = itr.next(); *//.next()* System.***out***.print(element + **" "**);  }  System.***out***.println();   itr.remove(); *//remove* **if** (itr.hasNext())  System.***out***.println(itr.next());  **else** System.***out***.println(**"No elements found"**);   *// -------------- The Methods Declared by List Iterator -------------------  // Modify objects being iterated* ListIterator litr = al.listIterator();   **while**(litr.hasNext()) { *//hasnext* Object element = litr.next(); *//next* litr.set(element + **"+"**); *//set* }  System.***out***.print(**"Modified contents of al: "**);  itr = al.iterator();  *// while(itr.hasNext()) { // Object element = itr.next(); // System.out.print(element + " "); // } // System.out.println(); // // // Now, display the list backwards // System.out.print("Modified list backwards: "); // // while(litr.hasPrevious()) { //hasprevious // Object element = litr.previous(); //previous (skips the last element) // System.out.print(element + " "); // } // System.out.println();  // while (litr.hasNext()){ // litr.add("Z"); // Object element = litr.next(); // System.out.println(element + "+"); // }* System.***out***.println(**"adding at beginning"**);  litr.add(**"Z"**);  **while** (litr.hasPrevious()){  Object element = litr.previous();  System.***out***.println(element + **"+"**);  }   } } | Original contents of al: C A E B D F  No elements found  Modified contents of al: adding at beginning  Z+  D++  B++  E++  A++  C++ |

**boolean remove(Object obj) & boolean removeAll(Collection c)**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  *// Create an array list* ArrayList al = **new** ArrayList();   *// add elements to the array list* al.add(**"C"**);  al.add(**"A"**);  al.add(**"E"**);  al.add(**"B"**);  al.add(**"D"**);  al.add(**"F"**);   **if** (al.remove(**"A"**)){  **for** (**int** i=0; i < al.size(); i++){  System.***out***.println(al.get(i));  }  }  System.***out***.println(**"not present"**);  **if** (al.remove(**"X"**)) {  **for** (**int** i = 0; i < al.size(); i++) {  System.***out***.println(al.get(i));  }  }   System.***out***.println(**"remove all function"**);  al.removeAll(al);  **for** (**int** i=0; i < al.size(); i++){  System.***out***.println(al.get(i));  }  } } | C  E  B  D  F  not present  remove all function |

**boolean retainAll(Collection c)**

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  ArrayList<Integer> arr = **new** ArrayList<Integer>();  arr.add(4);  arr.add(6);  arr.add(8);  arr.add(10);  arr.add(12);   ArrayList<Integer> arr1 = **new** ArrayList<Integer>();  arr1.add(3);  arr1.add(5);  arr1.add(6);  arr1.add(9);  arr1.add(11);   System.***out***.println(**"retains only the elements "** +  **"that are same"** +  **" in both Arraylists"** +  **" removes others"**);  arr.retainAll(arr1);  **for** (Integer i : arr)  System.***out***.println(i);   System.***out***.println(**"here's the proof that"** +  **" now arr has only the matched"** +  **" element"**);  **for** (**int** i = 0; i < arr.size(); i++)  System.***out***.println(arr.get(i));   System.***out***.println(**"when arr1 has all the items"**);  **for** (**int** i = 0; i < arr1.size(); i++)  System.***out***.println(arr1.get(i));  } } | retains only the elements that are same in both Arraylists removes others  6  here's the proof that now arr has only the matched element  6  when arr1 has all the items  3  5  6  9  11 |

.toArray(<Object>)

|  |  |
| --- | --- |
| **package** com.company;  **import** java.util.\*; **public class** Main {   **public static void** main(String[] args) {  ArrayList<Integer> arr = **new** ArrayList<Integer>();  arr.add(4);  arr.add(6);  arr.add(8);  arr.add(10);  arr.add(12);   **for** (Integer x : arr)  System.***out***.println(x);   *//converting an arraylist into an array* System.***out***.printf(**"converting in array"**);  Integer array[] = **new** Integer[arr.size()];  array = arr.toArray(array);   **for** (Integer y : array)  System.***out***.println(y);  } } | 4  6  8  10  12  converting in array4  6  8  10  12 |