|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **[next →](https://www.javatpoint.com/Sorting-in-collection-framework)**[**← prev**](https://www.javatpoint.com/java-enummap)  Java Collections class  Java collection class is used exclusively with static methods that operate on or return collections.  It inherits Object class.  The important points about Java Collections class are:   * Java Collection class supports the **polymorphic algorithms** that operate on collections. * Java Collection class throws a **NullPointerException** if the collections or class objects   provided to them are null.  Collections class declaration  Let's see the declaration for java.util.Collections class.   1. **public** **class** Collections **extends** Object  |  |  |  |  | | --- | --- | --- | --- | | **SN** | **Modifier & Type** | **Methods** | **Descriptions** | | 1) | static <T> boolean | [addAll()](https://www.javatpoint.com/java-collections-addall-method) | It is used to adds all of the specified elements  to the specified collection. | | 2) | static <T> Queue<T> | [asLifoQueue()](https://www.javatpoint.com/java-collections-aslifoqueue-method) | It returns a view of a Deque as a Last-in-first-out  (LIFO) Queue. | | 3) | static <T> int | [binarySearch()](https://www.javatpoint.com/java-collections-binarysearch-method) | It searches the list for the specified object and  returns their position in a sorted list. | | 4) | static <E> Collection<E> | [checkedCollection()](https://www.javatpoint.com/java-collections-checkedcollection-method) | It is used to returns a dynamically typesafe view  of the specified collection. | | 5) | static <E> List<E> | [checkedList()](https://www.javatpoint.com/java-collections-checkedlist-method) | It is used to returns a dynamically typesafe view  of the specified list. | | 6) | static <K,V> Map<K,V> | [checkedMap()](https://www.javatpoint.com/java-collections-checkedmap-method) | It is used to returns a dynamically typesafe view  of the specified map. | | 7) | static <K,V> NavigableMap<K,V> | [checkedNavigableMap()](https://www.javatpoint.com/java-collections-checkednavigablemap-method) | It is used to returns a dynamically typesafe view  of the specified navigable map. | | 8) | static <E> NavigableSet<E> | [checkedNavigableSet()](https://www.javatpoint.com/java-collections-checkednavigableset-method) | It is used to returns a dynamically typesafe view  of the specified navigable set. | | 9) | static <E> Queue<E> | [checkedQueue()](https://www.javatpoint.com/java-collections-checkedqueue-method) | It is used to returns a dynamically typesafe view  of the specified queue. | | 10) | static <E> Set<E> | [checkedSet()](https://www.javatpoint.com/java-collections-checkedset-method) | It is used to returns a dynamically typesafe view  of the specified set. | | 11) | static <K,V> SortedMap<K,V> | [checkedSortedMap()](https://www.javatpoint.com/java-collections-checkedsortedmap-method) | It is used to returns a dynamically typesafe view  of the specified sorted map. | | 12) | static <E> SortedSet<E> | [checkedSortedSet()](https://www.javatpoint.com/java-collections-checkedsortedset-method) | It is used to returns a dynamically typesafe view  of the specified sorted set. | | 13) | static <T> void | [copy()](https://www.javatpoint.com/java-collections-copy-method) | It is used to copy all the elements from one list  into another list. | | 14) | static boolean | [disjoint()](https://www.javatpoint.com/java-collections-disjoint-method) | It returns true if the two specified collections have  no elements in common. | | 15) | static <T> Enumeration<T> | [emptyEnumeration()](https://www.javatpoint.com/java-collections-emptyenumeration-method) | It is used to get an enumeration that has no  elements. | | 16) | static <T> Iterator<T> | [emptyIterator()](https://www.javatpoint.com/java-collections-emptyiterator-method) | It is used to get an Iterator that has no elements. | | 17) | static <T> List<T> | [emptyList()](https://www.javatpoint.com/java-collections-emptylist-method) | It is used to get a List that has no elements. | | 18) | static <T> ListIterator<T> | [emptyListIterator()](https://www.javatpoint.com/java-collections-emptylistiterator-method) | It is used to get a List Iterator that has no  elements. | | 19) | static <K,V> Map<K,V> | [emptyMap()](https://www.javatpoint.com/java-collections-emptymap-method) | It returns an empty map which is immutable. | | 20) | static <K,V> NavigableMap<K,V> | [emptyNavigableMap()](https://www.javatpoint.com/java-collections-emptynavigablemap-method) | It returns an empty navigable map which is  immutable. | | 21) | static <E> NavigableSet<E> | [emptyNavigableSet()](https://www.javatpoint.com/java-collections-emptynavigableset-method) | It is used to get an empty navigable set which  is immutable in nature. | | 22) | static <T> Set<T> | [emptySet()](https://www.javatpoint.com/java-collections-emptyset-method) | It is used to get the set that has no elements. | | 23) | static <K,V> SortedMap<K,V> | [emptySortedMap()](https://www.javatpoint.com/java-collections-emptysortedmap-method) | It returns an empty sorted map which is immutable. | | 24) | static <E> SortedSet<E> | [emptySortedSet()](https://www.javatpoint.com/java-collections-emptysortedset-method) | It is used to get the sorted set that has no  elements. | | 25) | static <T> Enumeration<T> | [enumeration()](https://www.javatpoint.com/java-collections-enumeration-method) | It is used to get the enumeration over the  specified collection. | | 26) | static <T> void | [fill()](https://www.javatpoint.com/java-collections-fill-method) | It is used to replace all of the elements of the  specified list with the specified elements. | | 27) | static int | [frequency()](https://www.javatpoint.com/java-collections-frequency-method) | It is used to get the number of elements in the  specified collection equal to the specified object. | | 28) | static int | [indexOfSubList()](https://www.javatpoint.com/java-collections-indexofsublist-method) | It is used to get the starting position of the first  occurrence of the specified target list within the  specified source list. It returns -1 if there is no  such occurrence in the specified list. | | 29) | static int | [lastIndexOfSubList()](https://www.javatpoint.com/java-collections-lastindexofsublist-method) | It is used to get the starting position of the last  occurrence of the specified target list within the  specified source list. It returns -1 if there is no  such occurrence in the specified list. | | 30) | static <T> ArrayList<T> | [list()](https://www.javatpoint.com/java-collections-list-method) | It is used to get an array list containing the  elements returned by the specified enumeration  in the order in which they are returned by the  enumeration. | | 31) | static <T extends Object & Comparable<? super T>> T | [max()](https://www.javatpoint.com/java-collections-max-method) | It is used to get the maximum value of the given  collection, according to the natural ordering of  its elements. | | 32) | static <T extends Object & Comparable<? super T>> T | [min()](https://www.javatpoint.com/java-collections-min-method) | It is used to get the minimum value of the given  collection, according to the natural ordering of its  elements. | | 33) | static <T> List<T> | [nCopies()](https://www.javatpoint.com/java-collections-ncopies-method) | It is used to get an immutable list consisting of **n**   copies of the specified object. | | 34) | static <E> Set<E> | [newSetFromMap()](https://www.javatpoint.com/java-collections-newsetfrommap-method) | It is used to return a set backed by the specified  map. | | 35) | static <T> boolean | [replaceAll()](https://www.javatpoint.com/java-collections-replaceall-method) | It is used to replace all occurrences of one  specified value in a list with the other specified  value. | | 36) | static void | [reverse()](https://www.javatpoint.com/java-collections-reverse-method) | It is used to reverse the order of the elements  in the specified list. | | 37) | static <T> Comparator<T> | [reverseOrder()](https://www.javatpoint.com/java-collections-reverseorder-method) | It is used to get the comparator that imposes the  reverse of the natural ordering on a collection  of objects which implement the Comparable  interface. | | 38) | static void | [rotate()](https://www.javatpoint.com/java-collections-rotate-method) | It is used to rotate the elements in the specified  list by a given distance. | | 39) | static void | [shuffle()](https://www.javatpoint.com/java-collections-shuffle-method) | It is used to randomly reorders the specified list  elements using a default randomness. | | 40) | static <T> Set<T> | [singleton()](https://www.javatpoint.com/java-collections-singleton-method) | It is used to get an immutable set which contains  only the specified object. | | 41) | static <T> List<T> | [singletonList()](https://www.javatpoint.com/java-collections-singletonlist-method) | It is used to get an immutable list which contains  only the specified object. | | 42) | static <K,V> Map<K,V> | [singletonMap()](https://www.javatpoint.com/java-collections-singletonmap-method) | It is used to get an immutable map, mapping  only the specified key to the specified value. | | 43) | static <T extends Comparable<? super T>>void | [sort()](https://www.javatpoint.com/java-collections-sort-method) | It is used to sort the elements presents in the  specified list of collection in ascending order. | | 44) | static void | [swap()](https://www.javatpoint.com/java-collections-swap-method) | It is used to swap the elements at the specified  positions in the specified list. | | 45) | static <T> Collection<T> | [synchronizedCollection()](https://www.javatpoint.com/java-collections-synchronizedcollection-method) | It is used to get a synchronized (thread-safe)  collection backed by the specified collection. | | 46) | static <T> List<T> | [synchronizedList()](https://www.javatpoint.com/java-collections-synchronizedlist-method) | It is used to get a synchronized (thread-safe)  collection backed by the specified list. | | 47) | static <K,V> Map<K,V> | [synchronizedMap()](https://www.javatpoint.com/java-collections-synchronizedmap-method) | It is used to get a synchronized (thread-safe)  map backed by the specified map. | | 48) | static <K,V> NavigableMap<K,V> | [synchronizedNavigableMap()](https://www.javatpoint.com/java-collections-synchronizednavigablemap-method) | It is used to get a synchronized (thread-safe)  navigable map backed by the specified navigable  map. | | 49) | static <T> NavigableSet<T> | [synchronizedNavigableSet()](https://www.javatpoint.com/java-collections-synchronizednavigableset-method) | It is used to get a synchronized (thread-safe)  navigable set backed by the specified navigable  set. | | 50) | static <T> Set<T> | [synchronizedSet()](https://www.javatpoint.com/java-collections-synchronizedset-method) | It is used to get a synchronized (thread-safe)  set backed by the specified set. | | 51) | static <K,V> SortedMap<K,V> | [synchronizedSortedMap()](https://www.javatpoint.com/java-collections-synchronizedsortedmap-method) | It is used to get a synchronized (thread-safe)  sorted map backed by the specified sorted map. | | 52) | static <T> SortedSet<T> | [synchronizedSortedSet()](https://www.javatpoint.com/java-collections-synchronizedsortedset-method) | It is used to get a synchronized (thread-safe)  sorted set backed by the specified sorted set. | | 53) | static <T> Collection<T> | [unmodifiableCollection()](https://www.javatpoint.com/java-collections-unmodifiablecollection-method) | It is used to get an unmodifiable view of the  specified collection. | | 54) | static <T> List<T> | [unmodifiableList()](https://www.javatpoint.com/java-collections-unmodifiablelist-method) | It is used to get an unmodifiable view of the  specified list. | | 55) | static <K,V> Map<K,V> | [unmodifiableMap()](https://www.javatpoint.com/java-collections-unmodifiablemap-method) | It is used to get an unmodifiable view of the  specified map. | | 56) | static <K,V> NavigableMap<K,V> | [unmodifiableNavigableMap()](https://www.javatpoint.com/java-collections-unmodifiablenavigablemap-method) | It is used to get an unmodifiable view of the  specified navigable map. | | 57) | static <T> NavigableSet<T> | [unmodifiableNavigableSet()](https://www.javatpoint.com/java-collections-unmodifiablenavigableset-method) | It is used to get an unmodifiable view of the  specified navigable set. | | 58) | static <T> Set<T> | [unmodifiableSet()](https://www.javatpoint.com/java-collections-unmodifiableset-method) | It is used to get an unmodifiable view of the  specified set. | | 59) | static <K,V> SortedMap<K,V> | [unmodifiableSortedMap()](https://www.javatpoint.com/java-collections-unmodifiablesortedmap-method) | It is used to get an unmodifiable view of the  specified sorted map. | | 60 | static <T> SortedSet<T> | [unmodifiableSortedSet()](https://www.javatpoint.com/java-collections-unmodifiablesortedset-method) | It is used to get an unmodifiable view of the  specified sorted set. |   Java Collections Example   1. **import** java.util.\*; 2. **public** **class** CollectionsExample { 3. **public** **static** **void** main(String a[]){ 4. List<String> list = **new** ArrayList<String>(); 5. list.add("C"); 6. list.add("Core Java"); 7. list.add("Advance Java"); 8. System.out.println("Initial collection value:"+list); 9. Collections.addAll(list, "Servlet","JSP"); 10. System.out.println("After adding elements collection value:"+list); 11. String[] strArr = {"C#", ".Net"}; 12. Collections.addAll(list, strArr); 13. System.out.println("After adding array collection value:"+list); 14. } 15. }   Output:  Initial collection value:[C, Core Java, Advance Java]  After adding elements collection value:[C, Core Java, Advance Java, Servlet, JSP]  After adding array collection value:[C, Core Java, Advance Java, Servlet, JSP, C#, .Net]  Java Collections Example: max()   1. **import** java.util.\*; 2. **public** **class** CollectionsExample { 3. **public** **static** **void** main(String a[]){ 4. List<Integer> list = **new** ArrayList<Integer>(); 5. list.add(46); 6. list.add(67); 7. list.add(24); 8. list.add(16); 9. list.add(8); 10. list.add(12); 11. System.out.println("Value of maximum element from the collection: "+Collections.max(list)); 12. } 13. }   Output:  Value of maximum element from the collection: 67  Java Collections Example: min()   1. **import** java.util.\*; 2. **public** **class** CollectionsExample { 3. **public** **static** **void** main(String a[]){ 4. List<Integer> list = **new** ArrayList<Integer>(); 5. list.add(46); 6. list.add(67); 7. list.add(24); 8. list.add(16); 9. list.add(8); 10. list.add(12); 11. System.out.println("Value of minimum element from the collection: "+Collections.min(list)); 12. } 13. }   Output:  Value of minimum element from the collection: 8 |