

**Collections methods in java**

The java.util.Collections class in Java provides a wide range of static utility methods for operating on collections such as Lists, Sets, and Maps. These methods implement polymorphic algorithms and are designed to work with any type of collection, but many are specifically optimized for Lists[[1]](#fn1)[[2]](#fn2)[[3]](#fn3).

Below is an overview of key methods available in the Collections class:

**Commonly Used Collections Methods**

* **addAll(Collection<? super T> c, T... elements)**
  + Adds all specified elements to the specified collection.
  + Example: Collections.addAll(list, "A", "B", "C");[[2]](#fn2)[[4]](#fn4)
* **copy(List<? super T> dest, List<? extends T> src)**
  + Copies all elements from the source list into the destination list.
  + The destination list must be at least as long as the source list, otherwise an IndexOutOfBoundsException is thrown.
  + Example: Collections.copy(dest, src);[[2]](#fn2)[[3]](#fn3)
* **sort(List<T> list)**
  + Sorts the specified list into ascending order, according to the natural ordering of its elements.
  + Example: Collections.sort(list);[[3]](#fn3)
* **sort(List<T> list, Comparator<? super T> c)**
  + Sorts the specified list according to the order induced by the specified comparator.
  + Example: Collections.sort(list, comparator);[[3]](#fn3)
* **reverse(List<?> list)**
  + Reverses the order of the elements in the specified list.
  + Example: Collections.reverse(list);[[3]](#fn3)
* **shuffle(List<?> list)**
  + Randomly permutes the specified list using a default source of randomness.
  + Example: Collections.shuffle(list);[[3]](#fn3)
* **rotate(List<?> list, int distance)**
  + Rotates the elements in the specified list by the specified distance.
  + Example: Collections.rotate(list, 2);[[3]](#fn3)
* **swap(List<?> list, int i, int j)**
  + Swaps the elements at the specified positions in the specified list.
  + Example: Collections.swap(list, 0, 1);[[3]](#fn3)
* **replaceAll(List<T> list, T oldVal, T newVal)**
  + Replaces all occurrences of one specified value in a list with another.
  + Example: Collections.replaceAll(list, "A", "B");[[3]](#fn3)
* **binarySearch(List<? extends Comparable<? super T>> list, T key)**
  + Searches the specified list for the specified object using the binary search algorithm.
  + The list must be sorted into ascending order according to the natural ordering of its elements.
  + Example: int index = Collections.binarySearch(list, "key");[[3]](#fn3)
* **frequency(Collection<?> c, Object o)**
  + Returns the number of elements in the specified collection equal to the specified object.
  + Example: int count = Collections.frequency(collection, "A");[[3]](#fn3)
* **disjoint(Collection c1, Collection c2)**
  + Returns true if the two specified collections have no elements in common.
  + Example: boolean result = Collections.disjoint(collection1, collection2);[[3]](#fn3)
* **min(Collection<? extends T> coll)**
  + Returns the minimum element of the given collection, according to the natural ordering of its elements.
  + Example: T min = Collections.min(collection);[[3]](#fn3)[[5]](#fn5)
* **max(Collection<? extends T> coll)**
  + Returns the maximum element of the given collection, according to the natural ordering of its elements.
  + Example: T max = Collections.max(collection);[[3]](#fn3)[[5]](#fn5)
* **min(Collection<? extends T> coll, Comparator<? super T> comp)**
  + Returns the minimum element of the given collection, according to the order induced by the specified comparator.
  + Example: T min = Collections.min(collection, comparator);[[5]](#fn5)
* **max(Collection<? extends T> coll, Comparator<? super T> comp)**
  + Returns the maximum element of the given collection, according to the order induced by the specified comparator.
  + Example: T max = Collections.max(collection, comparator);[[5]](#fn5)
* **fill(List<? super T> list, T obj)**
  + Replaces all of the elements of the specified list with the specified element.
  + Example: Collections.fill(list, "X");[[5]](#fn5)

**Summary Table**

|  |  |
| --- | --- |
| Method Name | Description |
| addAll | Adds elements to a collection |
| copy | Copies elements from one list to another |
| sort | Sorts a list (natural or comparator order) |
| reverse | Reverses the order of elements in a list |
| shuffle | Randomly permutes elements in a list |
| rotate | Rotates elements in a list |
| swap | Swaps two elements in a list |
| replaceAll | Replaces all occurrences of a value in a list |
| binarySearch | Searches a sorted list using binary search |
| frequency | Counts occurrences of an element in a collection |
| disjoint | Checks if two collections have no common elements |
| min/max | Finds minimum/maximum element (natural or comparator order) |
| fill | Replaces all elements of a list with a specified value |

These methods make the Collections class a powerful utility for manipulating collections in Java[[1]](#fn1)[[2]](#fn2)[[3]](#fn3).

⁂

1. <https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html>

1. <https://jenkov.com/tutorials/java-collections/collections.html>

1. <https://www.javaguides.net/2018/07/java-util-collections-class-methods-guide.html>

1. <https://www.tutorialspoint.com/java/util/java_util_collections.htm>

1. <https://courses.cs.washington.edu/courses/cse341/98au/java/jdk1.2beta4/docs/api/java/util/Collections.html>