JAVA.UTIL.COLLECTIONS CLASS

Introduction

The **java.util.Collections** class consists exclusively of static methods that operate on or return collections.Following are the important points about Collections:

- It contains polymorphic algorithms that operate on collections, "wrappers", which return a new collection backed by a specified collection.
- The methods of this class all throw a NullPointerException if the collections or class objects provided to them are null.

Class declaration

Following is the declaration for **java.util.Collections** class:

```
public class Collections
   extends Object
```

Field

Following are the fields for **java.util.Collections** class:

- **static List EMPTY_LIST** -- This is the empty list *immutable*.
- **static Map EMPTY_MAP** -- This is the empty map *immutable*.
- **static Set EMPTY_SET** -- This is the empty set *immutable*.

Class methods

S.N.	Method & Description
1	static <T> boolean addAll*Collection < ?superT > c, T. . . elements* This method adds all of the specified elements to the specified collection.
2	static <T> Queue<T> asLifoQueue*Deque < T > deque* This method returns a view of a Deque as a Last-in-first-out *Lifo* Queue.
3	static <T> int binarySearch*List < ?extendsComparable < ?superT >> list, Tkey* This method searches the specified list for the specified object using the binary search algorithm.
4	static <T> int binarySearch*List < ?extendsT > list, Tkey, Comparator < ?superT < c* This method searches the specified list for the specified object using the binary search algorithm.
5	static <E> Collection<E> checkedCollection*Collection < E > c, Class < E > type*

This method returns a dynamically typesafe view of the specified collection.

6

static <E> List<E> checkedList*List < E > list, Class < E > type*

This method returns a dynamically typesafe view of the specified list.

7

static <K,V> Map<K,V> checkedMap*Map < K, V > m, Class < K > keyType, Class < V > valueType*

This method returns a dynamically typesafe view of the specified map.

8

static <E> Set<E> checkedSet*Set < E > s, Class < E > type*

This method returns a dynamically typesafe view of the specified set.

9

static <K,V> SortedMap<K,V> checkedSortedMap
SortedMap < K, V > m, Class < K > keyType, Class < V > valueType

This method returns a dynamically typesafe view of the specified sorted map.

10

static <E> SortedSet<E>checkedSortedSet*SortedSet < E > s, Class < E > type*

This method returns a dynamically typesafe view of the specified sorted set.

11

static <T> void copy*List < ?superT > dest, List < ?extendsT > src*

This method copies all of the elements from one list into another.

12

static boolean disjoint*Collection < ? > c1, Collection < ? > c2*

This method returns true if the two specified collections have no elements in common.

13

static <T> List<T> emptyList

This method returns the empty list *immutable*.

14

static <K,V> Map<K,V> emptyMap

This method returns the empty map *immutable*.

15

static <T> Set<T> emptySet

This method returns the empty set *immutable*.

16

static <T> Enumeration<T> enumeration*Collection < T > c*

This method returns an enumeration over the specified collection.

17

static <T> void fill*List < ?superT > list, Tobj*

This method replaces all of the elements of the specified list with the specified element.

18

static int frequency*Collection < ? > c, Objecto*

This method returns the number of elements in the specified collection equal to the specified object.

19

static int indexOfSubList*List < ? > source, List < ? > target*

This method returns the starting position of the first occurrence of the specified target list within the specified source list, or -1 if there is no such occurrence.

20

static int lastIndexOfSubList*List < ? > source, List < ? > target*

This method returns the starting position of the last occurrence of the specified target list within the specified source list, or -1 if there is no such occurrence.

21

static <T> ArrayList<T> list*Enumeration < T > e*

This method returns an array list containing the elements returned by the specified enumeration in the order they are returned by the enumeration.

22

static <T extends Object & Comparable<? super T> >T max*Collection < ?extendsT > coll*

This method returns the maximum element of the given collection, according to the natural ordering of its elements.

23

static <T> T max*Collection < ?extendsT > coll, Comparator < ?superT > comp*

This method returns the maximum element of the given collection, according to the order induced by the specified comparator.

24

static <T extends Object & Comparable<? super T>>T min*Collection < ?extendsT > coll*

This method Returns the minimum element of the given collection, according to the natural ordering of its elements.

25

static <T> T min*Collection < ?extendsT > coll, Comparator < ?superT > comp*

This method returns the minimum element of the given collection, according to the order induced by the specified comparator.

26

static <T> List<T> nCopies*intn, To*

This method returns an immutable list consisting of n copies of the specified object.

27

static <E> Set<E> newSetFromMap*Map < E, Boolean > map*

This method returns a set backed by the specified map.

28

static <T> boolean replaceAll*List < T > list, ToldVal, TnewVal*

This method replaces all occurrences of one specified value in a list with another.

29

static void reverse*List < ? > list*

This method reverses the order of the elements in the specified list.

30

static <T> Comparator<T> reverseOrder

This method returns a comparator that imposes the reverse of the natural ordering on a collection of objects that implement the Comparable interface.

31

static <T> Comparator<T> reverseOrder*Comparator < T > cmp*

This method returns a comparator that imposes the reverse ordering of the specified comparator.

32

static void rotate*List < ? > list, intdistance*

This method rotates the elements in the specified list by the specified distance.

33

static void shuffle*List < ? > list*

This method randomly permutes the specified list using a default source of randomness.

34

static void shuffle*List < ? > list, Randomrnd*

This method randomly permute the specified list using the specified source of randomness.

35

static <T> Set<T> singleton*To*

This method returns an immutable set containing only the specified object.

36

static <T> List<T> singletonList*To*

This method returns an immutable list containing only the specified object.

37

static <K,V> Map<K,V> singletonMap*Kkey, Vvalue*

This method returns an immutable map, mapping only the specified key to the specified value.

38

static <T extends Comparable<? super T>> void sort*List < T > list*

This method sorts the specified list into ascending order, according to the natural ordering of its elements.

39

static <T> void sort*List < T > list, Comparator < ?superT > c*

This method sorts the specified list according to the order induced by the specified comparator.

40

static void swap*List < ? > list, inti, intj*

This method swaps the elements at the specified positions in the specified list.

41

static <T> Collection<T> synchronizedCollection*Collection < T > c*

This method returns a synchronized *thread − safe* collection backed by the specified collection.

42

static <T> List<T> synchronizedList*List < T > list*

This method returns a synchronized *thread − safe* list backed by the specified list.

43

static <K,V> Map<K,V> synchronizedMap*Map < K, V > m*

This method returns a synchronized *thread − safe* map backed by the specified map.

44

static <T> Set<T> synchronizedSet*Set < T > s*

This method returns a synchronized *thread − safe* set backed by the specified set.

45

static <K,V> SortedMap<K,V> synchronizedSortedMap*SortedMap < K, V > m*

This method returns a synchronized *thread − safe* sorted map backed by the specified sorted map.

46

static <T> SortedSet<T> synchronizedSortedSet*SortedSet < T > s*

This method returns a synchronized *thread − safe* sorted set backed by the specified sorted set.

47

static <T> Collection<T> unmodifiableCollection*Collection < ?extendsT > c*

This method returns an unmodifiable view of the specified collection.

48

static <T> List<T> unmodifiableList*List < ?extendsT > list*

This method returns an unmodifiable view of the specified list.

49

static <K,V> Map<K,V> unmodifiableMap*Map < ?extendsK, ?extendsV > m*

This method returns an unmodifiable view of the specified map.

50 static <T> Set<T> unmodifiableSet(*Set < ?extendsT > s*)

This method returns an unmodifiable view of the specified set.

51 static <K,V> SortedMap<K,V> unmodifiableSortedMap(*SortedMap < K, ?extendsV > m*)

This method returns an unmodifiable view of the specified sorted map.

52 static <T> SortedSet<T> unmodifiableSortedSet(*SortedSet < T > s*)

This method returns an unmodifiable view of the specified sorted set.

Methods inherited

This class inherits methods from the following classes:

- java.util.Object