Java Collection: ArrayList Exercises

**1.** Write a Java program to create a new array list, add some colors (string) and print out the collection. 

**2.** Write a Java program to iterate through all elements in a array list. 

**3.** Write a Java program to insert an element into the array list at the first position. 

**4.** Write a Java program to retrieve an element (at a specified index) from a given array list. 

**5.** Write a Java program to update specific array element by given element. 

**6.** Write a Java program to remove the third element from a array list. 

**7.** Write a Java program to search an element in a array list. 

**8.** Write a Java program to sort a given array list. 

**9.** Write a Java program to copy one array list into another. 

**10.** Write a Java program to shuffle elements in a array list. 

**11.** Write a Java program to reverse elements in a array list. 

**12.** Write a Java program to extract a portion of a array list. 

**13.** Write a Java program to compare two array lists. 

**14.** Write a Java program of swap two elements in an array list. 

**15.** Write a Java program to join two array lists. 

**16.** Write a Java program to clone an array list to another array list. 

**17.** Write a Java program to empty an array list. 

**18.** Write a Java program to test an array list is empty or not. 

**19.** Write a Java program to trim the capacity of an array list the current list size. 

**20.** Write a Java program to increase the size of an array list. 

**21.** Write a Java program to replace the second element of a ArrayList with the specified element. 

**22.** Write a Java program to print all the elements of a ArrayList using the position of the elements. 

Java Collection: LinkedList Exercises

**1.** Write a Java program to append the specified element to the end of a linked list. 

**2.** Write a Java program to iterate through all elements in a linked list. 

**3.** Write a Java program to iterate through all elements in a linked list starting at the specified position. 

**4.** Write a Java program to iterate a linked list in reverse order. 

**5.** Write a Java program to insert the specified element at the specified position in the linked list. 

**6.** Write a Java program to insert elements into the linked list at the first and last position. 

**7.** Write a Java program to insert the specified element at the front of a linked list. 

**8.** Write a Java program to insert the specified element at the end of a linked list. 

**9.** Write a Java program to insert some elements at the specified position into a linked list. 

**10.** Write a Java program to get the first and last occurrence of the specified elements in a linked list. 

**11.** Write a Java program to display the elements and their positions in a linked list. 

**12.** Write a Java program to remove a specified element from a linked list. 

**13.** Write a Java program to remove first and last element from a linked list. 

**14.** Write a Java program to remove all the elements from a linked list. 

**15.** Write a Java program of swap two elements in an linked list. 

**16.** Write a Java program to shuffle the elements in a linked list. 

**17.** Write a Java program to join two linked lists. 

**18.** Write a Java program to clone an linked list to another linked list. 

**19.** Write a Java program to remove and return the first element of a linked list. 

**20.** Write a Java program to retrieve but does not remove, the first element of a linked list. 

**21.** Write a Java program to retrieve but does not remove, the last element of a linked list. 

**22.** Write a Java program to check if a particular element exists in a linked list. 

**23.** Write a Java program to convert a linked list to array list. 

**24.** Write a Java program to compare two linked lists. 

**25.** Write a Java program to test an linked list is empty or not. 

**26.** Write a Java program to replace an element in a linked list. 

Java Collection: HashSet Exercises

**1.** Write a Java program to append the specified element to the end of a hash set. 

**2.** Write a Java program to iterate through all elements in a hash list. 

**3.** Write a Java program to get the number of elements in a hash set. 

**4.** Write a Java program to empty an hash set. 

**5.** Write a Java program to test a hash set is empty or not. 

**6.** Write a Java program to clone a hash set to another hash set. 

**7.** Write a Java program to convert a hash set to an array. 

**8.** Write a Java program to convert a hash set to a tree set. 

**9.** Write a Java program to convert a hash set to a List/ArrayList. 

**10.** Write a Java program to compare two hash set. 

**11.** Write a Java program to compare two sets and retain elements which are same on both sets. 

**12.** Write a Java program to remove all of the elements from a hash set. 

Java Collection: TreeSet Exercises

**1.** Write a Java program to create a new tree set, add some colors (string) and print out the tree set. 

**2.** Write a Java program to iterate through all elements in a tree set. 

**3.** Write a Java program to add all the elements of a specified tree set to another tree set. 

**4.** Write a Java program to create a reverse order view of the elements contained in a given tree set. 

**5.** Write a Java program to get the first and last elements in a tree set. 

**6.** Write a Java program to clone a tree set list to another tree set. 

**7.** Write a Java program to get the number of elements in a tree set. 

**8.** Write a Java program to compare two tree sets. 

**9.** Write a Java program to find the numbers less than 7 in a tree set. 

**10.** Write a Java program to get the element in a tree set which is greater than or equal to the given element. 

**11.** Write a Java program to get the element in a tree set which is less than or equal to the given element. 

**12.** Write a Java program to get the element in a tree set which is strictly greater than or equal to the given element. 

**13.** Write a Java program to get an element in a tree set which is strictly less than the given element. 

**14.** Write a Java program to retrieve and remove the first element of a tree set. 

**15.** Write a Java program to retrieve and remove the last element of a tree set. 

**16.** Write a Java program to remove a given element from a tree set. 

Java Collection: PriorityQueue Exercises

**1.** Write a Java program to create a new priority queue, add some colors (string) and print out the elements of the priority queue. 

**2.** Write a Java program to iterate through all elements in priority queue. 

**3.** Write a Java program to add all the elements of a priority queue to another priority queue. 

**4.** Write a Java program to insert a given element into a priority queue. 

**5.** Write a Java program to remove all the elements from a priority queue. 

**6.** Write a Java program to count the number of elements in a priority queue. 

**7.** Write a Java program to compare two priority queues. 

**8.** Write a Java program to retrieve the first element of the priority queue. 

**9.** Write a Java program to retrieve and remove the first element. 

**10.** Write a Java program to convert a priority queue to an array containing all of the elements of the queue. 

**11.** Write a Java program to convert a Priority Queue elements to a string representation. 

**12.** Write a Java program to change priorityQueue to maximum priorityqueue. 

Java Collection: HashMap Exercises

**1.** Write a Java program to associate the specified value with the specified key in a HashMap. 

**2.** Write a Java program to count the number of key-value (size) mappings in a map. 

**3.** Write a Java program to copy all of the mappings from the specified map to another map. 

**4.** Write a Java program to remove all of the mappings from a map. 

**5.** Write a Java program to check whether a map contains key-value mappings (empty) or not. 

**6.** Write a Java program to get a shallow copy of a HashMap instance. 

**7.** Write a Java program to test if a map contains a mapping for the specified key. 

**8.** Write a Java program to test if a map contains a mapping for the specified value. 

**9.** Write a Java program to create a set view of the mappings contained in a map. 

**10.** Write a Java program to get the value of a specified key in a map. 

**11.** Write a Java program to get a set view of the keys contained in this map. 

**12.** Write a Java program to get a collection view of the values contained in this map. 

Java Collection: TreeMap Exercises

**1.** Write a Java program to associate the specified value with the specified key in a Tree Map. 

**2.** Write a Java program to copy a Tree Map content to another Tree Map. 

**3.** Write a Java program to search a key in a Tree Map. 

**4.** Write a Java program to search a value in a Tree Map. 

**5.** Write a Java program to get all keys from the given a Tree Map. 

**6.** Write a Java program to delete all elements from a given Tree Map. 

**7.** Write a Java program to sort keys in Tree Map by using comparator. 

**8.** Write a Java program to get a key-value mapping associated with the greatest key and the least key in a map. 

**9.** Write a Java program to get the first (lowest) key and the last (highest) key currently in a map. 

**10.** Write a Java program to get a reverse order view of the keys contained in a given map. 

**11.** Write a Java program to get a key-value mapping associated with the greatest key less than or equal to the given key. 

**12.** Write a Java program to get the greatest key less than or equal to the given key. 

**13.** Write a Java program to get the portion of a map whose keys are strictly less than a given key. 

**14.** Write a Java program to get the portion of this map whose keys are less than (or equal to, if inclusive is true) a given key. 

**15.** Write a Java program to get the least key strictly greater than the given key. Return null if there is no such key. 

**16.** Write a Java program to get a key-value mapping associated with the greatest key strictly less than the given key. Return null if there is no such key. 

**17.** Write a Java program to get the greatest key strictly less than the given key. Return null if there is no such key. 

**18.** Write a Java program to get NavigableSet view of the keys contained in a map. 

**19.** Write a Java program to remove and get a key-value mapping associated with the least key in a map. 

**20.** Write a Java program to remove and get a key-value mapping associated with the greatest key in this map. 

**21.** Write a Java program to get the portion of a map whose keys range from a given key (inclusive), to another key (exclusive). 

**22.** Write a Java program to get the portion of a map whose keys range from a given key to another key. 

**23.** Write a Java program to get a portion of a map whose keys are greater than or equal to a given key. 

**24.** Write a Java program to get a portion of a map whose keys are greater than to a given key. 

**25.** Write a Java program to get a key-value mapping associated with the least key greater than or equal to the given key. Return null if there is no such key. 

**26.** Write a Java program to get the least key greater than or equal to the given key. Returns null if there is no such key.