**Collections**

The Collections Framework consists of three parts:

|  |
| --- |
| 1. ***interfaces***, the abstract data types that the framework supports. 2. ***implementations***, the concrete versions of these interfaces. 3. ***algorithms***, the predefined actions that can be defined on  either the interfaces or their implementations. |
| 1. All implementations are unsynchronized or synchronized. 2. All implementations are serializable and cloneable 3. All implementations support having null elements. |
| The predefined algorithms for supporting the framework are found in the  Collections and Arrays classes.   |  | | --- | | The framework consists of four core interfaces with two specializations for sorting | | 1. ***Collection*** 2. ***List*** 3. ***Set*** 4. ***Map*** 5. ***SortedSet*** 6. ***SortedMap*** |   *Interface type and its implementation*   |  |  | | --- | --- | | **Interface Type** | **Implemented by** | | Set | HashSet, LinkedHashSet, EnumSet | | SortedSet | TreeSet | | List | Vector, Stack, ArrayList, LinkedList | | Queue | PriorityQueue, LinkedList | | Map | Hashtable, HashMap, LinkedHashMap, WeakHashMap, IdentityHashMap | | SortedMap | TreeMap |   ***Conversion Of Different Collection Data Types***  Implementations of the Collection interface normally have a constructor that accepts a  Collection object. This enables you to convert a Collection to a different type of Collection,  such as a Queue to a List, or a List to a Set, etc.  Here are the constructors of some implementations:    **public**ArrayList (Collection c)      **public**HashSet (Collection c)      **public**LinkedList (Collection c)  **Example : The following code implements Queue using LinkList.**  **import**java.util.ArrayList; **import**java.util.LinkedList; **import**java.util.List; **import**java.util.Queue;  **public class**Test {   **public static void**main(String[] args) {     Queue queue = **new**LinkedList();     queue.add("Hello");     queue.add("World");     List list = **new**ArrayList(queue);     System.out.println(list);   } }  Output  Hello World  Example : Implement List though ArrayList and set through hashset  **import**java.util.ArrayList; **import**java.util.HashSet; **import**java.util.List; **import**java.util.Set; **public class**MainClass {   **public static void**main(String[] args) {     List myList = **new**ArrayList();     myList.add("Hello");     myList.add("World");     myList.add("World");     Set set = **new**HashSet(myList);     System.out.println(set);   } } |

**Making Your Objects Comparable and Sortable**

|  |
| --- |
| You can make objects comparable by implementing the java.lang.Comparable and  java.util.Comparator interfaces. |
| Classes such as java.lang.String, java.util.Date, and primitive wrapper classes all  implement java.lang.Comparable. |

**Example : Sort Array through Arrays.sort method**

**import**java.util.Arrays;  
**public class T**est {  
  **public static void**main(String[] args) {  
    **int**[] intArray = **new int**[] { 5, 4, 3, 2, 1 };  
    Arrays.sort(intArray);  
    **for**(**int**i : intArray) {  
      System.out.println(i);  
    }  
  }  
}

output

1

2

3

4

5

**Example : Collections.synchronized methods**

**import**java.util.ArrayList;  
**import**java.util.Collection;  
**import**java.util.Collections;  
**import**java.util.HashMap;  
**import**java.util.HashSet;  
**import**java.util.List;  
**import**java.util.Map;  
**import**java.util.Set;  
  
**public class**MainClass {  
  **public static void**main(String[] args) {  
    Collection c = Collections.synchronizedCollection(**new**ArrayList());  
    List list = Collections.synchronizedList(**new**ArrayList());  
    Set s = Collections.synchronizedSet(**new**HashSet());  
    Map m = Collections.synchronizedMap(**new**HashMap());  
  }  
}

**Example : Get Synchronized List from ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**import**java.util.List;  
**public class**Test {  
  **public static void**main(String[] args) {  
    ArrayList arrayList = **new**ArrayList();  
    List list = Collections.synchronizedList(arrayList);  }  
}

**Example :Sort elements of ArrayList use Generics**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**public class T**est {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("1");  
    arrayList.add("3");  
    arrayList.add("5");  
    arrayList.add("2");  
    arrayList.add("4");  
    Collections.sort(arrayList);  
    **for**(String str: arrayList)  
      System.out.println(str);  
  }  
}

Example : Copy Elements of ArrayList to Java Vector

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**import**java.util.Vector;  
**public class**Main {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("1");  
    arrayList.add("2");  
    arrayList.add("3");  
    arrayList.add("4");  
    arrayList.add("5");  
  
    Vector<String> v = **new**Vector<String>();  
  
    v.add("A");  
    v.add("B");  
    v.add("D");  
    v.add("E");  
    v.add("F");  
    v.add("G");  
    v.add("H");  
  
    System.out.println(v);  
    Collections.copy(v, arrayList);  
    System.out.println(v);  
  }  
}

**Exampl : Copy Elements of One ArrayList to Another ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**public class**Test

 {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList1 = **new**ArrayList<String>();  
    arrayList1.add("1");  
    arrayList1.add("2");  
    arrayList1.add("3");  
  
    ArrayList<String> arrayList2 = **new**ArrayList<String>();  
  
    arrayList2.add("One");  
    arrayList2.add("Two");  
    arrayList2.add("Three");  
    arrayList2.add("Four");  
    arrayList2.add("Five");  
    System.out.println(arrayList2);  
    Collections.copy(arrayList2, arrayList1);  
    System.out.println(arrayList2);  
  }  
}

Example : Find maximum element of ArrayList

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**class**Test {  
  **public static void**main(String[] args) {  
    ArrayList<Integer> arrayList = **new**ArrayList<Integer>();  
    arrayList.add(**new**Integer("3"));  
    arrayList.add(**new**Integer("1"));  
    arrayList.add(**new**Integer("8"));  
    arrayList.add(**new**Integer("3"));  
    arrayList.add(**new**Integer("5"));  
  
    Object obj = Collections.max(arrayList);  
    System.out.println(obj);  
  }  
}

**Example : Get Enumeration over ArrayList. Enmeration are used to read ArrayList Sequentially**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**import**java.util.Enumeration;  
**class**Test {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("A");  
    arrayList.add("B");  
    arrayList.add("D");  
    arrayList.add("E");  
    arrayList.add("F");  
  
    Enumeration e = Collections.enumeration(arrayList);  
    **while**(e.hasMoreElements())  
      System.out.println(e.nextElement());  
  }  
}

**Example : Perform Binary Search on ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
  
**public class**Main {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("1");  
    arrayList.add("4");  
    arrayList.add("2");  
    arrayList.add("5");  
    arrayList.add("3");  
  
    Collections.sort(arrayList);  
    System.out.println("Sorted ArrayList contains : " + arrayList);  
    **int**index = Collections.binarySearch(arrayList, "4");  
    System.out.println("Element found at : " + index);  
  }  
}

**Example : Replace All Elements Of ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**class T**est {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("A");  
    arrayList.add("B");  
    arrayList.add("D");  
  
    System.out.println(arrayList);  
    Collections.fill(arrayList, "REPLACED");  
    System.out.println(arrayList);  
  }  
}

Output A B D

REPLACED REPLACED REPLACED

**Example: Replace all occurrences of specified element of ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**class**Test {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("A");  
    arrayList.add("B");  
    arrayList.add("A");  
    arrayList.add("C");  
    arrayList.add("D");  
    System.out.println(arrayList);  
    Collections.replaceAll(arrayList, "A", "Replace All");  
    System.out.println(arrayList);  
  }  
}  
/\*  
[A, B, A, C, D]  
[Replace All, B, Replace All, C, D]  
\*/

**Example : Reverse order of all elements of ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**class**Test {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("A");  
    arrayList.add("B");  
    arrayList.add("C");  
    arrayList.add("D");  
    arrayList.add("E");  
    System.out.println(arrayList);  
    Collections.reverse(arrayList);  
    System.out.println(arrayList);  
  }  
}

**Example : Shuffle elements of ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**class T**est {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("A");  
    arrayList.add("B");  
    arrayList.add("C");  
    arrayList.add("D");  
    arrayList.add("E");  
    System.out.println(arrayList);  
  
    Collections.shuffle(arrayList);  
    System.out.println(arrayList);  
  }  
}  
/\*  
[A, B, C, D, E]  
[E, D, A, C, B]  
\*/

**Example : Swapping elements of ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**class**Test {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("A");  
    arrayList.add("B");  
    arrayList.add("C");  
    arrayList.add("D");  
    arrayList.add("E");  
    System.out.println(arrayList);  
    Collections.swap(arrayList, 0, 4);  
    System.out.println(arrayList);  
  }  
}  
/\*  
[A, B, C, D, E]  
[E, B, C, D, A]  
\*/

**Example : Create an empty collection object**

**import**java.util.Collections;  
**import**java.util.Date;  
**import**java.util.List;  
**import**java.util.Map;  
**import**java.util.Set;  
  
**class Test**{  
  **public static void**main(String args[]) {  
    List list = Collections.EMPTY\_LIST;  
    Set set = Collections.EMPTY\_SET;  
    Map map = Collections.EMPTY\_MAP;  
    List<String> s = Collections.emptyList();  
    Set<Long> l = Collections.emptySet();  
    Map<Date, String> d = Collections.emptyMap();  
  }  
}

**Example : Filling ArrayList Elements with common data using**

**Collections.fill() method**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**import**java.util.List;  
**class**Test {  
  **public static void**main(String[] args) {  
    List list = **new**ArrayList();  
    **for**(**int**i = 0; i < 10; i++)  
      list.add("");  
    Collections.fill(list, "Hello");  
    System.out.println(list);  
  }  
}

Hello Hello Hello Hello Hello Hello Hello Hello Hello

**Example: Create and demonstrate an immutable collection**

**import**java.util.ArrayList;  
**import**java.util.Collection;  
**import**java.util.Collections;  
**import**java.util.List;  
**class**Test {  
  **public static void**main(String args[]) {  
    List<Character> list = **new**ArrayList<Character>();  
    list.add('X');  
    System.out.println("Element added to list: " + list.get(0));  
    Collection<Character> immutableCol = Collections.unmodifiableCollection(list);  
    immutableCol.add('Y');  
  }  
}

Example : Shifting Elements in an Array: Shift all elements right by one

**import**java.util.Arrays;  
**public class**Main {  
  **public static void**main(String[] argv) **throws**Exception {  
    **int**[] array = { 1, 2, 3 };  
        System.arraycopy(array, 0, array, 1, array.length - 1);  
    System.out.println(Arrays.toString(array));  
  }  
}  
//[1, 1, 2]

**ArrayList Class**

|  |
| --- |
| The ArrayList class is the Collection Framework's replacement for the Vector class. |
| The main difference is: ArrayList is not synchronized by default, whereas Vector is.  Both maintain their data in an ordered fashion. |
| Ordered means the elements are held according to the positional index inserted,  sorted refers to the comparison of element values for ordering. |
| The ArrayList and Vector provide quick, random access to elements at a cost of  slower insertion and deletion of those elements not at the end of the list. |
| If you need to frequently add and delete elements from the middle of the list,  consider using a LinkedList.  **Creating an ArrayList**   1. For the first two constructors, an empty array list is created. 2. The initial capacity is ten unless explicitly specified by using  the second constructor.   **public**ArrayList() **public**ArrayList (**int**initialCapacity)  For Sun's reference implementation, the formula to increase capacity is  newCapacity= (oldCapacity \* 3)/2 + 1.  **Example A boolean is being stored and then retrieved from an ArrayList**  **import**java.util.ArrayList; **public class**Main {   **public static void**main(String... args) {     ArrayList<Boolean> list = **new**ArrayList<Boolean>();     list.add(**true**);     **boolean**flag = list.get(0);   } }  **Example: Creating a new ArrayList from another collection**  **import**java.util.ArrayList; **import**java.util.Arrays; **import**java.util.List; **class**Test {   **public static void**main(String args[]) **throws**Exception {     String elements[] = { "A", "B", "C", "D" };     List list = **new**ArrayList(Arrays.asList(elements));     System.out.println(list);   } } |

**Example : Adding Single Elements**

Adding a single element to the list by calling the add() method:

**public boolean**add(Object element)  
  
**public boolean**add(**int**index, Object element)

**Example : Add elements in array List**

**import**java.util.ArrayList;  
**import**java.util.List;  
**public class**Test {  
  **public static void**main(String args[]) **throws**Exception {  
    // Create/fill collection  
    List list = **new**ArrayList();  
    list.add("A");  
    list.add("B");  
    list.add("C");  
    System.out.println(list);  
  }  
}

**Example: Adding elements in the middle of a List**

**import**java.util.ArrayList;  
**import**java.util.List;  
**public class**MainClass {  
  **public static void**main(String args[]) **throws**Exception {  
    List list = **new**ArrayList();  
    list.add("A");  
    list.add("B");  
    list.add("C");  
    list.add(1, "G");  
    System.out.println(list);  
  }  
}

Output

A G B C

**Example : Append all elements of other Collection to ArrayList**

**import**java.util.ArrayList;  
**import**java.util.Vector;  
**class Test** {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("1");  
    arrayList.add("2");  
    arrayList.add("3");  
  
    Vector<String> v = **new**Vector<String>();  
    v.add("4");  
    v.add("5");  
  
    // append all elements of Vector to ArrayList  
    arrayList.addAll(v);  
  
    **for**(String str : arrayList)  
      System.out.println(str);  
  }  
}  
/\*  
1  
2  
3  
4  
5  
\*/

**Example : To create a read-only list, use the unmodifiableList() method of the Collections class**

**import**java.util.ArrayList;  
**import**java.util.Collections;  
**import**java.util.List;  
**public class Test** {  
  **public static void**main(String args[]) **throws**Exception {  
    List list = **new**ArrayList();  
    list.add("A");  
    list.add("B");  
    list.add("C");  
    list = Collections.unmodifiableList(list);  
    list.add(1, "G");  
  
    System.out.println(list);  
  
  }  
}

Output

Exception in thread "main" java.lang.UnsupportedOperationException

at java.util.Collections$UnmodifiableList.add(Collections.java:1160)

at MainClass.main(MainClass.java:14)

**Example : Getting an Element**

**import**java.util.ArrayList;  
**import**java.util.List;  
**public class**Test {  
  **public static void**main(String args[]) **throws**Exception {  
    List list = **new**ArrayList();  
    list.add("A");  
    list.add("B");  
    list.add("C");  
    System.out.println(list.get(2));  
  
  }  
}

**Example: Get Sub List of ArrayList**

**import**java.util.ArrayList;  
**import**java.util.List;  
**public class**Test {  
  **public static void**main(String[] args) {  
    ArrayList<String> arrayList = **new**ArrayList<String>();  
    arrayList.add("1");  
    arrayList.add("2");  
    arrayList.add("3");  
    arrayList.add("4");  
    arrayList.add("5");  
    List lst = arrayList.subList(1, 3);  
    **for**(**int**i = 0; i < lst.size(); i++)  
      System.out.println(lst.get(i));  
    // remove one element from sub list  
    Object obj = lst.remove(0);  
    System.out.println(obj + " is removed");  
  
    **for**(String str: arrayList)  
      System.out.println(str);

 }  
}

**Removing a Single Element**

|  |
| --- |
| Use the remove() method to remove a single element: |
| 1. public boolean remove(Object element) 2. Removes a single element by checking for equality, 3. Returns true if the object was found and removed from the list, 4. Otherwise, false is returned. |
| If the list contains duplicates, the first element in the list that matches the element will be removed. |
| If removal is not supported, you'll get an UnsupportedOperationException. |
| If the index passed in is outside the valid range of elements, an  IndexOutOfBoundsException is thrown. |

**Example : Remove element from ArrayList using Index and value**

**import**java.util.ArrayList;  
**import**java.util.List;  
**public class**Test {  
  **public static void**main(String args[]) **throws**Exception {  
    List list = **new**ArrayList();  
    list.add("A");  
    list.add("B");  
    list.add("C");  
    System.out.println(list.remove(0));  
    System.out.println(list.remove("B"));  
    System.out.println(list);  
  }  
}

A

true

[C]

**Example: Fetching Elements with iterator**

**public**Iterator iterator()  
**public**ListIterator listIterator()  
**public**ListIterator listIterator(**int**index)

Using an iterator from a List is like using any other iterator. The order of the elements in the list is preserved.

**import**java.util.Arrays;  
**import**java.util.Iterator;  
**import**java.util.List;  
**public class T**est {  
  **public static void**main(String[] a) {  
    List list = Arrays.asList(**new**String[] { "A", "B", "C", "D" });  
    Iterator iter = list.iterator();  
    **while**(iter.hasNext()) {  
      System.out.println(iter.next());  
    }  
  }  
}

A

B

C

D

**Example : Checking for Existence: public boolean contains(Object element)**

**import**java.util.Arrays;  
**import**java.util.List;  
**class**Test {  
  **public static void**main(String[] a) {  
    List list = Arrays.asList(**new**String[] { "A", "B", "C", "D" });  
    System.out.println(list.contains("C"));  
  }  
}

**Example : Checking for Position**

**public int**indexOf(Object element)  
**public int**lastIndexOf(Object element)

|  |
| --- |
| return -1 if the element is not found. |
| To find all of the positions for a single element in an ArrayList, you'll need to convert  the list to a Vector and use the versions of indexOf() or lastIndexOf().  **import**java.util.Arrays; **import**java.util.List; **class**Test {   **public static void**main(String[] a) {     List list = Arrays.asList(**new**String[] { "A", "B", "C", "D" });     System.out.println(list.lastIndexOf("A"));   } }  **Example: Checking Capacity**   |  | | --- | | The capacity is the number of elements the array list can hold before the internal  data structure has to resize. | | Use the ensureCapacity() method to check that the internal data structure has  enough capacity before adding elements: |   **public void**ensureCapacity(**int**minimumCapacity)  **import**java.util.ArrayList; **class Test** {   **public static void**main(String[] a) {     ArrayList list = **new**ArrayList();     list.add("A");     list.ensureCapacity(10);     System.out.println(list.size());   } }  **Example : Remove duplicate items from an ArrayList using Hashset conversion**  **import**java.util.ArrayList; **import**java.util.HashSet; **import**java.util.List; **public class Test** {   **public static void**main(String[] argv) {     List<String> arrayList1 = **new**ArrayList<String>();     arrayList1.add("A");     arrayList1.add("A");     arrayList1.add("B");     arrayList1.add("B");     arrayList1.add("B");     arrayList1.add("C");     HashSet<String> hashSet = **new**HashSet<String>(arrayList1);     List<String> arrayList2 = **new**ArrayList<String>(hashSet);     **for**(Object item : arrayList2)       System.out.println(item);   } } /\*A B C \*/  **Example : Looping through a Collection object: while loop, iterator, and for each**  **import**java.util.ArrayList; **import**java.util.Iterator; **class Test** {   **public static void**main(String[] args) {     ArrayList<String> list = **new**ArrayList<String>();     list.add("Monday");     list.add("Tuesdag");     list.add("Wednesday");     list.add("Thursday");     list.add("Friday");     list.add("Saturday");     list.add("Sunday");     Iterator<String> iterator = **null**;     iterator = list.iterator();     **while**(iterator.hasNext()) {       String element = iterator.next();       System.out.println(element);     }     **for**(iterator = list.iterator(); iterator.hasNext();) {       String element = iterator.next();       System.out.println(element);     }      **for**(String element : list) {       System.out.println(element);     }   } } /\* Monday Tuesdag Wednesday Thursday Friday Saturday Sunday Monday Tuesdag Wednesday Thursday Friday Saturday Sunday Monday Tuesdag Wednesday Thursday Friday Saturday Sunday \*/  **Example : If an ArrayList contains a given item**  **import**java.util.ArrayList; **import**java.util.List; **class Test** {   **public static void**main(String[] args) {     List list = **new**ArrayList();     list.add("Item 1");     list.add("Item 2");     **if**(list.contains("Item 1")) {       System.out.println("True");     } **else**{       System.out.println("False");     }   } } |

**LinkedList Class**

|  |
| --- |
| The LinkedList class is a doubly linked list, which internally maintains references to  the previous and next element at each node in the list. |
| Creating a LinkedList |
| 1. public LinkedList(): creating an empty list 2. public LinkedList(Collection col): copy constructor |

**Example: Get first and last elements from LinkedList**

**import**java.util.LinkedList;  
**class Test** {  
  **public static void**main(String[] args) {  
    LinkedList<String> lList = **new**LinkedList<String>();  
    lList.add("1");  
    lList.add("2");  
    lList.add("3");  
    lList.add("4");  
    lList.add("5");  
    System.out.println("First element of LinkedList is : " + lList.getFirst());  
    System.out.println("Last element of LinkedList is : " + lList.getLast());  
  }  
}

**Example : Get elements from LinkedList**

**import**java.util.LinkedList;  
**class Test** {  
  **public static void**main(String[] args) {  
    LinkedList<String> lList = **new**LinkedList<String>();  
    lList.add("1");  
    lList.add("2");  
    lList.add("3");  
    lList.add("4");  
    lList.add("5");  
    **for**(String str: lList) {  
      System.out.println(str);  
    }  
  }  
}

**Example : Get SubList from LinkedList**

**import**java.util.LinkedList;  
**import**java.util.List;  
**class Test** {  
  **public static void**main(String[] args) {  
    LinkedList<String> lList = **new**LinkedList<String>();  
    lList.add("1");  
    lList.add("2");  
    lList.add("3");  
    lList.add("4");  
    lList.add("5");  
    System.out.println(lList);  
    List lst = lList.subList(1, 4);  
    System.out.println(lst);  
    lst.remove(2);  
    System.out.println(lst);  
    System.out.println(lList);  
  }  
}

**Example : Adding Elements: add a single element**

|  |
| --- |
| Adding the element to the end of the list unless an index is specified. |
| |  | | --- | | **public boolean**add(Object element) **public boolean**add(**int**index, Object element) | |
| To Treat the linked list as a stack or queue: |
| |  | | --- | | **public boolean**addFirst(Object element) **public boolean**addLast(Object element) | |
| The addLast() method is equivalent to call add(), with no index. |
| |  | | --- | | **import**java.util.LinkedList; **class**Test {   **public static void**main(String[] a) {     LinkedList list = **new**LinkedList();     list.add("A");     list.add("B");     list.add("C");     list.add("D");     list.addFirst("X");     list.addLast("Z");     System.out.println(list);   } } | |

**Example : Removing Elements**

|  |  |
| --- | --- |
| |  | | --- | | **public**Object removeFirst() **public**Object removeLast() | |
| |  | | --- | | **import**java.util.LinkedList; **class**Test {   **public static void**main(String[] a) {      LinkedList list = **new**LinkedList();     list.add("A");     list.add("B");     list.add("C");     list.add("D");     list.addFirst("X");     list.addLast("Z");     System.out.println(list);     System.out.println(list.getFirst());     System.out.println(list.getLast());     list.removeFirst();     list.removeLast();     System.out.println(list);   } } | |

[X, A, B, C, D, Z]

X

Z

[A, B, C, D]

**Example : Making a stack from a LinkedList (Stack as an Adaptor class)**

**import**java.util.LinkedList;  
**class**Test {  
  **public static void**main(String[] args) {  
    StackL stack = **new**StackL();  
    **for**(**int**i = 0; i < 10; i++)  
      stack.push(i);  
    System.out.println(stack.top());  
    System.out.println(stack.top());  
    System.out.println(stack.pop());  
    System.out.println(stack.pop());  
    System.out.println(stack.pop());  
  }  
  
}  
  
**class**StackL {  
  
  **private**LinkedList list = **new**LinkedList();  
  **public void**push(Object v) {  
    list.addFirst(v);  
  }  
  **public**Object top() {  
    **return**list.getFirst();  
  }  
  **public**Object pop() {  
    **return**list.removeFirst();  
  }  
}

9

9

9

8

7

**Example Making a queue from a LinkedList (Queue as an Adaptor class)**

**import**java.util.LinkedList;  
**class Test** {  
  **public static void**main(String[] args) {  
    Queue queue = **new**Queue();  
    **for**(**int**i = 0; i < 10; i++)  
      queue.put(Integer.toString(i));  
    **while**(!queue.isEmpty())  
      System.out.println(queue.get());  
  }  
  
}  
**class**Queue {  
  **private**LinkedList list = **new**LinkedList();  
  **public void**put(Object v) {  
    list.addFirst(v);  
  }  
  **public**Object get() {  
    **return**list.removeLast();  
  }  
  **public boolean**isEmpty() {  
    **return**list.isEmpty();  
  }  
}

0

1

2

3

4

5

6

7

8

9

**Example :Search elements of LinkedList**

**import**java.util.LinkedList;  
**class Test** {  
  **public static void**main(String[] args) {  
    LinkedList<String> lList = **new**LinkedList<String>();  
    lList.add("1");  
    lList.add("2");  
    lList.add("3");  
    lList.add("4");  
    lList.add("5");  
    lList.add("2");  
  
    System.out.println(lList.indexOf("2"));  
  
    System.out.println(lList.lastIndexOf("2"));  
  
  }  
}

**Example :Replace an Element of LinkedList**

**import**java.util.LinkedList;  
**public class**Main {  
  **public static void**main(String[] args) {  
    LinkedList<String> lList = **new**LinkedList<String>();  
    lList.add("1");  
    lList.add("2");  
    lList.add("3");  
    lList.add("4");  
    lList.add("5");  
    System.out.println(lList);  
    lList.set(3, "Replaced");  
    System.out.println(lList);  
  }  
}

**HashSet**

|  |
| --- |
| 1. A Set represents a mathematical set. 2. It is a Collection that, unlike List, does not allow duplicates. 3. There must not be two elements of a Set, say e1 and e2, such that e1.equals(e2). 4. The add method of Set returns false if you try to add a duplicate element. |
| **Example : Creating HashSet**   |  | | --- | | **import**java.util.HashSet; **import**java.util.Set; **class**Test {   **public static void**main(String[] a) {     Set set = **new**HashSet();     set.add("Hello");     **if**(set.add("Hello")) {       System.out.println("addition successful");     } **else**{       System.out.println("addition failed");     }   } } | |

addition failed

**Example :An easy way to initialize a set without manually adding each element**

**import**java.util.Arrays;  
**import**java.util.HashSet;  
**import**java.util.Set;  
**class**Test {  
  **public static void**main(String[] a) {  
    String elements[] = { "A", "B", "C", "D", "E" };  
    Set set = **new**HashSet(Arrays.asList(elements));  
    System.out.println(set);  
  
  }  
  
}

[D, A, C, B, E]

**TreeSet Class**

1. The other concrete Set implementation is the TreeSet.
2. A TreeSet keeps its elements ordered internally.
3. The tree is balanced, it's a red-black tree.
4. Having a balanced tree guarantees a quick o(log n) search time at the cost of a more time-intensive insertion (and deletion).
5. Elements added to the tree must be orderable.

|  |
| --- |
| Red-black tree rules refresher: |
| 1. Every node in the tree is either black or red. 2. The root is always black. 3. If a node is red, its children must be black. 4. Every path from the root to a leaf (or null child) must contain the same number of black nodes. |

**Example : Creating a TreeSet**

**import**java.util.Collections;  
**import**java.util.Set;  
**import**java.util.TreeSet;  
**class Test** {  
  **public static void**main(String args[]) **throws**Exception {  
    String elements[] = { "A", "C", "D", "G", "F" };  
    Set set = **new**TreeSet();  
    **for**(**int**i = 0, n = elements.length; i < n; i++) {  
      set.add(elements[i]);  
    }  
    System.out.println(set);

}  
}

**Example : Check if a particular value exists in TreeSet**

**import**java.util.TreeSet;  
**class Test** {  
  **public static void**main(String[] args) {  
    TreeSet<String> tSet = **new**TreeSet<String>();  
    tSet.add("1");  
    tSet.add("2");  
    tSet.add("3");  
    tSet.add("4");  
    tSet.add("5");  
    **boolean**blnExists = tSet.contains("3");  
    System.out.println("3 exists in TreeSet ? : " + blnExists);  
  }  
}

**Vector**

Java vectors: dynamically sized arrays with synchronized access.

**Example : Creating Vector**

**import**java.util.Vector;  
**class**Test {  
  **public static void**main(String args[]) {  
    Vector v = **new**Vector(5);  
    **for**(**int**i = 0; i < 10; i++) {  
      v.add(i);  
    }  
    System.out.println(v);  
  }  
}

**Example : Search an element of Java Vector**

**import**java.util.Vector;  
**class Test** {  
  **public static void**main(String[] args) {  
    Vector<String> v = **new**Vector<String>();  
    v.add("1");  
    v.add("2");  
    v.add("3");  
    v.add("4");  
    v.add("5");  
    v.add("1");  
    v.add("2");  
  
    System.out.println(v.contains("3"));  
    System.out.println(v.indexOf("5"));  
    System.out.println(v.lastIndexOf("2"));  
  }  
}