Collection Framework Overview

The CF in java provides set of interfaces and classes to manage the group of objects.

It has readymade implementations of data structures.

* Framework: A predefined structure which provides reusable designs.
* Iterable Interface: Root interface which allows traversing through elements using iterators.
* Collection Interface: Extends the iterable. It defines the common methods like adding, removing and checking the size of the collection.

[Collection (Java SE 21 & JDK 21)](https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Collection.html)

* List Interface

Extends the collection interface and represents an ordered collection of elements.

* Allows the duplicate elements
* Maintains the insertion order
* List Implementations
* ArrayList
* LinkedList
* Vector

1. ArrayList

Characteristics:

* Uses dynamic array internally
* It provides faster random access using indices
* Not synchronized, so it’s not thread-safe
* package arrayList;  
    
  import java.util.ArrayList;  
    
  public class ArrayListExample {  
   public static void main(String[] args) {  
   ArrayList<String> names = new ArrayList<>();  
    
   // Add elements  
   names.add("Ramesh");  
   names.add("Krishna");  
   names.add("Govinda");  
   names.add("Gopal");  
    
   // Access element  
   System.*out*.println("First Name: "+ names.get(0));  
    
    
   //Iterating the elements  
   for(int i=0; i < names.size();i++){  
   System.*out*.println(names.get(i));  
   }  
    
   //Removing an element  
   names.remove("Ramesh");  
    
   System.*out*.println("After removing an element: "+ names);  
   }  
  }

[ArrayList (Java SE 21 & JDK 21)](https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/ArrayList.html#add(E))

Methods:

* add() : adds an element to the list
* get(int index) : retrieves the element at that index
* remove(int index)/ remove(object o): Removes an element by index or object
* size(): returns the size of list.
* Contains(Object o): To check if a list contains the specified element
* Clear(): Removes all elements.
* [**set**](https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/ArrayList.html#set(int,E))(int index, [**E**](https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/ArrayList.html) element) : Replaces the element at the specified position in this list with the specified element.
* LinkedList  
  Characteristics:

- Internally uses doubly linked list structure

- Efficient for insertions and deletions

[LinkedList (Java SE 21 & JDK 21)](https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/LinkedList.html)  
  
package linkedListExample;  
  
import java.util.LinkedList;  
  
public class LinkedListExample {  
 public static void main(String[] args) {  
 LinkedList<String> tasks = new LinkedList<>();  
  
 // add elements  
 tasks.add("Read");  
 tasks.add("Write");  
 tasks.add("Exercise");  
  
 //Add elements at specified positions  
 tasks.addFirst("Wake Up");  
 tasks.addLast("Sleep");  
  
 //To access the first task  
 System.*out*.println("First Task: "+ tasks.getFirst());  
  
 //Iterate elements  
 for(String task: tasks){  
 System.*out*.println(task);  
 }  
  
 //to remove elements  
 tasks.removeFirst();  
 tasks.removeLast();  
  
 System.*out*.println("After removal:"+ tasks);  
 }  
}

Methods of LinkedList:

* addFirst(): add element at the beginning
* addLast() : add elelement at last
* getFirst()/getLast(): retrieves the first or last element
* add(int index, E element) : Insert an element at specified position.
* Vector:

Characteristics:

* Similar to arraylist, but synchronized(thread-safe)
* Slower performance compared to arraylist due synchronization.
* It is automatically resizable.
* package vectorExample;  
    
  import java.util.Vector;  
    
  public class VectorExample {  
   public static void main(String[] args) {  
   Vector<Integer> numbers = new Vector<>();  
   // add elements  
   numbers.add(10);  
   numbers.add(20);  
   numbers.add(30);  
    
   // to access elements  
   System.*out*.println("First Element: "+ numbers.get(0));  
    
   // to iterate the elements  
   for (Integer number: numbers){  
   System.*out*.println(number);  
   }  
    
   // Remove an element  
   numbers.remove(20);  
   System.*out*.println("after removal:" + numbers);  
   }  
  }