**How to Sort Vector using Collections.sort in java – Example**

[**JAVA COLLECTIONS**](http://beginnersbook.com/category/java-collections/)

[**Vector**](http://beginnersbook.com/2013/12/vector-in-java/) maintains the insertion order which means it displays the elements in the same order, in which they got added to the Vector. In this example, we will see how to sort Vector elements in ascending order by using[**Collections.sort()**](http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#sort(java.util.List)). The Steps are as follows:  
1) Create a Vector object  
2) Add elements to the Vector using add(Element e) method  
3) Sort it using Collections.sort(Vector object)  
4) Display the sorted elements list.

import java.util.Collections;

import java.util.Vector;

public class SortingVectorExample {

public static void main(String[] args) {

// Create a Vector

Vector<String> vector = new Vector<String>();

//Add elements to Vector

vector.add("Walter");

vector.add("Anna");

vector.add("Hank");

vector.add("Flynn");

vector.add("Tom");

// By Default Vector maintains the insertion order

System.out.println("Vector elements before sorting: ");

for(int i=0; i < vector.size(); i++){

//get(i) method fetches the element from index i

System.out.println(vector.get(i));

}

// Collection.sort() sorts the collection in ascending order

Collections.sort(vector);

//Display Vector elements after sorting using Collection.sort

System.out.println("Vector elements after sorting: :");

for(int i=0; i < vector.size(); i++){

System.out.println(vector.get(i));

}

}

}

**Output:**

Vector elements before sorting:

Walter

Anna

Hank

Flynn

Tom

Vector elements after sorting: :

Anna

Flynn

Hank

Tom

Walter