import java.util.Scanner;

class LinearSearchExample

{

public static void main(String args[])

{

int i, num, item, array[];

//To capture user input

Scanner input = new Scanner(System.in);

System.out.println("Enter number of elements:");

num = input.nextInt();

//Creating array to store the all the numbers

array = new int[num];

System.out.println("Enter " + num + " integers");

//Loop to store each numbers in array

for (i = 0; i < num; i++)

array[i] = input.nextInt();

System.out.println("Enter the search value:");

item = input.nextInt();

for (i = 0; i < num; i++)

{

if (array[i] == item)

{

System.out.println(item+" is present at location "+(i+1));

/\*Item is found so to stop the search and to come out of the

\* loop use break statement.\*/

break;

}

}

if (i == num)

System.out.println(item + " doesn't exist in array.");

}

}

Output 1:

Enter number of elements:

6

Enter 6 integers

22

33

45

1

3

99

Enter the search value:

45

45 is present at location 3

**Binary search**

import java.util.Scanner;

class BinarySearchExample

{

public static void main(String args[])

{

int i, num, item, array[], first, last, middle;

//To capture user input

Scanner input = new Scanner(System.in);

System.out.println("Enter number of elements:");

num = input.nextInt();

//Creating array to store the all the numbers

array = new int[num];

System.out.println("Enter " + num + " integers");

//Loop to store each numbers in array

for (i = 0; i < num; i++)

array[i] = input.nextInt();

System.out.println("Enter the search value:");

item = input.nextInt();

first = 0;

last = num - 1;

middle = (first + last)/2;

while( first <= last )

{

if ( array[middle] < item )

first = middle + 1;

else if ( array[middle] == item )

{

System.out.println(item + " found at location " + (middle + 1) + ".");

break;

}

else

{

last = middle - 1;

}

middle = (first + last)/2;

}

if ( first > last )

System.out.println(item + " is not found.\n");

}

}

Output :

Enter number of elements:

7

Enter 7 integers

4

5

66

77

8

99

0

Enter the search value:

77

77 found at location 4.

**public** **class** BubbleSortExample {

**static** **void** bubbleSort(**int**[] arr) {

**int** n = arr.length;

**int** temp = 0;

**for**(**int** i=0; i < n; i++){

**for**(**int** j=1; j < (n-i); j++){

**if**(arr[j-1] > arr[j]){

                                 //swap elements

                                 temp = arr[j-1];

                                 arr[j-1] = arr[j];

                                 arr[j] = temp;

                         }

                 }

         }

    }

**public** **static** **void** main(String[] args) {

**int** arr[] ={3,60,35,2,45,320,5};

                System.out.println("Array Before Bubble Sort");

**for**(**int** i=0; i < arr.length; i++){

                        System.out.print(arr[i] + " ");

                }

                System.out.println();

                bubbleSort(arr);//sorting array elements using bubble sort

                System.out.println("Array After Bubble Sort");

**for**(**int** i=0; i < arr.length; i++){

                        System.out.print(arr[i] + " ");

                }

        }

}

**public** **class** MyMergeSort

{

**void** merge(**int** arr[], **int** beg, **int** mid, **int** end)

{

**int** l = mid - beg + 1;

**int** r = end - mid;

intLeftArray[] = **new** **int** [l];

intRightArray[] = **new** **int** [r];

**for** (**int** i=0; i<l; ++i)

LeftArray[i] = arr[beg + i];

**for** (**int** j=0; j<r; ++j)

RightArray[j] = arr[mid + 1+ j];

**int** i = 0, j = 0;

**int** k = beg;

**while** (i<l&&j<r)

{

**if** (LeftArray[i] <= RightArray[j])

{

arr[k] = LeftArray[i];

i++;

}

**else**

{

arr[k] = RightArray[j];

j++;

}

k++;

}

**while** (i<l)

{

arr[k] = LeftArray[i];

i++;

k++;

}

**while** (j<r)

{

arr[k] = RightArray[j];

j++;

k++;

}

}

**void** sort(**int** arr[], **int** beg, **int** end)

{

**if** (beg<end)

{

**int** mid = (beg+end)/2;

sort(arr, beg, mid);

sort(arr , mid+1, end);

merge(arr, beg, mid, end);

}

}

**public** **static** **void** main(String args[])

{

intarr[] = {90,23,101,45,65,23,67,89,34,23};

MyMergeSort ob = **new** MyMergeSort();

ob.sort(arr, 0, arr.length-1);

System.out.println("\nSorted array");

**for**(**int** i =0; i<arr.length;i++)

{

    System.out.println(arr[i]+"");

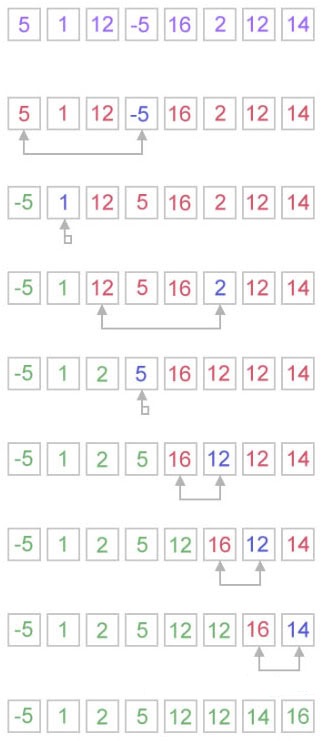
}

}

}

Selection Sort in Java

We can create a java program to sort array elements using selection sort. In selection sort algorithm, we search for the lowest element and arrange it to the proper location. We swap the current element with the next lowest number.



## **How does selection sort work?**

The selection sort algorithm works in a very simple way. It maintains two subarray for the given array.

* The subarray is already sorted.
* And the second subarray is unsorted.

With every iteration of selection sort, an element is picked from the unsorted subarray and moved to the sorted subarray.

**import** java.util.Scanner;

**public** **class** SelectionSortExample2

{

**public** **static** **void** main(String args[])

   {

**int** size, i, j, temp;

**int** arr[] = **new** **int**[50];

       Scanner scan = **new** Scanner(System.in);

       System.out.print("Enter Array Size : ");

       size = scan.nextInt();

       System.out.print("Enter Array Elements : ");

**for**(i=0; i<size; i++)

       {

           arr[i] = scan.nextInt();

       }

       System.out.print("Sorting Array using Selection Sort Technique..\n");

**for**(i=0; i<size; i++)

       {

**for**(j=i+1; j<size; j++)

           {

**if**(arr[i] > arr[j])

               {

                   temp = arr[i];

                   arr[i] = arr[j];

                   arr[j] = temp;

               }

           }

       }

       System.out.print("Now the Array after Sorting is :\n");

**for**(i=0; i<size; i++)

       {

           System.out.print(arr[i]+ "  ");

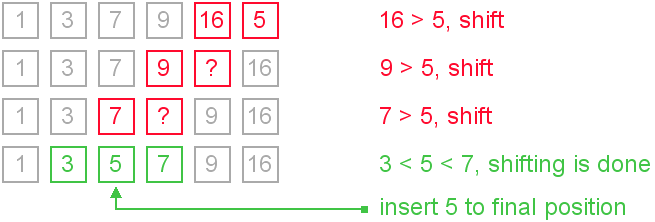
       }

   }

}

# Insertion Sort in Java

We can create a java program to sort array elements using insertion sort. Insertion is good for small elements only because it requires more time for sorting large number of elements.



**public** **class** InsertionSortExample {

**public** **static** **void** insertionSort(**int** array[]) {

**int** n = array.length;

**for** (**int** j = 1; j < n; j++) {

**int** key = array[j];

**int** i = j-1;

**while** ( (i > -1) && ( array [i] > key ) ) {

                array [i+1] = array [i];

                i--;

            }

            array[i+1] = key;

        }

    }

**public** **static** **void** main(String a[]){

**int** size, i, j, temp;

**int** arr[] = **new** **int**[50];

       Scanner scan = **new** Scanner(System.in);

       System.out.print("Enter Array Size : ");

       size = scan.nextInt();

       System.out.print("Enter Array Elements : ");

**for**(i=0; i<size; i++)

       {

           arr[i] = scan.nextInt();

       }

        System.out.println("Before Insertion Sort");

**for**(**int** i:arr){

            System.out.print(i+" ");

        }

        System.out.println();

        insertionSort(arr);//sorting array using insertion sort

        System.out.println("After Insertion Sort");

**for**(**int** i:arr){

            System.out.print(i+" ");

        }

    }

}