**Quick sort:**

import java.util.Arrays;

class Quicksort {

// method to find the partition position

static int partition(int array[], int low, int high) {

// choose the rightmost element as pivot

int pivot = array[high];

// pointer for greater element

int i = (low - 1);

// traverse through all elements

// compare each element with pivot

for (int j = low; j < high; j++) {

if (array[j] <= pivot) {

// if element smaller than pivot is found

// swap it with the greater element pointed by i

i++;

// swapping element at i with element at j

int temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

// swapt the pivot element with the greater element specified by i

int temp = array[i + 1];

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

array[high] = temp;

// return the position from where partition is done

return (i + 1);

}

static void quickSort(int array[], int low, int high) {

if (low < high) {

// find pivot element such that

// elements smaller than pivot are on the left

// elements greater than pivot are on the right

int pi = partition(array, low, high);

// recursive call on the left of pivot

quickSort(array, low, pi - 1);

// recursive call on the right of pivot

quickSort(array, pi + 1, high);

}

}

}

// Main class

class Main {

public static void main(String args[]) {

int[] data = { 8, 7, 2, 1, 0, 9, 6 };

System.out.println("Unsorted Array");

System.out.println(Arrays.toString(data));

int size = data.length;

// call quicksort() on array data

Quicksort.quickSort(data, 0, size - 1);

System.out.println("Sorted Array in Ascending Order ");

System.out.println(Arrays.toString(data));

}

}

**Output:**

Unsorted Array

[8, 7, 2, 1, 0, 9, 6]

Sorted Array in Ascending Order

[0, 1, 2, 6, 7, 8, 9]