**QuickSort**

// Java implementation of QuickSort

import java.io.\*;

class GFG{

// A utility function to swap two elements

static void swap(int[] arr, int i, int j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

/\* This function takes last element as pivot, places

the pivot element at its correct position in sorted

array, and places all smaller (smaller than pivot)

to left of pivot and all greater elements to right

of pivot \*/

static int partition(int[] arr, int low, int high)

{

// pivot

int pivot = arr[high];

// Index of smaller element and

// indicates the right position

// of pivot found so far

int i = (low - 1);

for(int j = low; j <= high - 1; j++)

{

// If current element is smaller

// than the pivot

if (arr[j] < pivot)

{

// Increment index of

// smaller element

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return (i + 1);

}

/\* The main function that implements QuickSort

arr[] --> Array to be sorted,

low --> Starting index,

high --> Ending index

\*/

static void quickSort(int[] arr, int low, int high)

{

if (low < high)

{

// pi is partitioning index, arr[p]

// is now at right place

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

// Separately sort elements before

// partition and after partition

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

// Function to print an array

static void printArray(int[] arr, int size)

{

for(int i = 0; i < size; i++)

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

System.out.println();

}

// Driver Code

public static void main(String[] args)

{

int[] arr = { 10, 7, 8, 9, 1, 5 };

int n = arr.length;

quickSort(arr, 0, n - 1);

System.out.println("Sorted array: ");

printArray(arr, n);

}

}