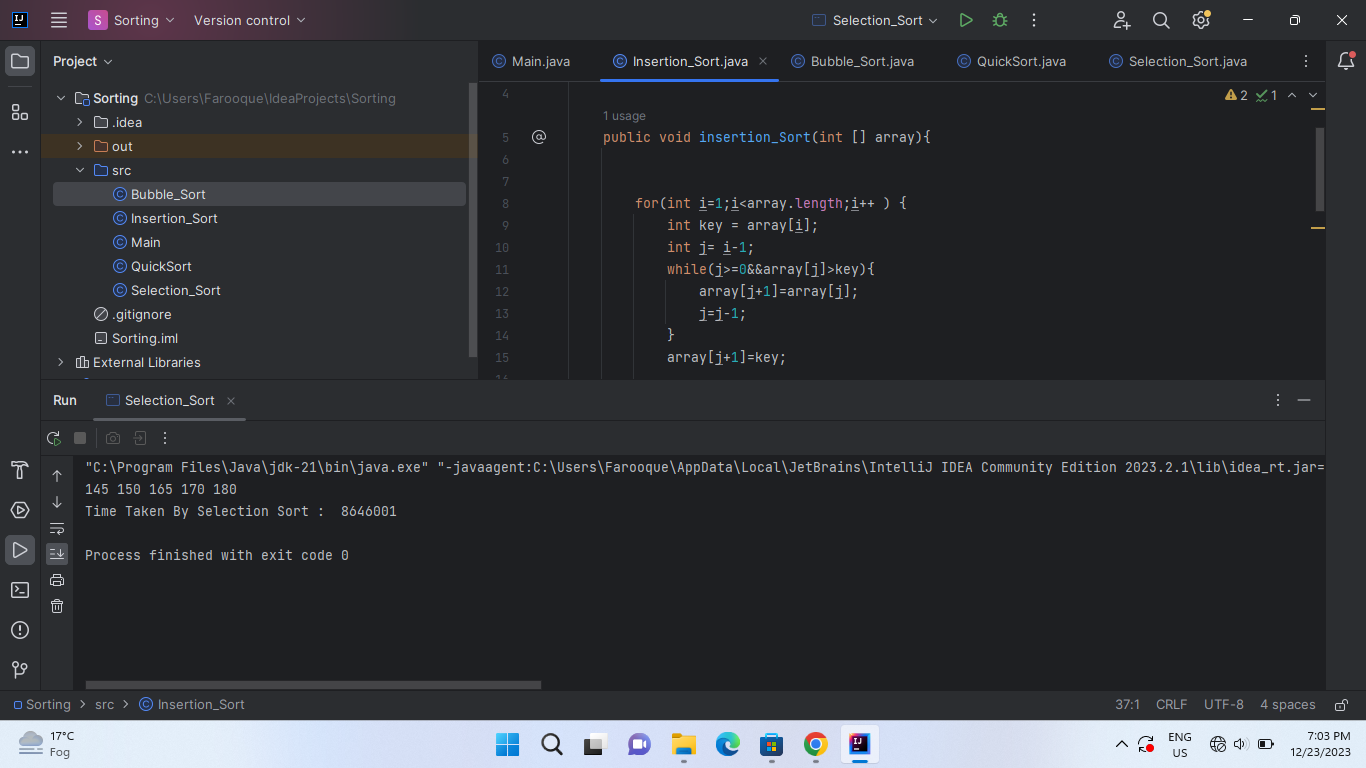
|  |  |
| --- | --- |
| **Student Name** | **Farooque Sajjad** |
| **Roll Number** | **22SW040** |
| **Section #** | **01** |
| **Lab #** | **04** |

***Question : No : 01***

**i)**

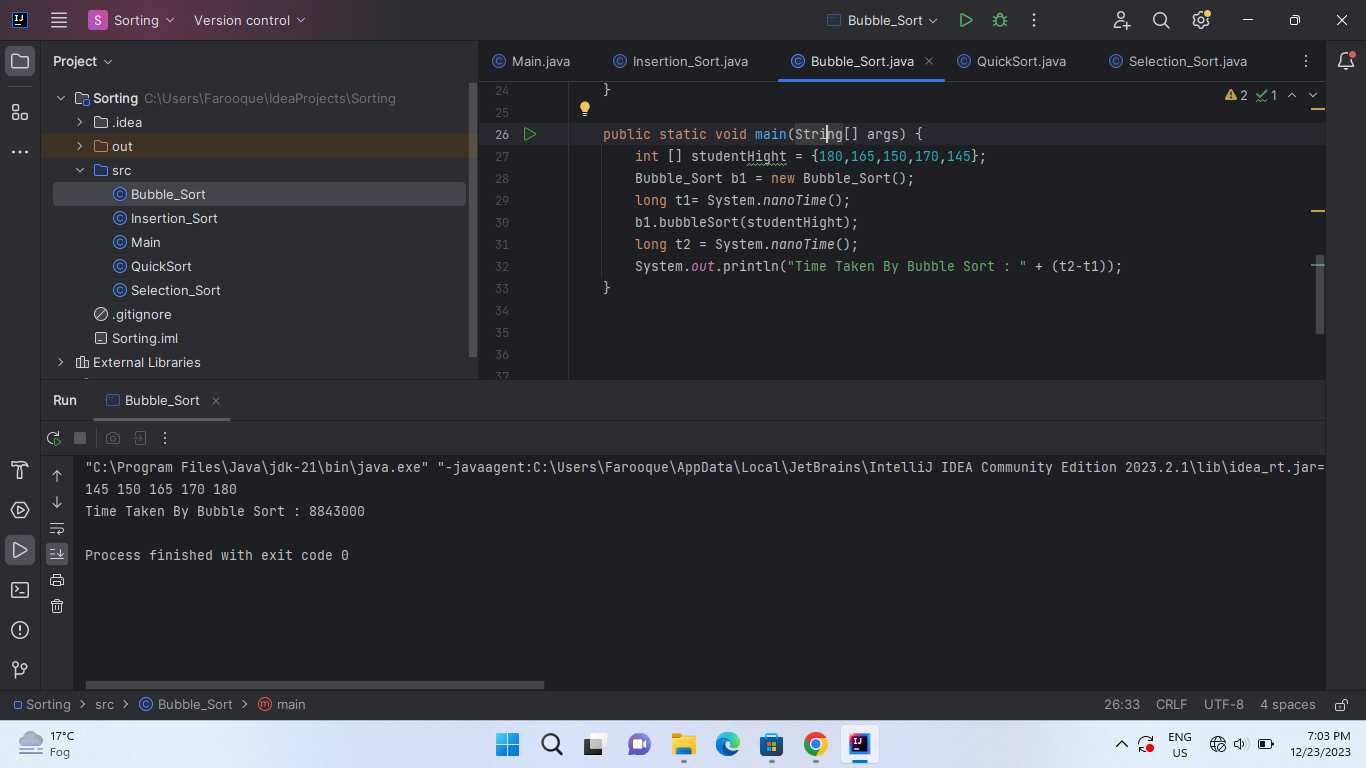
import java.util.Scanner;  
  
public class Selection\_Sort {  
 public void selectionSort(int [] array){  
 for(int i=0;i<array.length-1;i++){  
 int minIndex=i;  
 for(int j=i+1;j<array.length;j++){  
 if(array[i]>array[j]){  
 minIndex=j;  
 }  
 }  
 int temp = array[minIndex];  
 array[minIndex]=array[i];  
 array[i]=temp;  
 }  
  
 for(int i=0;i<array.length;i++){  
 System.*out*.print(array[i]+" ");  
 }  
 System.*out*.println();  
  
  
 }  
  
 public static void main(String[] args) {  
 int [] studentHight = {180,165,150,170,145};  
 Selection\_Sort s1= new Selection\_Sort();  
 long t1 = System.*nanoTime*();  
 s1.selectionSort(studentHight);  
 long t2 = System.*nanoTime*();  
 System.*out*.println("Time Taken By Selection Sort : " + (t2-t1));  
 }  
}



**ii)**

import java.util.Scanner;  
  
public class Bubble\_Sort {  
  
  
 public void bubbleSort(int [] array){  
  
 for(int i=0;i<array.length-1;i++){  
 for(int j=0;j<array.length-1-i;j++){  
 if(array[j]>array[j+1]){  
 int temp = array[j+1];  
 array[j+1]=array[j];  
 array[j]=temp;  
 }  
 }  
 }  
 for(int i=0;i<array.length;i++){  
 System.*out*.print(array[i]+" ");  
 }  
 System.*out*.println();  
  
  
  
 }  
  
 public static void main(String[] args) {  
 int [] studentHight = {180,165,150,170,145};  
 Bubble\_Sort b1 = new Bubble\_Sort();  
 long t1= System.*nanoTime*();  
 b1.bubbleSort(studentHight);  
 long t2 = System.*nanoTime*();  
 System.*out*.println("Time Taken By Bubble Sort : " + (t2-t1));  
 }  
  
  
  
  
}

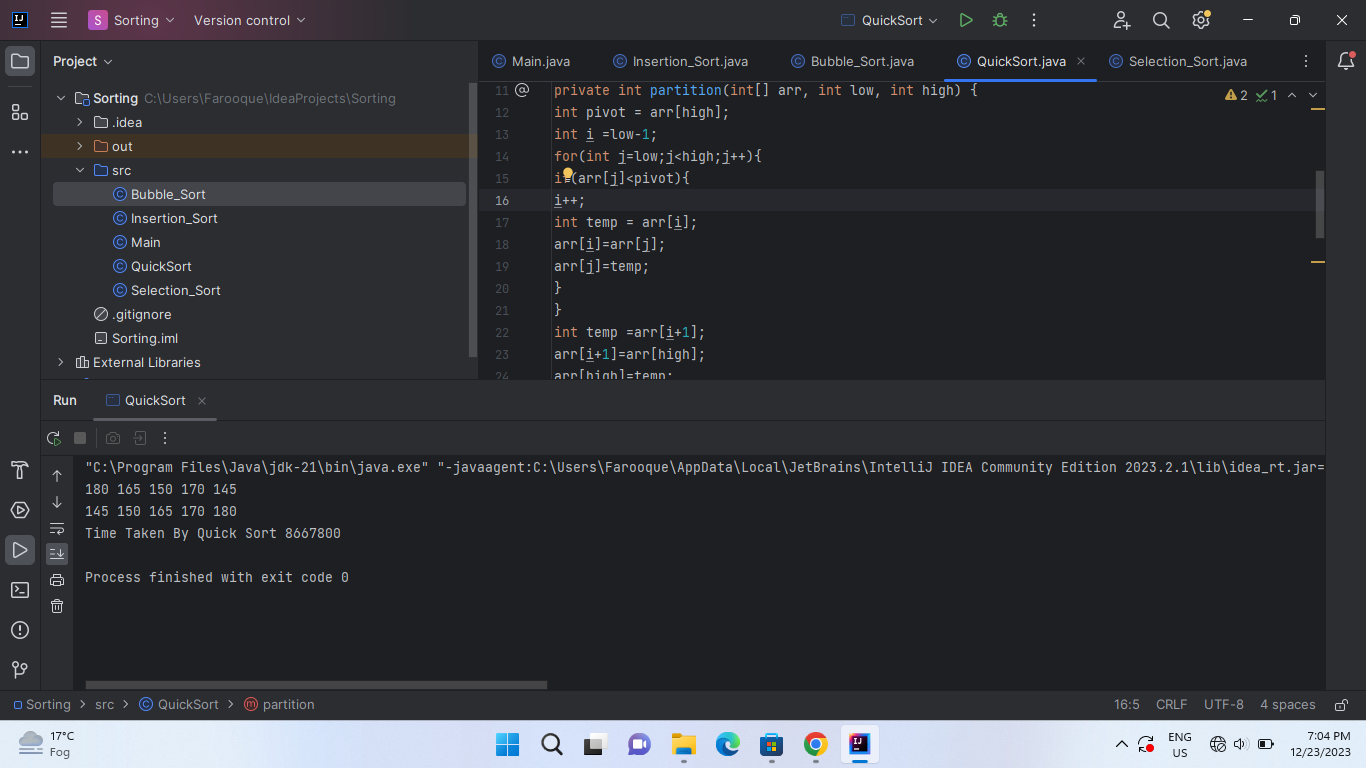
**OutPut**



**iii)**

import java.util.Scanner;  
  
public class QuickSort {  
public void quickSort(int [] arr, int low , int high ){  
if(low<high){  
int pivotIndex = partition(arr,low,high);  
quickSort(arr,low,pivotIndex-1);  
quickSort(arr,pivotIndex+1,high);  
}  
}  
private int partition(int[] arr, int low, int high) {  
int pivot = arr[high];  
int i =low-1;  
for(int j=low;j<high;j++){  
if(arr[j]<pivot){  
i++;  
int temp = arr[i];  
arr[i]=arr[j];  
arr[j]=temp;  
}  
}  
int temp =arr[i+1];  
arr[i+1]=arr[high];  
arr[high]=temp;  
return i+1;  
}  
public void printArray (int [] arr){  
for(int i=0;i<arr.length;i++){  
System.*out*.print(arr[i]+ " ");  
}  
System.*out*.println();  
}  
public static void main(String[] args) {  
  
QuickSort q1 = new QuickSort();  
int [] studentHight = {180,165,150,170,145};  
long t1 = System.*nanoTime*();  
q1.printArray(studentHight);  
q1.quickSort(studentHight,0,studentHight.length-1);  
q1.printArray(studentHight);  
long t2 = System.*nanoTime*();  
System.*out*.println("Time Taken By Quick Sort " + (t2-t1));  
}  
}

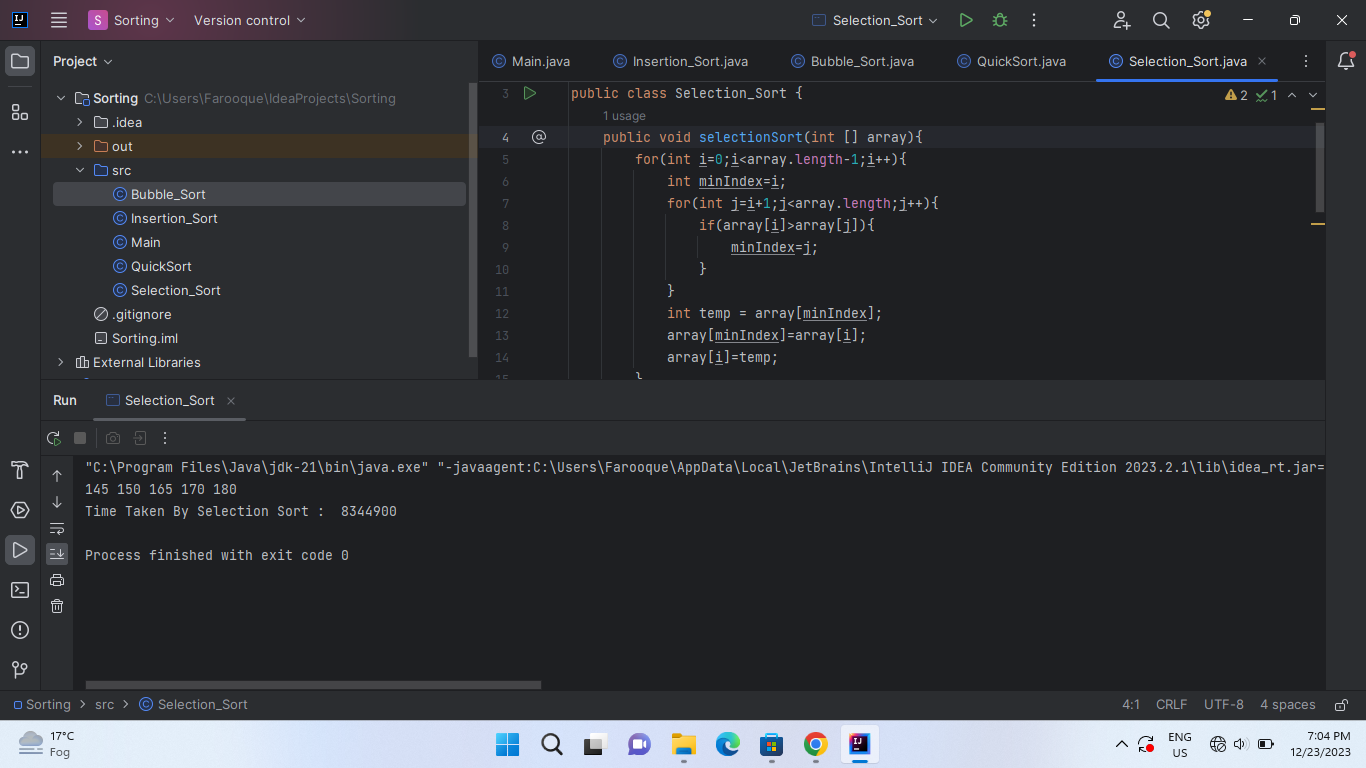
**Output**



***iv)***

public class Insertion\_Sort {  
public void insertion\_Sort(int [] array){  
for(int i=1;i<array.length;i++ ) {  
int key = array[i];  
int j= i-1;  
while(j>=0&&array[j]>key){  
array[j+1]=array[j];  
 j=j-1;  
 }  
 array[j+1]=key;  
  
  
 }  
 for(int i=0;i<array.length;i++){  
 System.*out*.print(array[i]+" ");  
 }  
 System.*out*.println();  
  
 }  
  
 public static void main(String[] args) {  
 int [] studentHight = {180,165,150,170,145};  
 Insertion\_Sort i1 = new Insertion\_Sort();  
 long t1= System.*nanoTime*();  
 i1.insertion\_Sort(studentHight);  
 long t2 = System.*nanoTime*();  
 System.*out*.println("Time Taken By Insertion Sort : " + (t2-t1));  
 }  
  
  
}

**Output**



***V***

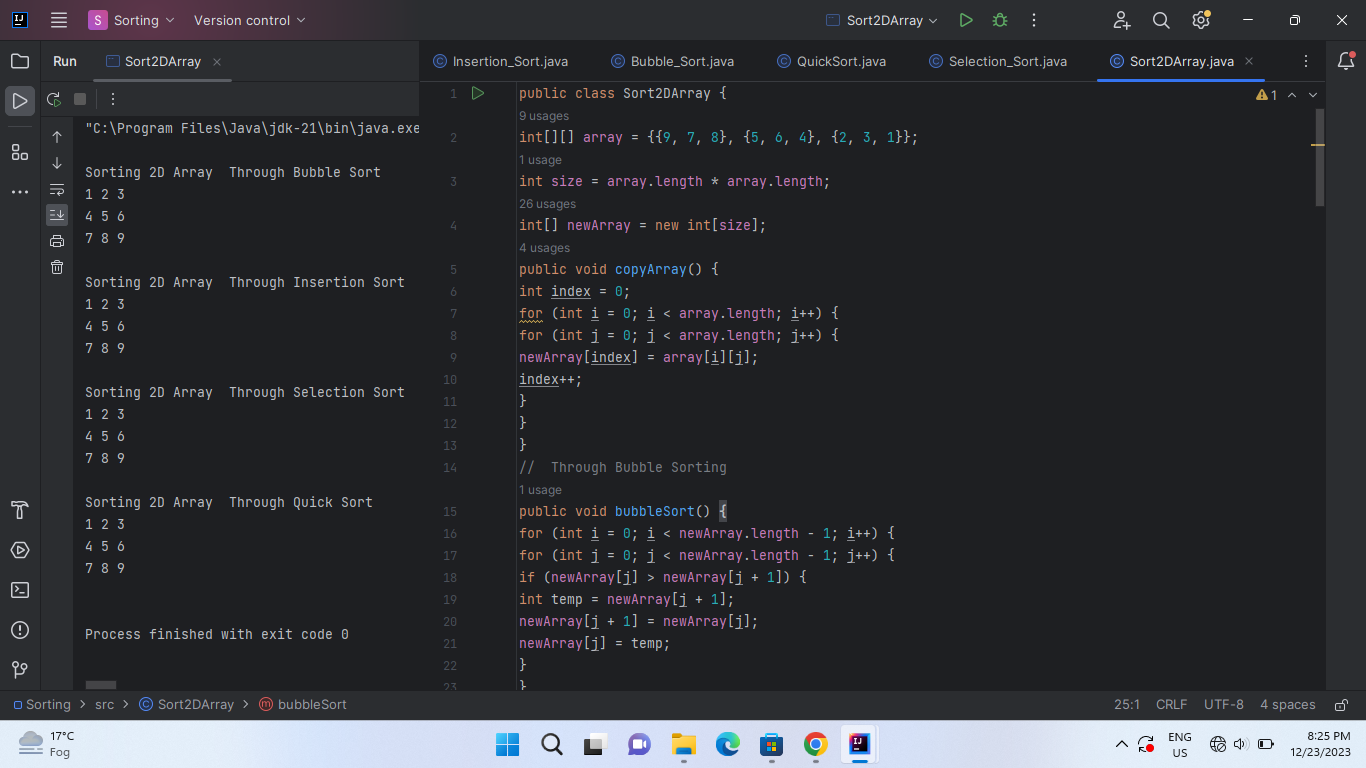
***Time of the each sorting algorithm is displayed above ,in the above displayed time the least time is taken by quick sort , b****ecause* ***it's really good at breaking down a big problem (sorting a big list) into smaller ones, making it easier and quicker to solve. It picks a special number (called a "pivot"), then arranges the other numbers around it in a way that helps to sort everything faster. This method of dividing and conquering makes Quick Sort faster compared to other sorting methods for most situations.***

***Question : No : 02***

***Coding***

public class Sort2DArray {  
int[][] array = {{9, 7, 8}, {5, 6, 4}, {2, 3, 1}};  
int size = array.length \* array.length;  
int[] newArray = new int[size];  
public void copyArray() {  
int index = 0;  
for (int i = 0; i < array.length; i++) {  
for (int j = 0; j < array.length; j++) {  
newArray[index] = array[i][j];  
index++;  
}  
}  
}  
// Through Bubble Sorting  
public void bubbleSort() {  
for (int i = 0; i < newArray.length - 1; i++) {  
for (int j = 0; j < newArray.length - 1; j++) {  
if (newArray[j] > newArray[j + 1]) {  
int temp = newArray[j + 1];  
newArray[j + 1] = newArray[j];  
newArray[j] = temp;  
}  
}  
}  
}  
// Through Insertion Sort  
 public void insertion\_Sort() {  
  
  
 for (int i = 1; i < newArray.length; i++) {  
 int key =newArray[i];  
 int j = i - 1;  
 while (j >= 0 && newArray[j] > key) {  
 newArray[j + 1] = newArray[j];  
 j = j - 1;  
 }  
 newArray[j + 1] = key;  
  
  
 }  
 }  
 // Through Quick Sort  
  
 public void quickSort(int[] arr, int low, int high) {  
  
 if (low < high) {  
 int pivotIndex = partition(arr, low, high);  
 quickSort(arr, low, pivotIndex - 1);  
 quickSort(arr, pivotIndex + 1, high);  
 }  
 }  
 private int partition(int[] arr, int low, int high) {  
 int pivot = arr[high];  
 int i = low - 1;  
 for (int j = low; j < high; j++) {  
 if (arr[j] < pivot) {  
 i++;  
 int temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
 }  
 }  
 int temp = arr[i + 1];  
 arr[i + 1] = arr[high];  
 arr[high] = temp;  
 return i + 1;  
 }  
 // Through Selection Sort  
 public void selectionSort() {  
 for (int i = 0; i < newArray.length - 1; i++) {  
 int minIndex = i;  
 for (int j = i + 1; j < newArray.length; j++) {  
 if (newArray[minIndex] > newArray[j]) { // not i always minIndex  
 minIndex = j;  
 }  
 }  
 int temp = newArray[minIndex];  
 newArray[minIndex] = newArray[i];  
 newArray[i] = temp;  
 }  
  
 }  
  
  
  
 public void copy2DArray () {  
 int index=0;  
 for (int i = 0; i < array.length;i++){  
 for(int j=0;j<array.length;j++){  
 array[i][j]=newArray[index];  
 index++;  
 }  
 }  
 }  
 public void print(){  
  
 for(int i=0;i<3;i++){  
 for(int j=0;j<3;j++){  
 System.*out*.print(array[i][j]+ " ");  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
 }  
  
 public static void main(String[] args) {  
 Sort2DArray s1 = new Sort2DArray();  
 Sort2DArray s2 = new Sort2DArray();  
 Sort2DArray s3 = new Sort2DArray();  
 Sort2DArray s4 = new Sort2DArray();  
 System.*out*.println();  
 System.*out*.println("Sorting 2D Array Through Bubble Sort ");  
 s1.copyArray();  
 s1.bubbleSort();  
 s1.copy2DArray();  
 s1.print();  
  
 System.*out*.println("Sorting 2D Array Through Insertion Sort ");  
 s2.copyArray();  
 s2.insertion\_Sort();  
 s2.copy2DArray();  
 s2.print();  
  
 System.*out*.println("Sorting 2D Array Through Selection Sort ");  
 s3.copyArray();  
 s3.selectionSort();  
 s3.copy2DArray();  
 s3.print();  
  
 System.*out*.println("Sorting 2D Array Through Quick Sort ");  
 s4.copyArray();  
 s4.quickSort(s4.newArray,0,s4.newArray.length-1);  
 s4.copy2DArray();  
 s4.print();  
  
  
  
  
 }  
  
  
  
}

***Output***

****