BML MUNJAL UNIVERSITY

BML MUNJAL UNIVERSITY™

FROM HERE TO THE WORLD

DSA LAB ASSIGNMENT

Name: Manikanta Mandala.

Sec: CSE-3.

Enrolment No. : 200C2030134

Problem Statement:

For a Given array M, do the following implementations -

1. Write a program to implement the Modified version of the bubble sort algorithm so that it terminates the outer loop when it detects that the array is sorted. Compare the running time of the modified algorithm with Original Bubble sort.

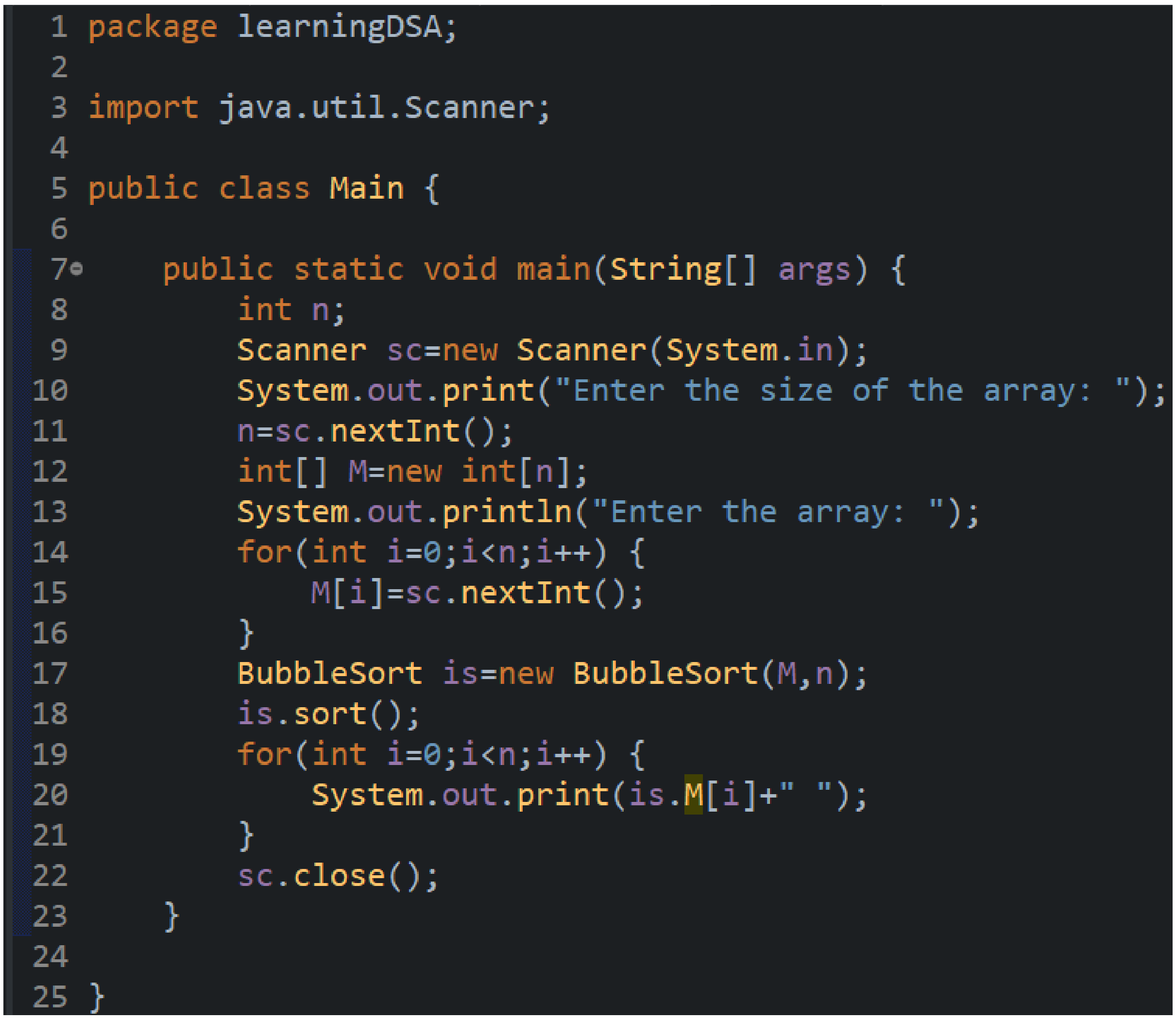
2. Implement Quick sort and Quick Sort 2 (Refer to PDF, Appendix E and F). Calculate the run time complexity of both the implementation and compare their performance in terms of best, average and worst time complexity.

Solution:

1.

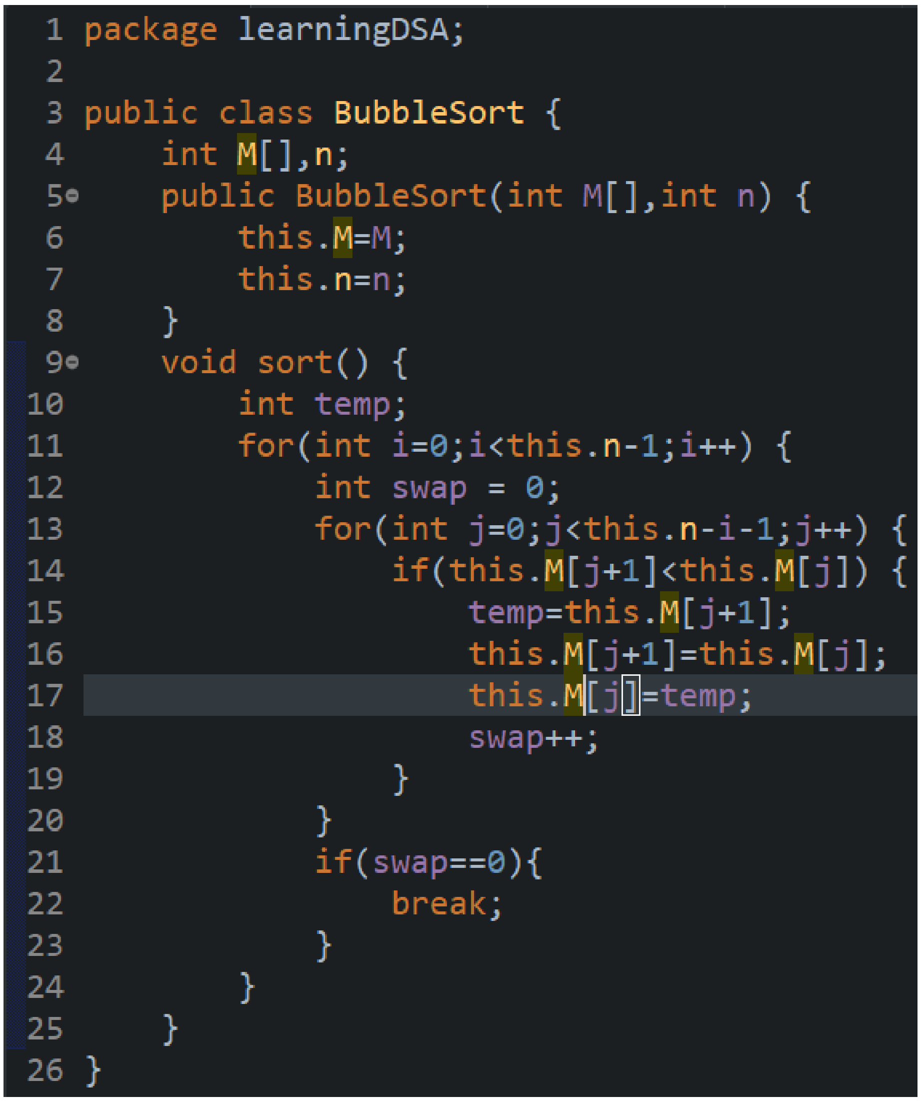
Modified version of Bubble sorting:

Main class with main function:



```java
package learningDSA;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int n;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        n=sc.nextInt();
        int[] M=new int[n];
        System.out.println("Enter the array: ");
        for(int i=0;i<n;i++) {
            M[i]=sc.nextInt();
        }
        BubbleSort is=new BubbleSort(M,n);
        is.sort();
        for(int i=0;i<n;i++) {
            System.out.print(is.M[i]+" ");
        }
        sc.close();
    }

}
```

BubbleSort class with sort method:



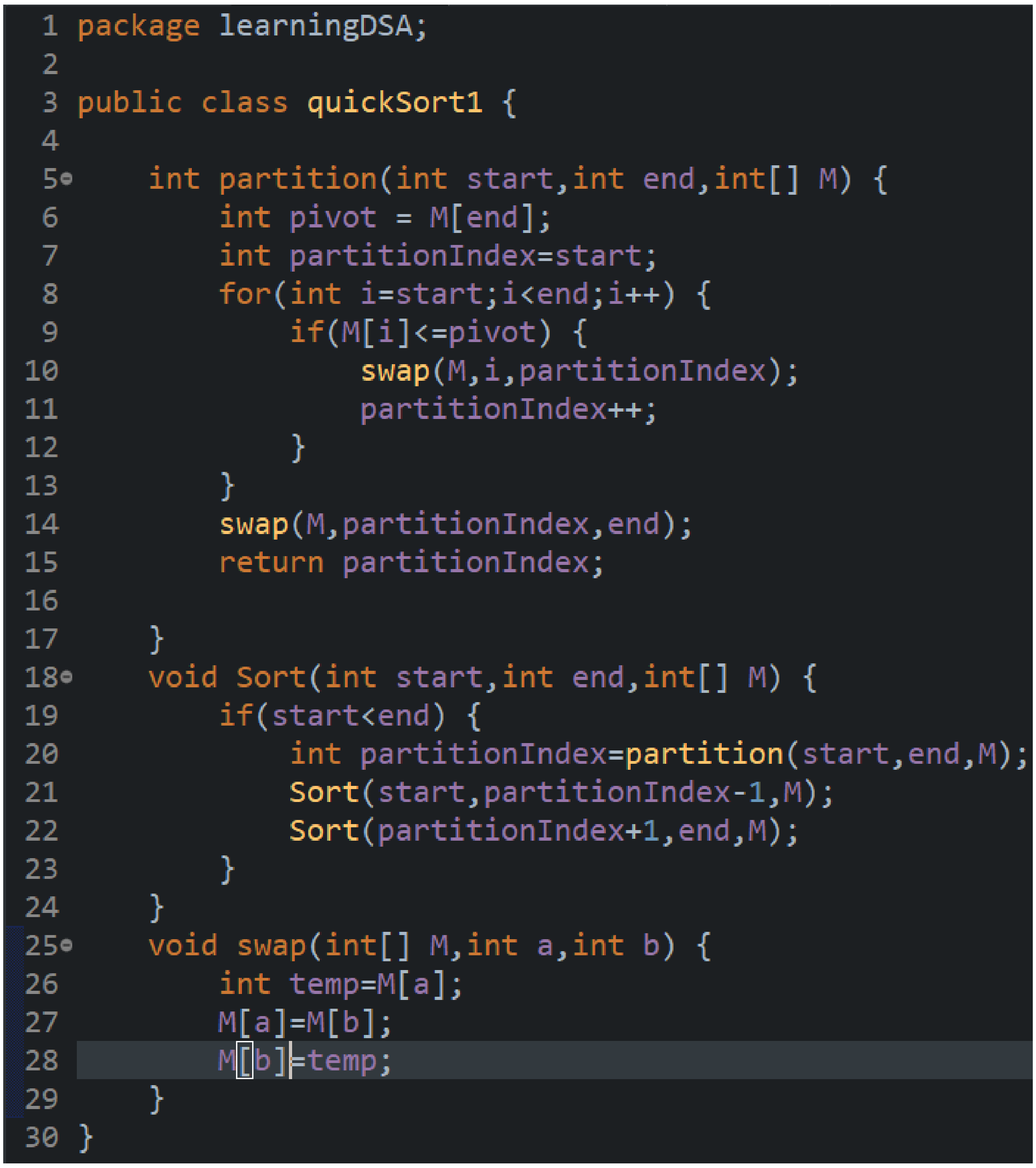
```java
package learningDSA;

public class BubbleSort {
    int M[],n;
    public BubbleSort(int M[],int n) {
        this.M=M;
        this.n=n;
    }
    void sort() {
        int temp;
        for(int i=0;i<this.n-1;i++) {
            int swap = 0;
            for(int j=0;j<this.n-i-1;j++) {
                if(this.M[j+1]<this.M[j]) {
                    temp=this.M[j+1];
                    this.M[j+1]=this.M[j];
                    this.M[j]=temp;
                    swap++;
                }
            }
            if(swap==0){
                break;
            }
        }
    }
}
```

Sorting name	Time complexity			Space complexity
	Worst case	Average case	Best case	Worst case
Original Bubble sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)
Modified Bubble sort	$O(n^2)$	$O(n^2)$	$O(n)$	O(1)

Here, the difference between Original and Modified Bubble sort is if condition that checks swap==0 every iteration of the outer loop. By this we can check whether the array is sorted before total completion of both the loops. So, the best-case time complexity will go down to $\Omega(n)$ instead of $\Omega(n^2)$.

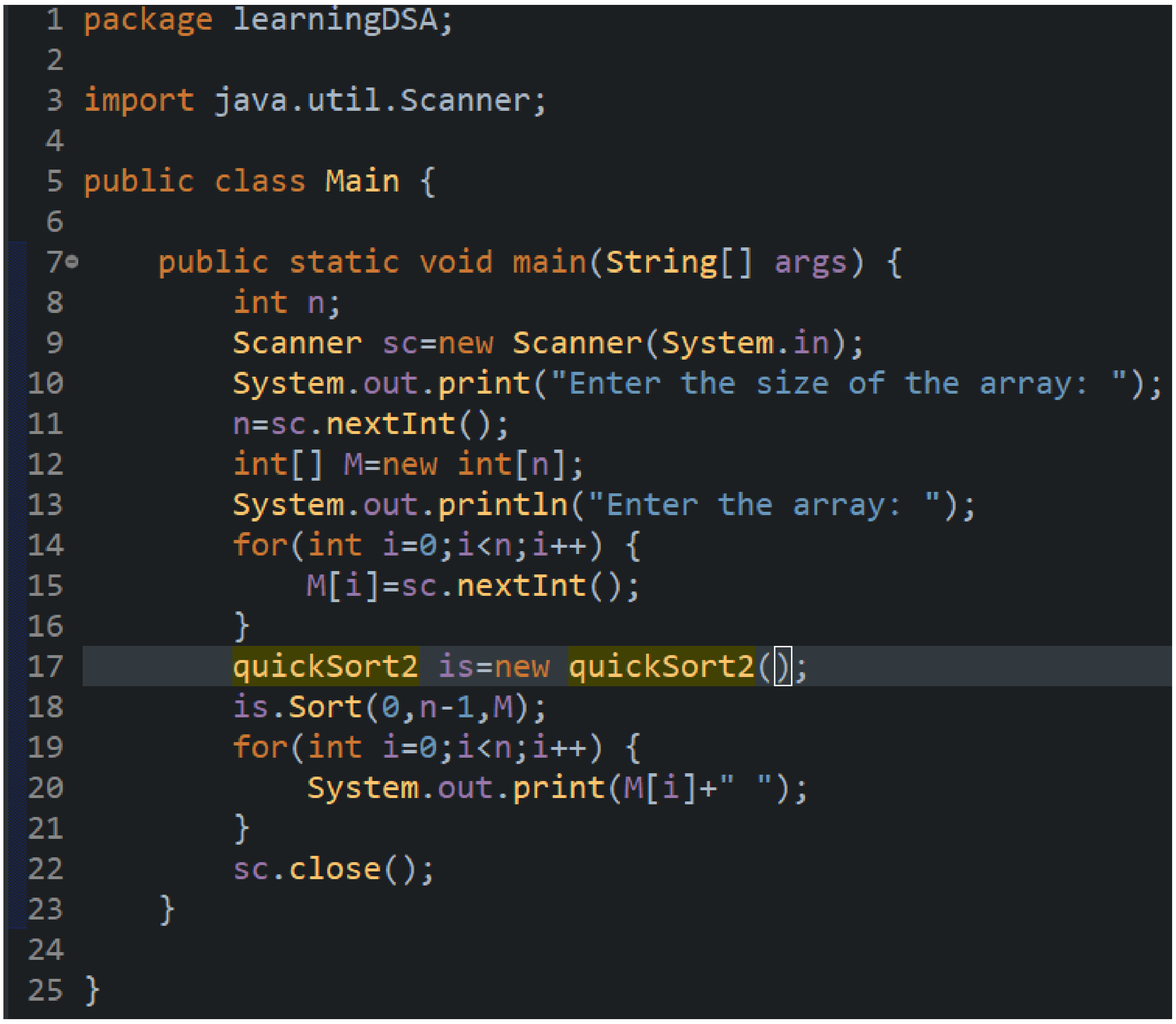
2.

quickSort1 class with sort method:



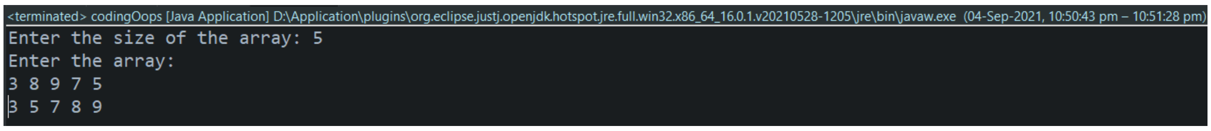
```java
package learningDSA;

public class quickSort1 {

    int partition(int start,int end,int[] M) {
        int pivot = M[end];
        int partitionIndex=start;
        for(int i=start;i<end;i++) {
            if(M[i]<=pivot) {
                swap(M,i,partitionIndex);
                partitionIndex++;
            }
        }
        swap(M,partitionIndex,end);
        return partitionIndex;

    }
    void Sort(int start,int end,int[] M) {
        if(start<end) {
            int partitionIndex=partition(start,end,M);
            Sort(start,partitionIndex-1,M);
            Sort(partitionIndex+1,end,M);
        }
    }
    void swap(int[] M,int a,int b) {
        int temp=M[a];
        M[a]=M[b];
        M[b]=temp;
    }
}
```

Main class with main method



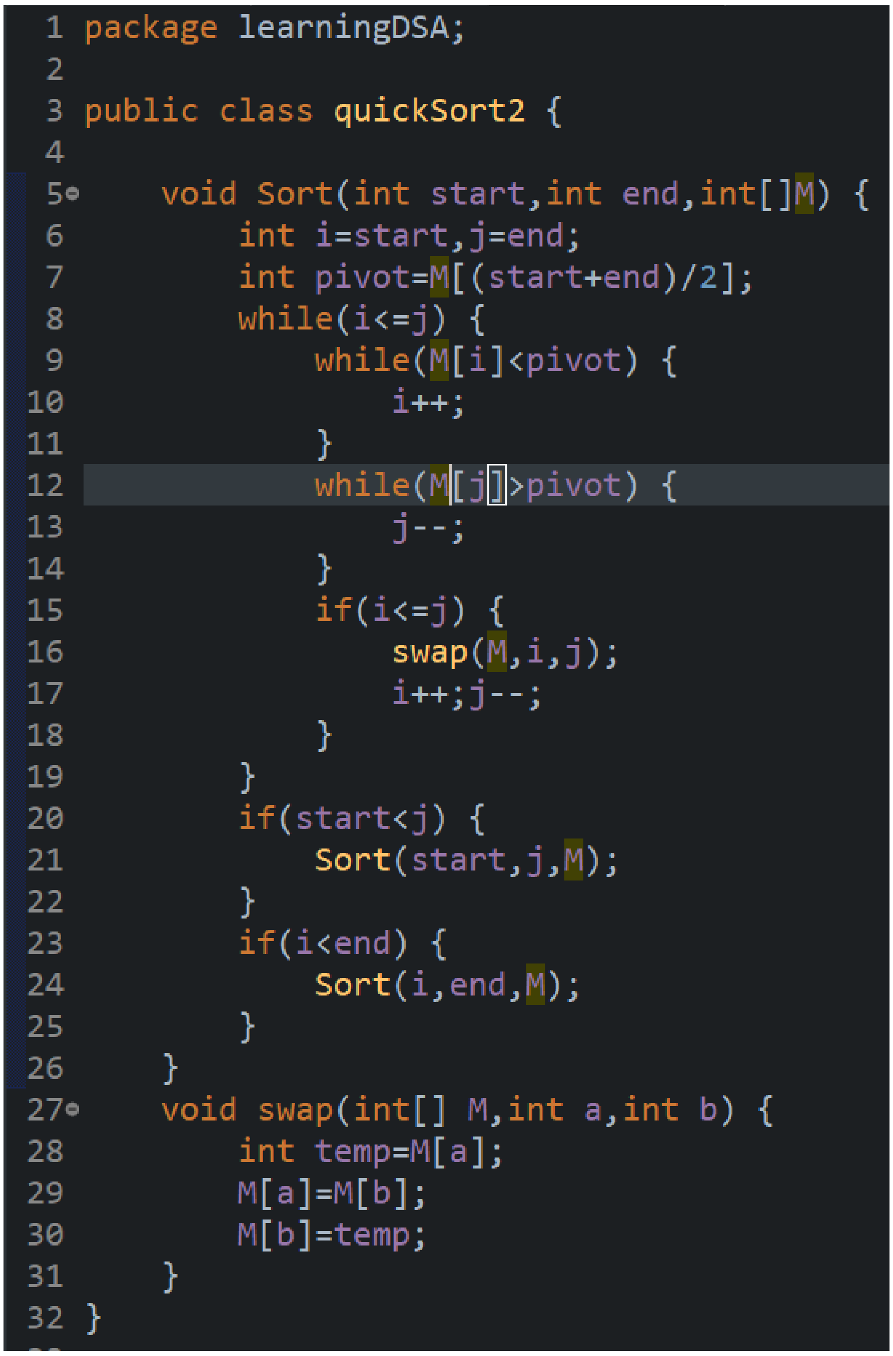
```java
package learningDSA;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int n;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        n=sc.nextInt();
        int[] M=new int[n];
        System.out.println("Enter the array: ");
        for(int i=0;i<n;i++) {
            M[i]=sc.nextInt();
        }
        quickSort2 is=new quickSort2();
        is.Sort(0,n-1,M);
        for(int i=0;i<n;i++) {
            System.out.print(M[i]+" ");
        }
        sc.close();
    }

}
```

Output:



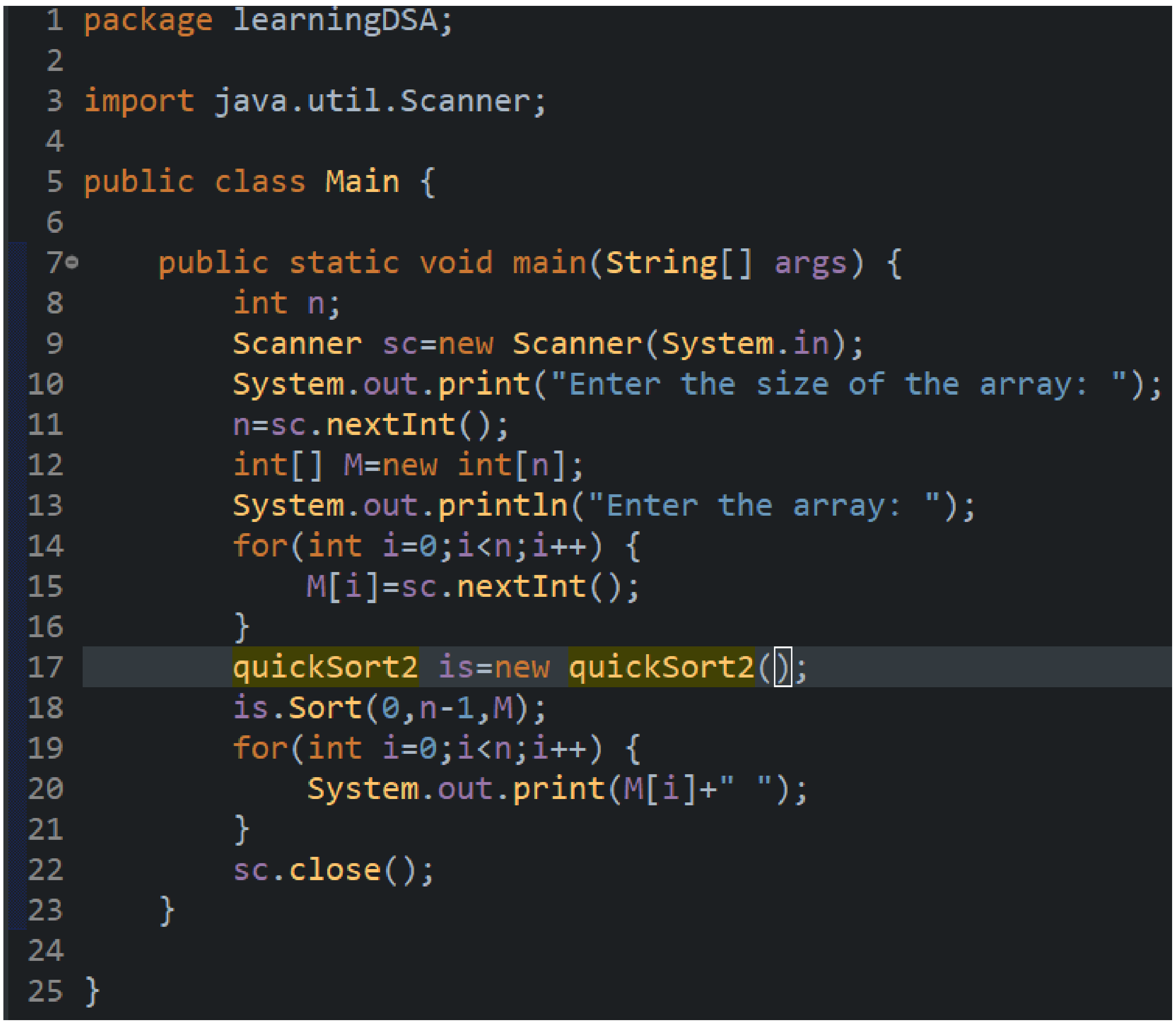
```
<terminated> codingOops [Java Application] D:\Application\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe  (04-Sep-2021, 10:50:43 pm – 10:51:28 pm)
Enter the size of the array: 5
Enter the array:
3 8 9 7 5
3 5 7 8 9
```

quickSort2 class with sort method:



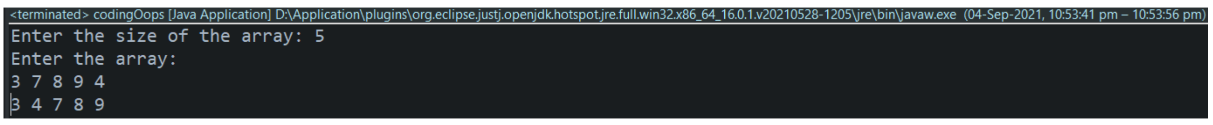
```java
package learningDSA;

public class quickSort2 {

    void Sort(int start,int end,int[]M) {
        int i=start,j=end;
        int pivot=M[(start+end)/2];
        while(i<=j) {
            while(M[i]<pivot) {
                i++;
            }
            while(M[j]>pivot) {
                j--;
            }
            if(i<=j) {
                swap(M,i,j);
                i++;j--;
            }
        }
        if(start<j) {
            Sort(start,j,M);
        }
        if(i<end) {
            Sort(i,end,M);
        }
    }
    void swap(int[] M,int a,int b) {
        int temp=M[a];
        M[a]=M[b];
        M[b]=temp;
    }
}
```

Main class with main method:



```java
package learningDSA;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int n;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        n=sc.nextInt();
        int[] M=new int[n];
        System.out.println("Enter the array: ");
        for(int i=0;i<n;i++) {
            M[i]=sc.nextInt();
        }
        quickSort2 is=new quickSort2();
        is.Sort(0,n-1,M);
        for(int i=0;i<n;i++) {
            System.out.print(M[i]+" ");
        }
        sc.close();
    }

}
```

Output:



```
<terminated> codingOops [Java Application] D:\Application\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe  (04-Sep-2021, 10:53:41 pm – 10:53:56 pm)
Enter the size of the array: 5
Enter the array:
3 7 8 9 4
3 4 7 8 9
```

Sorting name	Time Complexity			Space Complexity
	Worst case	Average case	Best case	Worst case
quickSort1	O(n^2)	O(nlog(n))	O(nlog(n))	O(1)
quickSort2	O(n^2)	O(nlog(n))	O(nlog(n))	O(1)

The differences between quickSort and quickSort2 are :

1.In quickSort, we are using three methods and in quickSort2, we are using two methods.

2.In quickSort, we are defining the end element as pivot but in quickSort2, we are using mid element as pivot element.