1. Create a Student class having roll no. , name, dept., marks. Use array of objects to store details of 5 students. List the name of the student a) having highest marks b) lowest marks c) Marks more than average.

2. Implement 2 sorting algorithms for sorting student array.

3. Use wrapper class for explaining auto-boxing and unboxing.

Code:

```
import java.util.Scanner;



class Student {

   int rollNo;

   String name;

   String dept;

   double marks;



   Student(int rollNo, String name, String dept, double marks) {

        this.rollNo = rollNo;

        this.name = name;

        this.dept = dept;

        this.marks = marks;

   }

}



class day8 {
```

```
public static void main(String[] args) {
    Student[] students = new Student[5];

    Scanner sc = new Scanner(System.in);
    for (int i = 0; i < 5; i++) {
        System.out.println();
        System.out.println("Enter details for student " + (i + 1));
        System.out.print("Roll No: ");
        int rollNo = sc.nextInt();
        System.out.print("Name: ");
        String name = sc.next();
        System.out.print("Department: ");
        String dept = sc.next();
        System.out.print("Marks: ");
        double marks = sc.nextDouble();

        students[i] = new Student(rollNo, name, dept, marks);
    }
    Student highestMarksStudent = students[0];
    for (int i = 1; i < students.length; i++) {
        if (students[i].marks > highestMarksStudent.marks) {
            highestMarksStudent = students[i];
        }
```

```
        }

        System.out.println(

                "Student with highest marks : " + highestMarksStudent.name
+ " - " + highestMarksStudent.marks);



        Student lowestMarksStudent = students[0];

        for (int i = 1; i < students.length; i++) {

            if (students[i].marks < lowestMarksStudent.marks) {

                lowestMarksStudent = students[i];

            }

        }

        System.out.println("Student with lowest marks : " +
lowestMarksStudent.name + " - " + lowestMarksStudent.marks);



        double totalMarks = 0;

        for (Student student : students) {

            totalMarks += student.marks;

        }

        double averageMarks = totalMarks / students.length;



        System.out.println("Students with marks more than average:");

        for (Student student : students) {

            if (student.marks > averageMarks) {

                System.out.println(student.name + " - " + student.marks);
```

```
        }
    }
    bubbleSort(students);
    System.out.println("Sorted by marks (Bubble Sort):");
    for (Student student : students) {
        System.out.println(student.name + " - " + student.marks);
    }

    selectionSort(students);
    System.out.println("Sorted by marks (Selection Sort):");
    for (Student student : students) {
        System.out.println(student.name + " - " + student.marks);
    }

    quicksort(students,0,4);
    System.out.println("Sorted by marks (Quick Sort):");
    for (Student student : students) {
        System.out.println(student.name + " - " + student.marks);
    }

    mergesort(students,0,4);
    System.out.println("Sorted by marks (Merge Sort):");
    for (Student student : students) {
```

```
            System.out.println(student.name + " - " + student.marks);
        }
    }

    private static void bubbleSort(Student[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j].marks < arr[j + 1].marks) {
                    Student temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

    private static void selectionSort(Student[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (arr[j].marks > arr[minIndex].marks) {
```

```
                    minIndex = j;
                }
            }
            Student temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
        }
    }

    static int partition(Student[] arr, int low, int high) {
        double pivot = arr[high].marks;
        int i = (low - 1);
        for (int j = low; j < high; j++) {
            if (arr[j].marks >= pivot) {
                i++;
                Student temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        Student temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
```

```
        return i + 1;
    }
    static void quicksort(Student[] arr, int low, int high)
    {
        if (low < high)
        {
            int pi = partition(arr, low, high);
            quicksort(arr, low, pi-1);
            quicksort(arr, pi+1, high);
        }
    }

    static void merge(Student arr[], int l, int m, int r)
    {
        int n1 = m - l + 1;
        int n2 = r - m;
        Student L[] = new Student[n1];
        Student R[] = new Student[n2];
        for (int i = 0; i < n1; ++i)
            L[i] = arr[l + i];
        for (int j = 0; j < n2; ++j)
            R[j] = arr[m + 1 + j];
```

```
        int i = 0, j = 0;
        int k = l;
        while (i < n1 && j < n2) {
            if (L[i].marks >= R[j].marks) {
                arr[k] = L[i];
                i++;
            }
            else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
```

```
    }

    static void mergesort(Student arr[], int l, int r)

    {

        if (l < r) {

            int m = l + (r - l) / 2;

            mergesort(arr, l, m);

            mergesort(arr, m + 1, r);

            merge(arr, l, m, r);

        }

    }

}
```

Output:

```
Enter details for student 1
Roll No: 16
Name: Ritabrata
Department: CSE
Marks: 89

Enter details for student 2
Roll No: 6
Name: Sayak
Department: CSE
Marks: 67

Enter details for student 3
Roll No: 15
Name: Subham
Department: CSE
Marks: 55

Enter details for student 4
Roll No: 13
Name: Parthiv
Department: CSE
Marks: 78

Enter details for student 5
Roll No: 20
Name: Chonk
Department: CSE
Marks: 76
```

```
Student with highest marks : Ritabrata - 89.0
Student with lowest marks : Subham - 55.0
Students with marks more than average:
Ritabrata - 89.0
Parthiv - 78.0
Chonk - 76.0
Sorted by marks (Bubble Sort):
Ritabrata - 89.0
Parthiv - 78.0
Chonk - 76.0
Sayak - 67.0
Subham - 55.0
Sorted by marks (Selection Sort):
Ritabrata - 89.0
Parthiv - 78.0
Chonk - 76.0
Sayak - 67.0
Subham - 55.0
Sorted by marks (Quick Sort):
Ritabrata - 89.0
Parthiv - 78.0
Chonk - 76.0
Sayak - 67.0
Subham - 55.0
Sorted by marks (Merge Sort):
Ritabrata - 89.0
Parthiv - 78.0
Chonk - 76.0
Sayak - 67.0
Subham - 55.0
linuxmint@jc610:~/ritabrata-java/pack$
```

Wrapper Class:

```
public class Wrapper {
   public static void main(String[] args) {

        //unboxing
        Double a = new Double(10);
        double x = a;
        System.out.println("unboxed value "+x);

        //doing unboxing implicitly
        Double g = new Double(50);
        double h = g.doubleValue();
        System.out.println("implicitly unboxed value "+h);

        //autboxed
        int c = 20;
        Integer b = c;
        System.out.println("autoboxed value "+b);

        //doing autoboxing implicitly
        double m = 56;
        Double n = Double.valueOf(m);
        System.out.println("implicitly autoboxed value "+n);

   }
}
```

Output:

```
linuxmint@jc610:~/ritabrata-java$ cd "/home/linuxmint/ritab
Note: day8wrapper.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
unboxed value 10.0
implicitly unboxed value 50.0
autoboxed value 20
implicitly autoboxed value 56.0
linuxmint@jc610:~/ritabrata-java/pack$
```