OOP Lab Program 3

3) a) Design, Develop and Implement a Java program to sort a list of elements in ascending and descending order and show exception handling.

```
Public class Sort {

        void sortAscend(int arr[], int n)
        {
                for (int i = 0; i < n-1; i++)
                for (int j = 0; j < n-i-1; j++)
                if (arr[j] > arr[j+1])
                {
                      int temp = arr[j];
                       arr[j] = arr[j+1];
                       arr[j+1] = temp;
                   }
        }
        void sortDescend(int arr[], int n)
        {

                for (int i = 0; i < n-1; i++)
                for (int j = 0; j < n-i-1; j++)
                if (arr[j] < arr[j+1])
                {
                       int temp = arr[j];
                       arr[j] = arr[j+1];
                       arr[j+1] = temp;
                   }
        }
        void printArray(int arr[], int n)
        {
              //int n = arr.length;
               for (int i=0; i<n; i++)
                    System.out.print(arr[i] + " ");
              System.out.println();
         }

}
Public class TestSort {

        Public static void main(String[] args) {

                int n;
                Scanner sc = new Scanner(System.in);
```

```
                Sort s = new Sort();
                 System.out.println("Input number of integers to sort");
                 n = sc.nextInt();
                 int array[] = new int[10];
                System.out.println("Enter " + n + " integers");

                for (int i = 0; i < n; i++)
                {
                    try {
                          array[i] = sc.nextInt();
                          }

                    catch(ArrayIndexOutOfBoundsException e){
                          System.out.println("Index out of bound");
                          System.exit(0);
                     }

                  }
               System.out.println("Entered Array is");
               s.printArray(array, n);

              s.sortAscend(array,n);
              System.out.println("Ascending order: ");
              s.printArray(array,n);

              s.sortDescend(array, n);
              System.out.println("Descending order: ");
              s.printArray(array,n);

              sc.close();

        }

}
```

3) b) Design, Develop and Implement a Java program to demonstrate multilevel inheritance by using super for calling the super class constructors. (Use Box, BoxWeight and BoxShipment classes)

```
class Box {
private double width;
private double height;
private double depth;

// construct clone of an object
Box(Box ob) { // pass object to constructor
```

```
width = ob.width;
height = ob.height;
depth = ob.depth;
}
// constructor used when all dimensions specified
Box(double w, double h, double d) {
width = w;
height = h;
depth = d;
}
// constructor used when no dimensions specified
Box() {
width = -1; // use -1 to indicate
height = -1; // an uninitialized
depth = -1; // box
}
// constructor used when cube is created
Box(double len) {
width = height = depth = len;
}
// compute and return volume
double volume() {
return width * height * depth;
}
}
// Add weight.
class BoxWeight extends Box {
double weight; // weight of box
// construct clone of an object
BoxWeight(BoxWeight ob) { // pass object to constructor
super(ob);
weight = ob.weight;
}
// constructor when all parameters are specified
BoxWeight(double w, double h, double d, double m) {
super(w, h, d); // call superclass constructor
weight = m;
}
// default constructor
BoxWeight() {
super();
weight = -1;
}

// constructor used when cube is created
BoxWeight(double len, double m) {
```

```
super(len);
weight = m;
}
}
// Add shipping costs.
class Shipment extends BoxWeight {
double cost;
// construct clone of an object
Shipment(Shipment ob) { // pass object to constructor
super(ob);
cost = ob.cost;
}
// constructor when all parameters are specified
Shipment(double w, double h, double d,
double m, double c) {
super(w, h, d, m); // call superclass constructor
cost = c;
}
// default constructor
Shipment() {
super();
cost = -1;
}
// constructor used when cube is created
Shipment(double len, double m, double c) {
super(len, m);
cost = c;
}
}
class DemoShipment {
public static void main(String args[]) {
Shipment shipment1 = new Shipment(10, 20, 15, 10, 3.41);
Shipment shipment2 =new Shipment(2, 3, 4, 0.76, 1.28);
double vol;
vol = shipment1.volume();
System.out.println("Volume of shipment1 is " + vol);
System.out.println("Weight of shipment1 is "+ shipment1.weight);

System.out.println("Shipping cost: $" + shipment1.cost);
System.out.println();
vol = shipment2.volume();
System.out.println("Volume of shipment2 is " + vol);
System.out.println("Weight of shipment2 is "+ shipment2.weight);
System.out.println("Shipping cost: $" + shipment2.cost);
}
}
```