College of Engineering, Thiruvananthapuram

Object Oriented Programming Lab

Anirudh A. V.

S2 CSE R2, Roll No. 11

Department of Computer Science

& Engineering

March 5, 2022

1 **Doubly Linked List**

1.1 Aim

To Write a Java program for the following

A) Create a doubly linked list of elements.
B) Delete a given element from the above list.
C) Display the contents of the list after deletion.

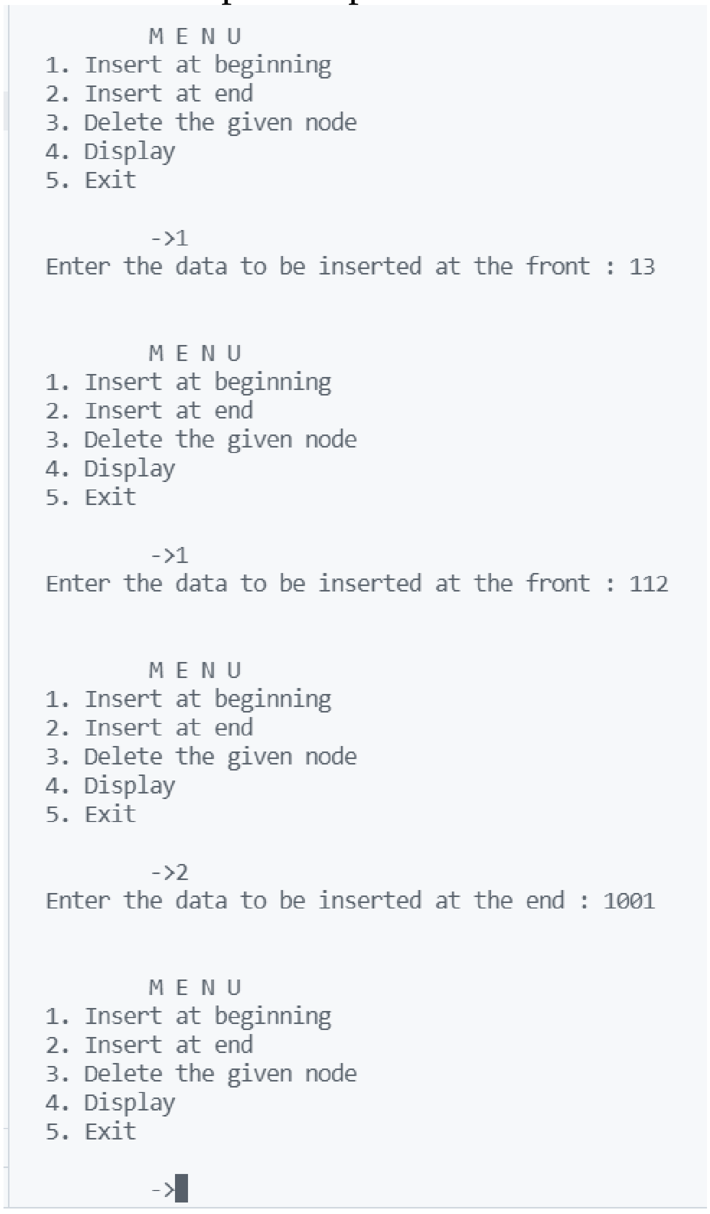
1.2 Algorithm

1. 1. Start
2. Create a class Node for representing the nodes withing the class LinkedList.
    i. Create data members (object of the same class) for pointing to the next and previous node.
3. Create a head node and assign it null.
4. Create the addNode(data) method which accepts the data to be added as a parameter and then creates a new node with a constructor of class Node.
5. Create a DeleteNode method which accepts the position of the node to be deleted.
6. It assigns the node as null and points the previous node to the node after it.
7. Create the display method to display the entire linked list after all addition operation and each deletion operation.
8. Stop

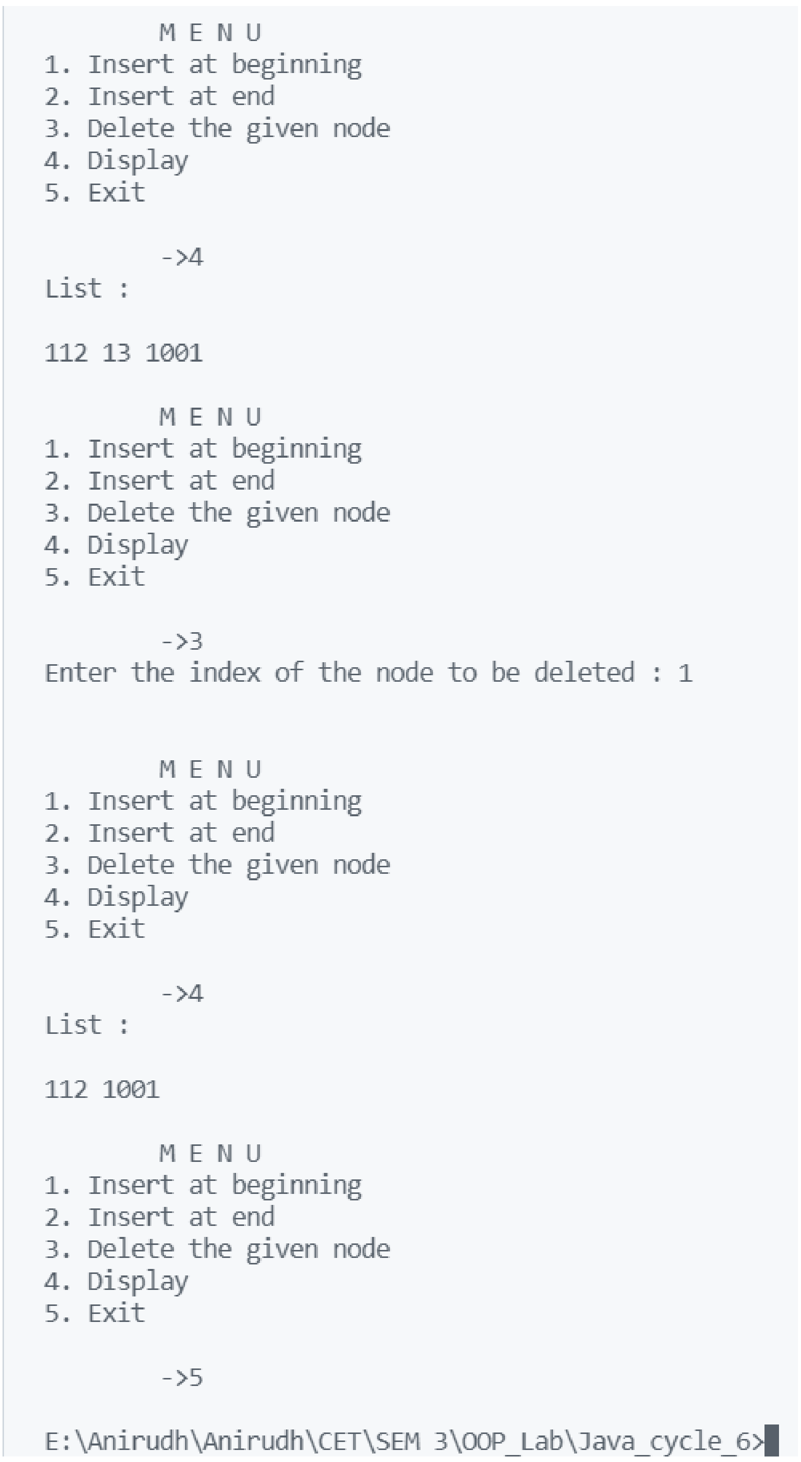
1.3 Code

```java
import java.util.Scanner;

public class LinkedList {

    static class node {
        int value;
        node next;

        node(int d) {
            value = d;
            next = null;
        }
    }

    node start = null;

    // public void finalize() {
    // System.out.println("object is garbage collected");
    // }

    public static LinkedList insert_beginning(LinkedList list, int data) {

        node Newnode = new node(data);
        if (list.start == null) {
            list.start = Newnode;
        } else {
            Newnode.next = list.start.next;
            list.start = Newnode;
        }

        return list;
    }

    public static LinkedList insert_end(LinkedList list, int data) {

        if (list.start == null) {
            System.out.println("The list is empty!!!");
        } else {
            node Newnode = new node(data);
            Newnode.next = null;

            node last = list.start;

            while (last.next != null) {
                last = last.next;
            }
            last.next = Newnode;
        }

        return list;
    }

    public static void traversal(LinkedList list) {

        int flag = 0;
        node temp = list.start;
        System.out.println("Linked list : ");

        while (temp.next != null) {
            System.out.printf("%d ", temp.value);
            temp = temp.next;
            flag = 1;
        }
        if (flag != 1) {
            System.out.println("List is empty!!");
        }


    }

    public static LinkedList delete(LinkedList list, int data) {

        if (list.start == null) {
            System.out.println("The list is empty!!!");
        } else {
            // node Newnode = new node(data);
            // Newnode.next = null;

            node temp1 = null, temp = list.start;
            // node temp1 = new node();
            while (temp.next != null) {
                temp1 = temp;
                temp = temp.next;
                if (temp.value == data) {
                    break;
                }
            }

            temp1.next = temp.next;
            temp = null;

            System.gc();

            // last.next = Newnode;
        }

        return list;
    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        LinkedList list = new LinkedList();

        int choice, response = 1;
        do {
            System.out.println("\t\nM E N U");
            System.out
                    .println("1. Insert at beginning\n2. Insert at end\n3.
Delete the given node\n4. Display\n5. Exit");
            System.out.printf("\n\t->");
            choice = sc.nextInt();
            int data;
            switch (choice) {
                case 1:
                    System.out.printf("Enter the data to be inserted at the
front : ");

                    data = sc.nextInt();
                    insert_beginning(list, data);
                    break;

                case 2:
                    System.out.printf("Enter the data to be inserted at the
end : ");

                    data = sc.nextInt();
                    insert_end(list, data);
                    break;

                case 3:
                    System.out.printf("Enter the data of the node to be
deleted : ");

                    data = sc.nextInt();
                    delete(list, data);
                    break;

                case 4:
                    traversal(list);
                    break;

                case 5:
                    System.exit(0);
                    break;

                default:
                    break;
            }
        } while (response == 1);
        sc.close();
    }
}
```

1.4 Sample Output



```
        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->1
Enter the data to be inserted at the front : 13


        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->1
Enter the data to be inserted at the front : 112


        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->2
Enter the data to be inserted at the end : 1001


        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->
```



```
        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->4
List :

112 13 1001

        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->3
Enter the index of the node to be deleted : 1


        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->4
List :

112 1001

        M E N U
1. Insert at beginning
2. Insert at end
3. Delete the given node
4. Display
5. Exit

        ->5

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_6>
```

2 Binary Search

2.1 Aim

To Write a Java program that implements the binary search algorithm

2.2 Algorithm

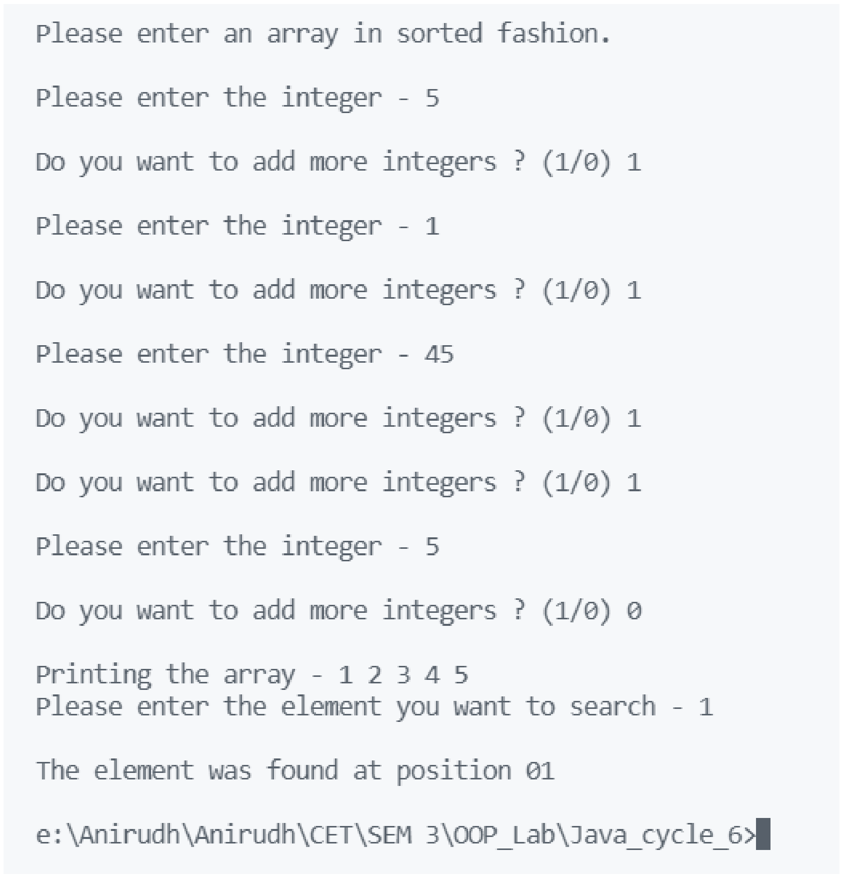
1. 1. Start
2. Create a function binarySearch() where array is passed along with the element to be searched and the low and high value of the array
3. Now check the condition whether the required element is the mid one where mid=low+(high-low)/2
4. If element <array(mid) then high = low-1
5. If element>array(mid) then low=mid+1
6. Print the result whether the element is found and their respective position
7. Stop

2.3 Code

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

class Binarysearch{
    public static void main(String args[]){
        ArrayList<Integer> arr = new ArrayList<Integer>();
        Scanner sc = new Scanner(System.in);
        int x;
        System.out.print("\nPlease enter an array in sorted fashion.\n");
        while(true)
        {
            System.out.print("\nPlease enter the integer - ");

            x= sc.nextInt();
```

```java
            arr.add(x);
            System.out.printf("\nDo you want to add more integers ? (1/0) ");
            x = sc.nextInt();
            if(x == 0)
            break;
        }
        Iterator<Integer> itr = arr.iterator();
        System.out.printf("\nPrinting the array - ");
        while(itr.hasNext())
        {
            x = itr.next();
            System.out.printf(x + " ");

        }

        System.out.printf("\nPlease enter the element you want to search - ");
        x = sc.nextInt();
        sc.close();

       // Performing binary search
        int start = 0, end = arr.size()-1, mid  = (start + end)/2, flag = 0;
        while(end >= start)
        {
            if(x > arr.get(mid))
            {
                start = mid +1;
                mid = (start + end)/2;
            }else if(x < arr.get(mid))
            {
                end = mid - 1;
                mid = (start + end)/2;

            }else{
                flag =  1;
                break;
            }
        }

        if(flag == 1)
        {
            System.out.println("\nThe element was found at position " +
mid+1);
        }
        else
        System.out.println("\nThe element was not found in the array");

    }
}
```

2.4 Sample Output



```
Please enter an array in sorted fashion.

Please enter the integer - 5

Do you want to add more integers ? (1/0) 1

Please enter the integer - 1

Do you want to add more integers ? (1/0) 1

Please enter the integer - 45

Do you want to add more integers ? (1/0) 1

Do you want to add more integers ? (1/0) 1

Please enter the integer - 5

Do you want to add more integers ? (1/0) 0

Printing the array - 1 2 3 4 5
Please enter the element you want to search - 1

The element was found at position 01

e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_6>
```

3 **Quick Sort**

3.1 Aim

To Write a Java program that implements Quick Sort algorithm for sorting a list of names in ascending order.

3.2 Algorithm

1. Start

2. Create a class Sort with a constructor that saves the array and a sort_now method for sorting the array.

3. Set the pivot as the first element and move the element till all the elements in its right and left are greater and smaller respectively. Recursively call this method by splitting the array again and again.

4. Accept the array in the main function and create an object of the class.

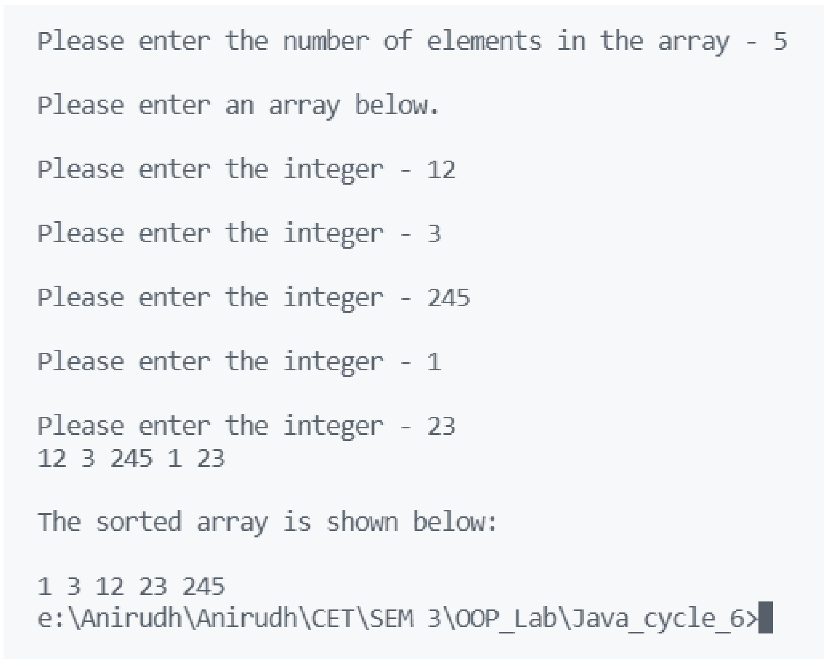
5. Call the sort method to sort the array. Then display it after sorting.

6. Stop

3.3 Code

```java
import java.util.Scanner;

class Sort{
    int arr[], n;
    Sort(int arr[], int n){

        this.arr = arr;
        this.n = n;
    }

    void sort_now(int arr[], int st, int end)
    {
        int pivot = st, i=st+1, j=end, temp;
        if(st >= end)
        return;
        try{
            do{
                if(arr[i]>arr[pivot]){
                    // j=end;
                    do{
                        if(arr[pivot]>arr[j]){
                            temp = arr[j];
                            arr[j] = arr[i];
                            arr[i] = temp;
                            break;
                        }
                        j-=1;
                    }while(i<=j);

                }
                i+=1;
            }while(i<=j);
        }catch(Exception e){
            return;
        }
        temp = arr[j];
        arr[j] = arr[pivot];
        arr[pivot] = temp;

        sort_now(arr, j+1, end);
        sort_now(arr, st, j-1);
    }
}

public class QuickSort {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("\nPlease enter the number of elements in the array - 
");
        int n = sc.nextInt(), x, arr[] = new int[n];

        System.out.print("\nPlease enter an array below.\n");

        for(int i=0; i<n; i++)
        {
            System.out.print("\nPlease enter the integer - ");
            x= sc.nextInt();
            arr[i] = x;
        }

        sc.close();
        Sort st = new Sort(arr, n);
        for(int p=0; p<n; p++)
        {
            System.out.printf("%d ", arr[p]);
        }
        System.out.println();
        st.sort_now(arr, 0, n-1);
        System.out.print("\nThe sorted array is shown below:\n\n");
        for(int i=0; i<n; i++)
        {
            System.out.printf("%d ", arr[i]);
        }

    }
}
```

3.4 Sample Output



```
Please enter the number of elements in the array - 5

Please enter an array below.

Please enter the integer - 12

Please enter the integer - 3

Please enter the integer - 245

Please enter the integer - 1

Please enter the integer - 23
12 3 245 1 23

The sorted array is shown below:

1 3 12 23 245
e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_6>
```

4 Thread Synchronization

4.1 Aim

To Write a Java program to implement Heap Sort algorithm using array.

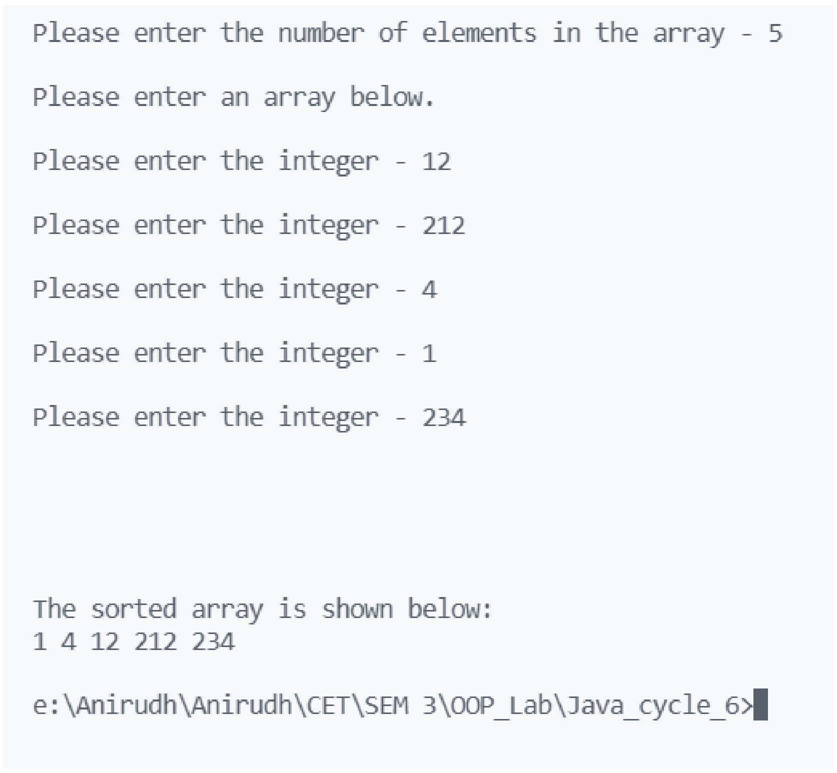
4.2 Algorithm

1. Start
2. Create the static method called Heapify which makes a heap from the given array.
3. Create the HeapSort method which accepts the array and the number of elements as parameters.
    1. Move from the bottom most branch upwards and called the heapify function recursively to create a heap from the given array.
    2. Now find the first element (which will also be the largest) and swap it with the end of the array. Call the heapify method again with n-1 elments of the array to recreate the heap.
    3. Proceed like this till n=1.
4. Accept the array in the main method and call the methods to obtain the sorted array.
5. Display the array to the user.
6. Stop

4.3 Code

```
import java.util.*;

public class HeapSort {


    static void heapify(int arr[], int n, int i){

        int largest = i, lc = 2*i+1, rc = 2*i+2;

        if(lc<n && arr[lc] > arr[largest]){
            largest = lc;
        }

        if(rc<n && arr[rc] > arr[largest]){
            largest = rc;
        }

        // Moving across the largest path if it exists
        if(i != largest){
            int temp = arr[largest];
            arr[largest] = arr[i];
            arr[i] = temp;

            heapify(arr, n, largest);
        }
        return;
    }

    static void Heapsort(int arr[], int n){

        // Initially heapify the tree. For this we start from the bottom most
parent (n/2 - 1)
        for(int i= n/2 - 1; i>=0; i--){
            heapify(arr, n, i);
        }


        // Due to heapify the largest elment is already at the top. So we're
going to push it out.
        int temp;

        // Now we extract the largest elements from the heap tree in an
iterative way.
        for(int i=n-1; i>0; i--)
        {

            temp = arr[0];
            arr[0] = arr[i];
            arr[i] = temp;
            System.out.println();
            heapify(arr, i, 0);
        }

    }

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("\nPlease enter the number of elements in the array -
");
        int n = sc.nextInt(), x, arr[] = new int[n];

        System.out.print("\nPlease enter an array below.\n");

        for(int i=0; i<n; i++)
        {
            System.out.print("\nPlease enter the integer - ");
            x= sc.nextInt();
            arr[i] = x;
        }

        sc.close();
        Heapsort(arr, n);

        System.out.println("\nThe sorted array is shown below:");

        // Printing the Sorted array
        for(int i=0; i<n; i++){
            System.out.printf("%d ", arr[i]);
        }
        System.out.println();
    }
}
```

4.4 Sample Output



```
Please enter the number of elements in the array - 5

Please enter an array below.

Please enter the integer - 12

Please enter the integer - 212

Please enter the integer - 4

Please enter the integer - 1

Please enter the integer - 234




The sorted array is shown below:
1 4 12 212 234

e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_6>
```