LAIBA SHAHID
2023F-BSE-247

LAB 05

SORTING ON LINEAR ARRAY

OBJECTIVE: To sort a linear array using Selection Sort, Bubble Sort and Merge Sort

LAB TASKS

1. **Write a program for Selection sort that sorts an array containing numbers, prints all the sort values of array each followed by its location.**

INPUT:

```
package lab05;
import java.util.Scanner;
public class Lab05 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(source: System.in);
        System.out.print(s: "Enter the number of elements in the array: ");
        int n = sc.nextInt();
        int[] A1 = new int[n];
        System.out.println(x: "Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            A1[i] = sc.nextInt();
        }
        for (int i = 0; i < n - 1; i++) {
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (A1[j] < A1[minIndex]) {
                    minIndex = j;
                }
            }
            if (minIndex != i) {
                int temp = A1[minIndex];
                A1[minIndex] = A1[i];
                A1[i] = temp;
            }
            System.out.println("Array after iteration " + (i + 1) + ":");
            for (int k = 0; k < n; k++) {
                System.out.println("Value: " + A1[k] + " , Index: " + k);
            }
            System.out.println();
        }
        System.out.println(x: "Sorted array:");
        for (int i = 0; i < n; i++) {
            System.out.println("Value: " + A1[i] + " , Index: " + i);
        }
        sc.close();
```

OUTPUT:

```
Enter the number of elements in the array:
4
Enter the elements of the array:
12
76
32
22
Array after iteration 1:
Value: 12 , Index: 0
Value: 76 , Index: 1
Value: 32 , Index: 2
Value: 22 , Index: 3

Array after iteration 2:
Value: 12 , Index: 0
Value: 22 , Index: 1
Value: 32 , Index: 2
Value: 76 , Index: 3

Array after iteration 3:
Value: 12 , Index: 0
Value: 22 , Index: 1
Value: 32 , Index: 2
Value: 76 , Index: 3
```

```
Sorted array:
Value: 12 , Index: 0
Value: 22 , Index: 1
Value: 32 , Index: 2
```

2. **Write a program that takes 10 numbers as input in an array. Sort the elements of array by using Bubble sort. Print each iteration of the sorting process**

INPUT:

```
public class Lab05 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(source: System.in);
        int[] arr = new int[10];
        System.out.println(x: "Enter 10 numbers:");
        for (int i = 0; i < 10; i++) {
            arr[i] = sc.nextInt();
        }
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            boolean swapped = false;

            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    // Swap array[j] and array[j + 1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            System.out.println("Array after iteration " + (i + 1) + ":");
            for (int k = 0; k < n; k++) {
                System.out.print(arr[k] + " ");
            }
            System.out.println();
            if (!swapped) break;
        }
        System.out.println(x: "Sorted array:");
        for (int num : arr) {
            System.out.print(num + " ");
        }
        sc.close();
    }
}
```

OUTPUT:

```
Enter 10 numbers:
5
6
2
9
2
6
4
1
7
0
Array after iteration 1:
5 2 6 2 6 4 1 7 0 9
Array after iteration 2:
2 5 2 6 4 1 6 0 7 9
Array after iteration 3:
2 2 5 4 1 6 0 6 7 9
Array after iteration 4:
2 2 4 1 5 0 6 6 7 9
Array after iteration 5:
2 2 1 4 0 5 6 6 7 9
Array after iteration 6:
2 1 2 0 4 5 6 6 7 9
Array after iteration 7:
1 2 0 2 4 5 6 6 7 9
Array after iteration 8:
1 0 2 2 4 5 6 6 7 9
Array after iteration 9:
0 1 2 2 4 5 6 6 7 9
```

```
Sorted array:
0 1 2 2 4 5 6 6 7 9 BUILD SUCCESSFUL (total time: 14 seconds)
```

3. Write a program that takes 10 random numbers in an array. Sort the elements of array by using Merge sort applying recursive technique. Print each iteration of the sorting process.

INPUT:

```
public class Lab05 {
    public static void main(String[] args) {
        int[] arr = new int[10];
        Random rr = new Random();
        System.out.println(x: "Unsorted array:");
        for (int i = 0; i < arr.length; i++) {
            arr[i] = rr.nextInt(bound: 100);
            System.out.print(arr[i] + " ");
        }
        System.out.println();
        mergeSort(arr, left:0, arr.length - 1);
        System.out.println(x: "Sorted array:");
        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
    public static void mergeSort(int[] arr, int left, int right) {
        if (left < right) {
            int mid = left + (right - left) / 2;

            mergeSort(arr, left, right:mid);
            mergeSort(arr, mid + 1, right);
            merge(arr, left, mid, right);
            System.out.println("Array after merging (" + left + " to " + right + "):");
            for (int i = left; i <= right; i++) {
                System.out.print(arr[i] + " ");
            }
            System.out.println();
        }
    }
```

```
    public static void merge(int[] arr, int left, int mid, int right) {
        int n1 = mid - left + 1;
        int n2 = right - mid;
        int[] leftArray = new int[n1];
        int[] rightArray = new int[n2];

        for (int i = 0; i < n1; i++) {
            leftArray[i] = arr[left + i];
        }
        for (int i = 0; i < n2; i++) {
            rightArray[i] = arr[mid + 1 + i];
        }
        int i = 0, j = 0, k = left;
        while (i < n1 && j < n2) {
            if (leftArray[i] <= rightArray[j]) {
                arr[k] = leftArray[i];
                i++;
            } else {
                arr[k] = rightArray[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = leftArray[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = rightArray[j];
            j++;
            k++;
        }
    }
}
```

OUTPUT:

```
Unsorted array:
41 48 35 69 48 0 61 14 87 73
Array after merging (0 to 1):
41 48
Array after merging (0 to 2):
35 41 48
Array after merging (3 to 4):
48 69
Array after merging (0 to 4):
35 41 48 48 69
Array after merging (5 to 6):
0 61
Array after merging (5 to 7):
0 14 61
Array after merging (8 to 9):
73 87
Array after merging (5 to 9):
0 14 61 73 87
Array after merging (0 to 9):
0 14 35 41 48 48 61 69 73 87
Sorted array:
0 14 35 41 48 48 61 69 73 87 BUILD SUCCESSFUL (total time: 0 seconds)
```

HOME TASK

1. **Declare an array of size n to store account balances. Initialize with values 0 to 100000 and sort Account No's according to highest balance values by using Quick sort, For e.g.: Account No. 3547 Balance 28000 Account No. 1245 Balance 12000**

INPUT:

```
import java.util.Random;
public class Lab05 {
    public static void main(String[] args) {
        int n = 10;
        int[] accountNumbers = new int[n];
        int[] balances = new int[n];
        Random random = new Random();

        System.out.println(x: "Initial Account Balances:");
        for (int i = 0; i < n; i++) {
            accountNumbers[i] = 1000 + i;
            balances[i] = random.nextInt(bound: 100001);
            System.out.println("Account No. " + accountNumbers[i] + " Balance " + balances[i]);
        }
        quickSort(accountNumbers, balances, low: 0, n - 1);
        System.out.println(x: "\nAccounts sorted by highest balance:");
        for (int i = 0; i < n; i++) {
            System.out.println("Account No. " + accountNumbers[i] + " Balance " + balances[i]);
        }
    }
    public static void quickSort(int[] accountNumbers, int[] balances, int low, int high) {
        if (low < high) {
            int partitionIndex = partition(accountNumbers, balances, low, high);

            quickSort(accountNumbers, balances, low, partitionIndex - 1);
            quickSort(accountNumbers, balances, partitionIndex + 1, high);
        }
    }
```

```
    public static int partition(int[] accountNumbers, int[] balances, int low, int high) {
        int pivot = balances[high];
        int i = low - 1;

        for (int j = low; j < high; j++) {
            if (balances[j] >= pivot) {
                i++;

                int tempBalance = balances[i];
                balances[i] = balances[j];
                balances[j] = tempBalance;

                int tempAccount = accountNumbers[i];
                accountNumbers[i] = accountNumbers[j];
                accountNumbers[j] = tempAccount;
            }
        }

        int tempBalance = balances[i + 1];
        balances[i + 1] = balances[high];
        balances[high] = tempBalance;

        int tempAccount = accountNumbers[i + 1];
        accountNumbers[i + 1] = accountNumbers[high];
        accountNumbers[high] = tempAccount;

        return i + 1;
    }
}
```

OUTPUT:

```
Initial Account Balances:
Account No. 1000 Balance 39435
Account No. 1001 Balance 31544
Account No. 1002 Balance 9661
Account No. 1003 Balance 49846
Account No. 1004 Balance 7977
Account No. 1005 Balance 73687
Account No. 1006 Balance 5833
Account No. 1007 Balance 4799
Account No. 1008 Balance 69956
Account No. 1009 Balance 86709

Accounts sorted by highest balance:
Account No. 1009 Balance 86709
Account No. 1005 Balance 73687
Account No. 1008 Balance 69956
Account No. 1003 Balance 49846
Account No. 1000 Balance 39435
Account No. 1001 Balance 31544
Account No. 1002 Balance 9661
Account No. 1004 Balance 7977
Account No. 1006 Balance 5833
Account No. 1007 Balance 4799
```

2. **Write a program which takes an unordered list of integers (or any other objects e.g. String), you have to rearrange the list in their natural order using merge sort.**

INPUT:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class Lab05 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(source: System.in);

        System.out.println(x: "Enter the number of elements:");
        int n = sc.nextInt();
        List<Integer> list = new ArrayList<>();

        System.out.println(x: "Enter the elements:");
        for (int i = 0; i < n; i++) {
            list.add(e: sc.nextInt());
        }
        list = mergeSort(list);
        System.out.println(x: "Sorted list:");
        for (int num : list) {
            System.out.print(num + " ");
        }
        sc.close();
    }
    public static List<Integer> mergeSort(List<Integer> list) {
        if (list.size() <= 1) {
            return list;
        }
        int mid = list.size() / 2;
        List<Integer> left = list.subList(fromIndex:0, toIndex: mid);
        List<Integer> right = list.subList(fromIndex:mid, toIndex: list.size());

        left = mergeSort(list:left);
        right = mergeSort(list:right);

        return merge(left, right);
    }
```

```
    public static List<Integer> merge(List<Integer> left, List<Integer> right) {
        List<Integer> result = new ArrayList<>();
        int i = 0, j = 0;

        while (i < left.size() && j < right.size()) {
            if (left.get(index: i) <= right.get(index: j)) {
                result.add(e: left.get(index: i));
                i++;
            } else {
                result.add(e: right.get(index: j));
                j++;
            }
        }
        while (i < left.size()) {
            result.add(e: left.get(index: i));
            i++;
        }
        while (j < right.size()) {
            result.add(e: right.get(index: j));
            j++;
        }
        return result;
    }
}
```

OUTPUT:

```
Enter the number of elements:
3
Enter the elements:
2
8
5
Sorted list:
2 5 8 BUILD SUCCESSFUL (total time: 6 seconds)
```

3. **You are given an unordered list of integers or strings. Write a program to Take this list as input. Sort it in natural order using Merge Sort. For integers, this means ascending order. For strings, this means alphabetical order. Print the sorted list.**

INPUT:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class Lab05 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(source: System.in);

        System.out.println(x: "Do you want to sort integers or strings? (type 'int' or 'string')");
        String choice = sc.nextLine();

        if (choice.equalsIgnoreCase(anotherString: "int")) {
            System.out.println(x: "How many integers do you want to sort?");
            int n = sc.nextInt();
            List<Integer> numbers = new ArrayList<>();
            System.out.println(x: "Enter the integers:");
            for (int i = 0; i < n; i++) {
                numbers.add(e: sc.nextInt());
            }
            List<Integer> sortedNumbers = mergeSort(list:numbers);
            System.out.println("Sorted Integers: " + sortedNumbers);
        } else if (choice.equalsIgnoreCase(anotherString: "string")) {
            System.out.println(x: "How many strings do you want to sort?");
            int n = sc.nextInt();
            sc.nextLine();
            List<String> words = new ArrayList<>();
            System.out.println(x: "Enter the strings:");
            for (int i = 0; i < n; i++) {
                words.add(e: sc.nextLine());
            }
            List<String> sortedWords = mergeSort(list:words);
            System.out.println("Sorted Strings: " + sortedWords);
        } else {
            System.out.println(x: "Invalid choice. Please restart the program.");
        }
```

```
        sc.close();
    }
    public static <T extends Comparable<T>> List<T> mergeSort(List<T> list) {
        if (list.size() <= 1) {
            return list;
        }
        int mid = list.size() / 2;
        List<T> left = mergeSort(new ArrayList<>(c: list.subList(fromIndex:0, toIndex: mid)));
        List<T> right = mergeSort(new ArrayList<>(c: list.subList(fromIndex:mid, toIndex: list.size())));

        return merge(left, right);
    }
    public static <T extends Comparable<T>> List<T> merge(List<T> left, List<T> right) {
        List<T> result = new ArrayList<>();
        int i = 0, j = 0;

        while (i < left.size() && j < right.size()) {
            if (left.get(index:i).compareTo(o: right.get(index:j)) <= 0) {
                result.add(e: left.get(i++));
            } else {
                result.add(e: right.get(j++));
            }
        }
        result.addAll(c: left.subList(fromIndex:i, toIndex: left.size()));
        result.addAll(c: right.subList(fromIndex:j, toIndex: right.size()));

        return result;
    }
}
```

OUTPUT:

```
Do you want to sort integers or strings? (type 'int' or 'string')
int
How many integers do you want to sort?
4
Enter the integers:
2
6
8
5
Sorted Integers: [2, 5, 6, 8]
```

```
Do you want to sort integers or strings? (type 'int' or 'string')
string
How many strings do you want to sort?
3
Enter the strings:
laiba
waniya
mariam
Sorted Strings: [laiba, mariam, waniya]
```

4. You are given a set of bank accounts, each with a unique account number and a balance. Write a Java program to Declare an array of size n to store account balances. Initialize each balance randomly with values between 0 and 100,000. Sort the accounts in descending order of their balances using Quick Sort. Print the sorted list in the format

INPUT:

```
import java.util.Random;
public class Lab05 {
    public static void main(String[] args) {
        int n = 5;
        int[] accountNumbers = new int[n];
        int[] balances = new int[n];
        Random random = new Random();

        System.out.println(x: "Initial Account Details:");
        for (int i = 0; i < n; i++) {
            accountNumbers[i] = 1000 + i;
            balances[i] = random.nextInt(bound: 100001);
            System.out.println("Account No: " + accountNumbers[i] + ", Balance: " + balances[i]);
        }
        quickSort(balances, accountNumbers, low: 0, n - 1);
        System.out.println(x: "\nSorted Account Details (Descending Order of Balance):");
        for (int i = 0; i < n; i++) {
            System.out.println("Account No: " + accountNumbers[i] + ", Balance: " + balances[i]);
        }
    }
    public static void quickSort(int[] balances, int[] accountNumbers, int low, int high) {
        if (low < high) {
            int pivotIndex = partition(balances, accountNumbers, low, high);
            quickSort(balances, accountNumbers, low, pivotIndex - 1);
            quickSort(balances, accountNumbers, pivotIndex + 1, high);
        }
    }
```

```
    public static int partition(int[] balances, int[] accountNumbers, int low, int high) {
        int pivot = balances[high];
        int i = low - 1;

        for (int j = low; j < high; j++) {
            if (balances[j] > pivot) {
                i++;
                swap(balances, accountNumbers, i, j);
            }
        }
        swap(balances, accountNumbers, i + 1, j: high);
        return i + 1;
    }
    public static void swap(int[] balances, int[] accountNumbers, int i, int j) {
        int tempBalance = balances[i];
        balances[i] = balances[j];
        balances[j] = tempBalance;

        int tempAccount = accountNumbers[i];
        accountNumbers[i] = accountNumbers[j];
        accountNumbers[j] = tempAccount;
    }
}
```

OUTPUT:

```
Initial Account Details:
Account No: 1000, Balance: 24200
Account No: 1001, Balance: 63584
Account No: 1002, Balance: 24449
Account No: 1003, Balance: 68459
Account No: 1004, Balance: 73699

Sorted Account Details (Descending Order of Balance):
Account No: 1004, Balance: 73699
Account No: 1003, Balance: 68459
Account No: 1001, Balance: 63584
Account No: 1002, Balance: 24449
Account No: 1000, Balance: 24200
```