LAB # 04

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.

```
//2022f-BSE-301
no usages
class SelectionSort {
    1 usage
    public static void selectionSort(int[] 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] < arr[minIndex]) {
                    minIndex = j;
                }
            }

            int temp = arr[minIndex];
            arr[minIndex] = arr[i];
            arr[i] = temp;

            System.out.println("Sorted Value: " + arr[i] + ", Location: " + i);
        }

        System.out.println("Sorted Value: " + arr[n-1] + ", Location: " + (n-1));
    }

    no usages
    public static void main(String[] args) {
        int[] arr = {63, 84, 25, 10, 21, 1, 80};

        System.out.println("Original Array:");
        for (int i = 0; i < arr.length; i++) {
            System.out.println("Value: " + arr[i] + ", Location: " + i);
        }

        selectionSort(arr);
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Original Array:
Value: 63, Location: 0
Value: 84, Location: 1
Value: 25, Location: 2
Value: 10, Location: 3
Value: 21, Location: 4
Value: 1, Location: 5
Value: 80, Location: 6
Sorted Value: 1, Location: 0
Sorted Value: 10, Location: 1
Sorted Value: 21, Location: 2
Sorted Value: 25, Location: 3
Sorted Value: 63, Location: 4
Sorted Value: 80, Location: 5
Sorted Value: 84, Location: 6

Process finished with exit code 0
```

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.

```java
//2022f-BSE-301
import java.util.Arrays;

no usages
class BubbleSort {
    no usages
    public static void main(String[] args) {
        int[] numbers = {5, 3, 9, 1, 7, 2, 8, 4, 6, 0};
        System.out.println("Original array: " + Arrays.toString(numbers));
        int n = numbers.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (numbers[j] > numbers[j + 1]) {

                    int temp = numbers[j];
                    numbers[j] = numbers[j + 1];
                    numbers[j + 1] = temp;
                }
            }
            System.out.println("After iteration " + (i + 1) + ": " + Arrays.toString(numbers));
        }

        System.out.println("Sorted array: " + Arrays.toString(numbers));
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Original array: [5, 3, 9, 1, 7, 2, 8, 4, 6, 0]
After iteration 1: [3, 5, 1, 7, 2, 8, 4, 6, 0, 9]
After iteration 2: [3, 1, 5, 2, 7, 4, 6, 0, 8, 9]
After iteration 3: [1, 3, 2, 5, 4, 6, 0, 7, 8, 9]
After iteration 4: [1, 2, 3, 4, 5, 0, 6, 7, 8, 9]
After iteration 5: [1, 2, 3, 4, 0, 5, 6, 7, 8, 9]
After iteration 6: [1, 2, 3, 0, 4, 5, 6, 7, 8, 9]
After iteration 7: [1, 2, 0, 3, 4, 5, 6, 7, 8, 9]
After iteration 8: [1, 0, 2, 3, 4, 5, 6, 7, 8, 9]
After iteration 9: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Sorted array: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Process finished with exit code 0
```

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

Nabeera Siddiqui 2022f-BSE-301

```java
//2022f-BSE-301
import java.util.Arrays;

no usages
class MergeSort {
    no usages
    public static void main(String[] args) {
        int[] numbers = {6, 1, 8, 2, 7, 5, 4};
        System.out.println("Original array: " + Arrays.toString(numbers));

        mergeSort(numbers, left: 0, right: numbers.length - 1);

        System.out.println("Sorted array: " + Arrays.toString(numbers));
    }

    3 usages
    public static void mergeSort(int[] arr, int left, int right) {
        if (left < right) {
            int mid = left + (right - left) / 2;

            mergeSort(arr, left, mid);
            mergeSort(arr, left: mid + 1, right);

            merge(arr, left, mid, right);
        }
    }
    1 usage
    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 j = 0; j < n2; ++j)
            rightArray[j] = arr[mid + 1 + j];

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

```
            j++;
        }
        k++;
        System.out.println("After iteration: " + Arrays.toString(arr));
    }

    while (i < n1) {
        arr[k] = leftArray[i];
        i++;
        k++;
        System.out.println("After iteration: " + Arrays.toString(arr));
    }

    while (j < n2) {
        arr[k] = rightArray[j];
        j++;
        k++;
        System.out.println("After iteration: " + Arrays.toString(arr));

    }
}
}
```

Output:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Original array: [6, 1, 8, 2, 7, 5, 4]
After iteration: [1, 1, 8, 2, 7, 5, 4]
After iteration: [1, 6, 8, 2, 7, 5, 4]
After iteration: [1, 6, 2, 2, 7, 5, 4]
After iteration: [1, 6, 2, 8, 7, 5, 4]
After iteration: [1, 6, 2, 8, 7, 5, 4]
After iteration: [1, 2, 2, 8, 7, 5, 4]
After iteration: [1, 2, 6, 8, 7, 5, 4]
After iteration: [1, 2, 6, 8, 7, 5, 4]
After iteration: [1, 2, 6, 8, 5, 5, 4]
After iteration: [1, 2, 6, 8, 5, 7, 4]
After iteration: [1, 2, 6, 8, 4, 7, 4]
After iteration: [1, 2, 6, 8, 4, 5, 4]
After iteration: [1, 2, 6, 8, 4, 5, 7]
After iteration: [1, 2, 6, 8, 4, 5, 7]
After iteration: [1, 2, 6, 8, 4, 5, 7]
After iteration: [1, 2, 4, 8, 4, 5, 7]
After iteration: [1, 2, 4, 5, 4, 5, 7]
After iteration: [1, 2, 4, 5, 6, 5, 7]
After iteration: [1, 2, 4, 5, 6, 7, 7]
After iteration: [1, 2, 4, 5, 6, 7, 8]
Sorted array: [1, 2, 4, 5, 6, 7, 8]

Process finished with exit code 0
```

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

```
//2022f-BSE-301
 9 usages
 class Account {
    3 usages
    int accountNo;
    5 usages
    int balance;

    1 usage
    public Account(int accountNo, int balance) {
        this.accountNo = accountNo;
        this.balance = balance;
    }
}

1 usage
class QuickSort {

    3 usages
    public static void quickSort(Account[] accounts, int low, int high) {
        if (low < high) {
            int pi = partition(accounts, low, high);

            quickSort(accounts, low, high: pi - 1);
            quickSort(accounts, low: pi + 1, high);
        }
    }

    1 usage
    private static int partition(Account[] accounts, int low, int high) {
        int pivot = accounts[high].balance;
        int i = (low - 1);

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

                Account temp = accounts[i];
                accounts[i] = accounts[j];
                accounts[j] = temp;
            }
        }
```

```
        Account temp = accounts[i + 1];
        accounts[i + 1] = accounts[high];
        accounts[high] = temp;

        return i + 1;
    }
}

no usages
public class Main {
    no usages
    public static void main(String[] args) {
        int n = 5;
        Account[] accounts = new Account[n];

        for (int i = 0; i < n; i++) {
            accounts[i] = new Account( accountNo: i + 1, (int) (Math.random() * 100000));
        }

        System.out.println("Unsorted accounts:");
        for (Account account : accounts) {
            System.out.println("Account No. " + account.accountNo + " Balance " + account.balance);
        }

        QuickSort.quickSort(accounts, low: 0, high: n - 1);
        System.out.println("\nSorted accounts:");
        for (Account account : accounts) {
            System.out.println("Account No. " + account.accountNo + " Balance " + account.balance);
        }
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Unsorted accounts:
Account No. 1 Balance 33689
Account No. 2 Balance 64423
Account No. 3 Balance 49941
Account No. 4 Balance 95435
Account No. 5 Balance 77201

Sorted accounts:
Account No. 4 Balance 95435
Account No. 5 Balance 77201
Account No. 2 Balance 64423
Account No. 3 Balance 49941
Account No. 1 Balance 33689

Process finished with exit code 0
```

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.

```java
//2022f-BSE-301
import java.util.*;

no usages
class MergeSort {

    3 usages
    public static void mergeSort(List<Comparable> list) {
        if (list.size() <= 1) {
            return;
        }

        int mid = list.size() / 2;
        List<Comparable> left = new ArrayList<>(list.subList(0, mid));
        List<Comparable> right = new ArrayList<>(list.subList(mid, list.size()));

        mergeSort(left);
        mergeSort(right);

        merge(list, left, right);
    }

    1 usage
    private static void merge(List<Comparable> result, List<Comparable> left, List<Comparable> right) {
        int i = 0, j = 0, k = 0;

        while (i < left.size() && j < right.size()) {
            if (left.get(i).compareTo(right.get(j)) <= 0) {
                result.set(k++, left.get(i++));
            } else {
                result.set(k++, right.get(j++));
            }
        }

        while (i < left.size()) {
            result.set(k++, left.get(i++));
        }

        while (j < right.size()) {
            result.set(k++, right.get(j++));
        }
    }

    no usages
    public static void main(String[] args) {
        List<Comparable> list = new ArrayList<>(Arrays.asList( 5, 1, 3, 2, 6, 4));

        System.out.println("Original List: " + list);
        mergeSort(list);
        System.out.println("Sorted List: " + list);
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ
Original List: [5, 1, 3, 2, 6, 4]
Sorted List: [1, 2, 3, 4, 5, 6]

Process finished with exit code 0
```