

LAB # 05

Sorting on Linear Array

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

Lab Task

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.

```
public class SelectionSort {  
    public static void selectionSort(int[] array) {  
        int n = array.length;  
        for (int i = 0; i < 4; i++) {  
            int minIndex = i;  
            for (int j = i + 1; j < n; j++) {  
                if (array[j] < array[minIndex]) {  
                    minIndex = j;  
                }  
            }  
            if (minIndex != i) {  
                // Swap  
                int temp = array[i];  
                array[i] = array[minIndex];  
                array[minIndex] = temp;  
            }  
            printArray(array, i);  
        }  
    }  
    public static void printArray(int[] array, int index) {  
        for (int i = 0; i < array.length; i++) {  
            System.out.print(array[i] + " ");  
        }  
        System.out.println(" (Index: " + index + ")");  
    }  
    public static void main(String[] args) {  
        int[] array = {50, 20, 80, 10, 40};  
        selectionSort(array);  
    }  
}
```

```
run:
10 20 80 50 40 (Index: 0)
10 20 80 50 40 (Index: 1)
10 20 40 50 80 (Index: 2)
10 20 40 50 80 (Index: 3)
BUILD SUCCESSFUL (total time: 0 s
```

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.

```
package laab5;
import java.util.Scanner;
public class Laab5 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int[] array = new int[10];
        System.out.println("Enter 10 numbers:");

        for (int i = 0; i < 10; i++) {
            array[i] = input.nextInt();
        }

        System.out.println("Bubble Sort Process:");
        for (int i = 0; i < array.length - 1; i++) {
            for (int j = 0; j < array.length - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    int temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }

            // Print iteration
            System.out.print("Iteration " + (i + 1) + ": ");
            for (int num : array) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }
}
```

```
run:
```

```
Enter 10 numbers:
```

```
2
```

```
3
```

```
5
```

```
24
```

```
19
```

```
392
```

```
277
```

```
44
```

```
15
```

```
12
```

```
Bubble Sort Process:
```

```
Iteration 1: 2 3 5 19 24 277 44 15 12 392
```

```
Iteration 2: 2 3 5 19 24 44 15 12 277 392
```

```
Iteration 3: 2 3 5 19 24 15 12 44 277 392
```

```
Iteration 4: 2 3 5 19 15 12 24 44 277 392
```

```
Iteration 5: 2 3 5 15 12 19 24 44 277 392
```

```
Iteration 6: 2 3 5 12 15 19 24 44 277 392
```

```
Iteration 7: 2 3 5 12 15 19 24 44 277 392
```

```
Iteration 8: 2 3 5 12 15 19 24 44 277 392
```

```
Iteration 9: 2 3 5 12 15 19 24 44 277 392
```

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.

```
package laab5;
import java.util.Scanner;
import java.util.Arrays;
import java.util.Random;
public class Laab5 {
    public static void main(String[] args) {
        int[] arr = new Random().ints(10, 0, 100).toArray();
        System.out.println("Original Array: " + Arrays.toString(arr));
        mergeSort(arr, 0, arr.length - 1);
    }

    private static void mergeSort(int[] arr, int left, int right) {
        if (left < right) {
            int mid = (left + right) / 2;
            mergeSort(arr, left, mid);
            mergeSort(arr, mid + 1, right);
            merge(arr, left, mid, right);
            System.out.println("Array after merging: " + Arrays.toString(arr));
        }
    }

    private static void merge(int[] arr, int left, int mid, int right) {
        int[] temp = new int[right - left + 1];
        int i = left, j = mid + 1, k = 0;

        while (i <= mid && j <= right) {
            if (arr[i] <= arr[j]) {
                temp[k++] = arr[i++];
            } else {
                temp[k++] = arr[j++];
            }
        }

        while (i <= mid) {
            temp[k++] = arr[i++];
        }
    }
}
```

```
while (i <= mid) {  
    temp[k++] = arr[i++];  
}  
  
while (j <= right) {  
    temp[k++] = arr[j++];  
}  
  
System.arraycopy(temp, 0, arr, left, temp.length);  
}
```

run:

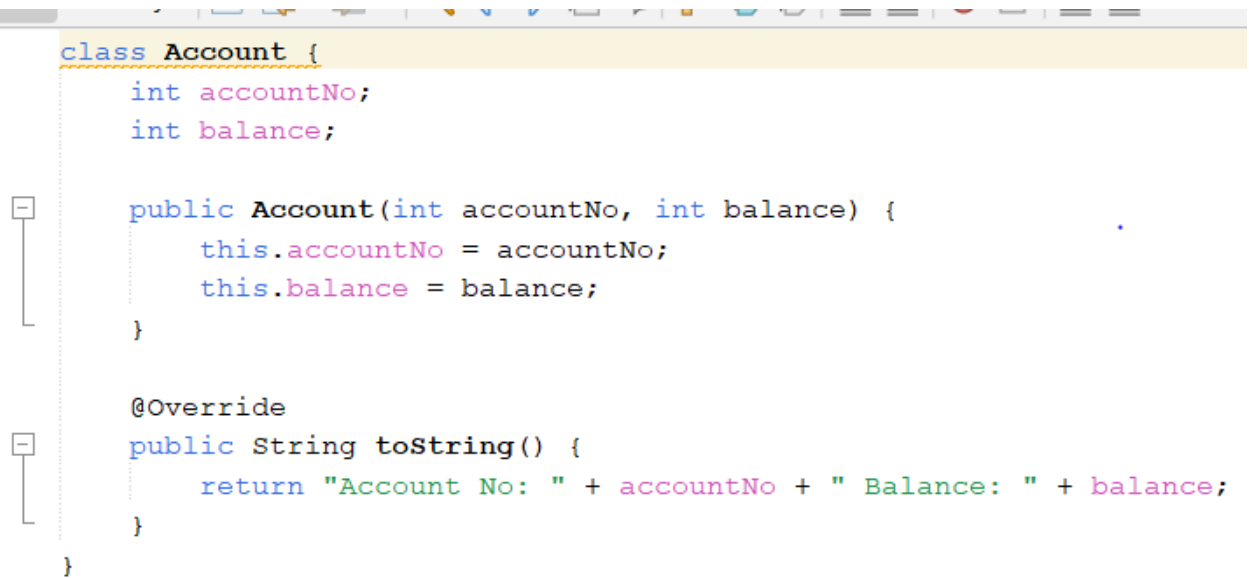
```
Original Array: [1, 73, 33, 42, 76, 13, 32, 5, 43, 15]  
Array after merging: [1, 73, 33, 42, 76, 13, 32, 5, 43, 15]  
Array after merging: [1, 33, 73, 42, 76, 13, 32, 5, 43, 15]  
Array after merging: [1, 33, 73, 42, 76, 13, 32, 5, 43, 15]  
Array after merging: [1, 33, 42, 73, 76, 13, 32, 5, 43, 15]  
Array after merging: [1, 33, 42, 73, 76, 13, 32, 5, 43, 15]  
Array after merging: [1, 33, 42, 73, 76, 5, 13, 32, 43, 15]  
Array after merging: [1, 33, 42, 73, 76, 5, 13, 32, 15, 43]  
Array after merging: [1, 33, 42, 73, 76, 5, 13, 15, 32, 43]  
Array after merging: [1, 5, 13, 15, 32, 33, 42, 43, 73, 76]  
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



```
class Account {  
    int accountNo;  
    int balance;  
  
    public Account(int accountNo, int balance) {  
        this.accountNo = accountNo;  
        this.balance = balance;  
    }  
  
    @Override  
    public String toString() {  
        return "Account No: " + accountNo + " Balance: " + balance;  
    }  
}
```

```
import java.util.Random;

public class QuickSortAccounts {

    public static void main(String[] args) {
        int n = 10;
        Account[] accounts = new Account[n];
        Random rand = new Random();

        for (int i = 0; i < n; i++) {
            accounts[i] = new Account(rand.nextInt(10000), rand.nextInt(100001));
        }

        System.out.println("Unsorted Accounts:");
        for (Account account : accounts) {
            System.out.println(account);
        }

        quickSort(accounts, 0, accounts.length - 1);

        System.out.println("\nSorted Accounts (Descending Order of Balance):");
        for (Account account : accounts) {
            System.out.println(account);
        }
    }

    private static void quickSort(Account[] arr, int low, int high) {
        if (low < high) {
            int pi = partition(arr, low, high);
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }

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

        for (int j = low; j < high; j++) {
```

```
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

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

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

    Account temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return i + 1;
}
```

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.


```
import java.util.Arrays;
import java.util.Random;
public class Laab5 {
    public static void main(String[] args) {
        int[] arr = {34, 12, 45, 2, 8, 19, 50, 29};
        System.out.println("Unordered List: " + Arrays.toString(arr));

        mergeSort(arr, 0, arr.length - 1);
        System.out.println("Sorted List (Natural Order): " + Arrays.toString(arr));
    }

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

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

            merge(arr, left, mid, right);
        }
    }

    private 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];

        System.arraycopy(arr, left, leftArray, 0, n1);
        System.arraycopy(arr, mid + 1, rightArray, 0, n2);

        int i = 0, j = 0, k = left;

        while (i < n1 && j < n2) {
            while (i < n1 && j < n2) {
                arr[k++] = (leftArray[i] <= rightArray[j]) ? leftArray[i++] : rightArray[j++];
            }

            while (i < n1) arr[k++] = leftArray[i++];
            while (j < n2) arr[k++] = rightArray[j++];
        }
    }
}
```

Unordered List: [34, 12, 45, 2, 8, 19, 50, 29]

Sorted List (Natural Order): [2, 8, 12, 19, 29, 34, 45, 50]

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.

```
import java.util.Scanner;
public class Laab5 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the number of elements:");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        System.out.println("Enter the elements (integers or strings):");
        String[] arr = new String[n];
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextLine();
        }

        mergeSort(arr, 0, arr.length - 1);

        System.out.println("Sorted List: " + Arrays.toString(arr));
    }

    private static void mergeSort(String[] arr, int left, int right) {
        if (left < right) {
            int mid = (left + right) / 2;
            mergeSort(arr, left, mid);
            mergeSort(arr, mid + 1, right);
            merge(arr, left, mid, right);
        }
    }

    private static void merge(String[] arr, int left, int mid, int right) {
        String[] temp = new String[right - left + 1];
        int i = left, j = mid + 1, k = 0;

        while (i <= mid && j <= right) {
```

```

        while (i <= mid && j <= right) {
            if (arr[i].compareTo(arr[j]) <= 0) {
                temp[k++] = arr[i++];
            } else {
                temp[k++] = arr[j++];
            }
        }

        while (i <= mid) temp[k++] = arr[i++];
        while (j <= right) temp[k++] = arr[j++];

        System.arraycopy(temp, 0, arr, left, temp.length);
    }
}

```

run:

Enter the number of elements:

5

Enter the elements (integers or strings):

apple

cherry

mosambi

pomogranette

grape

Sorted List: [apple, cherry, grape, mosambi, pomogranette]

ENTER CHOICE TO CONTINUE (1-3) : 3

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

```
import java.util.Random;
public class QuickSortBankAccounts {
    public static void main(String[] args) {
        int n = 10;
        BankAccount[] accounts = new BankAccount[n];
        Random r= new Random();

        for (int i = 0; i < n; i++) {
            accounts[i] = new BankAccount(r.nextInt(10000), r.nextInt(100001));
        }

        System.out.println("Unsorted Accounts:");
        for (BankAccount account : accounts) {
            System.out.println(account);
        }

        quickSort(accounts, 0, accounts.length - 1);

        System.out.println("\nSorted Accounts (Descending Order of Balance):");
        for (BankAccount account : accounts) {
            System.out.println(account);
        }
    }

    private static void quickSort(BankAccount[] arr, int low, int high) {
        if (low < high) {
            int pi = partition(arr, low, high);
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }

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

        for (int j = low; j < high; j++) {
```

```
    for (int j = low; j < high; j++)  
        if (arr[j].balance > pivot) {  
            i++;  
            BankAccount temp = arr[i]  
            arr[i] = arr[j];  
            arr[j] = temp;  
        }  
    }  
  
    BankAccount temp = arr[i + 1];  
    arr[i + 1] = arr[high];  
    arr[high] = temp;  
  
    return i + 1;  
}
```

Unsorted Accounts:

Account No: 8754 Balance: 21691
Account No: 4847 Balance: 20653
Account No: 5907 Balance: 75284
Account No: 3860 Balance: 69468
Account No: 1936 Balance: 64899
Account No: 7164 Balance: 69545
Account No: 913 Balance: 50034
Account No: 6917 Balance: 40787
Account No: 3214 Balance: 20536
Account No: 1435 Balance: 38908

Sorted Accounts (Descending Order of Balance):

Account No: 5907 Balance: 75284
Account No: 7164 Balance: 69545
Account No: 3860 Balance: 69468
Account No: 1936 Balance: 64899
Account No: 913 Balance: 50034
Account No: 6917 Balance: 40787
Account No: 1435 Balance: 38908
Account No: 8754 Balance: 21691
Account No: 4847 Balance: 20653
Account No: 3214 Balance: 20536