PRAKTIKUM 11
ALGORITMA PENGURUTAN (BUBBLE SORT)

A. TUJUAN PEMBELAJARAN

1. Memahami step by step algoritma pengurutan *bubble sort.*
2. Mampu mengimplementasikan algoritma pengurutan *bubble sort* dengan berbagai macam parameter berupa tipe data primitif atau tipe Generic.
3. Mampu mengimplementasikan algoritma pengurutan *bubble sort* secara ascending dan descending.

B. DASAR TEORI

Algoritma *Bubble Sort*

Metode gelembung (*bubble sort*) sering juga disebut dengan metode penukaran (*exchange sort*) adalah metode yang mengurutkan data dengan cara membandingkan masing-masing elemen, kemudian melakukan penukaran bila perlu. Metode ini mudah dipahami dan diprogram, tetapi bila dibandingkan dengan metode lain yang kita pelajari, metode ini merupakan metode yang paling tidak efisien.

Proses pengurutan metode gelembung ini menggunakan dua kalang. Kalang pertama melakukan pengulangan dari elemen ke 1 sampai dengan elemen ke N-1 (misalnya variable i), sedangkan kalang kedua melakukan pengulangan menurun dari elemen ke N-1 sampai elemen ke i (misalnya variable j). Pada setiap pengulangan, elemen ke j-1 dibandingkan dengan elemen ke j. Apabila data ke j-1 lebih besar daripada data ke j, dilakukan penukaran.

Algoritma gelembung dapat dituliskan sebagai berikut :

1 i ← 0

2 selama (i < N) kerjakan baris 3 sampai dengan 7

3 j ← N - 1

4 Selama (j >= i) kerjakan baris 5 sampai dengan 7

5 Jika (Data[j-1] > Data[j]) maka tukar Data[j-1] dengan Data[j]

6 $j \leftarrow j - 1$

7 $i \leftarrow i + 1$

Untuk lebih memperjelas langkah-langkah algoritma gelembung dapat dilihat pada tabel 6.3. Proses pengurutan pada tabel 6.3 dapat dijelaskan sebagai berikut:

- Pada saat i=1, nilai j diulang dari 9 sampai dengan 1. Pada pengulangan pertama Data[9] dibandingkan Data[8], karena 20 < 31 maka Data[9] dan Data[8] ditukar. Pada pengulangan kedua Data[8] dibandingkan Data[7], karena 20 > 15 maka proses dilanjutkan. Demikian seterusnya sampai j=1.
- Pada saat i=2, nilai j diulang dari 9 sampai dengan 2. Pada pengulangan pertama Data[9] dibandingkan Data[8], karena 31 > 20 maka proses dilanjutkan. Pada pengulangan kedua Data[8] dibandingkan Data[7], karena 20 < 23 Data[8] dan Data[7] ditukar. Demikian seterusnya sampai j=2.
- Pada saat i=3, nilai j diulang dari 9 sampai dengan 3. Pada pengulangan pertama Data[9] dibandingkan Data[8], karena 31 > 23 maka proses dilanjutkan. Pada pengulangan kedua Data[8] dibandingkan Data[7], karena 23 > 20 maka proses dilanjutkan. Demikian seterusnya sampai j=3.
- Dan seterusnya sampai dengan i=8.

Tabel 1 Proses Pengurutan dengan Metode Gelembung

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=1; j=9	12	35	9	11	3	17	23	15	**31**	**20**
j=8	12	35	9	11	3	17	23	**15**	**20**	31
j=7	12	35	9	11	3	17	**23**	**15**	20	31
j=6	12	35	9	11	3	**17**	**15**	23	20	31
j=5	12	35	9	11	**3**	**15**	17	23	20	31
j=4	12	35	9	**11**	**3**	15	17	23	20	31
j=3	12	35	**9**	**3**	11	15	17	23	20	31
j=2	12	**35**	**3**	9	11	15	17	23	20	31

j=1	**12**	**3**	35	9	11	15	17	23	20	31
i=2; j=9	3	12	35	9	11	15	17	23	**20**	**31**
j=8	3	12	35	9	11	15	17	**23**	**20**	31
j=7	3	12	35	9	11	15	**17**	**20**	23	31
j=6	3	12	35	9	11	**15**	**17**	20	23	31
j=5	3	12	35	9	**11**	**15**	17	20	23	31
j=4	3	12	35	**9**	**11**	15	17	20	23	31
j=3	3	12	**35**	**9**	11	15	17	20	23	31
j=2	3	**12**	**9**	35	11	15	17	20	23	31
i=3; j=9	3	9	12	35	11	15	17	20	**23**	**31**
j=8	3	9	12	35	11	15	17	**20**	**23**	31
j=7	3	9	12	35	11	15	**17**	**20**	23	31
j=6	3	9	12	35	11	**15**	**17**	20	23	31
j=5	3	9	12	35	**11**	**15**	17	20	23	31
j=4	3	9	12	**35**	**11**	15	17	20	23	31
j=3	3	9	**12**	**11**	35	15	17	20	23	31
i=4; j=9	3	9	11	12	35	15	17	20	**23**	**31**
j=8	3	9	11	12	35	15	17	**20**	**23**	31
j=7	3	9	11	12	35	15	**17**	**20**	23	31
j=6	3	9	11	12	35	**15**	**17**	20	23	31
j=5	3	9	11	12	**35**	**15**	17	20	23	31
j=4	3	9	11	**12**	**15**	35	17	20	23	31
i=5; j=9	3	9	11	12	15	35	17	20	**23**	**31**
j=8	3	9	11	12	15	35	17	**20**	**23**	31
j=7	3	9	11	12	15	35	**17**	**20**	23	31
j=6	3	9	11	12	15	**35**	**17**	20	23	31
j=5	3	9	11	12	**15**	**17**	35	20	23	31
i=6; j=9	3	9	11	12	15	17	35	20	**23**	**31**
j=8	3	9	11	12	15	17	35	**20**	**23**	31
j=7	3	9	11	12	15	17	**35**	**20**	23	31
j=6	3	9	11	12	15	**17**	**20**	35	23	31
i=7; j=9	3	9	11	12	15	17	20	35	**23**	**31**
j=8	3	9	11	12	15	17	20	**35**	**23**	31

j=7	3	9	11	12	15	17	**20**	**23**	35	31
i=8; j=9	3	9	11	12	15	17	20	23	**35**	**31**
j=8	3	9	11	12	15	17	20	**23**	**31**	35
i=9 j=9	3	9	11	12	15	17	20	23	**31**	**35**
Akhir	3	9	11	12	15	17	20	23	31	35

C. TUGAS PENDAHULUAN

Jelaskan algoritma pengurutan *bubble sort* secara *ascending* dengan data 5 6 3 1 2

D. PERCOBAAN

Percobaan 1 : *Bubble sort* secara *ascending* dengan data int.

```
public class BubbleDemo {
    public static void bubbleSort(int[] arr) {
        int i=1, j, n = arr.length;
        int temp;

        while (i<n){
            j = n-1 ;
            while(j>=i){
                if (arr[j-1]>arr[j]){
                    temp = arr[j];
                    arr[j] = arr[j-1];
                    arr[j-1] = temp;
                }
                j = j - 1;
            }
            i = i + 1;
        }
    }

    public static void tampil(int data[]) {
        for (int i = 0; i < data.length; i++) {
            System.out.print(data[i] + " ");
        }
        System.out.println();
    }
}

public class MainBubble {
    public static void main(String[] args) {
        int A[] = {10,6,8,3,1};
        BubbleDemo.tampil(A);
        BubbleDemo.bubbleSort(A);
        BubbleDemo.tampil(A);
    }
```

```
}
```

Percobaan 2 : *Bubble sort* secara *descending* dengan data int.

```
public class BubbleDemo {
    public static void bubbleSort(int[] arr) {
        int i=1, j, n = arr.length;
        int temp;

        while (i<n){
            j = n-1 ;
            while(j>=i){
                if (arr[j-1] < arr[j]){
                    temp = arr[j];
                    arr[j] = arr[j-1];
                    arr[j-1] = temp;
                }
                j = j - 1;
            }
            i = i + 1;
        }
    }

    public static void tampil(int data[]) {
        for (int i = 0; i < data.length; i++) {
            System.out.print(data[i] + " ");
        }
        System.out.println();
    }
}

public class MainBubble {
    public static void main(String[] args) {
        int A[] = {10,6,8,3,1};
        BubbleDemo.tampil(A);
        BubbleDemo.bubbleSort(A);
        BubbleDemo.tampil(A);
    }
}
```

Percobaan 3 : *Bubble sort* secara *ascending* dengan data double.

```
public class BubbleDemo {
    public static void bubbleSort(double[] arr) {
        int i=1, j, n = arr.length;
        double temp;

        while (i<n){
            j = n-1 ;
            while(j>=i){
                if (arr[j-1]>arr[j]){
                    temp = arr[j];
```

```
                    arr[j] = arr[j-1];
                    arr[j-1] = temp;
                }
                j = j - 1;
            }
            i = i + 1;
        }
    }

    public static void tampil(double data[]) {
        for (int i = 0; i < data.length; i++) {
            System.out.print(data[i] + " ");
        }
        System.out.println();
    }
}

public class MainBubble2 {
    public static void main(String[] args) {
        double A[] = {10.3,6.2,8.4,3.6,1.1};
        BubbleDemo.tampil(A);
        BubbleDemo.bubbleSort(A);
        BubbleDemo.tampil(A);
    }
}
```

E. LATIHAN

1. Buatlah program sorting Bubble dengan parameter array Integer (class Wrapper) !

```
public static void bubbleSort(Integer[] A){…}
```

2. Buatlah program sorting Bubble dengan parameter array Double (class Wrapper) !

```
public static void bubbleSort(Double[] A){…}
```

3. Buatlah fungsi tampil() untuk menampilkan data.

```
public static<T> void tampil(T data[]){ }
```

4. Lakukan pengujian fungsi bubbleSort(), dengan membuat fungsi main() sebagai berikut :

```
public class Demo1 {
    public static void main(String[] args) {
        //Data Integer
        Integer arr3[] = {1,5,6,2,8,9};
        BubbleDemo.bubbleSort(arr3);
        BubbleDemo.tampil(arr3);

        //data Double
        Double arr4[] = {1.3,5.2,6.6,2.7,8.8,9.1};
```

```
        BubbleDemo.bubbleSort(arr4);
        BubbleDemo.tampil(arr4);
    }
}
```

5. Buatlah program sorting Bubble dengan parameter array Number !

```
public static<T extends Number> void bubbleSort(T[] A){…}
```

6. Lakukan pengujian fungsi bubbleSort(), dengan membuat fungsi main() sebagai berikut :

```
public class Demo2 {
    public static void main(String[] args) {
        Float arr5[] = {1.3f,5.2f,6.6f,2.7f,8.8f,9.1f};
        BubbleDemo.bubbleSort(arr5);
        BubbleDemo.tampil(arr5);

        Byte arr6[] = {6,7,11,1,3,2};
        BubbleDemo.bubbleSort(arr6);
        BubbleDemo.tampil(arr6);
    }
}
```

7. Buatlah program sorting Bubble dengan parameter array yang memenuhi T extends Comparable !

```
public static <T extends Comparable> void bubbleSort2(T[] arr) {
}
```

8. Lakukan pengujian fungsi bubbleSort(), dengan membuat fungsi main() sebagai berikut :

```
public class Demo3 {
    public static void main(String[] args) {
        //data String
        String arr7[]=
{"jeruk","anggur","belimbing","jambu","kelengkeng"};
        BubbleDemo.bubbleSort2(arr7);
        BubbleDemo.tampil(arr7);
    }
}
```

9. Buatlah class Mahasiswa dengan variable nrp dan nama yang memiliki tipe String ! Class Mahasiswa mengimplementasikan interface Comparable, selanjutnya implementasikan fungsi abstract compareTo(), untuk membandingkan dua objek mahasiswa berdasarkan nrp.

```
public class Mahasiswa implements Comparable <Mahasiswa> {
```

```
    private String nrp ;
    private String nama ;
    @Override
    public int compareTo(Mahasiswa o) {…}

    @Override
    public String toString() {…}
}
```

10. Lakukan pengujian fungsi bubbleSort() lagi, sebelumnya tambahkan pada fungsi main() seperti di bawah ini !

```
public class Demo4 {
    public static void main(String[] args) {
        Mahasiswa arr8[] = {new Mahasiswa("02", "Budi"), new
Mahasiswa("01", "Andi"), new Mahasiswa("04", "Udin"), new
Mahasiswa("03", "Candra")};
        BubbleDemo.bubbleSort2(arr8);
        BubbleDemo.tampil(arr8);
    }
}
```

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.