LAPORAN

Struktur Data Linear
Praktikum 4 : Buble Sort & Selection Sort

`

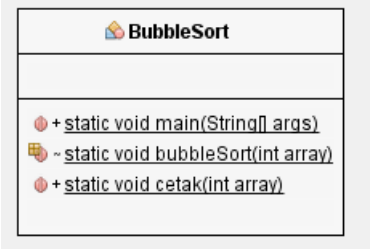
NAMA : Johanes Yogtan WR

NIM : 215314105

Program Studi INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA

1. Buble Sort

   - Diagram UML



   - Input (screenshot)
     Ascending :

```
package Modul3_Pengurutan;
public class BubbleSort {
    public static void main(String[] args) {
        int array[] = {5, 8, 26, 15, 11, 31};
        System.out.println("DATA SEBELUM DIURUTKAN");
        cetak(array);

        bubbleSort(array);
        System.out.println("\nDATA SESUDAH DIURUTKAN");
        cetak(array);
        System.out.println("");
    }
    static void bubbleSort(int array[]) {
        int jum;
        int n = array.length;
        for (int i = 0; i < n ; i++) {
            for (int j = 1; j < n - 1; j++) {
                if (array[j - 1] > array[j]) {
                    jum = array[j - 1];
                    array[j - 1] =  array[j];
                    array[j] = jum;
                }
            }
        }
    }
    public static void cetak(int array[]) {
       for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }

}
```

Descending :

```
package Modul3_Pengurutan;
public class BubbleSort {
    public static void main(String[] args) {
        int array[] = {5, 8, 26, 15, 11, 31};
        System.out.println("DATA SEBELUM DIURUTKAN");
        cetak(array);

        bubbleSort(array);
        System.out.println("\nDATA SESUDAH DIURUTKAN");
        cetak(array);
        System.out.println("");
    }
    static void bubbleSort(int array[]) {
        int jum;
        int n = array.length;
         for (int i = 0; i < n + 1; i++)
            for (int j = 0; j < n - 1; j++)
                if (array[j] <array[j + 1]) {
                    jum = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = jum;
                }
            }


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

}
```

- Output (screenshot)

Ascending :

```
run:
DATA SEBELUM DIURUTKAN
5 8 26 15 11 31
DATA SESUDAH DIURUTKAN
5 8 11 15 26 31
BUILD SUCCESSFUL (total time: 0 seconds)
```

Descending :

```
run:
DATA SEBELUM DIURUTKAN
5 8 26 15 11 31
DATA SESUDAH DIURUTKAN
31 26 15 11 8 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Ilustrasi

Iterasi 1

X[0]	Dengan	X[1]	(5 dengan 8)	Tak Berubah
X[1]	Dengan	X[2]	(8 dengan 26)	Tidak Berubah
X[2]	Dengan	X[3]	(26 dengan 15)	Ditukar
X[3]	Dengan	X[4]	(26 dengan 11)	Ditukar
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

Iterasi 2

X[0]	Dengan	X[1]	(5 dengan 8)	Tak Berubah
X[1]	Dengan	X[2]	(8 dengan 15)	Tidak Berubah
X[2]	Dengan	X[3]	(15 dengan 11)	Ditukar
X[3]	Dengan	X[4]	(15 dengan 26)	Tidak Berubah
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

Iterasi 3 **SAMPAI** Iterasi 6

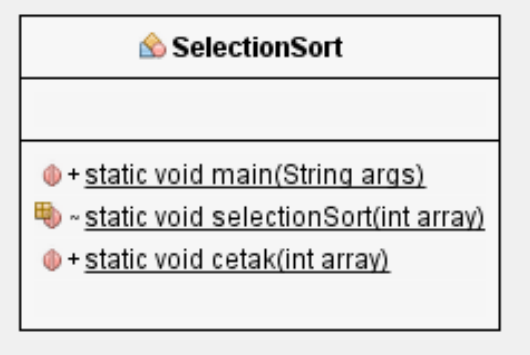
X[0]	Dengan	X[1]	(5 dengan 8)	Tak Berubah
X[1]	Dengan	X[2]	(8 dengan 11)	Tidak Berubah
X[2]	Dengan	X[3]	(11 dengan 15)	Tidak Berubah
X[3]	Dengan	X[4]	(15 dengan 26)	Tidak Berubah
X[4]	Dengan	X[5]	(26 dengan 31)	Tidak Berubah

- Penjelasan terkait ilustrasi (minimal proses dari 2 iterasi)

Program akan melakukan perulangan bersangkar, perulangan pertama untuk menjalankan iterasi dan perulangan kedua untuk menjalankan indek data setiap iterasi. Selama perulangan kedua akan menjalankan fungsi if, apakah array [j – 1] / array 0 lebih besar dari array 1. Jika iya, indek 0 akan disimpan ke variabel sementara dan data indek 1 ditukar ke indek 0, sehingga data indek 0 ditukar ke indek 1. jika tidak, array 1 akan berjalan membandingkan dengan data lainnya.

Seperti iterasi 1 dan 2 diatas, indek 0 dan 1 di bandingkan, ternyata 5 tidak lebih besar dari 8, sehingga data tidak ditukar, dilanjutkan indek 1 dan 2, ternyata 8 tidak lebih besar dari 15, sehingga data tidak ditukar, dilanjutkan indek 2 dan 3, ternyata 15 lebih besar dari 11, sehingga data indek 3 pindah ke 2 dan sebaliknya “ditukar”, dilanjutkan indek 2 tadi yang sudah dipindahkan untuk membandingkan, indek 3 dan 4, ternyata 15 tidak lebih besar dari 26, sehingga data tidak ditukar, dilanjutkan indek 4 dan 5, ternyata 26 tidak lebih besar dari 31, sehingga data tidak berubah. Begitu seterusnya hingga perulangan 1 selesai dan data tidak ditukar.

2. Selection Sort
   - Diagram UML



   - Input (screenshot)
   - Ascending :

```
package Modul3_Pengurutan;
public class SelectionSort {
    public static void main(String args[]) {
        int array[] = {5, 8, 26, 15, 11, 31};
        System.out.println("DATA SEBELUM DIURUTKAN");
        cetak(array);

        selectionSort(array);
        System.out.println("\nDATA SESUDAH DIURUTKAN");
        cetak(array);
        System.out.println("");
    }
    static void selectionSort(int array[]) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {
            int indek = i;
            for (int j = i + 1; j < n; j++) {
                if (array[j] < array[indek]) {
                    indek = j;
                }
            }
            int jum = array[indek];
            array[indek] = array[i];
            array[i] = jum;
        }
    }

    public static void cetak(int array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

Descending :

```
package Modul3_Pengurutan;
public class SelectionSort {
    public static void main(String args[]) {
        int array[] = {5, 8, 26, 15, 11, 31};
        System.out.println("DATA SEBELUM DIURUTKAN");
        cetak(array);

        selectionSort(array);
        System.out.println("\nDATA SESUDAH DIURUTKAN");
        cetak(array);
        System.out.println("");
    }
    static void selectionSort(int array[]) {
        int n = array.length;
        for (int i = 0; i < array.length - 1; i++) {
            int indek = i;
            for (int j = i + 1; j < n; j++) {
                if (array[j] > array[indek]) {
                    indek = j;
                }
            }
            int jum;
            if (array[i] < array[indek]) {
                jum = array[i];
                array[i] = array[indek];
                array[indek] = jum;
            }
        }
    }
    public static void cetak(int array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

- Output (screenshot)
- Ascending :

```
run:
DATA SEBELUM DIURUTKAN
5 8 26 15 11 31
DATA SESUDAH DIURUTKAN
5 8 11 15 26 31
BUILD SUCCESSFUL (total time: 0 seconds)
```

Descending :

```
run:
DATA SEBELUM DIURUTKAN
5 8 26 15 11 31
DATA SESUDAH DIURUTKAN
31 26 15 11 8 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Ilustrasi

Iterasi 1

X[0]	Dengan	X[1]	(5 dengan 8)	Tidak Berubah
X[1]	Dengan	X[2]	(5 dengan 26)	Tidak Berubah
X[2]	Dengan	X[3]	(5 dengan 15)	Tidak Berubah
X[3]	Dengan	X[4]	(5 dengan 11)	Tidak Berubah
X[4]	Dengan	X[5]	(5 dengan 31)	Tidak Berubah

Iterasi 2

X[0]	Dengan	X[1]	(8 dengan 26)	Tidak Berubah
X[1]	Dengan	X[2]	(8 dengan 15)	Tidak Berubah
X[2]	Dengan	X[3]	(8 dengan 11)	Tidak Berubah
X[3]	Dengan	X[4]	(8 dengan 31)	Tidak Berubah

Iterasi 3

X[0]	Dengan	X[1]	(26 dengan 15)	Tidak Berubah
X[1]	Dengan	X[2]	(26 dengan 11)	Ditukar
X[2]	Dengan	X[3]	(11 dengan 31)	Tidak Berubah

Iterasi 4 – Data Terurut

- Penjelasan terkait ilustrasi (minimal proses dari 2 iterasi)

Program akan melakukan perulangan bersangkar, perulangan pertama untuk menjalankan iterasi dan menyimpan indek yang sedang dijalankan dan perulangan kedua untuk menjalankan indek data setiap iterasi. Selama perulangan kedua akan menjalankan fungsi if, apakah array [j] / array 0 lebih besar dari array 1. Jika iya, indek 0 akan disimpan ke variabel sementara , jika tidak, array akan berjalan membandingkan dengan data lainnya.

Seperti iterasi 0 dan 1 diatas, indek 1 dan 0 di bandingkan, ternyata 8 tidak lebih kecil dari 5, sehingga data tidak ditukar, dilanjutkan indek 2 dan 0, ternyata 26 tidak lebih kecil dari 5, sehingga data tidak ditukar, dilanjutkan indek 3 dan 0, ternyata 15 tidak lebih kecil dari 5, sehingga data tidak ditukar", dilanjutkan indek 4 dan 0, ternyata 11 tidak lebih kecil dari 5, sehingga data tidak ditukar, dilanjutkan indek 5 dan 0, ternyata 31 tidak lebih kecil dari 5, sehingga data tidak ditukar. Begitu seterusnya hingga perulangan 1 selesai dan data tidak ditukar. Kita lanjutkan iterasi 3 yang dimana datanya tertukar, ketika perulangan dilakukan dan kondisi if tercapai data yang dibandingkan akan disimpan ke variabel semntara hingga nantinya akan ditukar.

3. Tabel Perbandingan Iterasi

Perbandingan Bublesort

	[0]	[1]	[2]	[3]	[4]	[5]	Jumlah Perbandingan (jumlah pertukaran yang terjadi)
Awal	5	8	26	15	11	31	-
Iterasi 1	5	8	15	11	26	31	2
Iterasi 2	5	8	11	15	26	31	1
Iterasi 3	5	8	11	15	26	31	0
Iterasi 4	5	8	11	15	26	31	0
Iterasi 5	5	8	11	15	26	31	0
Iterasi 6	5	8	11	15	26	31	0

Perbandingan Selectionsort

	[0]	[1]	[2]	[3]	[4]	[5]	Jumlah Perbandingan (jumlah pertukaran yang terjadi)
Awal	5	8	26	15	11	31	-
Iterasi 1	5	8	26	15	11	31	0
Iterasi 2	5	8	26	15	11	31	0
Iterasi 3	5	8	11	15	26	31	1
Iterasi 4	5	8	11	15	26	31	0
Iterasi 5	5	8	11	15	26	31	0
Iterasi 6	5	8	11	15	26	31	0

4. Tabel Perbandingan Waktu

Perbandingan Bublesort

- Input (screenshot)

```
package Modul3_Pengurutan;
public class BubbleSortNano {

    public static void main(String[] args) {
        int array[] = new int[1000];
        System.out.print("Random Number     : ");
        for (int i = 0; i < array.length; i++) {
            double random = Math.random() * 100;
            array[i] = (int) random;
            System.out.print(array[i] + " ");
        }
        bubbleSort(array);
        System.out.println("");
        System.out.print("Ascending Number : ");
        cetak(array);
        System.out.println("\nWaktu            : " + System.nanoTime());
    }
    static void bubbleSort(int array[]) {
        int jum;
        int n = array.length;
        for (int i = 0; i < n; i++) {
            for (int j = 1; j < n - 1; j++) {
                if (array[j - 1] > array[j]) {
                    jum = array[j - 1];
                    array[j - 1] = array[j];
                    array[j] = jum;
                }
            }
        }
    }
    public static void cetak(int array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");

        }
    }
}
```

N	Nano Time
1000	422720732263600
10.000	422731631573200
100.000	422763511537400
1.000.000	Too long wait…
10.000.000	Too long wait…

Perbandingan Selectionsort

- Input (screenshot)

```
package Modul3_Pengurutan;
public class SelectionSortNano {
    public static void main(String args[]) {
        int array[] = new int[1000];
        System.out.print("Random Number     : ");
        for (int i = 0; i < array.length; i++) {
            double random = Math.random() * 100;
            array[i] = (int) random;
            System.out.print(array[i] + " ");
        }
        selectionSort(array);
        System.out.println("");
        System.out.print("Ascending Number : ");
        cetak(array);
        System.out.println("\nWaktu            : " + System.nanoTime());
    }
    static void selectionSort(int array[]) {
        int n = array.length;
        for (int i = 0; i < n - 1; i++) {
            int indek = i;
            for (int j = i + 1; j < n; j++) {
                if (array[j] < array[indek]) {
                    indek = j;
                }
            }
            int jum = array[indek];
            array[indek] = array[i];
            array[i] = jum;
        }
    }
    public static void cetak(int array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

N	Nano Time
1000	423043719675000
10.000	423066932097500
100.000	423098386696400
1.000.000	Too long wait…
10.000.000	Too long wait…