LAPORAN HASIL PRAKTIKUM

Algoritma dan Struktur Data

Praktikum 5 Sorting

Alexsa Fitria Ayu Siswoyo.

244107020020

1E

Program Studi Teknik Informatika

Jurusan Teknologi Informasi

Politeknik Negeri Malang

2025

Langkah Praktikum 1 (Mengimplementasikan Sorting Menggunakan Objet)

a. SORTING – BUBBLE SORT

1. Buat folder baru dengan nama Praktikum05.
2. Buat class Sorting < No Presensi >, kemudian tambahkan atribut sebagai berikut:

```
public class Sorting02 {
    int [ ] data ;
    int jumData ;
}
```

3. Buatlah konstruktor dengan parameter Data[] dan jmlData.

```
Sorting02 ( int Data [ ] , int jmlDat )
{
    jumData = jmlDat ;
    data = new int [ jmlDat ] ;
    for ( int i = 0 ; i < jumData ; i
++ ) {
        data [ i ] = Data [ i ] ;
    }
}
```

4. Buatlah method bubbleSort bertipe void dan deklarasikan isinya menggunakan algoritma Bubble Sort.

```
void bubbleSort() {
    int temp = 0;
    for (int i = 0; i < jumData - 1; i++) {
        for (int j = 1; j < jumData - i; j++) {
            if (data[j - 1] > data[j]) {
                temp = data[j];
                data[j] = data[j - 1];
                data[j - 1] = temp;
            }
        }
    }
}
```

5. Buatlah method tampil bertipe void dan deklarasikan isi method tersebut.

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

6. Buat class SortingMain < No Presensi > kemudian deklarasikan array dengan nama a[] kemudian isi array tersebut

```
int a[] = {20, 10, 2, 7, 12};
```

7. Buatlah objek baru dengan nama dataurut1 yang merupakan instansiasi dari class Sorting, kemudian isi parameternya

```
Sorting02 dataurut1 = new Sorting02(a,
a.length);
```

8. Lakukan pemanggilan method `bubbleSort` dan `tampil`

```
System.out.println("Data awal 1");
dataurut1.tampil();
dataurut1.bubbleSort();
System.out.println("Data sudah
diurutkan dengan BUBBLE SORT (ASC)");
dataurut1.tampil();
```

9. Jalankan program, dan amati hasilnya!

```
d478cfbfffe0e7\redhat.java\jdt_ws\Praktikum-ASD_6ab0b5a7\bin' 'praktik
ingMain02'
Data awal 1
20 10 2 7 12
Data sudah diurutkan dengan BUBBLE SORT (ASC)
2 7 10 12 20
PS C:\Users\User\OneDrive\KAMPUS\KELAS\1E\SEM 2\Praktikum-ASD>
```

b. SORTING – SELECTION SORT

1. Pada class Sorting < No Presensi > yang sudah dibuat di praktikum sebelumnya, tambahkan method SelectionSort yang mengimplementasikan pengurutan menggunakan algoritma selection sort.

```
void SelectionSort() {
    for (int i = 0; i < jumData - 1; i++) {
        int min = i;
        for (int j = i + 1; j < jumData; j++) {
            if (data[j] < data[min]) {
                min = j;
            }
        }
        int temp = data[i];
        data[i] = data[min];
        data[min] = temp;
    }
}
```

2. Deklarasikan array dengan nama b[] pada kelas SortingMain < No Presensi > kemudian isi array tersebut

```
int b[] = {30, 20, 2, 8, 14};
```

3. Buatlah objek baru dengan nama dataurut2 yang merupakan instansiasi dari class Sorting, kemudian isi parameternya

```
Sorting02 dataurut2 = new Sorting02(b, b.length);
```

4. Lakukan pemanggilan method SelectionSort dan tampil

```
System.out.println("Data awal 2");
dataurut2.tampil();
dataurut2.SelectionSort();
System.out.println("Data sudah diurutkan dengan SELECTION
SORT (ASC)");
dataurut2.tampil();
```

5. Jalankan program dan amati hasilnya!

```
'

Data awal 1
20 10 2 7 12
Data sudah diurutkan dengan BUBBLE SORT (ASC)
2 7 10 12 20
Data awal 2
30 20 2 8 14
Data sudah diurutkan dengan SELECTION SORT (ASC)
2 8 14 20 30
PS C:\Users\User\OneDrive\KAMPUS\KELAS\1E\SEM 2\Praktikum-ASD>
```

c. SORTING – INSERTION SORT

1. Pada class Sorting < No Presensi > yang sudah dibuat di praktikum sebelumnya tambahkan method insertionSort yang mengimplementasikan pengurutan menggunakan algoritma insertion sort.

```
void insertionSort() {
    for (int i = 1; i <= data.length - 1; i++) {
        int temp = data[i];
        int j = i - 1;
        while (j >= 0 && data[j] > temp) {
            data[j + 1] = data[j];
            j--;
        }
        data[j + 1] = temp;
    }
}
```

2. Deklarasikan array dengan nama c[] pada kelas SortingMain < No Presensi > kemudian isi array tersebut

```
int c[] = {40, 10, 4, 9, 3};
```

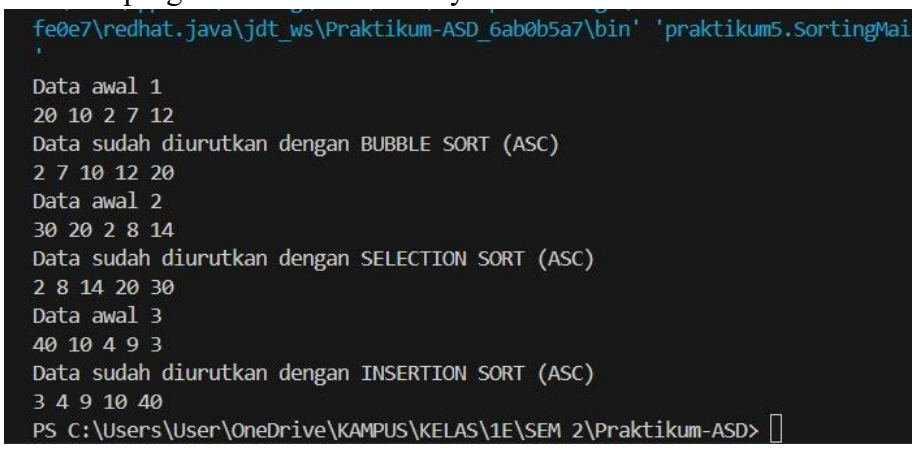
3. Buatlah objek baru dengan nama dataurut3 yang merupakan instansiasi dari class Sorting, kemudian isi parameternya

```
Sorting02 dataurut3 = new Sorting02(c,
c.length);
```

4. Lakukan pemanggilan method `insertionSort` dan `tampil`

```
System.out.println("Data awal 3");
dataurut3.tampil();
dataurut3.insertionSort();
System.out.println("Data sudah
diurutkan dengan INSERTION SORT
(ASC)");
dataurut3.tampil();
```

5. Jalankan program dan amati hasilnya!



PERTANYAAN

1. Jelaskan fungsi kode program berikut:

```
if ( data [ j - 1 ] > data [ j ] ) {
    temp = data [ j ] ;
    data [ j ] = data [ j - 1 ] ;
    data [ j - 1 ] = temp ;
}
```

Jawaban:
Kode program tersebut merupakan *Swapping* (operasi pertukaran) nilai antara dua elemen yang berdekatan pada Array data. Fungsi dari kode program tersebut yaitu:

a. **`if ( data [ j - 1 ] > data [ j ] ) :`** Baris ini memeriksa apakah elemen sebelum indeks j lebih besar dari elemen pada indeks j.
b. **`temp = data [ j ] ;`**: Nilai elemen pada indeks j disimpan sementara dalam variabel temp.
c. **`data [ j ] = data [ j - 1 ] ;`**: Elemen pada indeks j kini diisi dengan nilai elemen sebelumnya (pada indeks j-1).
d. **`data [ j - 1 ] = temp ;`**: Elemen pada indeks j-1 sekarang diisi dengan nilai yang sebelumnya disimpan di temp (nilai asli dari elemen pada indeks j).

2. Tunjukkan kode program yang merupakan algoritma pencarian nilai minimum pada selection sort!
Jawaban:

```
for (int j = i + 1; j < jumData; j++) {
    if (data[j] < data[min]) {
        min = j;
    }
}
```

3. Pada Insertion sort, jelaskan maksud dari kondisi pada perulangan while (j >= 0 && data[j] > temp)
Jawaban:
Kondisi while (j >= 0 && data[j] > temp) dalam perulangan Insertion Sort memiliki dua tujuan utama agar loop terus berjalan: pertama, menjaga agar indeks j tetap berada dalam batas array (tidak kurang dari 0) untuk menghindari kesalahan akses; kedua, membandingkan elemen data[j] dengan nilai temp dan melanjutkan perulangan selama elemen saat ini lebih besar dari nilai yang akan disisipkan.

4. Pada Insertion sort, apakah tujuan dari perintah data[j + 1] = data[j]?
Jawaban:
Tujuan dari data[j + 1] = data[j] pada Insertion Sort adalah untuk mengosongkan posisi di sebelah kanan elemen data[j] dengan cara menggeser data[j] itu sendiri ke kanan. Proses penggeseran ini terjadi berulang kali dalam loop while (j >= 0 && data[j] > temp) selama indeks j tidak negatif dan nilai data[j] lebih besar dari nilai temp yang sedang disisipkan.

Langkah Praktikum 2 (Sorting Menggunakan Array of Object)

Mengurutkan Data Mahasiswa Berdasarkan IPK (Selection Sort)

1. Buatlah class dengan nama **Mahasiswa<No Presensi>**.
2. Untuk lebih jelasnya class tersebut dapat dilihat pada potongan kode di bawah ini:

```
public class Mahasiswa02 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    // Konstruktor default
    public Mahasiswa02() {
    }

    // Konstruktor Berparameter (dibuat agar ada yang nama
var parameter inputnya sama ada yang tidak)
    public Mahasiswa02(String nm, String name, String kls,
double ip) {
        nim = nm;
        nama = name;
        ipk = ip;
        kelas = kls;
    }

    void tampilInfo() {
        System.out.println("Nama  : " + nama);
        System.out.println("NIM   : " + nim);
        System.out.println("Kelas : " + kelas);
        System.out.println("IPK   : " + ipk);
    }
}
```

3. Buat class **MahasiswaBerprestasi<No Presensi>** seperti di bawah ini!

```
public class MahasiswaBerprestasi02 {
    Mahasiswa02[] listMhs = new Mahasiswa02[5];
    int idx;
}
```

4. Tambahkan method **tambah()** di dalam class tersebut! Method **tambah()** digunakan untuk menambahkan objek dari class **Mahasiswa** ke dalam atribut **listMhs**.

```
void tambah(Mahasiswa m) {
    if (idx < listMhs.length) {
        listMhs[idx] = m;
        idx++;
    } else {
        System.out.println("x: data sudah penuh");
    }
}
```

5. Tambahkan method **tampil()** di dalam class tersebut! Method **tampil()** digunakan untuk menampilkan semua data mahasiswa-mahasiswa yang ada di dalam class tersebut! Perhatikan penggunaan sintaks for yang agak berbeda dengan for yang telah dipelajari sebelumnya, meskipun secara konsep sebenarnya mirip.

```
void tampil () {
    for ( Mahasiswa02 m : listMhs ) {
        m.tampilInformasi ( ) ;
        System.out.println ( " ------------------------------
-- " ) ;
    }
}
```

6. Tambahkan method **bubbleSort()** di dalam class tersebut!

```
void bubbleSort() {
    for (int i = 0; i < listMhs.length - 1; i++) {
        for (int j = 1; j < listMhs.length - i; j++) {
            if (listMhs[j].ipk > listMhs[j - 1].ipk) {
                Mahasiswa tmp = listMhs[j];
                listMhs[j] = listMhs[j - 1];
                listMhs[j - 1] = tmp;
            }
        }
    }
}
```

7. Buat class **MahasiswaDemo<No Presensi>**, kemudian buatlah sebuah objek **MahasiswaBerprestasi** dan buatlah 5 objek mahasiswa kemudian tambahkan semua objek mahasiswa tersebut dengan memanggil fungsi **tambah** pada objek [1] **MahasiswaBerprestasi**. Silakan dipanggil fungsi **tampil()** untuk melihat semua data yang telah dimasukkan, urutkan data tersebut dengan memanggil fungsi **bubbleSort()** dan yang terakhir panggil fungsi **tampil()** kembali.

```
public static void main(String[] args) {
    MahasiswaBerprestasi list = new MahasiswaBerprestasi();

    Mahasiswa m1 = new Mahasiswa("123", "Zidan", "2A", 3.2);
    Mahasiswa m2 = new Mahasiswa("124", "Ayu", "2A", 3.5);
    Mahasiswa m3 = new Mahasiswa("125", "Sofi", "2A", 3.1);
    Mahasiswa m4 = new Mahasiswa("126", "Sita", "2A", 3.9);
    Mahasiswa m5 = new Mahasiswa("127", "Miki", "2A", 3.7);

    list.tambah(m1);
    list.tambah(m2);
    list.tambah(m3);
    list.tambah(m4);
    list.tambah(m5);
```

```
    System.out.println("Data mahasiswa sebelum sorting:");
    list.tampil();

    System.out.println("\nData Mahasiswa setelah sorting
berdasarkan IPK (DESC):");
    list.bubbleSort();
    list.tampil();
}
```

```
PS C:\Users\User\OneDrive\KAMPUS\KELAS\1E\SEM 2\Praktikum-ASD> & 'C:\Program Files\Java\j
Users\User\AppData\Roaming\Code\User\workspaceStorage\15abba0384ebaaa242d478cfbfffe0e7\red

Data mahasiswa sebelum sorting :
 Nama : Zidan
 NIM : 123
 Kelas : 2A
 IPK : 3.2
 ----------------------------
 Nama : Ayu
 NIM : 124
 Kelas : 2A
 IPK : 3.5
 ----------------------------
 Nama : Sofi
 NIM : 125
 Kelas : 2A
 IPK : 3.1
 ----------------------------
 Nama : Sita
 NIM : 126
 Kelas : 2A
 IPK : 3.9
 ----------------------------
 Nama : Miki
 NIM : 127
 Kelas : 2A
 IPK : 3.7
 ----------------------------
Data Mahasiswa setelah sorting berdasarkan IPK ( DESC ) :
 Nama : Sita
 NIM : 126
 Kelas : 2A
 IPK : 3.9
 ----------------------------
 Nama : Miki
 NIM : 127
 Kelas : 2A
 IPK : 3.7
 ----------------------------
 Nama : Ayu
 NIM : 124
 Kelas : 2A
 IPK : 3.5
 ----------------------------
 Nama : Zidan
 NIM : 123
 Kelas : 2A
 IPK : 3.2
 ----------------------------
 Nama : Sofi
 NIM : 125
 Kelas : 2A
 IPK : 3.1
 ----------------------------
PS C:\Users\User\OneDrive\KAMPUS\KELAS\1E\SEM 2\Praktikum-ASD>
```

PERTANYAAN

1. Perhatikan perulangan di dalam bubbleSort() di bawah ini:

```
for (int i = 0; i < listMhs.length - 1; i++) {
    for (int j = 1; j < listMhs.length - i; j++) {
        // ... isi perulangan ...
    }
}
```

   a. Mengapa syarat dari perulangan i adalah i < listMhs.length - 1?

      Kondisi i < listMhs.length - 1 pada perulangan luar Bubble Sort memastikan bahwa setiap iterasi menempatkan satu elemen terbesar (atau terkecil) pada posisi akhirnya. Setelah i iterasi, i elemen terakhir (atau pertama) sudah terurut, sehingga perulangan luar hanya perlu berjalan hingga indeks kedua terakhir. Setelah n-1 iterasi (dengan n sebagai panjang array), seluruh daftar pasti terurut.

   b. Mengapa syarat dari perulangan j adalah j < listMhs.length - i?

      Kondisi j < listMhs.length - i pada perulangan dalam Bubble Sort memastikan bahwa setelah i iterasi luar, i elemen terakhir sudah terurut. Akibatnya, perulangan dalam pada iterasi ke-i hanya perlu memproses elemen hingga indeks listMhs.length - i - 1, menghindari perbandingan dengan bagian array yang sudah terurut.

   c. Jika banyak data di dalam listMhs adalah 50, maka berapa kali perulangan i akan berlangsung? Dan ada berapa Tahap bubble sort yang ditempuh?

      Ketika listMhs berisi 50 elemen, perulangan luar dalam algoritma Bubble Sort akan dieksekusi 49 kali, dengan nilai i bergerak dari 0 sampai 48. Setiap eksekusi perulangan luar ini merupakan satu tahap (*pass*) dalam proses Bubble Sort, sehingga totalnya ada 49 tahap yang diperlukan untuk mengurutkan data.

2. Modifikasi program di atas di mana data mahasiswa bersifat dinamis (input dari keyboard) yang terdiri dari nim, nama, kelas, dan ipk!

```
    public MahasiswaBerprestasi02(int kapasitas) {
        listMhs = new Mahasiswa02[kapasitas];
        idx = 0;
    }
```

```
package praktikum5;

import java.util.Scanner;

public class MahasiswaDemo02 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah mahasiswa: ");
        int jumlahMahasiswa = input.nextInt();
        input.nextLine(); // Consume newline

        MahasiswaBerprestasi02 list = new
MahasiswaBerprestasi02(jumlahMahasiswa);

        for (int i = 0; i < jumlahMahasiswa; i++) {
            System.out.println("\nMasukkan data mahasiswa
ke-" + (i + 1) + ":");
            System.out.print("NIM   : ");
            String nim = input.nextLine();
            System.out.print("Nama  : ");
            String nama = input.nextLine();
            System.out.print("Kelas : ");
            String kelas = input.nextLine();
            System.out.print("IPK   : ");
            double ipk = input.nextDouble();
            input.nextLine(); // Consume newline

            Mahasiswa02 m = new Mahasiswa02(nim, nama,
kelas, ipk);
            list.tambah(m);
        }

        System.out.println("\nData mahasiswa sebelum
sorting:");
        list.tampil();

        System.out.println("\nData Mahasiswa setelah
sorting berdasarkan IPK (DESC):");
        list.bubbleSort();
        list.tampil();

        input.close();
    }
}
```

Mengurutkan Data Mahasiswa Berdasarkan IPK Menggunaka Insertion Sort

1. Lihat kembali class `MahasiswaBerprestasi`, dan tambahkan method `selectionSort()` di dalamnya! Method ini juga akan melakukan proses sorting secara ascending, tetapi menggunakan pendekatan selection sort.

```
void selectionSort ( ) {
    for ( int i = 0 ; i < listMhs.length - 1 ; i ++ ) {
        int idxMin = i ;
        for ( int j = i + 1 ; j < listMhs.length ; j ++ ) {
            if ( listMhs [ j ] .ipk < listMhs [ idxMin ]
.ipk ) {
                idxMin = j ;
            }
        }
        Mahasiswa tmp = listMhs [ idxMin ] ;
        listMhs [ idxMin ] = listMhs [ i ] ;
        listMhs [ i ] = tmp ;
    }
}
```

2. Setelah itu, buka kembali class MahasiswaDemo, dan di dalam method main() tambahkan baris program untuk memanggil method selectionSort() tersebut, kemudian panggil method tampil() untuk menampilkan data yang sudah diurutkan!

```
System.out.println ( " Data yang sudah terurut menggunakan SELECTION SORT ( ASC ) " )
;
list.selectionSort ( ) ;
list.tampil ( ) ;
```

3. Coba jalankan kembali class MahasiswaDemo, dan amati hasilnya! Apakah kini data mahasiswa telah tampil urut menaik berdasar ipk?

```
Data yang sudah terurut menggunakan SELECTION SORT ( ASC )
Nama : Sofi
NIM : 125
Kelas : 2A
IPK : 3.1
------------------------------
Nama : Zidan
NIM : 123
Kelas : 2A
IPK : 3.2
------------------------------
Nama : Ayu
NIM : 124
Kelas : 2A
IPK : 3.5
------------------------------
Nama : Miki
NIM : 127
Kelas : 2A
IPK : 3.7
------------------------------
Nama : Sita
NIM : 126
Kelas : 2A
IPK : 3.9
------------------------------
Data yang sudah terurut menggunakan INSERTION SORT ( ASC )
Nama : Sofi
NIM : 125
Kelas : 2A
IPK : 3.1
------------------------------
Nama : Zidan
NIM : 123
Kelas : 2A
IPK : 3.2
------------------------------
Nama : Ayu
NIM : 124
Kelas : 2A
IPK : 3.5
------------------------------
Nama : Miki
NIM : 127
Kelas : 2A
IPK : 3.7
------------------------------
Nama : Sita
NIM : 126
Kelas : 2A
IPK : 3.9
------------------------------
PS C:\Users\User\OneDrive\KAMPUS\KELAS\1E\SEM 2\Praktikum-ASD>
```

PERTANYAAN

1. Ubahlah fungsi pada InsertionSort sehingga fungsi ini dapat melaksanakan proses sorting dengan cara descending.

```
    void insertionSort ( ) {
        for ( int i = 1 ; i < listMhs.length ; i ++ ) {
            Mahasiswa02 temp = listMhs [ i ] ;
            int j = i ;
            while ( j > 0 && listMhs [ j - 1 ] .ipk <
temp.ipk ) {
                listMhs [ j ] = listMhs [ j - 1 ] ;
                j-- ;
            }
            listMhs [ j ] = temp ;
        }
    }
```

Perubahan yang dilakukan

- Pada baris while (j > 0 && listMhs[j - 1].ipk < temp.ipk), tanda **>``** (lebih besar dari) telah diubah menjadi **<`** (lebih kecil dari).

Perubahan pada kondisi while menyebabkan loop terus berjalan jika IPK elemen di sebelah kiri lebih kecil dari IPK elemen yang sedang diproses. Proses ini secara efektif menggeser elemen-elemen ber-IPK rendah ke kanan, membuka ruang bagi elemen ber-IPK tinggi untuk ditempatkan di bagian awal, yang menghasilkan urutan descending berdasarkan IPK.