JOBSHEET VI SORTING (BUBBLE, SELECTION, DAN INSERTION SORT)

Ka Abi Muhammad Robit Fuad/12/TI-1B/244107020163

6.5 Tujuan Praktikum

Setelah melakukan praktikum ini diharapkan mahasiswa mampu:

a. Mahasiswa mampu membuat algoritma sorting menggunakan bubble sort, selection sort dan insertion sort

b. Mahasiswa mampu menerapkan algoritma sorting menggunakan bubble sort, selection sort dan insertion sort pada program

6.6 Praktikum

1 - Mengimplementasikan Sorting menggunakan object

Waktu : 60 menit

6.2.1 Langkah Praktikum

1 a. SORTING – BUBBLE SORT

1. Buat folder baru bernama Jobsheet6 di dalam repository Praktikum ASD

2. Buat class Sorting, kemudian tambahkan atribut sebagai berikut:

```
int[]data;
int jumData;
```

3. Buatlah konstruktor dengan parameter Data[] dan jmlDat

```
Sorting12 (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-1; 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

```
public static void main(String[] args) {
    int a[]={20,10,2,7,12};
    Sorting12 dataurut1 = new Sorting12(a, a.length);
    System.out.println(x:"Data awal 1");
    dataurut1.tampil();
    dataurut1.bubbleSort();
    System.out.println(x:"Data sudah diurutkan dengan BUBBLE SORT (ASC)");
    dataurut1.tampil();
}
```

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

```
Sorting12 dataurut1 = new Sorting12(a, a.length);
```

8. Lakukan pemanggilan method bubbleSort dan tampil

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

9. Jalankan program, dan amati hasilnya!

6.2.2 Verifikasi Hasil Percobaan

```
Data awal 1
20 10 2 7 12
Data sudah diurutkan dengan BUBBLE SORT (ASC)
2 7 10 20 12
PS C:\Users\kaabi\OneDrive\Desktop\Praktikum ASD\Jobsheet 6>
```

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

```
Sorting12 dataurut2 = new Sorting12(b, b.length);
```

4. Lakukan pemanggilan method SelectionSort dan tampil

```
dataurut2.tampil();
dataurut2.SelectionSort();
System.out.println(x:"Data s
dataurut2.tampil();
```

5. Jalankan program dan amati hasilnya!

6.2.3 Verifikasi Hasil Percobaan

```
Data awal 2
30 20 2 8 14
Data sudah diurutkan dengan SELECTION SORT (ASC)
2 8 14 20 30
PS C:\Users\kaabi\OneDrive\Desktop\Praktikum ASD\Jobshe
```

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

```
Sorting12 dataurut3 = new Sorting12(c, c.length);
```

4. Lakukan pemanggilan method insertionSort dan tampil

```
dataurut3.tampil();
dataurut3.insertionSort();
System.out.println(x:"Data
dataurut3.tampil();
```

5. Jalankan program dan amati hasilnya!

6.2.4 Verifikasi Hasil Percobaan

```
Data awal 3
40 10 4 9 3
Data sudah diurutkan dengan INSERTION SORT (ASC)
3 4 9 10 40
PS C:\Users\kaabi\OneDrive\Desktop\Praktikum ASD\Jobs
```

6.2.5 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;
}
```

- if (data[j-1] > data[j]) {
    - Kode ini memeriksa apakah elemen sebelum elemen saat ini (**data[j-1]**) lebih besar dari elemen saat ini (**data[j]**).
    - Jika kondisi ini benar, berarti kedua elemen perlu ditukar untuk mengurutkannya.

  temp = data[j];

  Variabel **temp** digunakan untuk menyimpan nilai **data[j]**. Ini dilakukan s ebelum **data[j]** akan ditimpa dengan nilai **data[j-1]**.

  data[j] = data[j-1];

  data[j-1] = temp;

    - Elemen **data[j]** diubah menjadi **data[j-1]**, yang melakukan penggantian nilai.
    - Kemudian, **data[j-1]** diubah menjadi nilai yang disimpan dalam **temp** — yaitu nilai awal dari **data[j]**.
    - Dengan cara ini, kedua elemen telah ditukar.

Kode ini bertujuan untuk membandingkan dan mengganti posisi dua elemen dalam array **data**, sehingga dapat membantu dalam pengurutan data.

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

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

- Loop j = i+1 → Memeriksa elemen setelah indeks i untuk mencari nilai terkecil.
- if (data[j] < data[min]) → Jika ditemukan elemen yang lebih kecil dari data[min], maka indeks min diperbarui ke j.
- Setelah loop selesai, min akan berisi indeks elemen terkecil dalam sisa array yang belum diurutkan.

3. Pada Insertion sort , jelaskan maksud dari kondisi pada perulangan

```
while (j >= 0 && data[j] > temp) {
   data[j + 1] = data[j];
   j--;
}
```

Maksud dari kondisi while (j >= 0 && data[j] > temp)

1. j >= 0 → Memastikan bahwa indeks j tidak keluar dari batas array saat membandingkan elemen sebelumnya.
2. data[j] > temp → Mengecek apakah elemen sebelumnya (data[j]) lebih besar dari elemen yang sedang diurutkan (temp). Jika iya, berarti elemen sebelumnya harus digeser ke kanan untuk memberi ruang bagi temp.

4. Pada Insertion sort, apakah tujuan dari perintah

data[j + 1] = data[j];
→ Menggeser elemen ke kanan jika lebih besar dari temp, agar ada ruang untuk menyisipkan temp pada posisi yang benar.

6.7 Praktikum 2- (Sorting Menggunakan Array of Object)

Waktu : 45 menit

6.3.2 Langkah-langkah Praktikum 2

1. Buatlah class dengan nama Mahasiswa<No Presensi>.

2. Untuk lebih jelasnya class tersebut dapat dilihat pada potongan kode di bawah ini

```
public class Mahasiswa12 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    Mahasiswa12(){

    }

    Mahasiswa12(String nm, String name, String kls, double ip){
        nim = nm;
        nama = name;
        ipk = ip;
        kelas = kls;
    }
    void tampilInformasi(){
        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 MahasiswaBerprestasi12 {
    Mahasiswa12[]listMhs = new Mahasiswa12 [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 (Mahasiswa12 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 (Mahasiswa12 m:listMhs){
        m.tampilInformasi();
        System.out.println(x:"--------------------");
    }
}
```

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) {
                Mahasiswa12 tmp = listMhs[j];
                listMhs[j]=listMhs[j-1];
                listMhs[j-1]=tmp;
            }
        }
    }
```

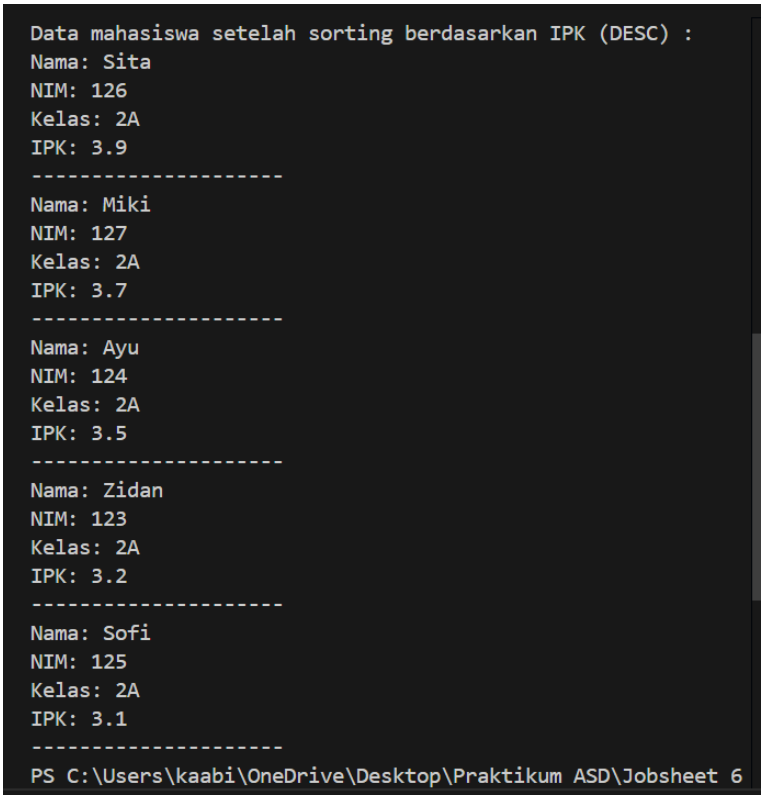
7. Buat class MahasiswaDemo<No Presensi>, kemudian buatlah sebuah objek

MahasiswaBerprestasi dan buatlah 5 objek mahasiswa kemudian tambahkan semua objek mahasiswa tersebut dengan memanggil fungsi tambah pada objek MahasiswaBerprestasi. Silakan dipanggil fungsi tampil() untuk melihat semua data yang telah dimasukan, urutkan data tersebut dengan memanggil fungsi bubbleSort() dan yang terakhir panggil fungsi tampil kembali.

```java
public static void main(String[] args) {
    MahasiswaBerprestasi12 list = new MahasiswaBerprestasi12();
    Mahasiswa12 m1 = new Mahasiswa12(nm:"123", name:"Zidan", kls:"2A", ip:3.2);
    Mahasiswa12 m2 = new Mahasiswa12(nm:"124", name:"Ayu", kls:"2A", ip:3.5);
    Mahasiswa12 m3 = new Mahasiswa12(nm:"125", name:"Sofi", kls:"2A", ip:3.1);
    Mahasiswa12 m4 = new Mahasiswa12(nm:"126", name:"Sita", kls:"2A", ip:3.9);
    Mahasiswa12 m5 = new Mahasiswa12(nm:"127", name:"Miki", kls:"2A", ip:3.7);

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

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

    System.out.println(x:"Data mahasiswa setelah sorting berdasarkan IPK (DESC) : ");
    list.bubbleSort();
    list.tampil();
}
```

6.3.3 Verifikasi Hasil Percobaan

```
obsheet 6_88ccf3f5\bin' 'MahasiswaDemo12'
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
----------------------
```



6.3.4 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++) {
```

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

Perulangan i mengontrol **jumlah iterasi utama (pass)** dalam **Bubble Sort**.

- i < listMhs.length - 1 digunakan karena setelah (n - 1) pass, seluruh elemen sudah terurut.
- Jika jumlah elemen dalam listMhs adalah n, maka kita hanya butuh (n - 1) iterasi karena di iterasi terakhir, elemen terkecil otomatis berada di tempat yang benar.

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

Perulangan j berfungsi untuk membandingkan dan menukar elemen dalam satu pass.

- Batas listMhs.length - i digunakan agar tidak membandingkan elemen yang sudah terurut di akhir array.

- Setiap pass memastikan satu elemen terbesar sudah ada di posisi benar, sehingga kita mengurangi jumlah iterasi yang diperlukan.

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

Berapa kali perulangan i berlangsung?

- Karena perulangan i berjalan hingga i < listMhs.length - 1, maka:

50−1=49 kali. Jadi, perulangan i berlangsung 49 kali.

Berapa tahap (pass) Bubble Sort yang ditempuh?

- Bubble Sort bekerja dalam pass, dan jumlah pass sama dengan jumlah iterasi i, yaitu 49 tahap (pass).

2. Modifikasi program diatas dimana data mahasiswa bersifat dinamis (input dari keyborad) yang terdiri dari nim, nama, kelas, dan ipk!

```
import java.util.Scanner;
public class MahasiswaDemo12 {
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print(s:"Masukkan jumlah mahasiswa: ");
        int jumlah = input.nextInt();
        input.nextLine();

        MahasiswaBerprestasi12 list = new MahasiswaBerprestasi12(jumlah);

        for (int i = 0; i < jumlah; i++) {
            System.out.println("Masukkan data mahasiswa ke-" + (i + 1) + ":");
            System.out.print(s:"NIM: ");
            String nim = input.nextLine();
            System.out.print(s:"Nama: ");
            String nama = input.nextLine();
            System.out.print(s:"Kelas: ");
            String kelas = input.nextLine();
```

6.4 Mengurutkan Data Mahasiswa Berdasarkan IPK (Selection Sort)

Waktu : 30 menit

Jika pada praktikum yang sebelumnya kita telah mengurutkan data mahasiwa berdasarkan IPK menggunakan Bubble Sort secara descending, pada kali ini kita akan mencoba untuk menambahkan fungsi pengurutan menggunakan Selection Sort.

6.4.1. Langkah-langkah Percobaan.

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;
        }
    }
    Mahasiswa12 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(x:"\nData mahas
list.tampil();

System.out.println(x:"Data yang su
list.selectionSort();
list.tampil();
```

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

6.4.2 Verifikasi Hasil Percobaan

Pastikan output yang ditampilkan sudah benar seperti di bawah ini

```
Data mahasiswa sebelum sorting:
Nama: Ali
NIM: 123
Kelas: 2B
IPK: 3.9
--------------------
Nama: ila
NIM: 124
Kelas: 2B
IPK: 3.1
--------------------
Nama: agus
NIM: 125
Kelas: 2b
IPK: 3.6
--------------------
Nama: tika
NIM: 126
Kelas: 2b
IPK: 3.3
--------------------
Nama: udin
NIM: 127
Kelas: 2b
IPK: 3.2
--------------------
```

```
Data yang sudah terurut menggunakan SELECTION SORT (ASC):
Nama: ila
NIM: 124
Kelas: 2B
IPK: 3.1
----------------------
Nama: udin
NIM: 127
Kelas: 2b
IPK: 3.2
----------------------
Nama: tika
NIM: 126
Kelas: 2b
IPK: 3.3
----------------------
Nama: agus
NIM: 125
Kelas: 2b
IPK: 3.6
----------------------
Nama: Ali
NIM: 123
Kelas: 2B
IPK: 3.9
----------------------
PS C:\Users\kaabi\OneDrive\Desktop\Praktikum ASD\Jobsheet 6>
```

6.4.3 Pertanyaan

Di dalam method selection sort, terdapat baris program seperti di bawah ini:

```
int idxMin=i;
    for (int j = i+1; j < listMhs.length; j++) {
      if (listMhs[j].ipk<listMhs[idxMin].ipk) {
        idxMin=j;
      }
    }
```

Untuk apakah proses tersebut, jelaskan!

- Menemukan indeks elemen dengan nilai IPK terkecil dalam bagian array yang belum diurutkan.
- Setelah indeks minimum ditemukan (idxMin), elemen pada indeks i dan idxMin akan ditukar agar nilai terkecil berada di depan.

6.5 Mengurutkan Data Mahasiswa Berdasarkan IPK Menggunakan Insertion Sort

Waktu : 30 menit

Yang terakhir akan diimplementasikan Teknik sorting menggunakan Insertion Sort, dengan mengurutkan IPK mahasiswa secara ascending.

6.5.1 Langkah-langkah Percobaan

1. Lihat kembali class MahasiswaBerprestasi, dan tambahkan method insertionSort() di dalamnya. Method ini juga akan melakukan proses sorting secara ascending, tetapi menggunakan pendekatan Insertion Sort.

```
void insertionSort() {
    for (int i = 1; i < listMhs.length; i++) {
        Mahasiswa12 temp = listMhs[i];
        int j = i;
        while (j > 0 && listMhs[j - 1].ipk > temp.ipk) {
            listMhs[j] = listMhs[j - 1];
            j--;
        }
        listMhs[j] = temp;
    }
```

2. Setelah itu, buka kembali class MahasiswaDemo, dan di dalam method main() tambahkan baris program untuk memanggil method insertionSort() dan tampil () tersebut!

```
System.out.println(x:"\nDa
list.tampil();

System.out.println(x:"Data
list.insertionSort();
list.tampil();
```

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

6.5.2 Verifikasi Hasil Percobaan

```
Data mahasiswa sebelum sorting:
Nama: ayu
NIM: 111
Kelas: 2c
IPK: 3.7
----------------------
Nama: dika
NIM: 222
Kelas: 2c
IPK: 3.0
----------------------
Nama: ila
NIM: 333
Kelas: 2c
IPK: 3.8
----------------------
Nama: susi
NIM: 444
Kelas: 2c
IPK: 3.1
----------------------
Nama: yayuk
NIM: 555
Kelas: 2c
IPK: 3.4
----------------------
Data yang sudah terurut menggunakan INSERTION SORT (ASC):
Nama: dika
NIM: 222
Kelas: 2c
IPK: 3.0
----------------------
Nama: susi
NIM: 444
Kelas: 2c
IPK: 3.1
----------------------
Nama: yayuk
NIM: 555
Kelas: 2c
IPK: 3.4
----------------------
Nama: ayu
NIM: 111
Kelas: 2c
IPK: 3.7
----------------------
Nama: ila
NIM: 333
Kelas: 2c
IPK: 3.8
----------------------
PS C:\Users\kaabi\OneDrive\Desktop\Praktikum ASD\Jobsheet 6>
```

6.5.3 Pertanyaan

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

```
void insertionSortDescending() {
    for (int i = 1; i < listMhs.length; i++) {
        Mahasiswa12 temp = listMhs[i];
        int j = i;
        while (j > 0 && listMhs[j - 1].ipk < temp.ipk) {
            listMhs[j] = listMhs[j - 1];
            j--;
        }
        listMhs[j] = temp;
    }
}
```

cukup ubah kondisi pada listmhs[j-1].ipk (yang awalnya > menjadi <)

6.6 Latihan Praktikum

Waktu : 45 Menit

Berdasarkan class diagram diatas buatlah menu dikelas main dengan pilihan menu:

1. Tambah data digunakan untuk menambahkan data dosen

2. Tampil data digunakan untuk menampilkan data seluruh dosen

3. Sorting ASC digunakan untuk mengurutkan data dosen berdasarkan usia dimulai dari dosen termuda ke dosen tertua menggunakan bublle Sort.

4. Sorting DSC digunakan untuk mengurutkan data dosen berdasarkan usia dimulai dari tertua ke dosen termuda dapat menggunakan algoritma selection sort atau insertion sort.

(jawaban langsung di kode program)