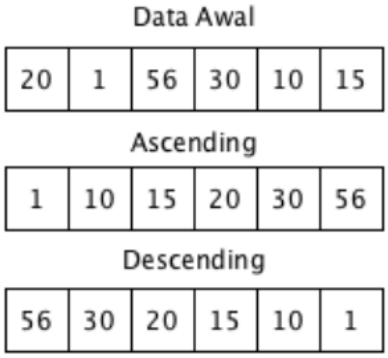
# JOBSHEET - 5
# SORTING

## 1. KOMPETENSI

1. Mahasiswa mampu memahami algoritma *sorting*.
2. Mahasiswa mampu membuat dan mendeklarasikan struktur algoritma *sorting*.
3. Mahasiswa mampu menerapkan dan mengimplementasikan algoritma *sorting*.

## 2. ULASAN TEORI

Sorting adalah suatu proses (operasi) mengurutkan data dalam suatu urutan yang dikehendaki. Terdapat dua jenis pengurutan, pengurutan data secara naik dikenal dengan istilah *ascending* dan pengurutan data secara menurun dikenal dengan istilah *descending.* Ilustrasi proses pengurutan dapat dilihat pada Gambar di bawah ini:




Gambar 5.1 Ilustrasi Proses Sorting

Dalam proses sorting (dalam modul ini hanya dibahas 4 metode), dikenal beberapa metode yang digunakan, antara lain:

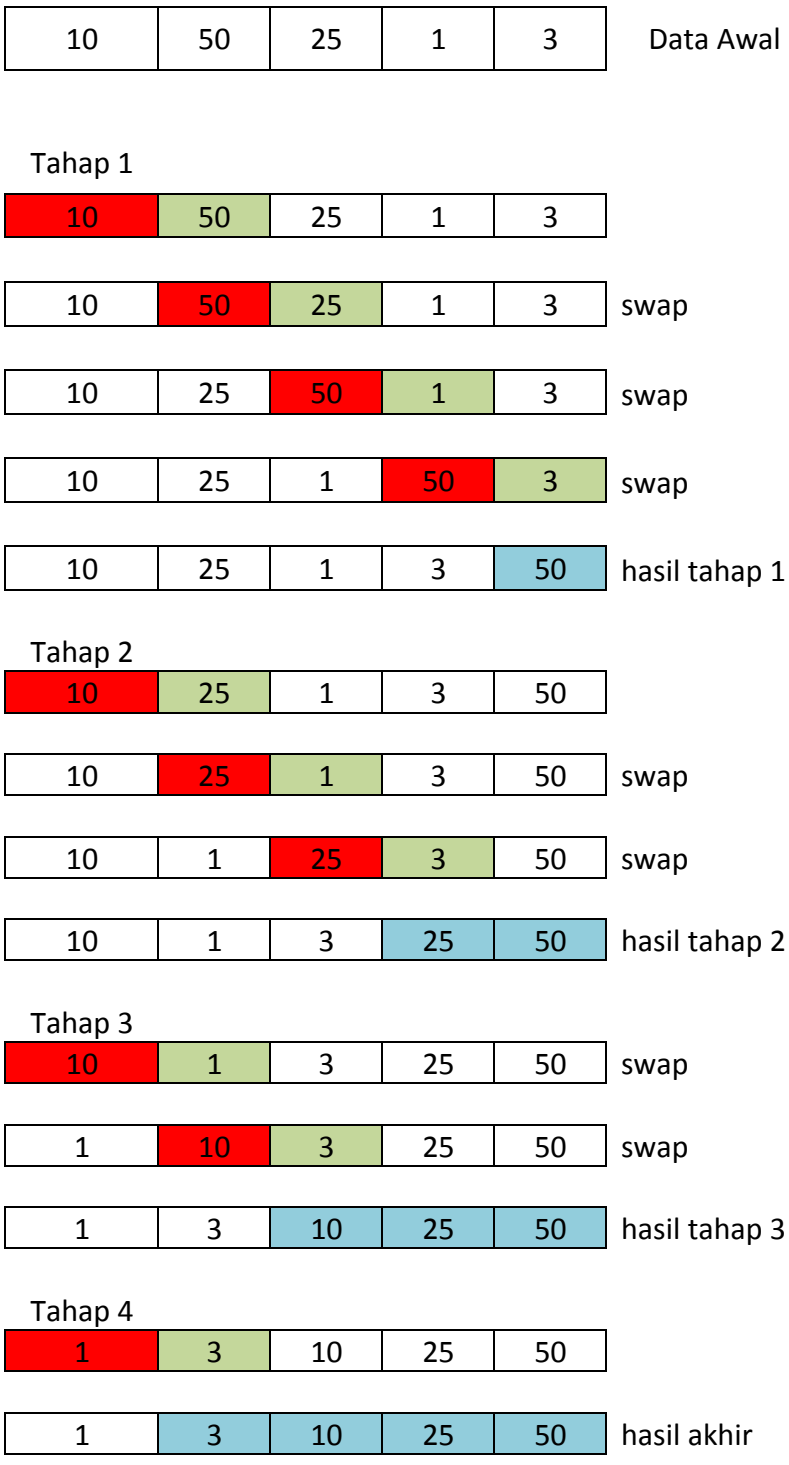
1. *Bubble Sort*
2. *Selection Sort*
3. *Insertion Sort*
4. *Merge Sort*

### Bubble Sort

*Bubble Sort* adalah salah satu metode *sorting* yang paling mudah diimplementasikan. Tetapi dengan metode ini, proses *sorting* kurang efektif jika dibandingkan dengan metode lain. Metode ini dilakukan dengan cara membandingkan data mulai dari data pertama dengan data-data setelahnya. Setiap kali pembandingan akan diikuti dengan proses penukaran (*swap*) jika nilai yang dibandingkan sesuai dengan model pengurutan (*ascending* atau *descending*).

Proses pembandingan serta penukaran akan diulang untuk data kedua (dibandingkan dengan data ketiga hingga sampai data terakhir. Begitu juga yang terjadi dengan data ketiga dibandingkan dengan data berikutnya. Proses ini akan berakhir sampai tidak ada data yang dibandingkan lagi.

Ilustrasi proses pengurutan dengan metode *Bubble Sort,* diilustrasikan pada Gambar 5.2.




**Gambar 5.2 Ilustrasi *bubble sort***

**Selection Sort**

Pengurutan dengan metode *Selection Sort* mengkombinasikan antara proses *sorting* dan *searching.* Metode ini memperbaiki pengurutan dengan metode *Bubble Sort* dengan cara mengurangi jumlah proses penukaran.

Metode ini akan mencari nilai terkecil dari deretan data terlebih dahulu untuk kemudian dilakukan proses penukaran (*swap*). Proses ini akan dilakukan sampai data terakhir. Ilustrasi proses pengurutan dapat dilihat pada Gambar 5.7.

| 10 | 50 | 25 | 1 | 3 |
|---|---|---|---|---|

| Proses | Swap | A[0] | A[1] | A[2] | A[3] | A[4] |
|---|---|---|---|---|---|---|
| | Data Awal | 10 | 50 | 25 | 1 | 3 |
| 1 | m=A[0], k=1 | 10 | 50 | 25 | 1 | 3 |
| 2 | m=A[1], k=3 | 1 | 30 | 25 | 10 | 3 |
| 3 | m=A[2], k=10 | 1 | 3 | 25 | 10 | 50 |
| 4 | m=A[3], k=50 | 1 | 3 | 10 | 25 | 50 |

**Gambar 5.7 Ilustrasi *selection sort***

**Insertion Sort**

Metode *Insertion Sort* dilakukan dengan cara menyisipkan (*insert*) suatu data pada posisi yang seharusnya. Langkah-langkah dalam proses ini, dijabarkan sebagai berikut:

1. Ambil satu data ke-*i*, simpan nilai ke dalam *temp* (*i* dimulai dari 2) .
2. Bandingkan nilai dari data *temp* dengan data yang ada di sebelah kiri satu per-satu.
3. Cek apakah data *temp* lebih kecil dari data di sebelah kiri.
4. Jika langkah nomor 3 bernilai benar, lakukan pergeseran data satu per-satu kemudian pada posisi yang tepat sisipkan data *temp*.
5. Ulangi langkah 1 sampai dengan 4, sehingga nilai *i* sama dengan data terakhir.

| Data Awal | 10 | 50 | 25 | 1 | 3 |
|---|---|---|---|---|---|

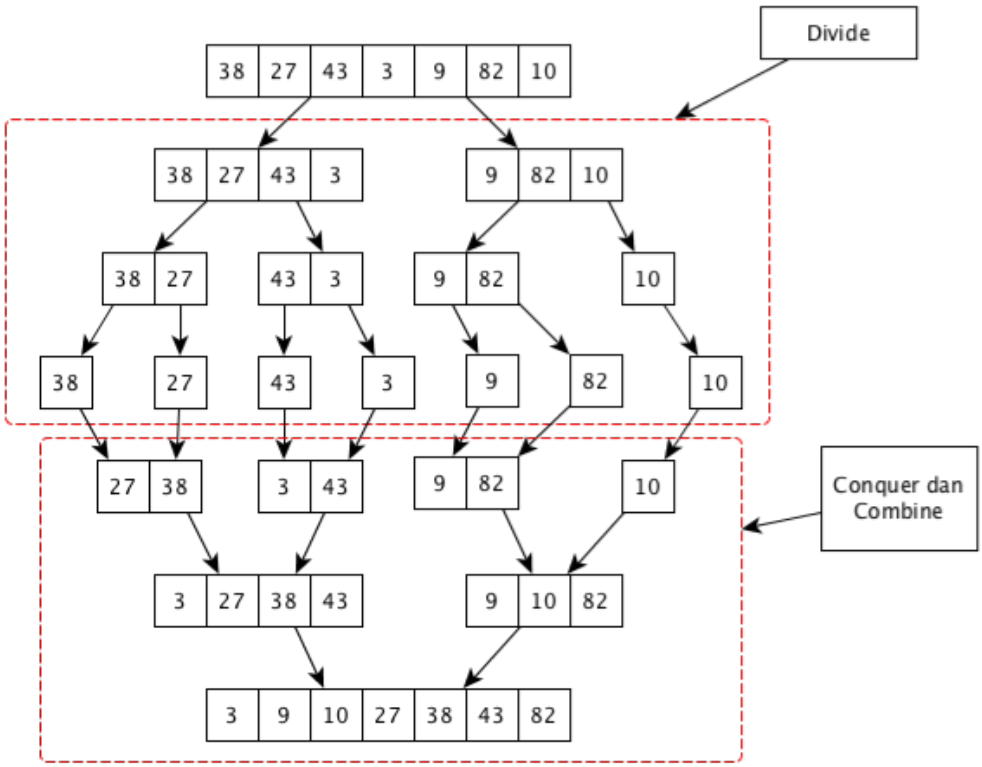
| Proses | Tampung | A[0] | A[1] | A[2] | A[3] | A[4] |
|---|---|---|---|---|---|---|
| | Data Awal | 10 | 50 | 25 | 1 | 3 |
| 1 | temp = A[1] | 10 | 50 | 25 | 1 | 3 |
| 2 | temp = A[2] | 10 | 25 | 50 | 1 | 3 |
| 3 | temp = A[3] | 1 | 10 | 25 | 50 | 3 |
| 4 | temp = A[4] | 1 | 3 | 10 | 25 | 50 |

**Gambar 5.8 Ilustrasi *insertion sort***

**Merge Sort**

Pengurutan dengan metode ini sering juga disebut dengan metode *Divide and Conquer*. Metode ini terdiri dari 3 tahapan. Divide membagi permasalahan atau koleksi data ke dalam bagian-bagian yang lebih kecil.

Conquer mengurutkan dari bagian yang paling kecil. Dan tahapan yang terakhir yaitu Combine, mengkombinasikan atau menggabungkan solusi dari bagian yang paling kecil sehingga menjadi solusi utama. Ilustrasi proses pengurutan yang terjadi dijabarkan pada Gambar 5.9.




**Gambar 5.9 Ilustrasi *merge sort***

## 1. LANGKAH PRAKTIKUM

Pada praktikum di jobsheet ini, kita akan membuat aplikasi untuk mengurutkan data mahasiswa berdasarkan ipk nya. Pada Praktikum 1 dibuat class Mahasiswa, selanjutnya pada Praktikum 2 akan dibuat class DaftarMahasiswaBerprestasi dimana di dalam class tersebut ada data mahasiswa-mahasiswa yang berprestasi. Pada class DaftarMahasiswaBerprestasi tersebut juga terdapat method untuk mengurutkan data mahasiswa berdasarkan ipk secara descending. Pada Praktikum 3, akan dibuat class Main.

### PRAKTIKUM 1 – Membuat Class Mahasiswa

1. Perhatikan diagram class Mahasiswa di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program class Mahasiswa.

| Mahasiswa |
| --- |
| nama: String<br>thnMasuk: int |

| umur: int<br>ipk: double |
|---|
| Mahasiswa(n: String, t: int, u: int, i: double)<br>tampil(): void |

2. Buat class Mahasiswa seperti di bawah ini!

```
1   package minggu5;
2
3   public class Mahasiswa {
4       String nama;
5       int thnMasuk, umur;
6       double ipk;
7
8       Mahasiswa(String n, int t, int u, double i){
9           nama = n;
10          thnMasuk = t;
11          umur = u;
12          ipk = i;
13      }
14
15      void tampil(){
16          System.out.println("Nama = "+nama);
17          System.out.println("Tahun Masuk = "+thnMasuk);
18          System.out.println("Umur = "+umur);
19          System.out.println("IPK = "+ipk);
20      }
21  }
```

## PRAKTIKUM 2 – Membuat Class DaftarMahasiswaBerprestasi

1. Di dalam class DaftarMahasiswaBerprestasi terdapat daftar mahasiswa-mahasiswa yang dinyatakan sebagai mahasiswa berprestasi. Di dalam class ini juga terdapat method yang digunakan untuk mengurutkan data mahasiswa berdasarkan nilai ipk. Selain itu, juga terdapat method untuk menampilkan semua data mahasiswa-mahasiswa yang berprestasi tersebut. Selain itu juga terdapat method untuk menambahkan data mahasiswa ke dalam daftar mahasiswa berprestasi. Perhatikan diagram class berikut ini!

| DaftarMahasiswaBerprestasi |
|---|
| listMhs: Mahasiswa[5]<br>idx: int |
| tambah(mhs: Mahasiswa): void<br>tampil(): void<br>bubbleSort(): void |

2. Buat class DaftarMahasiswaBerprestasi seperti di bawah ini!

```
1   package minggu5;
2
3   public class DafraMahasiswaBerprestasi {
4       Mahasiswa listMhs[] = new Mahasiswa[5];
5       int idx;
6
7       //setelah ini tuliskan method tambah()
8
9       //setelah ini tuliskan method tampil()
10
11      //setelah ini tuliskan method bubbleSort()
12  }
```

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

```
7       //setelah ini tuliskan method tambah()
8       void tambah(Mahasiswa m){
9           if(idx<listMhs.length){
10              listMhs[idx] = m;
11              idx++;
12          }else{
13              System.out.println("Data sudah penuh!!");
14          }
15      }
```

4. 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, meskipn secara konsep sebenarnya mirip.

```
17      //setelah ini tuliskan method tampil()
18      void tampil(){
19          for(Mahasiswa m : listMhs){
20              m.tampil();
21              System.out.println("------------------------");
22          }
23      }
```

5. Tambahkan method bubbleSort() di dalam class tersebut!

```
25      //setelah ini tuliskan method bubbleSort()
26      void bubbleSort(){
27          for(int i=0; i<listMhs.length-1; i++){
28              for(int j=1; j<listMhs.length-i; j++){
29                  if(listMhs[j].ipk > listMhs[j-1].ipk){
30                      //di bawah ini proses swap atau penukaran
31                      Mahasiswa tmp = listMhs[j];
32                      listMhs[j] = listMhs[j-1];
33                      listMhs[j-1] = tmp;
34                  }
35              }
36          }
37      }
```

6. Sampai tahap ini class DaftarMahasiswaBerprestasi telah lengkap.

## PRAKTIKUM 3 – Membuat Class Main

1. Buat class Main dan didalamnya buat method main() seperti di bawah ini!

```
1  package minggu5;
2  import java.util.Scanner;
3
4  public class Main {
5      public static void main(String[] args) {
6
7      }
8  }
```

2. **Di dalam method main()**, buatlah 2 objek dari class Scanner, sebuah objek DaftarMahasiswaBerprestasi dan deklarasikan variabel jumlahMhs senilai 5.!

```
6          Scanner s = new Scanner(System.in);
7          Scanner s1 = new Scanner(System.in);
8          DaftarMahasiswaBerprestasi data = new DaftarMahasiswaBerprestasi();
9          int jumMhs = 5;
```

3. Kemudian lakukan perulangan sebanyak 5 kali menggunakan for, untuk memasukan data nama, umur, tahun masuk dan ipk dari tiap mahasiswa, dan menambahkan objek mahasiswa ke dalam objek daftar mahasiswa berprestasi!

```
11         for(int i=0;i<jumMhs;i++){
12             System.out.print("Nama = ");
13             String nama =  s1.nextLine();
14             System.out.print("Thn masuk = ");
15             int thn =  s.nextInt();
16             System.out.print("Umur = ");
17             int umur =  s.nextInt();
18             System.out.print("IPK = ");
19             double ipk =  s.nextDouble();
20
21             Mahasiswa m = new Mahasiswa(nama,thn,umur,ipk);
22             data.tambah(m);
23         }
```

4. Coba tampilkan data-data mahsiswa yang telah masuk dalam daftar mahasiswa berprestasi!

```
25             System.out.println("Data mahasiswa sebelum sorting = ");
26             data.tampil();
```

Coba jalankan dan amati hasilnya. Apakah semua data mahasiswa telah tampil?

5. Panggil method bubbleSort() dan tampilkan kembali hasilnya!

```
27         System.out.println("Data mahasiswa setelah sorting desc berdasar ipk= ");
28         data.bubbleSort();
29         data.tampil();
```

Coba jalankan dan amati kembali hasilnya! Apakah semua data kini telah terurut secaar menurun berdasar ipk?

**PRAKTIKUM 4 – Menambahkan Proses Selection Sort di dalam Class DaftarMahasiswaBerprestasi**

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

```
39      //setelah ini tuliskan method selectionSort()
40      void selectionSort(){
41          for(int i=0; i<listMhs.length-1; i++){
42              int idxMin = i;
43              for(int j=i+1; j<listMhs.length; j++){
44                  if(listMhs[j].ipk < listMhs[idxMin].ipk){
45                      idxMin = j;
46                  }
47              }
48              //swap
49              Mahasiswa tmp = listMhs[idxMin];
50              listMhs[idxMin] = listMhs[i];
51              listMhs[i] = tmp;
52          }
53      }
```

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

```
30              System.out.println("Data mahasiswa setelah sorting asc berdasar ipk= ");
31              data.selectionSort();
32              data.tampil();
```

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

**4. PERTANYAAN**

1) Terdapat di method apakah proses bubble sort?
2) Terdapat di method apakah proses selection sort?
3) Apakah yang dimaksud proses swap? Tuliskan potongan program untuk melakukan proses swap tersebut!
4) Di dalam method bubbleSort(), terdapat baris program seperti di bawah ini:

```
29                  if(listMhs[j].ipk > listMhs[j-1].ipk){
30                      //di bawah ini proses swap atau penukaran
31                      Mahasiswa tmp = listMhs[j];
32                      listMhs[j] = listMhs[j-1];
33                      listMhs[j-1] = tmp;
34                  }
35              }
```

Untuk apakah proses tersebut?

5) Perhatikan perulangan di dalam bubbleSort() di bawah ini:

```
27              for(int i=0; i<listMhs.length-1; i++){
28                  for(int j=1; j<listMhs.length-i; j++){
```

a. Apakah perbedaan antara kegunaan perulangan i dan perulangan j?
b. Mengapa syarat dari perulangan i adalah i<listMhs.length-1 ?
c. Mengapa syarat dari perulangan j adalah j<listMhs.length-i ?

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

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

```
42              int idxMin = i;
43              for(int j=i+1; j<listMhs.length; j++){
44                  if(listMhs[j].ipk < listMhs[idxMin].ipk){
45                      idxMin = j;
46                  }
47              }
```

Untuk apakah proses tersebut?

## 5. TUGAS

1. Tambah method **insertionSort()** di dalam class **DaftarMahasiswaBerprestasi**, untuk mengurutkan secara ascending berdasar ipk! Gunakan pseudocode di bawah ini sebagai bantuan!

```
Jika terdapat array data dengan panjang array
n, index terkecil 0, index terbesar n-1
for i = 1 to n-1
      key = data[i]
      j = i-1
      while j>=0 and A[j]>key do
            A[j+1] = A[j]
            j--
      endwhile
      A[j+1] = key
endfor
```

2. Dalam suatu Keranjang Belanja, terdapat data Barang seperti di bawah ini:

| Nama Barang | Jumlah Barang | Harga Satuan |
|---|---|---|
| Kertas A4 | 3 | 2000 |
| Pensil | 2 | 100 |
| Penghapus | 1 | 500 |
| Spidol | 3 | 200 |

Dengan mengacu pada praktikum yang telah dilakukan:

a. Buatlah class **Barang**, class **KeranjangBelanja** dan class **Main**.
b. Di dalam class **KeranjangBelanja**, buat method pengurutan **ascending** berdasar **harga satuan**, menggunakan **bubble sort**
c. Di dalam class KeranjangBelanja, buat method pengurutan **ascending** berdasar **harga total per barang**, menggunakan **selection sort**

| Barang |
|---|
| nama: String<br>jumlah: int<br>hargaSatuan: int |
| Barang(n: String, j: int, hs: int) |

| |
|---|
| tampil(): void<br>hitungHargaTotal(): int |

| KeranjangBelanja |
|---|
| listBarang: Barang[4]<br>idx: int |
| tambah(brg: Barang): void<br>tampil(): void<br>bubbleSort(): void<br>selectionSort(): void |

3. Tambah method **mergeSort()** di dalam class **KeranjangBelanja**, yang akan melakukan proses pengurutan **ascending** (menaik) berdasarkan **jumlah barang**!