LAPORAN HASIL PRAKTIKUM
ALGORITMA STRUKTUR DATA
JOBSHEET 6

NAMA : JIRO AMMAR WAFI

NIM : 244107020190

KELAS : 1-E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLINEMA

2025

Praktikum 1

Mengimplementasikan Sorting menggunakan object

A. Sorting - Bubble Sort

1. Membuat class Sorting berdasarkan absen, dan membuat konstruktor berparameter:

```
package Jobsheet6;

public class Sorting14 {
    int[] data;
    int jumData;


    public Sorting14 (int Data[], int jmlDat) {
        jumData = jmlDat;
        data = new int[jmlDat];
        for (int i=0; i<jumData; i++){
            data[i] = Data[i];
        }
    }
}
```

2. Method bubbleSort() dengan kembalian void, disini adalah tahap sorting secara ascending yang menyimpan nilai yang lebih kecil ke variabel sementara, index yang memiliki nilai yang lebih besar ditukar ke index yang memiliki nilai kecil, dan index kecil tersebut menyimpan nilai dari variabel sementara tadi:

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

3. Method tampil dengan kembalian void, fungsi method ini untuk menampilkan seluruh data dalam array menggunakan loop for:

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

4. class SortingMain, mendeklarasi serta menginisialisasi sampel array, dan membuat objek baru menggunakan kons. berparameter yang berisi array dan panjang array dari class tersebut:

```
public class SortingMain14 {
    Run | Debug
    public static void main(String[] args) {
        int a[] = {20, 10, 2, 7, 12};
        Sorting14 dataUrut1 = new Sorting14(a, a.length);
    }
}
```

5. Memanggil method tampil dan bubbleSort, proses ini menampilkan terlebih dahulu isi array yang belum di susun, kemudian melakukan sorting secara ascending menggunakan bubble sorting. Hasil akhirnya akan menampilkan isi array yang telah disusun:

```
System.out.println(x:"Data Awal 1");
dataUrut1.tampil(); dataUrut1.bubbleSort();
System.out.println(x:"Data sudah diurutkan dengan BU
dataUrut1.tampil();
```

6. Verifikasi Kode = error pada pemilihan IF:

```
Data Awal 1
20 10 2 7 12
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index -1 out of bou
nds for length 5
        at Jobsheet6.Sorting14.bubbleSort(Sorting14.java:20)
        at Jobsheet6.SortingMain14.main(SortingMain14.java:9)
```

7. Menemukan error, salah inisialisasi nilai loop:

```
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;
    }
}
```

8. Verifikasi Kode = sukses:

```
Data Awal 1
20 10 2 7 12
Data sudah diurutkan dengan BUBBLE SORT (ASCENDING)
2 7 10 12 20
```

B. Sorting - Selection Sort

1. Method Selection Sort, mencari nilai minimum dengan nested loop yang menggunakan nilai variabel sebagai indeksnya, dan membandingkan antar dua variabel untuk menemukan nilai paling minimum diantara isi array, bertukar posisi index, setelah iterasi pertama, index array yang telah memiliki nilai paling minimum tidak akan di cek kembali, begitu seterusnya:

```
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. Deklarasi array (b), membuat objek baru (dataUrut2), dan memanggil method:

```
int b[] = {30, 20, 2, 8, 14};
Sorting14 dataUrut2 = new Sorting14(b, b.length);
System.out.println(x:"Data Awal 2");
dataUrut2.tampil(); dataUrut2.SelectionSort();
System.out.println(x:"Data sudah diurutkan dengan SELE
dataUrut2.tampil();
```

3. Verifikasi kode = sukses:

```
===========
Data Awal 2
30 20 2 8 14
Data sudah diurutkan dengan SELECTION SORT (ASCENDING)
2 8 14 20 30
```

C. Sorting - Insertion Sort

1. Insertion Sort, method ini memindahkan satu elemen ke posisi yang benar ke dalam bagian array yang telah terurut, mencari nilai yang lebih besar dari nilai array yang telah terurut sebelumnya, dan menukar indeksnya:

```
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. Deklarasi array (c), membuat objek baru (dataUrut3), dan memanggil method:

```
int c[] = {40, 10, 4, 9, 3};
Sorting14 dataUrut3 = new Sorting14(c, c.length);
System.out.println(x:"Data Awal 3");
dataUrut3.tampil(); dataUrut3.SelectionSort();
System.out.println(x:"Data sudah diurutkan dengan INSE
dataUrut3.tampil();
```

3. Verifikasi Kode = sukses:

```
===========
Data Awal 3
40 10 4 9 3
Data sudah diurutkan dengan INSERTION SORT (ASCENDING)
3 4 9 10 40
```

PERTANYAAN P1

1. Pada proses tersebut, nilai pada array data dengan indeks j-1, dibandingkan dengan nilai pada indeks ke j, jika hasilnya true, maka nilai yang menjadi pembandingnya (terkecil) akan disimpan ke dalam variabel sementara, kemudian nilai yang terbesar akan bertukar indeks ke indeks yang memiliki nilai terkecil sebelumnya. Setelah itu, indeks yang bertukar posisi ke sebelah kiri akan menyimpan nilai dari vaiabel sementara.

2. Dalam kode ini, dilakukan pencarian nilai minimum yang akan menukar posisi indeks apabila ditemukan hasilnya true.

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

3. Pada kondisi tersebut, dilakukan pengecekan kondisi apakah iterasi bisa dijalankan untuk pertama kali hingga update selanjutnya yang akan terus mengurangi iterasi dan membandingkan nilai pada indeks saat ini dengan nilai yang merupakan nilai terurut (sorted) apakah lebih besar atau tidak.

4. tujuan dari perintah tersebut adalah perintah selanjutnya apabila kondisi dari iterasi terpenuhi, yang akan menukar nilai array berdasarkan posisi indeks, nilai terbesar akan dipindahkan ke sisi kanan, sedangkan nilai terkecil akan dipindahkan ke sisi kiri.

Praktikum 2

Sorting menggunakan Array Of Object

1. Membuat class, konstruktor default dan parameter, serta method didalamnya:

```java
public class Mahasiswa14 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    Mahasiswa14 () {}

    Mahasiswa14 (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);
    }
}
```

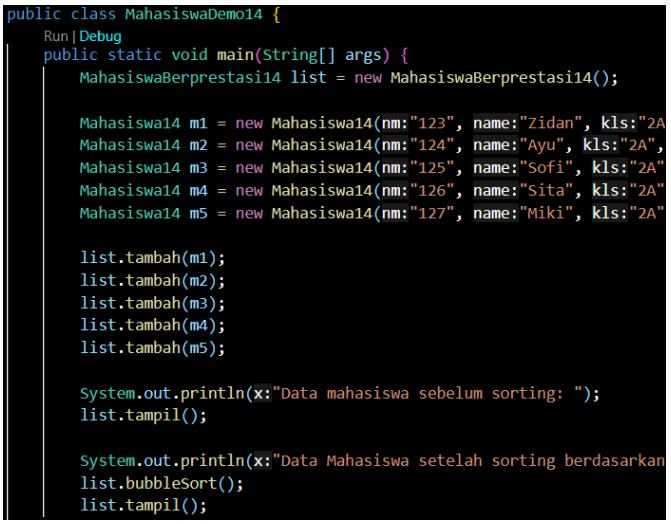
2. Membuat class MahasiswaBerprestasi, deklarasi array dan membuat method tambah() untuk menambahkan objek:

```java
public class MahasiswaBerprestasi14 {
    Mahasiswa14 [] listMhs = new Mahasiswa14[5];
    int idx;

    void tambah(Mahasiswa14 m){
        if (idx<listMhs.length){
            listMhs[idx]=m;
            idx++;
        } else {
            System.out.println(x:"Data sudah penuh");
        }
    }
}
```

3. Tambahan method tampil() untuk menampilkan informasi yang disimpan dalam array of object, dan bubbleSort() untuk sorting berdasarkan nilai IPK.

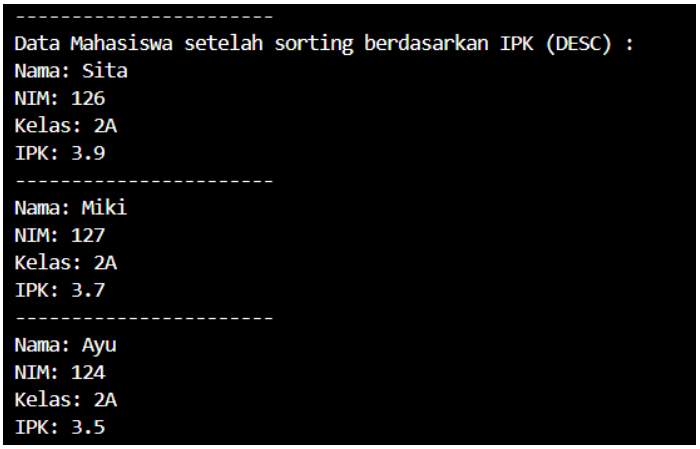
```
void tampil () {
    for (Mahasiswa14 m : listMhs) {
        m.tampilInformasi();
        System.out.println(x:"----------------------");
    }
}

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){
                Mahasiswa14 tmp = listMhs[j];
                listMhs[j] = listMhs[j-1];
                listMhs[j-1] = tmp;
            }
        }
    }
}
```

4. Membuat 5 objek di class baru yaitu MahasiswaDemo14 dan memanggil berbagai method dari class Mahasiswa14:



```
public class MahasiswaDemo14 {
    Run | Debug
    public static void main(String[] args) {
        MahasiswaBerprestasi14 list = new MahasiswaBerprestasi14();

        Mahasiswa14 m1 = new Mahasiswa14(nm:"123", name:"Zidan", kls:"2A
        Mahasiswa14 m2 = new Mahasiswa14(nm:"124", name:"Ayu", kls:"2A",
        Mahasiswa14 m3 = new Mahasiswa14(nm:"125", name:"Sofi", kls:"2A"
        Mahasiswa14 m4 = new Mahasiswa14(nm:"126", name:"Sita", kls:"2A"
        Mahasiswa14 m5 = new Mahasiswa14(nm:"127", name:"Miki", kls:"2A"

        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
        list.bubbleSort();
        list.tampil();
```

5. Verifikasi Kode=sukses:



```
----------------------
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
```

PERTANYAAN P2

1. Sub pertanyaan:
   A. Iterasi pertama diperlukan untuk memastikan bahwa seluruh elemen dalam array telah urut, dan setelah mencapai iterasi listMhs.length-1, seluruh elemen sudah berada di posisi yang tepat.
   B. Iterasi kedua ialah untuk proses perbandingan dan menukar antar elemen, dan mengurangi bagian yang perlu diperiksa (tidak perlu mengecek dari awal lagi).
   C. Perulangan i akan berjalan sebanyak 49 kali (50-1), dan bubble sort berjalan sebanyak 49 kali juga.

2. Modifikasi program :
   A. jumlah data yang dapat diterima bersifat statis, menyimpan nilai input menggunakan array 1 dimensi;

```
Scanner in = new Scanner (System.in);
MahasiswaBerprestasi14 list = new MahasiswaBerprestasi14();
byte bykData = 5;
String[] nim = new String[bykData];
String[] nama = new String[bykData];
String[] kelas = new String[bykData];
double[] ipk = new double[bykData];
```

   B. Looping untuk meminta input sebanyak nilai yang telah ditentukan;

```
System.out.println(x:">>> START INPUT");
for (int i = 0; i < bykData; i++) {
    System.out.println("\nMasukkan data mahasiswa ke-" + (i + 1) + ":");

    System.out.print(s:"NIM: ");
    nim[i] = in.nextLine();

    System.out.print(s:"Nama: ");
    nama[i]= in.nextLine();

    System.out.print(s:"Kelas: ");
    kelas[i]= in.nextLine();

    System.out.print(s:"IPK: ");
    ipk[i]= in.nextDouble();
    in.nextLine();
}
```

   C. Membuat objek menggunakan parameter dari input:

```
System.out.println(x:">>> END INPUT");
Mahasiswa14 m1 = new Mahasiswa14(nim[0], nama[0], kelas[0], ipk[0]); list.tambah(m1);
Mahasiswa14 m2 = new Mahasiswa14(nim[1], nama[1], kelas[1], ipk[1]); list.tambah(m2);
Mahasiswa14 m3 = new Mahasiswa14(nim[2], nama[2], kelas[2], ipk[2]); list.tambah(m3);
Mahasiswa14 m4 = new Mahasiswa14(nim[3], nama[3], kelas[3], ipk[3]); list.tambah(m4);
Mahasiswa14 m5 = new Mahasiswa14(nim[4], nama[4], kelas[4], ipk[4]); list.tambah(m5);
```

PERCOBAAN 3

Mengurutkan Data Mahasiswa Berdasarkan IPK (Selection Sort)

1. Menambahkan method SelectionSort() yang akan mengurutkan secara ascending:

```
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;
            }
        }
        Mahasiswa14 tmp = listMhs[idxMin];
        listMhs[idxMin]=listMhs[i];
        listMhs[i] = tmp;
    }
}
```

2. Memanggil method untuk sorting dan menampilkan data yang telah terurut:

```
System.out.println(x:"\n>>> Sorting Ascending dengan SELECTION SORT");
list.SelectionSort();
list.tampil();
```

3. Verifikasi Kode = sukses:

```
>>> Sorting Ascending dengan SELECTION SORT
Nama: A
NIM: 1
Kelas: A
IPK: 1.0
----------------------
Nama: R
NIM: R
Kelas: R
IPK: 1.5
----------------------
Nama: B
NIM: 2
Kelas: B
IPK: 2.0
----------------------
Nama: T
NIM: 7
Kelas: T
IPK: 3.9
----------------------
Nama: C
NIM: 4
Kelas: C
IPK: 4.0
----------------------
```

PERTANYAAN P3

1. `int idxMin=i`

   Pada proses ini, dipilih indeks ke i dari outer loop untuk dipilih sebagai angka yang mewakili nilai minimum;

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

   Inner loop memilih indeks perbandingan (bukan indeks yang telah dipilih sebelumnya sebagai nilai minimum), indeks tersebut digunakan untuk dibandingkan nilainya dengan indeks minimum. Jika true, indeks minimum akan digantikan dengan indeks perbandingan. Perbandingan ini akan dilanjutkan terus menerus sebanyak panjang array tersebut.

PERCOBAAN 4

Mengurutkan Data Mahasiswa Berdasarkan IPK Menggunakan Insertion Sort

1. Menambahkan method InsertionSort():

```
void InsertionSort() {
    for (int i=1; i<listMhs.length; i++) {
        Mahasiswa14 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. Baris perintah untuk cetak data:

```
System.out.println(x:"\n>>> Sorting Ascending dengan INSERTION SORT");
list.InsertionSort();
list.tampil();
```

3. Verifikasi Kode = sukses:

```
>>> Sorting Ascending dengan INSERTION SORT
Nama: KEEMPAT
NIM: 4
Kelas: D
IPK: 1.1
----------------------
Nama: KEDUA
NIM: 2
Kelas: B
IPK: 1.7
----------------------
Nama: PERTAMA
NIM:  1
Kelas: A
IPK: 2.9
----------------------
Nama: KETIGA
NIM: 3
Kelas: C
IPK: 3.2
----------------------
Nama: KELIMA
NIM: 5
Kelas: E
IPK: 3.4
----------------------
```

PERTANYAAN P4

1. Mengubah pembandingnya menjadi lebih kecil (<)

```
void InsertionSort() {
    for (int i=1; i<listMhs.length; i++) {
        Mahasiswa14 temp = listMhs[i];
        int j = i;
            while (j>0 && listMhs[j-1].ipk < temp.ipk) {
                listMhs[j]=listMhs[j-1];
                j--;
            }
        listMhs[j]=temp;
    }
}
```

Hasil:

```
>>> Sorting Descending dengan INSERTION SORT
Nama: B
NIM: B
Kelas: B
IPK: 10.0
----------------------
Nama: E
NIM: E
Kelas: E
IPK: 8.0
----------------------
Nama: D
NIM: D
Kelas: D
IPK: 6.0
----------------------
Nama: A
NIM: A
Kelas: A
IPK: 3.0
----------------------
Nama: C
NIM: C
Kelas: C
IPK: 2.0
----------------------
```

PRAKTIKUM AKHIR

Hasil Compiling Kode:

1. Menu 1 - Menambahkan data dengan input dinamis:

```
>>> Pilihan Menu
1. Tambah Data
2. Tampil Data
3. Sorting Ascending - Bubble Sort
4. Sorting Descending - Selection Sort
5. END
Pilih: 1

>>> Tambah Data
Masukkan data dosen ke-1
Kode: AA
Nama: AA
Jenis kelamin (L/P): L
Usia: 21
----------------------
Masukkan data dosen ke-2
Kode: BB
Nama: BB
Jenis kelamin (L/P): P
Usia: 24
```

2. Menu 2 - Menampilkan data yang telah di input:

```
>>> Pilihan Menu
1. Tambah Data
2. Tampil Data
3. Sorting Ascending - Bubble Sort
4. Sorting Descending - Selection Sort
5. END
Pilih: 2

>>> Tampilkan Data
Nama Dosen: AA
Kode: AA
Jenis Kelamin: Laki-laki
Usia: 21
----------------------
Nama Dosen: BB
Kode: BB
Jenis Kelamin: Perempuan
Usia: 24
----------------------
```

3. Menu 3 - Sorting Ascending dengan Bubble Sort:

```
>>> Pilihan Menu
1. Tambah Data
2. Tampil Data
3. Sorting Ascending - Bubble Sort
4. Sorting Descending - Selection Sort
5. END
Pilih: 3

>>> Sorting Ascending - Bubble Sort
Nama Dosen: AA
Kode: AA
Jenis Kelamin: Laki-laki
Usia: 21
-----------------------
Nama Dosen: BB
Kode: BB
Jenis Kelamin: Perempuan
Usia: 24
-----------------------
```

4. Menu 4 - Sorting Descending menggunakan Selection Sort:

```
>>> Pilihan Menu
1. Tambah Data
2. Tampil Data
3. Sorting Ascending - Bubble Sort
4. Sorting Descending - Selection Sort
5. END
Pilih: 4

>>> Sorting Descending - Selection Sort
Nama Dosen: BB
Kode: BB
Jenis Kelamin: Perempuan
Usia: 24
-----------------------
Nama Dosen: AA
Kode: AA
Jenis Kelamin: Laki-laki
Usia: 21
-----------------------
```

5. Menu 5 - Mengakhiri program secara langsung:

```
>>> Pilihan Menu
1. Tambah Data
2. Tampil Data
3. Sorting Ascending - Bubble Sort
4. Sorting Descending - Selection Sort
5. END
Pilih: 5
```