Nama : Siti Mutmainah

Kelas : TI-1B

Absen : 21

Mata Kuliah : ALSD

Praktikum 1 - Mengimplementasikan Sorting menggunakan object

Langkah-langkah:

a) SORTING – BUBBLE SORT

1. Buat class Sorting21.java, kemudian tambah atribut

```
public class Sorting21 {
    int[] data;
    int jumData;
```

2. Buat konstruktor dengan parameter Data[] dan jmlData

```
    Sorting21(int[] Data, int jmlData) {
        jumData=jmlData;
        data=new int[jmlData];
        for(int i=0; i<jumData; i++){
            data[i]=Data[i];
        }
    }
```

3. 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;
            }
        }
```

4. 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();
    }
```

5. Buat class SortingMain21 kemudian deklarasikan array dengan nama a[] kemudian isi array tersebut

```
public class SotingMain21 {
    Run | Debug
    public static void main(String[] args) {
        int[] a = {20, 10, 2, 7, 12};
```

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

```
Sorting21 dataurut1 = new Sorting21(a, a.length);
```

7. Lakukan pemanggilan method bubbleSort dan tampil

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

8. Hasil percobaan

```
PS D:\ASD\Praktikum-ASD\Jobsheet6> cd "d:\ASD\P
gMain21.java } ; if ($?) { java SotingMain21 }
Data awal 1
20 10 2 7 12
Data sudah diurutkan dengan BUBBLE SORT (ASC)
2 7 10 12 20
```

b) SORTING – SELECTION SORT

1. Pada class Sorting21 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 SortingMain21 kemudian isi array tersebut

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

3. Buatlah objek baru dengan nama dataurut2 yang merupakan instansiasi dari class Sorting21, kemudian isi parameternya

```
Sorting21 dataurut2 = new Sorting21(b, b.length);
```

4. Lakukan pemanggilan method SelectionSort dan tampil

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

5. Hasil percobaan

```
Data awal 2
30 20 2 8 14
Data sudah diurutkan dengan SELECTION SORT (ASC)
2 8 14 20 30
```

c) SORTING – INSERTION SORT

1. Pada class Sorting 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 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

```
Sorting21 dataurut3 = new Sorting21(c, c.length);
```

4. Lakukan pemanggilan method insertionSort dan tampil

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

5. Hasil percobaan

```
Data awal 3
40 10 4 9 3
Data sudah diurutkan dengan INSERTION SORT (ASC)
3 4 9 10 40
```

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;
}
```

   Jawab: Kode ini adalah bagian dari algoritma sorting yang melakukan pertukaran (swap) antara dua elemen dalam array. Jika elemen sebelumnya (data[j-1]) lebih besar dari elemen saat ini (data[j]), maka keduanya akan ditukar posisinya. Tujuannya adalah agar elemen yang lebih kecil berpindah ke posisi lebih awal (kiri), sehingga membantu dalam proses pengurutan data.

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

   Jawab:

```
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)
```

   Jawab: Kondisi ini memastikan bahwa elemen yang lebih besar dari temp akan digeser ke kanan agar temp bisa dimasukkan ke posisi yang tepat. (j >= 0) Mengecek apakah indeks j masih dalam batas array (tidak negatif). (data[j] > temp) Jika elemen saat ini lebih besar dari temp, maka elemen ini harus digeser ke kanan untuk memberi ruang bagi temp.

4. Pada Insertion sort, apakah tujuan dari perintah

```
data[j+1]= data[j];
```

   Jawab: Perintah ini berfungsi untuk menggeser elemen ke kanan agar ada tempat bagi elemen yang akan disisipkan (temp).

Praktikum 2 - Sorting Menggunakan Array of Object

Langkah-langkah:

Mengurutkan Data Mahasiswa Berdasarkan IPK (Bubble Sort)

1. Buatlah class dengan nama Mahasiswa21. Lengkapi atribut dan tambahkan konstruktor berparameter serta method tampilInformasi

```java
public class Mahasiswa21 {
        String nim;
        String nama;
        String kelas;
        double ipk;

        Mahasiswa21() {}

        Mahasiswa21(String nm, String name, String kls, double ip) {
            nim = nm;
            nama = name;
            kelas = kls;
            ipk = ip;
        }

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

2. Buat class MahasiswaBerprestasi21, buat array

```java
public class MahasiswaBerprestasi21 {
        Mahasiswa21[] listMhs = new Mahasiswa21[5];
        int idx;
```

3. Tambahkan method tambah() di dalam class tersebut! Method tambah() digunakan untuk menambahkan objek dari class Mahasiswa21 ke dalam atribut listMhs

```java
void tambah(Mahasiswa21 m) {
    if (idx < listMhs.length) {
        listMhs[idx] = m;
        idx++;
    } else {
        System.out.println(x:"Data sudah penuh");
    }
}
```

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, meskipun secara konsep sebenarnya mirip

```java
void tampil() {
    for (Mahasiswa21 m : listMhs) {
        m.tampilInformasi();
        System.out.println(x:"------------------------------");
    }
}
```

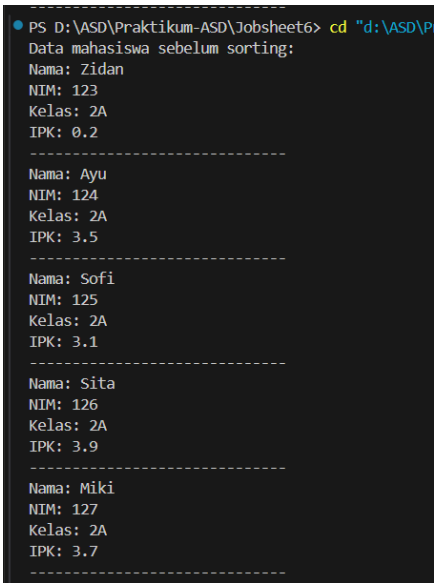
5. Tambahkan method bubbleSort() di dalam class tersebut

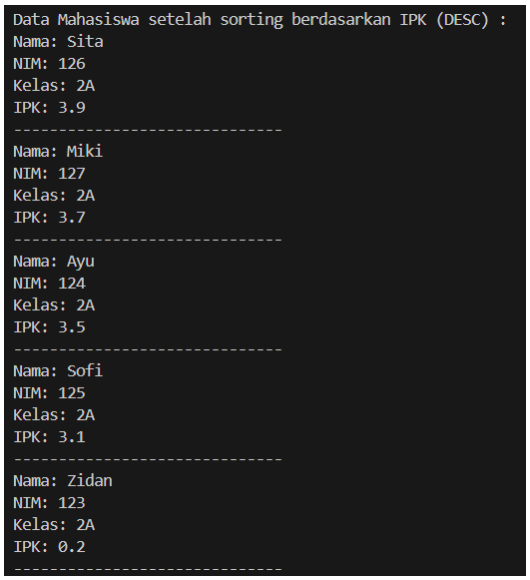
```
    void bubbleSort() {
        for (int i = 0; i < listMhs.length - 1; i++) {
            for (int j = 1; j < listMhs.length - i; j++) {
                if (listMhs[j].ipk > listMhs[j - 1].ipk) {
                    Mahasiswa21 tmp = listMhs[j];
                    listMhs[j] = listMhs[j - 1];
                    listMhs[j - 1] = tmp;
                }
            }
        }
    }
}
```

6. Buat class MahasiswaDemo21, 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

```
public class MahasiswaDemo21 {
    Run | Debug
    public static void main(String[] args) {
        MahasiswaBerprestasi21 list = new MahasiswaBerprestasi21();

        Mahasiswa21 m1 = new Mahasiswa21(nm:"123",name:"Zidan",kls:"2A",ip:.2);
        Mahasiswa21 m2 = new Mahasiswa21(nm:"124",name:"Ayu",kls:"2A",ip:3.5);
        Mahasiswa21 m3 = new Mahasiswa21(nm:"125", name:"Sofi",kls:"2A",ip:3.1);
        Mahasiswa21 m4 = new Mahasiswa21(nm:"126", name:"Sita",kls:"2A",ip:3.9);
        Mahasiswa21 m5 = new Mahasiswa21(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();
    }

}
```

7. Hasil percobaan





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 adalah i<listMhs.length-1 ?
   Jawab: Karena Bubble Sort membandingkan elemen berpasangan dan menggeser elemen terbesar ke akhir setiap iterasi. Setelah n-1 iterasi, semua elemen sudah berada di posisi yang benar. Jadi, tidak perlu melakukan iterasi terakhir karena elemen terakhir pasti sudah berada di tempatnya
   b) Mengapa syarat dari perulangan j adalah j adalah j<listMhs.length-1 ?
   Jawab: Karena setiap kali selesai satu iterasi luar (i), elemen terbesar akan berada di akhir array. Jadi, bagian yang sudah terurut tidak perlu dicek lagi, sehingga jumlah perbandingan berkurang setiap iterasi
   c) Jika banyak data di dalam listMhs adalah 50, maka berapakali perulangan i akan berlangsung? Dan ada berapa Tahap bubble sort yang ditempuh?
   Jawab: Perulangan I sebanyak 49 kali (50-1), tahap bubble sort n(n-1)/2 = 50(49)/2 hasilnya 1225
2. Modifikasi program diatas dimana data mahasiswa bersifat dinamis (input dari keyborad) yang terdiri dari nim, nama, kelas, dan ipk!
   Jawab:

```
public class MahasiswaBerprestasi21 {
        Mahasiswa21[] listMhs = new Mahasiswa21[5];
        int idx = 0;
```

```java
import java.util.Scanner;
public class MahasiswaDemo21 {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        MahasiswaBerprestasi21 list = new MahasiswaBerprestasi21();

        for (int i = 0; i < 5; i++) {
            System.out.println("Masukkan data mahasiswa ke-" + (i + 1));
            System.out.print(s:"NIM: ");
            String nim = sc.nextLine();
            System.out.print(s:"Nama: ");
            String nama = sc.nextLine();
            System.out.print(s:"Kelas: ");
            String kelas = sc.nextLine();
            System.out.print(s:"IPK: ");
            double ipk = sc.nextDouble();
            sc.nextLine();

            Mahasiswa21 m = new Mahasiswa21(nim, nama, kelas, ipk);
            list.tambah(m);
        }

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

        Accept | Reject | Show Diff
        System.out.println(x:"Data Mahasiswa setelah sorting berdasarkan IPK (DESC):");
        list.bubbleSort();
        list.tampil();

    }
}
```

Langkah-langkah:

Mengurutkan Data Mahasiswa Berdasarkan IPK (Selection Sort)

1. Lihat kembali class MahasiswaBerprestasi21, dan tambahkan method selectionSort() di dalamnya! Method ini juga akan melakukan proses sorting secara ascending, tetapi menggunakan pendekatan selection sort

```java
    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;
                }
            }
            Mahasiswa21 tmp = listMhs[idxMin];
            listMhs[idxMin] = listMhs[i];
            listMhs[i] = tmp;
        }
    }
}
```

2. buka kembali class MahasiswaDemo21, 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.printlx:n("Data mahasiswa setelah sorting menggunakan SELECTION SORT (ASC):");
list.selectionSort();
list.tampil();
```

3. Hasil percobaan

```
PS D:\ASD\Praktikum-ASD\Jobsheet6> cd "d:\
Masukkan data mahasiswa ke-1
NIM: 123
Nama: ali
Kelas: 2b
IPK: 3,9
Masukkan data mahasiswa ke-2
NIM: 124
Nama: ila
Kelas: 2b
IPK: 3,1
Masukkan data mahasiswa ke-3
NIM: 125
Nama: agus
Kelas: 2b
IPK: 3,6
Masukkan data mahasiswa ke-4
NIM: 126
Nama: tika
Kelas: 2b
IPK: 127
Masukkan data mahasiswa ke-5
NIM: 127
Nama: udin
Kelas: 2b
IPK: 3,2
```

```
------------------------------
Data mahasiswa setelah sorting menggunakan SELECTION SORT (ASC):
Nama: ila
NIM: 124
Kelas: 2b
IPK: 3.1
------------------------------
Nama: udin
NIM: 127
Kelas: 2b
IPK: 3.2
------------------------------
Nama: agus
NIM: 125
Kelas: 2b
IPK: 3.6
------------------------------
Nama: ali
NIM: 123
Kelas: 2b
IPK: 3.9
------------------------------
Nama: tika
NIM: 126
Kelas: 2b
IPK: 127.0
------------------------------
```

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!

Jawab: kode ini untuk mencari elemen dengan IPK terkecil dalam sisa daftar yang belum diurutkan. Setelah ketemu, nanti elemen itu akan ditukar ke posisi yang seharusnya. Proses ini diulang sampai seluruh daftar terurut secara ascending.

- int idxMin = i;
  Menyimpan indeks awal sebagai nilai minimum sementara
- for (int j = i+1; j < listMhs.length; j++)
  Mengecek elemen-elemen setelah indeks i, mencari apakah ada nilai yang lebih kecil dari yang sekarang dianggap minimum

- if (listMhs[j].ipk < listMhs[idxMin].ipk)
  Jika ditemukan IPK yang lebih kecil dari nilai minimum sementara, maka idxMin diperbarui dengan indeks elemen tersebut

Langkah-langkah:

Mengurutkan Data Mahasiswa Berdasarkan IPK Menggunakan Insertion Sort

1. Lihat kembali class MahasiswaBerprestasi21, 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++) {
            Mahasiswa21 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 MahasiswaDemo21, dan di dalam method main() tambahkan baris program untuk memanggil method insertionSort() dan tampil () tersebut

```
System.out.println(x:"Data yang sudah terurut menggunakan INSERTION SORT (ASC):");
list.insertionSort();
list.tampil();
```

3. Hasil percobaan

```
PS D:\ASD\Praktikum-ASD\Jobsheet6> cd "d:\
  ; if ($?) { javac MahasiswaDemo21.java }
o21 }
Masukkan data mahasiswa ke-1
NIM: 111
Nama: ayu
Kelas: 2c
IPK: 3,7
Masukkan data mahasiswa ke-2
NIM: 222
Nama: dika
Kelas: 2c
IPK:
3,0
Masukkan data mahasiswa ke-3
NIM: 333
Nama: ila
Kelas: 2c
IPK: 3,8
Masukkan data mahasiswa ke-4
NIM: 444
Nama: sus
Kelas: 2c
IPK: 3,1
Masukkan data mahasiswa ke-5
NIM: 555
Nama: yayuk
Kelas: 3c
IPK: 3,4
```

```
Data yang sudah terurut menggunakan INSERTION SORT (ASC):
Nama: dika
NIM: 222
Kelas: 2c
IPK: 3.0
------------------------------
Nama: sus
NIM: 444
Kelas: 2c
IPK: 3.1
------------------------------
Nama: yayuk
NIM: 555
Kelas: 3c
IPK: 3.4
------------------------------
Nama: ayu
NIM: 111
Kelas: 2c
IPK: 3.7
------------------------------
Nama: ila
NIM: 333
Kelas: 2c
IPK: 3.8
------------------------------
```

PERTANYAAN

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

```
while (j > 0 && listMhs[j - 1].ipk < temp.ipk) {
```

Hasil percobaan

```
Data yang sudah terurut menggunakan INSERTION SORT (ASC):
Nama: ila
NIM: 222
Kelas: 1a
IPK: 4.0
-----------------------------
Nama: iin
NIM: 333
Kelas: 1a
IPK: 3.8
-----------------------------
Nama: agus
NIM: 111
Kelas: 1a
IPK: 3.5
-----------------------------
Nama: iyes
NIM: 444
Kelas: 1a
IPK: 3.1
-----------------------------
Nama: seli
NIM: 555
Kelas: 1a
IPK: 3.0
-----------------------------
```