Nama : Muhammad Hikmal Al-Ghifary

Kelas : TI – 1B

Matkul : Praktikum Algoritma dan Struktur Data

SORTING (BUBBLE, SELECTION, DAN INSERTION SORT)

Percobaan 1: Mengimplementasikan Sorting menggunakan object

A. Bubble Sort

1. Buat folder baru bernama Jobsheet6 di dalam repository Praktikum ASD
2. Buat class `Sorting`, kemudian tambahkan atribut
3. Buatlah konstruktor dengan parameter Data[] dan jmlDat
4. Buatlah method bubbleSort bertipe void dan deklarasikan isinya menggunakan algoritma Bubble Sort.
5. Buatlah method tampil bertipe void dan deklarasikan isi method tersebut.

```java
public class Sorting15 {
    int [] data;
    int jumData;

    Sorting15 (int Data[], int jmlDat){
        jumData = jmlDat;
        data = new int [jmlDat];
        for (int i = 0; i < jumData; i++) {
            data[i]=Data[i];
        }
    }

    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;
                }
            }
        }
    }

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

        System.out.println();
    }
}
```

6. Buat class `SortingMain` kemudian deklarasikan array dengan nama a[] kemudian isi array tersebut

7. Buatlah objek baru dengan nama dataurut1 yang merupakan instansiasi dari class Sorting, kemudian isi parameternya
8. Lakukan pemanggilan method bubbleSort dan tampil

```java
public class SortingMain15 {
    Run | Debug
    public static void main(String[] args) {
        int a [] = {20,10,2,7,12};

        Sorting15 dataurut1 = new Sorting15(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();
    }
}
```

9. 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
PS C:\Users\Muhammad Hikmal AG\OneDrive\Docume
```

B. Selection Sort

1. Pada class Sorting yang sudah dibuat di praktikum sebelumnya tambahkan method `SelectionSort` yang mengimplementasikan pengurutan menggunakan algoritma selection sort.

```java
    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` kemudian isi array tersebut
3. Buatlah objek baru dengan nama dataurut2 yang merupakan instansiasi dari class Sorting, kemudian isi parameternya
4. Lakukan pemanggilan method `SelectionSort` dan tampil

```
18        Sorting15 dataurut2 = new Sorting15(b, b.length);
19
20        System.out.println(x:"Data awal 2");
21        dataurut2.tampil();
22        dataurut2.selectionSort();
23        System.out.println(x:"Data sudah diurutkan dengan SELECTION SORT (ASC)");
24        dataurut2.tampil();
```

5. Jalankan program dan amati hasilnya!

```
Data awal 2
30 20 2 8 14
Data sudah diurutkan dengan SELECTION SORT (ASC)
2 8 14 20 30
PS C:\Users\Muhammad Hikmal AG\OneDrive\Documents
```

C. Insertion Sort

1. Pada class Sorting yang sudah dibuat di praktikum sebelumnya tambahkan method insertionSort yang mengimplementasikan pengurutan menggunakan algoritma insertion sort.
2. Deklarasikan array dengan nama c[] pada kelas `SortingMain` kemudian isi array tersebut

```
49    void insertionSort(){
50        for (int i = 0; i < data.length; i++) {
51            int temp = data[i];
52            int j = i-1;
53            while (j>= 0 && data[j]>temp) {
54                data[j+1] = data[j];
55                j--;
56            }
57            data[j+1] = temp;
58        }
59    }
```

3. Buatlah objek baru dengan nama dataurut3 yang merupakan instansiasi dari class Sorting, kemudian isi parameternya
4. Lakukan pemanggilan method `insertionSort` dan tampil

```
27          int c [] = {40,10,4,9,3};
28
29          Sorting15 dataurut3 = new Sorting15(c, c.length);
30
31          System.out.println(x:"Data awal 3");
32          dataurut3.tampil();
33          dataurut3.insertionSort();
34          System.out.println(x:"Data sudah diurutkan menggunakan INSERTION SORT (ASC)");
35          dataurut3.tampil();
```

5. Jalankan program dan amati hasilnya!

```
Data awal 3
40 10 4 9 3
Data sudah diurutkan menggunakan INSERTION SORT (ASC)
3 4 9 10 40
PS C:\Users\Muhammad Hikmal AG\OneDrive\Documents\POL
```

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;
}
```

Kode tersebut digunakan sebagai media untuk menukar posisi pada bubble sort. Setiap array dicek dengan pembagian 1 elemen dibandingkan dengan elemen lainnya. Jika kondisi **data[j-1] > data[j]** terpenuhi, maka nilai kedua elemen tersebut akan ditukar, sehingga elemen dengan nilai yang lebih kecil akan berada di indeks yang lebih awal.

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

```
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;
        }
```

Nilai awal dari variabel min diset pada i, kemudian dilakukan perbandingan elemen lainnya yang dianggap minimum. Jika ditemukan nilai yang lebih kecil, maka nilai min akan diganti melalui kode **min = j**.

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

```
while (j>=0 && data[j]>temp)
```

Kode j>=0 berguna untuk memastikan bahwa indeks yang dibandingkan tetap berada dalam batas array. Jika elemen pada indeks j lebih besar dari temp, maka elemen akan digeser ke kanan untuk memberi ruang pada temp. Perulangan ini akan terus berjalan hingga ditemukan posisi yang sesuai untuk menyisipkan temp.

4. **Pada Insertion sort, apakah tujuan dari perintah**
```
data[j+1]= data[j];
```

Kode ini digunakan untuk menggeser elemen dalam array ke kanan agar dapat menyisipkan elemen baru di posisi yang sesuai. Saat ditemukan elemen yang lebih besar dari temp, elemen tersebut akan digeser ke indeks yang lebih tinggi (j+1). Pergeseran ini dilakukan secara bertahap hingga ditemukan posisi yang tepat untuk menyisipkan elemen temp dalam array yang sudah terurut.

Percobaan 2: Mengurutkan Data Mahasiswa Berdasarkan IPK

A. Bubble Sort

1. Buatlah class dengan nama `Mahasiswa`.
2. Untuk lebih jelasnya class tersebut dapat dilihat pada potongan kode di bawah ini

```java
public class Mahasiswa15 {
    String nim, nama, kelas;
    double ipk;

    Mahasiswa15() {}

    Mahasiswa15(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);
    }
}
```

3. Buat class `MahasiswaBerprestasi`
4. Tambahkan method tambah() di dalam class tersebut! Method tambah() digunakan untuk menambahkan objek dari class Mahasiswa ke dalam atribut listMhs.
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.
6. Tambahkan method `bubbleSort()` di dalam class tersebut!

```java
public class MahasiswaBerprestasi15 {
    Mahasiswa15 [] listMhs = new Mahasiswa15[5];
    int idx;

    void tambah(Mahasiswa15 m) {
        if (idx < listMhs.length) {
            listMhs[idx] = m;
            idx++;
        } else {
            System.out.println(x:"data sudah penuh");
        }
    }

    void tampil() {
        for (Mahasiswa15 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-1; j++) {
                if (listMhs[j].ipk > listMhs[j-1].ipk) {
                    Mahasiswa15 tmp = listMhs [j];
                    listMhs [j] = listMhs [j-1];
                    listMhs [j-1] = tmp;
                }
            }
        }
    }
}
```

7. Buat class `MahasiswaDemo`, 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 class MahasiswaDemo15 {
    Run | Debug
    public static void main(String[] args) {
        MahasiswaBerprestasi15 list = new MahasiswaBerprestasi15();
        Mahasiswa15 m1 = new Mahasiswa15(nm:"123", name:"Zidan", kls:"2A", ip:3.2);
        Mahasiswa15 m2 = new Mahasiswa15(nm:"124", name:"Ayu", kls:"2A", ip:3.5);
        Mahasiswa15 m3 = new Mahasiswa15(nm:"125", name:"Sofi", kls:"2A",ip:3.1);
        Mahasiswa15 m4 = new Mahasiswa15(nm:"126", name:"Sita", kls:"2A",ip:3.9);
        Mahasiswa15 m5 = new Mahasiswa15(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();
    }
}
```

8. Verifikasi hasil

```
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: Ayu
NIM: 124
Kelas: 2A
IPK: 3.5
--------------------------------------
Nama: Zidan
NIM: 123
Kelas: 2A
IPK: 3.2
--------------------------------------
Nama: Sofi
```

PERTANYAAN

1. **Perhatikan perulangan di dalam `bubbleSort()` di bawah ini:**
   a. **Mengapa syarat dari perulangan i adalah i<listMhs.length-1 ?**

      Perulangan pertama i digunakan untuk melakukan n-1 tahap perulangan, di mana n adalah jumlah elemen dalam array. Karena pada setiap iterasi elemen terbesar yang belum terurut akan dipindahkan ke posisinya yang benar, setelah n-1 iterasi, seluruh elemen sudah dalam urutan yang benar. Oleh karena itu, i harus kurang dari listMhs.length - 1.

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

      Perulangan j digunakan untuk membandingkan dan menukar elemen-elemen dalam array agar elemen dengan nilai terbesar bergerak ke posisi akhir. Pada iterasi pertama (i = 0), elemen terbesar dipindahkan ke posisi terakhir, sehingga tidak perlu dibandingkan lagi di iterasi berikutnya. Oleh karena itu, pada iterasi ke-i, batas atas perulangan dalam (j) dikurangi sebesar i, karena i elemen terbesar sudah berada di tempat yang benar.

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

      Perulangan i akan berlangsung sebanyak listMhs.length - 1 = 50 - 1 = 49 kali.

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

   // Penambahan pada class Mahasiswa Berprestasi

```
MahasiswaBerprestasi15(int jumlah) {
    listMhs = new Mahasiswa15[jumlah];
    idx = 0;
}
```

// Perubahan pada class Mahasiswa Demo

```
import java.util.Scanner;

public class MahasiswaDemo15 {
    Run | Debug
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        MahasiswaBerprestasi15 list = new MahasiswaBerprestasi15(jumlah);

        for (int i = 0; i < jumlah; 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();

            Mahasiswa15 m = new Mahasiswa15(nim, nama, kelas, ipk);
            list.tambah(m);
        }
```

B. Selection 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;
            }
        }

        Mahasiswa15 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:"Data yang sudah terurut menggunakan SELECTION SORT (ASC): ");
list.selectionSort();
list.tampil();
```

3. Coba jalankan kembali class MahasiswaDemo, dan amati hasilnya!

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

PERTANYAAN

1. **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 apa kode tersebut, jelaskan!

- idxMin dideklarasikan untuk menyimpan indeks pertama dari array yang belum disorting. Untuk awal, diisi dengan nilai i

- Loop for untuk mencari nilai ipk yang lebih kecil, dimulai dari indeks ke i+1.
- If berfungsi sebagai media pertukaran nilai idxMin. Jika ditemukan nilai ipk yang lebih kecil dari idxMin, maka nilainya akan diganti dengan nilai tersebut.

C. Insertion Sort

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++) {
        Mahasiswa15 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:"Data yang sudah terurut menggunakan INSERTION SORT (ASC): ");
list.insertionSort();
list.tampil();
```

3. Coba jalankan kembali class MahasiswaDemo, dan amati hasilnya!

```
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\Muhammad Hikmal AG\OneDrive\Documents\POLINEMA
```

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++) {
        //Mahasiswa15 temp = listMhs[i];
        //int j=i;
        //while (j>0 && listMhs[j-1].ipk>temp.ipk) {
            //listMhs[j] = listMhs[j-1];
            //j--;
        //}
        //listMhs[j] = temp;
    //}

    for (int i = 0; i < listMhs.length; i++) {
        Mahasiswa15 temp = listMhs[i];
        int j=i;
        while (j>0 && listMhs[j-1].ipk < temp.ipk) { // perubahan
            listMhs[j] = listMhs[j-1];
            j--;
        }

        listMhs[j] = temp;
    }
}
```