Nama : Dicky Darmawan

Kelas : TI 1B

NIM : 244107020037

Praktikum 1 - Mengimplementasikan Sorting menggunakan object

Langkah Praktikum 1

a. SORTING – BUBBLE SORT

1. Buat folder baru bernama Jobsheet6 di dalam repository Praktikum ASD
2. Buat class Sorting<No Presensi>, kemudian tambahkan atribut sebagai berikut:

```
public class Sorting08 {

    int[] data;
    int jumData;
```

3. Buatlah konstruktor dengan parameter Data[] dan jmlDat

```
    Sorting08(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 - i; 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 class SortingMain08 {
    public static void main(String[] args) {
        int a[] = {20, 10, 2, 7, 12};
```

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

```
Sorting08 dataurut1 = new Sorting08(a, a.length);
```

8. 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();
```

9. Jalankan program, dan amati hasilnya!

Verifikasi Hasil Percobaan

```
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 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 classSorting, kemudian isi parameternya

```
Sorting08 dataurut2 = new Sorting08(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 (
dataurut2.tampil();
```

5. Jalankan program dan amati hasilnya!

Verifikasi 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<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; 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

```
Sorting08 dataurut3 = new Sorting08(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 (
dataurut3.tampil();
```

5. Jalankan program dan amati hasilnya!

Verifikasi 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 tersebut!

```
if (data[j   int[] data   j]) {
    temp = data[j];
    data[j] = data[j - 1];
    data[j - 1] = temp;
```

   if (data[j - 1] > data[j]), mengecek apakah elemen sebelumnya (data[j - 1]) lebih besar daripada elemen saat ini (data[j]). jika benar, maka dilakukan pertukaran (swap) antara data[j] dan data[j - 1].

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;
    }
}
```

3. Pada Insertion sort, jelaskan maksud dari kondisi pada perulangan while (j>0 && data[j]>temp)

   Selama j lebih besar dari 0 dan data[j] lebih besar dari temp maka program di dalam while akan terus dijalankan

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

   Untuk menggeser elemen ke kanan agar ada tempat kosong untuk menyisipkan elemen yang sedang diproses

Praktikum 2- (Sorting Menggunakan Array of Object)

Langkah Praktikum 2 - Mengurutkan Data Mahasiswa Berdasarkan IPK (Bubble Sort)

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 Mahasiswa08 {

    String nim, nama, kelas;
    double ipk;

    Mahasiswa08() {

    }

    public Mahasiswa08(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 MahasiswaBerprestasi08 {

    Mahasiswa08[] listMhs = new Mahasiswa08[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(Mahasiswa08 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 (Mahasiswa08 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; i++) {
        for (int j = 1; j < listMhs.length - i; j++) {
            if (listMhs[j].ipk > listMhs[j - 1].ipk) {
                Mahasiswa08 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.

```
MahasiswaBerprestasi08 list = new MahasiswaBerprestasi08();
Mahasiswa08 m1 = new Mahasiswa08("123", "Zidan", "2A", 3.2);
Mahasiswa08 m2 = new Mahasiswa08("124", "Ayub", "2A", 3.5);
Mahasiswa08 m3 = new Mahasiswa08("125", "Sofi", "2A", 3.1);
Mahasiswa08 m4 = new Mahasiswa08("126", "Sita", "2A", 3.9);
Mahasiswa08 m5 = new Mahasiswa08("127", "Miki", "2A", 3.7);

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

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

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

Verifikasi Hasil Percobaan

```
Data mahasiswa sebelum sorting  Data mahasiswa setelah sorting berdasarkan IPK (DESC)
Nama: Zidan                     Nama: Zidan
NIM: 123                        NIM: 123
Kelas: 2A                       Kelas: 2A
IPK: 3.2                        IPK: 3.2
----------------------------   ----------------------------
Nama: Ayub                      Nama: Ayub
NIM: 124                        NIM: 124
Kelas: 2A                       Kelas: 2A
IPK: 3.5                        IPK: 3.5
----------------------------   ----------------------------
Nama: Sofi                      Nama: Sofi
NIM: 125                        NIM: 125
Kelas: 2A                       Kelas: 2A
IPK: 3.1                        IPK: 3.1
----------------------------   ----------------------------
Nama: Sita                      Nama: Sita
NIM: 126                        NIM: 126
Kelas: 2A                       Kelas: 2A
IPK: 3.9                        IPK: 3.9
----------------------------   ----------------------------
Nama: Miki                      Nama: Miki
NIM: 127                        NIM: 127
Kelas: 2A                       Kelas: 2A
IPK: 3.7                        IPK: 3.7
----------------------------   ----------------------------
```

Pertanyaan

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

       Karena setelah n-1 tahap array sudah pasti terurut, maka cukup melakukan perulangan hingga i < listMhs.length - 1. Jika jumlah data dalam array adalah 5, maka hanya perlu 4 tahap sorting, karena elemen terakhir secara otomatis sudah berada di posisi yang benar.

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

       Karena jumlah perbandingan berkurang setiap tahap, sehingga batas perulangan j dikurangi dengan nilai i

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

       Perulangan i akan berlangsung sebanyak 49 kali (karena i < 50 - 1)
2. Modifikasi program diatas dimana data mahasiswa bersifat dinamis (input dari keyborad) yang terdiri dari nim, nama, kelas, dan ipk!

```java
Scanner input = new Scanner(System.in);
MahasiswaBerprestasi08 list = new MahasiswaBerprestasi08();

for (int i = 0; i < 5; 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();
    System.out.print(s:"IPK: ");
    String ip = input.nextLine();
    double ipk = Double.parseDouble(ip);
```

Mengurutkan Data Mahasiswa Berdasarkan IPK (Selection Sort)

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; i++) {
        int idxMin = i;
        for (int j = i + 1; j < listMhs.length; j++) {
            if (listMhs[j].ipk < listMhs[idxMin].ipk) {
                idxMin = j;
            }
        }
        Mahasiswa08 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!
   Coba jalankan kembali class MahasiswaDemo, dan amati hasilnya! Apakah kini data mahasiswa telah tampil urut menaik berdasar ipk?

```
System.out.println(x:"Data yang sudah terurut menggunakan SELECTION SORT
list.SelectionSort();
list.tampil();
```

Verifikasi Hasil Percobaan

```
Data mahasiswa sebelum sorting
Nama: DICKY
NIM: 123
Kelas: 1B
IPK: 3.85
----------------------------
Nama: RAMA
NIM: 124
Kelas: 1B
IPK: 3.7
----------------------------
Nama: SAM
NIM: 125
Kelas: 1B
IPK: 3.89
----------------------------
Nama: KADEK
NIM: 126
Kelas: 1B
IPK: 3.83
----------------------------
Nama: SITA
NIM: 127
Kelas: 1B
IPK: 3.7
----------------------------
```

```
Data yang sudah terurut menggunakan SELECTION SORT (ASC)
Nama: RAMA
NIM: 124
Kelas: 1B
IPK: 3.7
----------------------------
Nama: SITA
NIM: 127
Kelas: 1B
IPK: 3.7
----------------------------
Nama: KADEK
NIM: 126
Kelas: 1B
IPK: 3.83
----------------------------
Nama: DICKY
NIM: 123
Kelas: 1B
IPK: 3.85
----------------------------
Nama: SAM
NIM: 125
Kelas: 1B
IPK: 3.89
----------------------------
```

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!

Int idxMIn = I, menyimpan indeks dari elemen minimum (nilai IPK terkecil) yang ditemukan. indeks elemen pertama dalam bagian yang belum terurut (i) dianggap sebagai elemen terkecil.

for (int j = i + 1; j < listMhs.length; j++), melakukan perulangan untuk mencari nilai terkecil dari array. j = i + 1 karena elemen indeks i sudah dianggap sebagai nilai minimum sementara.

if (listMhs[j].ipk < listMhs[idxMin].ipk), jika ditemukan elemen yang lebih kecil dari listMhs[idxMin].ipk maka akan elemen j akan diperbarui ke idxMin.

Mengurutkan Data Mahasiswa Berdasarkan IPK menggunakan Insertion Sort

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++) {
        Mahasiswa08 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
list.insertionSort();
list.tampil();
```

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

Verifikasi Hasil Percobaan

```
Data mahasiswa sebelum sorting
Nama: DICKY
NIM: 123
Kelas: 1B
IPK: 3.85
----------------------------
Nama: RAMA
NIM: 124
Kelas: 1B
IPK: 3.7
----------------------------
Nama: SAM
NIM: 125
Kelas: 1B
IPK: 3.89
----------------------------
Nama: KADEK
NIM: 126
Kelas: 1B
IPK: 3.83
----------------------------
Nama: SITA
NIM: 127
Kelas: 1B
IPK: 3.7
----------------------------
Data yang sudah terurut menggunakan INSERTION SORT (ASC)
Nama: MISEL
NIM: 126
Kelas: 1B
IPK: 3.8
----------------------------
Nama: FIKA
NIM: 124
Kelas: 1B
IPK: 3.81
----------------------------
Nama: JUAN
NIM: 125
Kelas: 1B
IPK: 3.83
----------------------------
Nama: DICKY
NIM: 123
Kelas: 1B
IPK: 3.85
----------------------------
Nama: PETRA
NIM: 127
Kelas: 1B
IPK: 3.86
----------------------------
```

Pertanyaan

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

```
void insertionSort() {
    for (int i = 1; i < listMhs.length; i++) {
        Mahasiswa08 temp = listMhs[i];
        int j = i;
        while (j > 0 && listMhs[j - 1].ipk < temp.ipk) {
            listMhs[j] = listMhs[j - 1];
            j--;
        }
        listMhs[j] = temp;
    }
}
```