Nama : Muhammad Hikmal Al-Ghifary

Kelas : TI – 1B

Matkul : Praktikum Algoritma dan Struktur Data

SEARCHING

Percobaan 1: Searching / Pencarian Menggunakan Algoritma Sequential Search

1. Pada pertemuan Jobsheet 7 ini akan menggunakan class Mahasiswa<no Presensi>, MahasiswaBerprestasi<no Presensi>, dan MahasiswaDemo <no presensi> pada pertemuan Jobsheet 6 sebelumnya
2. Buat folder baru bernama Jobsheet7 di dalam repository Praktikum ASD, kemudian buka ketiga class dari Jobsheet 6 tersebut dan copy ke folder Jobsheet 7
3. Tambahkan method sequentialSearching bertipe integer dengan parameter cari bertipe double pada class MahasiswaBerprestasi<no presensi>. Kemudian Deklarasikan isi method sequentialSearching dengan algoritma pencarian data menggunakan teknik sequential searching.
4. Buatlah method tampilPosisi bertipe void dan Deklarasikan isi dari method tampilPosisi pada class MahasiswaBerprestasi<no presensi>.
5. Pada class MahasiswaBerprestasi<no presensi>, buatlah method tampilDataSearch bertipe void dan Deklarasikan isi dari method tampilDataSearch .

```
76      int sequentialSearching(double cari) {
77          int posisi = -1;
78          for(int j = 0; j < listMhs.length; j++) {
79              if(listMhs[j].ipk == cari) {
80                  posisi = j;
81                  break;
82              }
83          }
84          return posisi;
85      }
86
87      void tampilPosisi(double x, int pos) {
88          if(pos != -1) {
89              System.out.println("Data Mahasiswa dengan IPK: " + x + " ditemukan pada index " + pos);
90          } else {
91              System.out.println("Data " + x + " tidak ditemukan");
92          }
93      }
94
95      void tampilDataSearch(double x, int pos) {
96          if(pos != -1) {
97              System.out.println("nim\t : " + listMhs[pos].nim);
98              System.out.println("nama\t : " + listMhs[pos].nama);
99              System.out.println("kelas\t : " + listMhs[pos].kelas);
100             System.out.println("ipk\t : " + listMhs[pos].ipk);
101         } else {
102             System.out.println("Data mahasiswa dengan IPK " + x + " tidak di temukan");
103         }
104     }
```

6. Pada class MahasiswaDemo , tambahkan kode program berikut ini untuk melakukan pencarian data dengan algoritma sequential searching.

```
        list.tampil();

       // System.out.println("Data Mahasiswa Sebelum Sorting: ");
       // list.tampil();

        //System.out.println("Data Mahasiswa setelah sorting berdasarkan IPK (DESC): ");
        //list.bubbleSort();
        //list.tampil();

        //System.out.println("Data yang sudah terurut menggunakan SELECTION SORT (ASC): ");
        //list.selectionSort();
        //list.tampil();

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


        //melakukan pencarian data
        System.out.println(x:"----------------------");
        System.out.println(x:"Pencarian Data");
        System.out.println(x:"----------------------");
        System.out.println(x:"Masukkan IPK mahasiswa yang dicari");
        System.out.print(s:"IPK: ");
        double cari = sc.nextDouble();

        System.out.println(x:"Menggunakan Sequential Searching");
        double posisi = list.sequentialSearching(cari);
        int pss = (int) posisi;
        list.tampilPosisi(cari, pss);
        list.tampilDataSearch(cari, pss);
    }
```

7. Verifikasi hasil percobaan

```
Nama: adi
NIM: 111
Kelas: 2
IPK: 3.6
-------------------------------------
Nama: tio
NIM: 222
Kelas: 2
IPK: 3.8
-------------------------------------
Nama: ila
NIM: 333
Kelas: 2
IPK: 3.0
-------------------------------------
Nama: lia
NIM: 444
Kelas: 2
IPK: 3.5
-------------------------------------
Nama: fia
NIM: 555
Kelas: 2
IPK: 3.3
-------------------------------------
----------------------
Pencarian Data
----------------------
Masukkan IPK mahasiswa yang dicari
IPK: 3,5
Menggunakan Sequential Searching
Data Mahasiswa dengan IPK: 3.5 ditemukan pada index 3
nim      : 444
nama     : lia
kelas    : 2
ipk      : 3.5
PS C:\Users\Muhammad Hikmal AG\OneDrive\Documents\POLI
```

PERTANYAAN

1. **Jelaskan perbedaan method tampilDataSearch dan tampilPosisi pada class MahasiswaBerprestasi!**
   - tampilDataSearch berfungsi untuk menampilkan posisi data jika ditemukan, serta menampilkan informasi lengkap mahasiswa tersebut (termasuk nim, nama, kelas, dan ipk).
   - tampilPosisi berfungsi untuk menampilkan informasi apakah data dengan IPK tertentu ditemukan atau tidak, serta di indeks berapa data tersebut jika ditemukan.
2. **Jelaskan fungsi break pada kode program dibawah ini!**

```
if (listMhs[j].ipk==cari){
    posisi=j;
    break;
}
```

Fungsi break pada kode program tersebut adalah untuk berhenti / keluar dari iterasi for saat ipk yang dicari ditemukan. Tanpa break, perulangan akan terus berjalan meskipun data sudah ditemukan, sehingga bisa menyebabkan program melakukan iterasi yang tidak perlu, memperlambat proses pencarian.

Percobaan 2: Searching / Pencarian Menggunakan Binary Search

1. Pada percobaan 6.2.1 (sequential search) tambahkan method findBinarySearch bertipe integer pada class MahasiswaBerprestasi. Kemudian Deklarasikan isi method findBinarySearch dengan algoritma pencarian data menggunakan teknik binary searching.

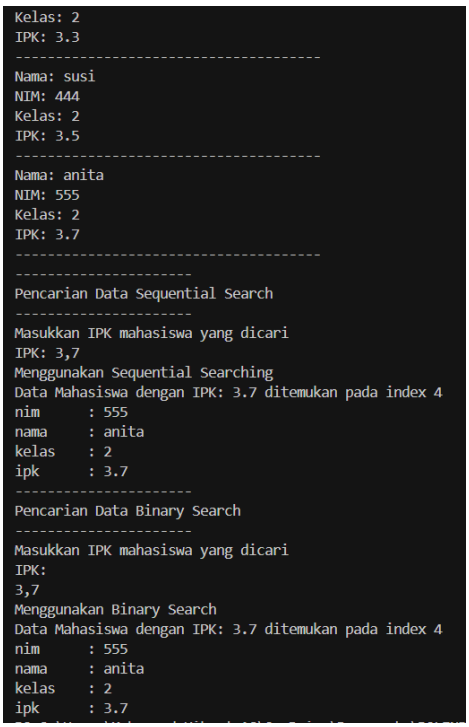
```
int findBinarySearch(double cari, int left, int right) {
    int mid;
    if (right >= left) {
        mid = (left + right)/2;
        if (cari == listMhs[mid].ipk) {
            return mid;
        } else if (listMhs[mid].ipk > cari) {
            return findBinarySearch(cari, left, mid -1);
        } else {
            return findBinarySearch(cari, mid + 1, right);
        }
    }

    return -1;
}
```

2. Panggil method findBinarySearch terdapat pada class MahasiswaBerprestasi di kelas MahasiswaDemo. Kemudia panggil method tampilPosisi dan tampilDataSearch

```
59          System.out.println(x:"----------------------");
60          System.out.println(x:"Pencarian Data Binary Search");
61          System.out.println(x:"----------------------");
62          System.out.println(x:"Masukkan IPK mahasiswa yang dicari");
63          System.out.print(s:"IPK: ");
64          double cari2 = sc.nextDouble();
65
66          System.out.println(x:"Menggunakan Binary Search");
67          double posisi2 = list.findBinarySearch(cari, left:0, jumlah-1);
68          int pss2 = (int) posisi2;
69          list.tampilPosisi(cari, pss2);
70          list.tampilDataSearch(cari, pss2);
```

3. Jalankan dan amati hasilnya (inputkan data IPK secara terurut -ASC seperti verifikasi hasil percobaan dibawah ini).



PERTANYAAN

1. **Tunjukkan pada kode program yang mana proses divide dijalankan!**

   ```
   mid = (left + right)/2;
   ```

   Kode tersebut membagi data menjadi 2 bagian (sub bagian)

2. **Tunjukkan pada kode program yang mana proses conquer dijalankan!**

   ```
   if (cari == listMhs[mid].ipk) {
       return mid;
   } else if (listMhs[mid].ipk > cari) {
       return findBinarySearch(cari, left, mid -1);
   } else {
       return findBinarySearch(cari, mid + 1, right);
   }
   ```

   Kode yang menyelesaikan sub masalah

3. **Jika data IPK yang dimasukkan tidak urut. Apakah program masih dapat berjalan? Mengapa demikian!**

   Program tidak akan berjalan dikarenakan binary search harus mengeksekusi data secara terurut, kemudian membaginya menjadi 2 sub bagian dengan nilai tengah sebagai acuan.

4. **Jika IPK yang dimasukkan dari IPK terbesar ke terkecil (missal : 3.8, 3.7, 3.5, 3.4, 3.2) dan elemen yang dicari adalah 3.2. Bagaimana hasil dari**

binary search? Apakah sesuai? Jika tidak sesuai maka ubahlah kode program binary search agar hasilnya sesuai

Modifikasi kode

```
int findBinarySearch(double cari, int left, int right) {
    if (right >= left) {
        int mid = (left + right) / 2;

        boolean ascending = listMhs[left].ipk < listMhs[right].ipk;

        if (listMhs[mid].ipk == cari) {
            return mid;
        }

        if (ascending) {
            if (listMhs[mid].ipk > cari) {
                return findBinarySearch(cari, left, mid - 1);
            } else {
                return findBinarySearch(cari, mid + 1, right);
            }
        } else {
            if (listMhs[mid].ipk < cari) {
                return findBinarySearch(cari, left, mid - 1);
            } else {
                return findBinarySearch(cari, mid + 1, right);
            }
        }
    }
    return -1;
}
```

5. **Modifikasilah program diatas yang mana jumlah mahasiswa yang diinputkan sesuai dengan masukan dari keyboard.**

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