MODUL

STRUKTUR DATA

Disusun oleh :

Tim Pembuat Modul Kurikulum 2019

(disadur dari Buku Ajar Struktur Data Menggunakan Java, AgungBP, 2017)

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER

AKAKOM

YOGYAKARTA

2019

KATA PENGANTAR

Sudah dapat dipastikan bahwa sebagai calon informatikawan yang nantinya akan bergelut di dunia IT, mahasiswa pasti akan dituntut banyak hal dalam penguasaan hardware maupun software, tidak hanya dalam pengoperasian program-program aplikasi mutakhir namun juga penguasaannya terhadap konsep pemrograman yang baik dan benar. Sebagai salah satu bentuk pembelajaran tersebut adalah dengan dilakukannya praktikum.

Praktikum Struktur Data adalah praktikum yang akan membantu mahasiswa memahami konsep-konsep pengelolaan data secara algoritmis. Mahasiswa akan belajar segala hal yang mendasar yang berkaitan tentang data, mulai dari pengenalan data, struktur penyimpanan data, pengelolaan data di dalam memori, hingga pemecahan permasalahan tentang data, dan modul ini diciptakan untuk membantu mahasiswa mempelajarinya.

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu membuat penyelesaian dari kasus-kasus tentang pengelolaan data dalam sebuah program berbahasa Java.

Modul ini masih akan terus disempurnakan seiring dengan berjalannya proses pembelajaran hingga mencapai kondisi ideal. Untuk itu kritik bagi perbaikan modul ini silakan kirimkan melalui email ***agung.b.prasetyo@gmail.com***.

Akhirnya penulis mengucapkan selamat berpraktikum, selamat berproses, dan jadilah yang terbaik.

Salam.

Tim Penyusun.

DAFTAR ISI

KATA PENGANTAR ii
DAFTAR ISI iii
MODUL 1 TIPE-TIPE DATA DALAM JAVA 1
MODUL 2 STRUKTUR PENYIMPAN BERBASIS RECORD & ARRAY OF RECORD 12
MODUL 3 PENGELOLAAN DATA PADA ARRAY : PENAMBAHAN & PENGHAPUSAN DATA 26
MODUL 4 PENGELOLAAN DATA PADA ARRAY: PENCARIAN DATA (SEARCHING) 39
MODUL 5 PENGELOLAAN DATA PADA ARRAY: PENGURUTAN (SORTING) 47
MODUL 6 PEMANFAATAN ARRAY SEBAGAI STACK (TUMPUKAN) DAN QUEUE (ANTRIAN) 63
MODUL 7 POINTER DALAM JAVA 72
MODUL 8 SINGLE LINKEDLIST (SENARAI BERANTAI TUNGGAL) 87
MODUL 9 SINGLE LINKEDLIST (LANJUTAN) 95
MODUL 10 DOUBLE LINKEDLIST (SENARAI BERANTAI GANDA) 102
MODUL 11 PENGURUAN DATA (SORTING) & PENCARIAN DATA (SEARCHING) LINKEDLIST 110
MODUL 12 COLLECTION 117
MODUL 13 POHON BINER 127
MODUL 14 HASHING PADA LARIK 134

MODUL 1
TIPE-TIPE DATA DALAM JAVA

CAPAIANPEMBELAJARAN

Mahasiswa dapat menggunakan berbagai tipe data untuk menyimpan data baik aphabetic, aplhanumerik, maupun boolean

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Setiap bahasa pemrograman memiliki tipe data yang spesifik. Tipe data akan digunakan untuk mendeklarasikan variable yang digunakan. Tipe data digunakan untuk menentukan bentuk data yag dapat ditampung oleh sebuah variabel .

Dalam java terdapat dua jenis tipe data. Yang pertama adalah tipe data *primitive* yang merupakan tipe data bawaan dari compiler java. Tipe data ini akan anda pelajari pada modul 1 ini.

Sedangkan tipe data yang kedua adalah tipe data *buatan* yang baru akan anda pelajari pada modul 2.

Dalam bahasa Java, *tipe data primitive* dibedakan menjadi tiga bagian yaitu :

1. **Tipe Data Alphabetic**
    - Char
    - String
2. **Tipe Data Alphanumeric**
    a. Tipe data Bilangan Bulat
    - Byte
    - Short
    - Int
    - Long

    b. Tipe Data Bilangan Pecahan
    - Float
    - Double
3. **Tipe Data Boolean**

1. Tipe Data Alphabetic

Char (Karakter)

Tipe data char merupakan tipe yang digunakan untuk menyatakan sebuah karakter, bisa berupa huruf/ tandabaca/ simbol, yang didefinisikan dengan diawali dan diakhiri dengan tanda ‘ (petik tunggal).

Untuk merepresentasikan semua karakter yang ada bahasa Java menggunakan karakter *Unicode* yaitu sekumpulan karakter umum yang terdapat pada semua bahasa, seperti English, Latin, Arab, Yunani dan lain-lainnya.Karakter *Unicode* yang membutuhkan ukuran 16-bit dan memiliki 1680 jenis karakter.

Berikut ini disajikan contoh program sederhana menggunakan tipe data char.

```
public class tipeData {
       public static void main(String[] args) {
              char   data1 = 'C';
              System.out.println("Nilai Char    : "+ data1);
       }
}
Hasil Eksekusi :
       Nilai Char    : C
       Press any key to continue . . .
```

Program 1.1 Contoh pemakaian `char`

Dalam Java, data bertipe char juga dapat diinputkan melalui keyboard untuk disimpan dalam sebuah variabel char. Namun java tidak menyediakan fungsi khusus untuk membaca masukan bertipe char sehingga perlu dibuat pembacaan karakter menggunakan pembacaan kode unicode menggunakan fungsi `System.in.read()` dan kemudian mengkonversinya menggunakan fungsi (char) untuk dapat membacanya. Berikut ini adalah program untuk membaca masukannya.

```
import java.util.Scanner;
public class inputViaKeyboard
{ public static void main(String[] args)
  {
       Scanner masukan = new Scanner(System.in);
       int   bacaTombol=0;
       char  huruf;
       System.out.print("Silakan masukkan sebuah huruf: ");
       try
       {   bacaTombol = System.in.read();      }
       catch(java.io.IOException e)
       {      }
       huruf = (char)bacaTombol;
       System.out.println("Huruf yang anda entri adalah : " + huruf);
  }
}
Hasil Eksekusi :
       Silakan masukkan sebuah huruf: Z ↵

       Huruf yang anda entri adalah : Z
       Press any key to continue . . .
```

Program 1.2 Contoh pemakaian `char` Via Keyboard

String

Tipe data String merupakan kumpulan dari tipe data char. Karena merupakan kumpulan char, maka tipe data String dapat digunakan untuk menyimpan kalimat.

Jika dilihat dari unsur pembentuknya, tipe data string bukan merupakan *tipe data primitif*, tetapi sudah merupakan sebuah *objek* yang berisi kumpulan tipe data char.

Berikut ini disajikan contoh program sederhana menggunakan tipe data string.

```
public class tipeData {
     public static void main(String[] args) {
          String data2 = "Namaku Agung Budi Prasetyo";
          System.out.println("Nilai String  : "+ data2);
     }
}
Hasil Eksekusi :
       Nilai String  : Namaku Agung Budi Prasetyo
       Press any key to continue . . .
```

Program 1.3 Contoh pemakaian `String`

Dalam bahasa java, data bertipe string juga dapat diinputkan melalui keyboard untuk disimpan dalam sebuah variabel string. Untuk keperluan tersebut java telah menyediakan sebuah fungsi untuk membaca masukan yaitu `next()` dan `nextline()`. Namun dari kedua fungsi di atas penulis lebih menyarankan untuk menggunakan fungsi `next()`.

Berikut ini adalah program untuk membaca masukannya.

```
import java.util.Scanner;
public class inputViaKeyboard
{
  public static void main(String[] args)
  {
       Scanner masukan = new Scanner(System.in);
       String kalimat;
       System.out.print("Silakan masukkan sebuah kalimat : ");
       kalimat = masukan.next();
       System.out.println("Kalimat yang anda entri adalah :" + kalimat);
  }
}
Hasil Eksekusi :
       Silakan masukkan sebuah kalimat: Namaku Agung Budi Prasetyo ↵

       Kalimat yang anda entri adalah : Namaku Agung Budi Prasetyo
       Press any key to continue . . .
```

Program 1.4 Contoh pemakaian `String` via keyboard

Untuk karakter-karakter yang tidak dapat diketikkan secara langsung melalui keyboard, java menyediakan beberapa *escape sequence* (pasangan karakter yang dianggap sebagai karakter tunggal). *Escape sequence* tidak dianggap sebagai *String*, melainkan tetap sebagai tipe karakter khusus. Berikut ini adalah beberapa contoh escape sequence.

Kode	Keterangan

\b	Backspace
\t	Tab
\n	Linefeed
\r	Carriage return
\f	Formfeed
\’	Petik tunggal
\”	Petik ganda
\ddd	Octal (dd= 0 s/d 377)
\xdd	Heksadesimal (dd=0 s/d FF atau ff)

Contoh penggunaan escape sequence

```
public class tipeData {
     public static void main(String[] args) {
          String kalimat = "Halo \"Agung Budi Prasetyo\" apa kabar.." ;
          System.out.println("Nilai String  : "+ kalimat);
     }
}
Hasil Eksekusi :
       Nilai String  : Halo "Agung Budi Prasetyo" apa kabar..
       Press any key to continue . . .
```

Program 1.5 Contoh pemakaian escape sequence

2. Tipe Data Aplhanumeric

A. Tipe Bilangan Bulat

Ada empat macam tipe bilangan bulat, di mana masing-masing memiliki jangkauan nilai yang berbeda yaitu byte, short, int dan long.

Tipe	Ukuran	Jangkauan Nilai
byte	8 bit	-128 s/d 127
short	16 bit	-32.768 s/d 32.767
int	32 bit	-2.147.483.648 s/d 2.147.483.647
long	64 bit	-9.223.372.036.854.775.808 s/d 9.223.372.036.854.775.807

Byte

Type data *byte* umumnya digunakan pada saat kita bekerja dengan sebuah data *stream* dari suatu file, memory, maupun jaringan koputer yaitu untuk kepeluan proses membaca/menulis berkas. Selain itu, tipe ini juga digunakan saat bekerja dengan data biner. Namun tidak jarang pula tipe byte digunakan untuk menyimpan bilangan yang tidak terlalu besar (bilangan di bawah bilangan 127, contohnya umur pegawai).

Short

Tipa data short memiliki ukuran yang sedikit lebih besar dibandingkan byte. Pada umumnya tipe data short digunakan pada komputer-komputer 16-bit yang memang memiliki ukuran bilangan yang terbatas. Tipe short digunakan pula pada aplikasi khusus yang memperhatikan penggunaan memori.

Int

Tipe data int merupakan tipe yang paling banyak dipakai dalam

merepresentasikan angka dalam Java. Hal ini karena tipe data int dianggap paling efisien dibandingkan dengan tipe-tipe integer lainnya. Tipe *Int* banyak digunakan untuk indeks dalam struktur pengulangan maupun dalam konstruksi sebuah *array.*Selain itu, secara teori setiap ekspresi yang melibatkan tipe integer *byte, short, int, long* semuanya itu akan dipromosikan ke *int* terlebih dahulu sebelum dilakukan proses perhitungan

Long

Tipe ini digunakan untuk kasus-kasus tertentu yang nilainya berada di luar rentang tipe int, karna tipe ini punya range paling tinggi dibanding Integer lainnya. Dengan kata lain, tipe long terpaksa digunakan jika data memiliki range diluar range int.

Semua bilangan bulat dalam Java secara default dianggap sebagai tipe int. Sedangkan bilangan yang ingin dikategorikan sebagai long harus diakhiri dengan huruf `L`. Misalnya : `18102006`**`L`**. Berikut ini disajikan contoh program sederhana menggunakan tipe data byte, short, int, dan long.

```
public class tipeData {
  public static void main(String[] args) {
        // Tipe data primitif
        byte   data3 = 34;
        short  data4 = 714;
        int    data5 = 2235641;
        long   data6 = 546767226531L;
        System.out.println("Nilai Byte    : "+ data3);
        System.out.println("Nilai Short   : "+ data4);
        System.out.println("Nilai Int     : "+ data5);
        System.out.println("Nilai Long    : "+ data6);
  }
}
Hasil Eksekusi :
  Nilai Byte    : 34
  Nilai Short   : 714
  Nilai Int     : 2235641
  Nilai Long    : 546767226531
  Press any key to continue . . .
```

Program 1.6 Contoh pemakaian byte, short, int dan long

Dalam bahasa java, data bertipe byte, short, int dan long juga dapat diinputkan melalui keyboard untuk disimpan dalam sebuah variabel dengan tipe yang sama. Untuk keperluan tersebut java telah menyediakan sebuah fungsi untuk membaca masukan yaitu `nextByte()` untuk tipe data byte, `nextShort()` untuk tipe data short, `nextInt()` untuk tipe data int, dan `nextLong()` untuk tipe data long.

Berikut ini adalah program untuk membaca masukannya.

```
import java.util.Scanner;
public class inputViaKeyboard
{
  public static void main(String[] args)
  {
   Scanner masukan = new Scanner(System.in);
   byte   bilanganByte;
   short  bilanganShort;
   int    bilanganInt;
   long   bilanganLong;
```

```
    System.out.print("Silakan masukkan bilangan bertipe byte: ");
    bilanganByte = masukan.nextByte();
    System.out.print("Silakan masukkan bilangan bertipe short: ");
    bilanganShort = masukan.nextShort();
    System.out.print("Silakan masukkan bilangan bertipe int: ");
    bilanganInt = masukan.nextInt();
    System.out.print("Silakan masukkan bilangan bertipe long: ");
    bilanganLong = masukan.nextLong();

    System.out.println("bilangan byte yang anda entri = " + bilanganByte);
    System.out.println("bilangan short yang anda entri = "+bilanganShort);
    System.out.println("bilangan int   yang anda entri = "+ bilanganInt);
    System.out.println("bilangan long  yang anda entri = "+ bilanganLong);
   }
}
Hasil Eksekusi :
    Silakan masukkan bilangan bertipe byte: 34 ↵
    Silakan masukkan bilangan bertipe short: 714 ↵
    Silakan masukkan bilangan bertipe int: 2235641 ↵
    Silakan masukkan bilangan bertipe long: 546767226531 ↵

    bilangan byte  yang anda entri adalah = 34
    bilangan short yang anda entri adalah = 714
    bilangan int   yang anda entri adalah = 2235641
    bilangan long  yang anda entri adalah = 546767226531
Press any key to continue . . .
```

Program 1.7 Contoh pemakaian `byte`, `short`, `int` dan `long` via keyboard

B. Tipe Bilangan Pecahan (Float dan double)

Ada dua tipe data yang berkaitan dengan bilangan pecahan yang sering juga diistilahkan dengan sebutan bilangan titik mengambang.

Tipe	Ukuran	Jangkauan Nilai
float	32 bit, presisi 6-7 digit	-3.4E38 s/d +3.4E38
double	64 bit, presisis 14-15 bit	-1.7E308 s/d +1.7E308

Semua bilangan pecahan atau desimal dalam Java tanpa diakhiri huruf **f** akan dianggap sebagai *double*. Sedangkan bilangan yang ingin dikategorikan sebagai *float* harus diakhiri dengan huruf **F**. Misalnya : `4.22F` atau `2.314f`.

```
public class tipeData {
   public static void main(String[] args) {
        // Tipe data primitif
        float  data7 = 1.733F; // tipe data pecahan
        double data8 = 4.967; // tipe data pecahan
        System.out.println("Nilai Float   : "+ data5);
        System.out.println("Nilai Double  : "+ data6);
   }
}

Hasil Eksekusi :
       Nilai Float   : 4.967
       Nilai Double  : 1.733
       Press any key to continue . . .
```

Program 1.8 Contoh pemakaian `float` dan `double`

3. Tipe Data Boolean

Dalam Java dikenal tipe data `boolean` yang terdiri dari dua nilai saja, yaitu `true` dan `false`. Boolean sangat penting dalam mengevaluasi suatu kondisi, dan sering digunakan untuk menentukan alur program.

```
public class tipeData {
     public static void main(String[] args) {
          // Tipe data primitif
          boolean data9 = true;
          boolean data10 = false;
          System.out.println("Nilai data9  : "+ data9);
          System.out.println("Nilai data10 : "+ data10);
     }
}
Hasil Eksekusi :
  Nilai data9  : true
  Nilai data10 : false
  Press any key to continue . . .
```

Program 1.9 Contoh pemakaian `boolean`

Tipe Data Array/ Larik

Selain ketiga tipe data di atas, dalam java terdapat juga tipe data Array/ Larik. Tipe data Array adalah tipe data untuk membuat variabel secara bersusun.

Dengan adanya tipe data array/ larik kita dapat membuat variabel-wariabel kembar yang bertipe data sama. Sebagai contoh dapat kita lihat program berikut ini.

```
import java.util.Scanner;
public class tipeDataArray
{
  public static void main(String[] args)
  {
       String hobi[] = new String[3];
       Scanner masukan = new Scanner(System.in);

       System.out.println("Silakan masukkan hobi (maks 3) : ");
       System.out.print("hobi ke-0 : "); hobi[0] = masukan.next();
       System.out.print("hobi ke-1 : "); hobi[1] = masukan.next();
       System.out.print("hobi ke-2 : "); hobi[2] = masukan.next();

       System.out.println("Hobi ke-0 anda adalah " + hobi[0]);
       System.out.println("Hobi ke-1 anda adalah " + hobi[1]);
       System.out.println("Hobi ke-2 anda adalah " + hobi[2]);
  }
}

Hasil Eksekusi :
    Silakan masukkan hobi (maks 3) :
    hobi ke-0 : musik ↵
    hobi ke-1 : mancing ↵
    hobi ke-2 : touring ↵

    Hobi ke-0 anda adalah musik
    Hobi ke-1 anda adalah mancing
    Hobi ke-2 anda adalah touring
```

```
Press any key to continue . . .
```

Program 1.10 Contoh pemakaian Array/ Larik

Pada contoh di atas terdapat tiga buah variabel hobi. Sebagaimana kita ketahui sangat jarang seseorang yang hanya memiliki satu hobi saja. Kebanyakan orang memiliki lebih dari satu hobi. Jika kita menggunakan variabel tunggal untuk menyimpan data hobi maka kita tidak dapat menyimpan hobi yang lebih dari satu.

Di sisi lain jika kita menggunakan variabel dengan penamaan yang berbeda untuk hobi—hobi yang ada (misal: `hobi_1, hobi_2, hobi_3`, dst) maka kita akan kesulitan sendiri pada saat kita akan mengelola data-data tersebut. Oleh karena itu tipe data array/ larik akan menjadi solusi yang tepat untuk keperluan tersebut.

PRAKTIK

1. Praktik 1 *(percobaan tentang tipe data integer)*

Tulislah dan eksekusilah program dibawah ini

```
public class pembagian{
     public static void main(String[] args){
          int banyaknyaApel = 5;
          int jumlahAnak = 2;
          int perolehan;
          perolehan = banyaknyaApel / jumlahAnak;
          System.out.println("Masing2 mendapat = " + perolehan);
     }
}
```

Program di atas digunakan untuk menghitung 5 dibagi 2 yang menghasilkan nilai 2.5. Sekarang eksekusilah program di atas. Berapa hasil yang diperoleh sewaktu program tersebut dieksekusi? Apakah hasilnya 2.5? Mengapa bisa demikian? Jelaskan dan simpulkan dalam laporan anda.

2. Praktik 2 *(percobaan tentang tipe data Long)*

Tulislah dan eksekusilah program dibawah ini

```
public class cobaLong{
       public static void main(String[] args){
              long coba = 1234567890123;
              System.out.println(coba);
       }
}
```

Apa yang terjadi sewaktu program di atas dieksekusi? Mengapa bisa demikian? Sekarang tambahkan "`L`" pada akhir angka pada baris 3. Apa yang terjadi? Mengapa bisa demikian? Jelaskan dalam laporan anda?

3. Praktik 3 *(percobaan tentang tipe data String dan Char)*

Tulislah dan eksekusilah program dibawah ini

```
public class cobaKalimat{
       public static void main(String[] args){
              char coba="HAI";
              System.out.println(coba);
       }
}
```

Apa yang terjadi sewaktu program di atas dieksekusi? Mengapa bisa demikian? Sekarang gantilah `char` pada baris 3 dengan `String`. Apa yang terjadi? Mengapa bisa demikian? Jelaskan dalam laporan anda?

4. Praktik 4 *(percobaan tentang lingkup variabel)*

Tulislah dan eksekusilah program dibawah ini

```
1. public class Variabel {
2.   static int a;
3.   public static void main(String[] args) {
4.        int x;
5.        x = 10;
6.        a = 2;
7.        System.out.println("Nilai a : " + a);
8.        {    int y;
9.             y = 5;
10.            System.out.println("Nilai x : " + x);
11.            System.out.println("Nilai a : " + a);
12.            {      int z;
13.                   z = 20;
14.                   System.out.println("Nilai x + y + z + a :"
15.                                + (x + y + z + a));
16.            }
17.            System.out.println("Nilai Z : " + Z);
18.            System.out.println("Nilai y : " + y);
19.       }
20.       System.out.println("Nilai Z : " + Z);
21.       System.out.println("Nilai y : " + y);
22.       System.out.println("Nilai x : " + x);
23.  }
24.}
```

Apakah yang terjadi sewaktu program tersebut dieksekusi? Mengapa bisa demikian? Coba sekarang hapuslah instruksi pada baris 17, 20 dan 21 kemudian eksekusi kembali program tersebut. Apa yang terjadi? Mengapa bisa demikian? Jelaskan dalam laporan anda.

5. Praktik 5 *(percobaan tentang menerima masukan keyboard)*

Tulislah dan eksekusilah program dibawah ini

```
import java.util.Scanner;
public class inputDataViaKeyboard
{
  public static void main(String[] args)
  {    String nama;
       String alamat;
       int    umur;
       char   jekel; //jenis kelamin
       String hobi[] = new String[3];
       float  ipk;

       Scanner masukan = new Scanner(System.in);
       int bacaTombol=0;

       System.out.print("Silakan masukkan nama anda : ");
       nama = masukan.next();

       System.out.print("Silakan masukkan alamat anda : ");
       alamat = masukan.next();

       System.out.print("Silakan masukkan umur anda : ");
       umur = masukan.nextInt();

       System.out.print("Silakan masukkan Jenis Kelamin anda : ");
       try
       {   bacaTombol = System.in.read();
       }
       catch(java.io.IOException e)
       {
       }
       jekel = (char)bacaTombol;

       System.out.println("Silakan masukkan hobi (maks 3) : ");
       System.out.print("hobi ke-0 : ");hobi[0] = masukan.next();
       System.out.print("hobi ke-1 : ");hobi[1] = masukan.next();
       System.out.print("hobi ke-2 : ");hobi[2] = masukan.next();

       System.out.print("Silakan masukkan IPK anda : ");
       ipk = masukan.nextFloat();

       System.out.println("Nama anda adalah " + nama);
       System.out.println("Nama alamat adalah " + alamat);
       System.out.println("Umur anda adalah " + umur);
       System.out.println("Jenis Kelamin anda adalah " + jekel);
       System.out.println("Hobi ke-0 anda adalah " + hobi[0]);
       System.out.println("Hobi ke-1 anda adalah " + hobi[1]);
       System.out.println("Hobi ke-2 anda adalah " + hobi[2]);
       System.out.println("IPK anda adalah " + ipk);
  }
}
```

Apakah yang terjadi jika program di atas di *run* ? Jelaskanlah dan simpulkan dalam laporan anda.

LATIHAN

1. Buatlah sebuah program sederhana untuk menerima masukan bertipe string dari keyboard berupa "*password*". Selanjutnya program akan mencocokan password tersebut. Jika password yang dimasukkan sama dengan *"AKAKOM"* maka akan tampil pesan *"password anda benar"* tetapi jika tidak akan tampil pesan *"password anda salah*".

TUGAS

1. Dengan menggunakan struktur data seperti pada praktik 5 (nama, alamat, umur, jenis kelamin, hobi (3 buah), IPK), buatlah sebuah program untuk memasukkan biodata minimal untuk 5 orang mahasiswa.

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 1-11, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 2
STRUKTUR PENYIMPAN BERBASIS RECORD DAN ARRAY OF RECORD

CAPAIANPEMBELAJARAN

Mahasiswa dapat membuat suatu struktur *record* (rekaman) dan *array of record* (rekaman dalam larik) untuk menyimpan data menggunakan bahasa java

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Pada percobaan terdahulu (modul 1) kita telah belajar bagaimana membuat media penyimpan (variabel) menggunakan tipe data *primitive* baik itu tipe alphabetic, tipe data numeric maupun tipe data *array/ larik*.

Pada modul 2 ini kita akan lebih banyak belajar bagaimana membuat media penyimpan berbasis *record (rekaman)*. Record sering juga disebut Obyek/ Simpul/ List/ Node/ Senarai. Dalam pembuatannya, record didefinisikan sebagai variabel ber*tipe data buatan* (harus dideklarasikan menggunakan class).

Sebelum membahas tipe data buatan ada baiknya kita memahami terlebih dahulu mengapa kita memerlukan struktur penyimpan berbasis *record (rekaman)* melalui topik penyimpan yang tidak terstuktur berikut ini.

A. Struktur Penyimpan Yang Tidak Terstruktur

Pada percobaan terdahulu (modul 1) kita telah belajar bagaimana membuat beberapa variabel berupa ***nama***, ***alamat***, ***umur***, ***jekel, hobi [ ]*** dan ***ipk*** seperti tersaji pada program 2.1. berikut ini.

```
import java.util.Scanner;
public class inputDataViaKeyboard
{
  public static void main(String[] args)
  {    String nama;
```

```
        String alamat;
        int    umur;
        char   jekel; //jenis kelamin
        String hobi[] = new String[3];
        float  ipk;

        Scanner masukan = new Scanner(System.in);
        int bacaTombol=0;

        System.out.print("Silakan masukkan nama anda : ");
        nama = masukan.next();

        System.out.print("Silakan masukkan alamat anda : ");
        alamat = masukan.next();

        System.out.print("Silakan masukkan umur anda : ");
        umur = masukan.nextInt();

        System.out.print("Silakan masukkan Jenis Kelamin anda : ");
        try
        {   bacaTombol = System.in.read();
        }
        catch(java.io.IOException e)
        {
        }
        jekel = (char)bacaTombol;
        System.out.println("Silakan masukkan hobi (maks 3) : ");
        System.out.print("hobi ke-0 : ");hobi[0] = masukan.next();
        System.out.print("hobi ke-1 : ");hobi[1] = masukan.next();
        System.out.print("hobi ke-2 : ");hobi[2] = masukan.next();
        System.out.print("Silakan masukkan IPK anda : ");
        ipk = masukan.nextFloat();

        System.out.println("Nama anda adalah " + nama);
        System.out.println("Nama alamat adalah " + alamat);
        System.out.println("Umur anda adalah " + umur);
        System.out.println("Jenis Kelamin anda adalah " + jekel);
        System.out.println("Hobi ke-0 anda adalah " + hobi[0]);
        System.out.println("Hobi ke-1 anda adalah " + hobi[1]);
        System.out.println("Hobi ke-2 anda adalah " + hobi[2]);
        System.out.println("IPK anda adalah " + ipk);
   }
}
```

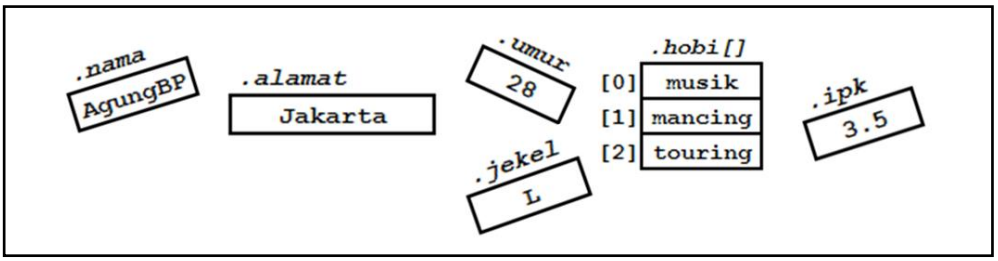
Hasil Eksekusi

```
        Silakan masukkan nama anda : AgungBP ↵
        Silakan masukkan alamat anda : Jakarta ↵
        Silakan masukkan umur anda : 28 ↵
        Silakan masukkan Jenis Kelamin anda : L ↵
        Silakan masukkan hobi (maks 3) :
        hobi ke-0 : musik ↵
        hobi ke-1 : mancing ↵
        hobi ke-2 : touring ↵
        Silakan masukkan IPK anda : 3.5 ↵

        Nama anda adalah AgungBP
        Nama alamat adalah Jakarta
        Umur anda adalah 28
        Jenis Kelamin anda L
        Hobi ke-0 anda musik
        Hobi ke-1 anda mancing
        Hobi ke-2 anda touring
```

```
IPK anda adalah 3.5

Press any key to continue . . .
```

Program 2.1

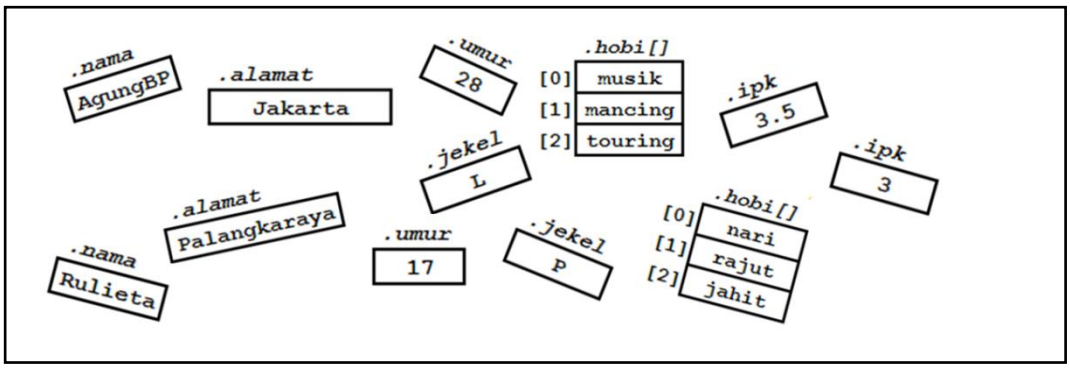
Apabila kita perhatikan dengan sungguh-sungguh program 2.1 di atas, meskipun terlihat seperti sebuah kesatuan variabel yang saling berhubungan, namun sebenarnya variabel ***nama***, ***alamat***, ***umur***, ***jekel, hobi [ ], ipk*** bukanlah suatu kesatuan yang utuh. Hal ini disebabkan karena masing-masing variabel tersebut dideklarasi secara terpisah menggunakan tipe data masing-masing sehingga tentu akan membentuk suatu struktur penyimpan yang terpisah pula sekalipun datanya adalah data milik satu orang **("AgungBP"; "Jakarta",28, 'L',"musik", "mancing", "touring" ,3.5)**. Akibat dari deklarasi variabel yang terpisah tersebut, maka struktur penyimpanan yang terjadi akan menjadi seperti yang diilustrasikan pada gambar 2.1 di bawah ini.



Gambar 2.1

Pada gambar 2.1 tersebut tidak dapat terlihat bahwa variabel ***nama*** memiliki hubungan langsung dengan variabel ***alamat*** dan ***umur***. Begitu juga dengan variabel ***ipk*** dan ***jekel***. Jadi, sekalipun variabel-variabel tersebut digunakan untuk menyimpan sebuah kesatuan data milik seorang pribadi **("AgungBP";"Jakarta",28, 'L',"musik", "mancing", "touring" ,3.5)**, namun secara struktur variabel-variabel tersebut merupakan variabel-variabel yang terpisah sehingga data-data tersebut belum dapat dikatakan sebuah *kesatuan data*. Nah, dengan demikian model penyimpan data seperti di atas dapat diartikan sebagai **model penyimpan data yang *tidak terstruktur***.

Apakah anda dapat membayangkan jika model penyimpanan data seperti di atas kita gunakan untuk menyimpan biodata milik lebih dari satu orang? Kira-kira yang terjadi akan seperti gambar 2.2 berikut ini

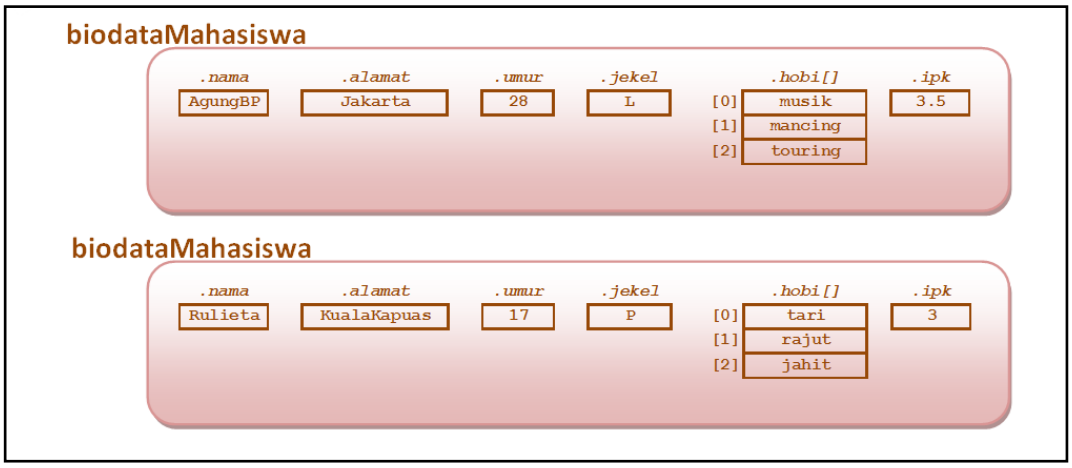


Gambar 2.2

Bila kita melihat gambar 2.2 di atas tentu kita akan kesulitan untuk menentukan keterkaitan data-data di atas. Kita tidak tahu berapa umur `AgungBP`, apakah `28` atau `17`? Siapakah yang memiliki hobi `rajut`, apakah `AgungBP` atau `Rulieta`? berapakah IPK dari `Rulieta`, dan seterusnya. Maka dapat kita simpulkan struktur penyimpan seperti di atas tidak layak digunakan untuk menyimpan biodata dalam jumlah banyak.

B. Struktur Penyimpan yang Terstruktur (Berbasis Record)

Lalu bagaimana dengan **penyimpan data yang *terstruktur*** ? Perhatikan ilustrasi pada gambar 2.3 berikut.



Gambar 2.3

Apabila dibandingkan dengan gambar 2.2, variabel-variabel pada gambar 2.3 tampak jauh lebih rapi dan terstruktur. Hal ini karena baik variabel ***nama***, ***alamat***, ***umur***, ***jekel, hobi [ ]*** maupun ***ipk***, kesemuanya diikat atau dibungkus dalam satu kesatuan oleh satu variabel induk yang bernama ***biodataMahasiswa***. Nah, variabel pembungkus inilah yang selanjutnya dikenal dengan istilah **obyek/ heap/ simpul/ node**. Pembahasan lebih jauh tentang obyek/ heap ini dapat anda ikuti pada modul 7.

Untuk dapat membuat struktur sebagaimana gambar 2.3 di atas, anda dapat mengikuti script dalam program 2.2 berikut ini.

```
import java.util.Scanner;
class formatBiodata
{   //bagian deklarasi struktur record ----------------------------------
    String nama;
    String alamat;
    int    umur;
    char   jekel;
    String hobi[] = new String[3];
    float  ipk;
}

public class strukturRekamanData
{   public static void main(String[] args)
    {   //bagian deklarasi record -------------------------------------
        formatBiodata    biodataMahasiswa = new formatBiodata();

        //bagian entri data melalui keyboard ---------------------------
        Scanner masukan = new Scanner(System.in);
        int bacaTombol=0;

        System.out.print("Silakan masukkan nama anda : ");
        biodataMahasiswa.nama = masukan.next();

        System.out.print("Silakan masukkan alamat anda : ");
        biodataMahasiswa.alamat = masukan.next();

        System.out.print("Silakan masukkan umur anda : ");
        biodataMahasiswa.umur = masukan.nextInt();

        System.out.print("Silakan masukkan Jenis Kelamin anda : ");
        try
        {   bacaTombol = System.in.read();
        }
        catch(java.io.IOException e)
        {      }
        biodataMahasiswa.jekel = (char)bacaTombol;

        System.out.println("Silakan masukkan hobi (maks 3) : ");
        System.out.print("hobi ke-0 : ");
        biodataMahasiswa.hobi[0] = masukan.next();
        System.out.print("hobi ke-1 : ");
        biodataMahasiswa.hobi[1] = masukan.next();
        System.out.print("hobi ke-2 : ");
        biodataMahasiswa.hobi[2] = masukan.next();

        System.out.print("Silakan masukkan IPK anda : ");
        biodataMahasiswa.ipk = masukan.nextFloat();

        System.out.println("Nama anda adalah "  + biodataMahasiswa.nama);
        System.out.println("Nama alamat adalah "+ biodataMahasiswa.alamat);
        System.out.println("Umur anda adalah "   + biodataMahasiswa.umur);
        System.out.println("Jenis Kelamin anda " + biodataMahasiswa.jekel);
        System.out.println("Hobi ke-0 anda "   + biodataMahasiswa.hobi[0]);
        System.out.println("Hobi ke-1 anda "   + biodataMahasiswa.hobi[1]);
        System.out.println("Hobi ke-2 anda "   + biodataMahasiswa.hobi[2]);
        System.out.println("IPK anda adalah "  + biodataMahasiswa.ipk);
    }
}
```

Hasil Eksekusi

```
Silakan masukkan nama anda : AgungBP ↵
Silakan masukkan alamat anda : Jakarta ↵
Silakan masukkan umur anda : 28 ↵
Silakan masukkan Jenis Kelamin anda : L ↵
Silakan masukkan hobi (maks 3) :
hobi ke-0 : musik ↵
hobi ke-1 : mancing ↵
hobi ke-2 : touring ↵
Silakan masukkan IPK anda : 3.5 ↵

Nama anda adalah Agung
Nama alamat adalah Jakarta
Umur anda adalah 28
Jenis Kelamin anda L
Hobi ke-0 anda musik
Hobi ke-1 anda mancing
Hobi ke-2 anda touring
IPK anda adalah 3.5

Press any key to continue . . .
```

Program 2.2

Dari program 2.2 di atas, walaupun hasil eksekusinya sama dengan program 2.1, namun secara struktur kedua program tersebut sangat jauh berbeda. Pada program 2.2 anda dapat melihat bahwa ada sebuah variabel bernama ***biodataMahasiswa*** yang berfungsi untuk menyatukan variabel yang lebih kecil yang berupa ***nama, alamat, umur, jekel***, ***hobi*** *[ ]* dan ***ipk***.

Jika kita lihat pada bagian deklarasi, variabel ***biodataMahasiswa*** ini tidaklah dibentuk menggunakan tipe data primitif seperti string, char, int, dan lainnya, tetapi dideklarasikan dalam tipe data bernama ***formatBiodata***. Tipe data ***formatBiodata*** ini tidaklah dikenal oleh Java karena memang bukanlah tipe bawaan dari java. Karena tipe data ini bukanlah bawaan java melainkan merupakan ciptaan programmer maka tipe data ini yang disebut dengan **tipe data buatan.**

Dalam pembuatannya, tipe data buatan dibentuk dengan melibatkan penggunaan **kelas (class)**, sedangkan variabelnya sendiri yaitu ***biodataMahasiswa*** disebut sebagai **obyek**. (anda akan mempelajari class dan obyek lebih jauh dalam Pemrograman Berorientasi Obyek). Variabel atau obyek yang bernama ***biodataMahasiswa*** ini akan menjadi pembungkus atau wadah bagi variabel-variabel yang lebih kecil yaitu ***nama, alamat, umur, jekel, hobi[ ]*** dan ***ipk***.

Dari tata cara penyebutannya, memang akhirnya penyebutan variabel ***nama, alamat, umur, jekel***, dan juga ***ipk*** tidaklah menjadi sederhana karena penyebutannya harus senantiasa didahului dengan menyebut nama obyeknya terlebih dahulu baru kemudian disusul dengan tanda titik (.) dan dilanjutkan dengan penyebutan nama variabelnya. Format baku penyebutannya mengikuti aturan sebagai berikut:

[nama obyek] • [nama variabel]

Sebagai contoh :

```
biodataMahasiswa.nama
biodataMahasiswa.alamat
biodataMahasiswa.umur
biodataMahasiswa.jekel
```

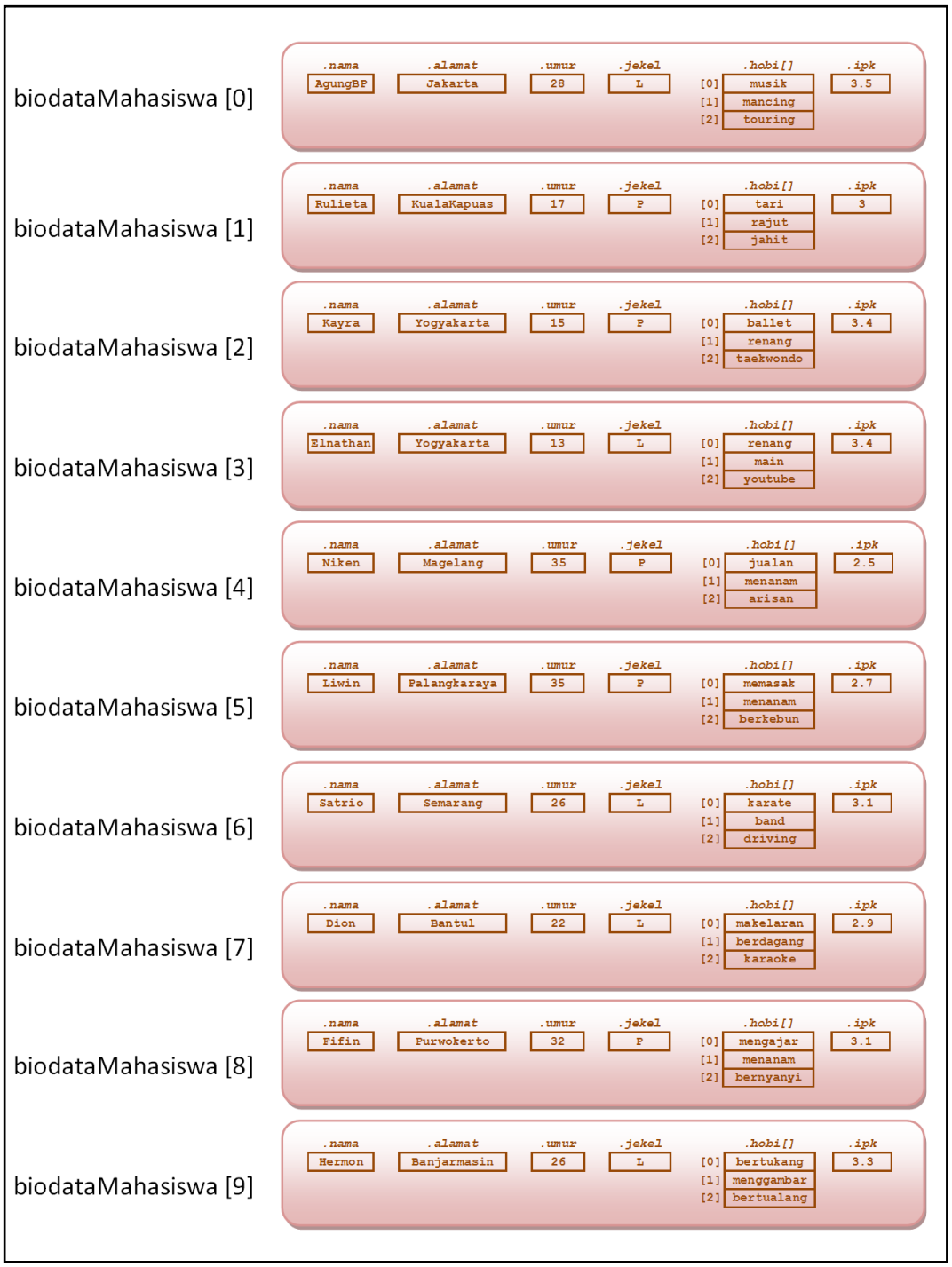
```
biodataMahasiswa.hobi[0]
biodataMahasiswa.hobi[1]
biodataMahasiswa.hobi[2]
biodataMahasiswa.ipk
```

Sekalipun terlihat lebih rumit, dengan menggunakan struktur penyimpan yang telah dipaparkan di atas kita telah dapat memastikan bahwa data-data *"AgungBP","Jakarta", 28,'L', "musik", "mancing", "touring" 3.5* yang tersimpan dalam variabel *nama, alamat, umur, jekel, hobi[0], hobi[1], hobi[2], ipk* adalah biodata milik seorang pribadi yang sama. Dengan demikian model penyimpan data tersebut sudah dapat diartikan merupakan ***model penyimpan data yang terstruktur***.

Karena fungsinya yang dapat membungkus atau menyatukan beberapa variabel yang terpisah maka konsep penyimpanan data seperti ini juga dapat disebut dengan konsep penyimpanan yang ***berbasis pada record (rekaman)***.

C. Struktur Penyimpan Berbasis Array of Record (Rekaman dalam Larik)

Nah, berikut ini kita akan membuat suatu susunan array (larik) yang terdiri dari record-record (rekaman) *biodataMahasiswa* sedemikian sehingga terbentuk sebuah struktur data sebagaimana tergambar dalam ilustrasi gambar 2.4 berikut ini.



Gambar 2.4

Untuk dapat membuat struktur array seperti di atas, anda dapat mengikuti script yang tersaji pada program 2.3 berikut ini.

```
import java.util.Scanner;
class formatBiodata
{   //bagian deklarasi struktur record ----------------------------------
    String nama;
```

```
    String alamat;
    int    umur;
    char   jekel;
    String hobi[] = new String[3];
    float  ipk;
}

class strukturRekamanData
{  public static void main(String[] args)
   {
      int N=5;
      //bagian deklarasi record berbasis LARIK -----------------------
      formatBiodata    biodataMahasiswa[] = new formatBiodata[5];
      biodataMahasiswa[0] = new formatBiodata();
      biodataMahasiswa[1] = new formatBiodata();
      biodataMahasiswa[2] = new formatBiodata();
      biodataMahasiswa[3] = new formatBiodata();
      biodataMahasiswa[4] = new formatBiodata();

      //bagian entri data ke dalam struktur larik -----------------------
      Scanner masukan = new Scanner(System.in);
      int bacaTombol=0;

      for (int i=0;  i<=N-1;  i++)
      {  System.out.print("Silakan masukkan nama anda : ");
         biodataMahasiswa[i].nama = masukan.next();

         System.out.print("Silakan masukkan alamat anda : ");
         biodataMahasiswa[i].alamat = masukan.next();

         System.out.print("Silakan masukkan umur anda : ");
         biodataMahasiswa[i].umur = masukan.nextInt();

         System.out.print("Silakan masukkan Jenis Kelamin anda : ");
         try
         {  bacaTombol = System.in.read();
         }
         catch(java.io.IOException e)
         {
         }
         biodataMahasiswa[i].jekel = (char)bacaTombol;

         System.out.println("Silakan masukkan hobi (maks 3) : ");
         System.out.print("hobi ke-0 : ");
         biodataMahasiswa[i].hobi[0] = masukan.next();
         System.out.print("hobi ke-1 : ");
         biodataMahasiswa[i].hobi[1] = masukan.next();
         System.out.print("hobi ke-2 : ");
         biodataMahasiswa[i].hobi[2] = masukan.next();

         System.out.print("Silakan masukkan IPK anda : ");
         biodataMahasiswa[i].ipk = masukan.nextFloat();
         System.out.println("");
      }

      //bagian menampilkan isi struktur Larik -------------------------
      System.out.println("-------------------------------------------");
      System.out.println("NAMA ALAMAT UMUR JEKEL HOBI1 HOBI2 HOBI3 IPK");
      System.out.println("-------------------------------------------");
      for (int i=0;  i<=N-1;  i++)
      {  System.out.print  (biodataMahasiswa[i].nama + "  ");
         System.out.print  (biodataMahasiswa[i].alamat + "  ");
```

```
        System.out.print   (biodataMahasiswa[i].umur + "  ");
        System.out.print   (biodataMahasiswa[i].jekel + "  ");
        System.out.print   (biodataMahasiswa[i].hobi[0] + "  ");
        System.out.print   (biodataMahasiswa[i].hobi[1] + "  ");
        System.out.print   (biodataMahasiswa[i].hobi[2] + "  ");
        System.out.println(biodataMahasiswa[i].ipk);
      }
      System.out.println("-------------------------------------------");
   }
}

Hasil Eksekusi :
-----------------------------------------------------------------------
   NAMA      ALAMAT        UMUR JEKEL  HOBI[0]  HOBI[1]   HOBI[2]   IPK
-----------------------------------------------------------------------
0. AgungBP   Jakarta       28   L      musik    mancing   touring   3.5
1. Rulieta   KualaKapuas   17   P      nari     rajut     jahit     3.0
2. Karti     Palangkaraya  18   P      renang   fbkan     masak     2.5
3. Hermon    Banjarmasin   25   L      kasti    bola      mancing   3.4
4. Anis      Surakarta     30   P      jualan   drama     nonton    0.3
5. Yanti     Purwokerto    21   P      lukis    musik     selfi     2.0
6. Handono   Bekasi        29   L      touring  musik     hiking    3.1
7. Anton     Jakarta       17   L      dangdut  jajan     turing    2.0
8. Mantri    Magelang      45   L      hiphop   wayang    game      3.1
9. Satrio    Yogyakarta    30   L      musik    drama     karate    3.3
-----------------------------------------------------------------------
Press any key to continue . . .
```

Program 2.3

Sedikit berbeda dengan program 2.2 sebelumnya, karena menggunakan larik (array) maka pada program 2.3 penyebutan nama obyek yang mendahului penyebutan variabel nama, alamat, umur, jenisKelaminnya, dan juga IPK, harus diikuti juga dengan tanda kurung array [ ].

[nama obyek] ***[indeks]*** ● *[nama variabel]*

Sebagai contoh :

```
biodataMahasiswa[0].nama
biodataMahasiswa[0].alamat
biodataMahasiswa[0].umur
biodataMahasiswa[0].jekel
biodataMahasiswa[0].hobi[0]
biodataMahasiswa[0].hobi[1]
biodataMahasiswa[0].hobi[2]
biodataMahasiswa[0].ipk
```

Pada program 2.3 di atas dibangun sebuah struktur penyimpan array yang berorientasi pada record (array of record). Program tersebut mampu menampung data mahasiswa sebanyak N = 5 orang.

D. Pembuatan Program Secara Modular (Fungsi)

Pada bagian ini kita akan memperbaiki program 2.3 menjadi lebil rapi. Apabila kita memperhatikan program tersebut dengan seksama tampak semua script ditulis di

dalam program utama (main). Hal tersebut bukanlah hal yang baik. Dengan program 2.4 kita akan mencoba mengelompok-kelompokkan beberapa bagian program sesuai dengan kegunaannya.

Apabila kita mencermati program-program yang dibuat sebelum ini terdapat 3 bagian besar. Yang pertama bagian **deklarasi**, kedua bagian **entri data**, dan ketiga bagian **menampilkan data**. Oleh karena itu pada program 2.4 berikut ini script program akan ditulis secara modular berdasarkan ketiga bagian di atas.

```
import java.util.Scanner;
class formatBiodata
{  //bagian deklarasi struktur record ---------------------------------
   String nama;
   String alamat;
   int    umur;
   char   jekel;
   String hobi[] = new String[3];
   float  ipk;
}
class pertemuan3
{  public static int N=5;

   //--------------------------------------------------
   //--- Fungsi untuk mengentri data ke dalam Larik ---
   //--------------------------------------------------
   public static void ngentriData(formatBiodata biodataMahasiswa[])
   {
      //bagian entri data ke dalam struktur larik ----------------
      Scanner masukan = new Scanner(System.in);
      int bacaTombol=0;

      for (int i=0;  i<=N-1;  i++)
      {  System.out.print("Silakan masukkan nama anda : ");
         biodataMahasiswa[i].nama = masukan.next();

         System.out.print("Silakan masukkan alamat anda : ");
         biodataMahasiswa[i].alamat = masukan.next();

         System.out.print("Silakan masukkan umur anda : ");
         biodataMahasiswa[i].umur = masukan.nextInt();

         System.out.print("Silakan masukkan Jenis Kelamin anda : ");
         try
         {  bacaTombol = System.in.read();
         }
         catch(java.io.IOException e)
         {
         }
         biodataMahasiswa[i].jekel = (char)bacaTombol;

         System.out.println("Silakan masukkan hobi (maks 3) : ");
         System.out.print("hobi ke-0 : ");
         biodataMahasiswa[i].hobi[0] = masukan.next();
         System.out.print("hobi ke-1 : ");
         biodataMahasiswa[i].hobi[1] = masukan.next();
         System.out.print("hobi ke-2 : ");
         biodataMahasiswa[i].hobi[2] = masukan.next();

         System.out.print("Silakan masukkan IPK anda : ");
         biodataMahasiswa[i].ipk = masukan.nextFloat();
```

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

    //-------------------------------------------------
    //---        Fungsi untuk menampilkan data        ---
    //-------------------------------------------------
    public static void tampilkanData(formatBiodata biodataMahasiswa[])
    {
      //bagian menampilkan isi struktur Larik -------------------------
      System.out.println("--------------------------------------------");
      System.out.println("NAMA ALAMAT UMUR  JEKEL HOBI1 HOBI2 HOBI3 IPK");
      System.out.println("--------------------------------------------");
      for (int i=0;  i<=N-1;  i++)
      {  System.out.print   (biodataMahasiswa[i].nama + "  ");
         System.out.print   (biodataMahasiswa[i].alamat + "  ");
         System.out.print   (biodataMahasiswa[i].umur + "  ");
         System.out.print   (biodataMahasiswa[i].jekel + "  ");
         System.out.print   (biodataMahasiswa[i].hobi[0] + "  ");
         System.out.print   (biodataMahasiswa[i].hobi[1] + "  ");
         System.out.print   (biodataMahasiswa[i].hobi[2] + "  ");
         System.out.println(biodataMahasiswa[i].ipk);
      }
      System.out.println("--------------------------------------------");

    }

    //-------------------------------------------------
    //---             Program Utama                   ---
    //-------------------------------------------------
    public static void main(String[] args)
    {  //bagian deklarasi record berbasis LARIK -----------------------
      formatBiodata    biodataMahasiswa[] = new formatBiodata[10];
      biodataMahasiswa[0] = new formatBiodata();
      biodataMahasiswa[1] = new formatBiodata();
      biodataMahasiswa[2] = new formatBiodata();
      biodataMahasiswa[3] = new formatBiodata();
      biodataMahasiswa[4] = new formatBiodata();
      ngentriData(biodataMahasiswa);
      tampilkanData(biodataMahasiswa);
    }
}

Hasil Eksekusi :
-----------------------------------------------------------------------
   NAMA      ALAMAT       UMUR JEKEL  HOBI[0]  HOBI[1]  HOBI[2]  IPK
-----------------------------------------------------------------------
0. AgungBP   Jakarta      28   L      musik    mancing  touring  3.5
1. Rulieta   KualaKapuas  17   P      nari     rajut    jahit    3.0
2. Karti     Palangkaraya 18   P      renang   fbkan    masak    2.5
3. Hermon    Banjarmasin  25   L      kasti    bola     mancing  3.4
4. Anis      Surakarta    30   P      jualan   drama    nonton   0.3
5. Yanti     Purwokerto   21   P      lukis    musik    selfi    2.0
6. Handono   Bekasi       29   L      touring  musik    hiking   3.1
7. Anton     Jakarta      17   L      dangdut  jajan    turing   2.0
8. Mantri    Magelang     45   L      hiphop   wayang   game     3.1
9. Satrio    Yogyakarta   30   L      musik    drama    karate   3.3
-----------------------------------------------------------------------
Press any key to continue . . .
```

Program 2.4

PRAKTIK

1. Praktek 1

Tuliskan dan eksekusilah program *2.1* hingga *2.4.* Catat hasilnya.

2. Praktek 2

Modifikasilah program *2.4.* agar dapat digunakan untuk mencatat seluruh data pada gambar 2.3 (Nilai N ditentukan secara statis sebanyak 10 orang, dengan data yang telah ditentukan).

LATIHAN

1. Modifikasilah program 2.4 agar dapat digunakan untuk memasukkan data dengan banyak record (N) **dinamis**. Banyak record (N) dientri oleh user **melalui keyboard** dengan maksimum 15 record

TUGAS

1. Modifikasilah program 2.4 agar dapat digunakan untuk memasukkan data dengan banyak record (N) dinamis. Banyak record (N) akan bertambah secara otomatis (++) apabila user menghendaki memasukkan data lagi. Maksimum record 20.
2. Buat program tentang biodata mahasiswa dengan field-field Nama, NoMhs, Nilai UTS dan Nilai UAS yang di tampilkan dalam menu sebagai berikut

Menu

1. Input
2. View
3. Exit

3.

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 12-24, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 3
PENGELOLAAN DATA PADA ARRAY/LARIK: PENAMBAHAN DAN PENGHAPUSAN DATA

CAPAIANPEMBELAJARAN

Mahasiswa dapat menambah data baru ke dalam larik dan dapat menghapus data tertentu dari dalam larik

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Menambah Data Baru Ke Dalam Larik

Menambah data baru ke dalam larik dapat dilakukan pada bagian depan, bagian tengah, dan bagian belakang.

Pada bagian ini akan dicontohkan data seperti tabel berikut ini.

	.nama	*.alamat*	*.umur*	*.jekel*	*.hobi [ ]*	*.IPK*
biodataMahasiswa [0]	AgungBP	Jakarta	28	L	*.hobi[0]* musik *.hobi[1]* mancing *.hobi[2]* touring	3.5
biodataMahasiswa [1]	Rulieta	KualaKapuas	17	P	*.hobi[0]* tari *.hobi[1]* rajut *.hobi[2]* jahit	3
biodataMahasiswa [2]	Kayra	Yogyakarta	15	P	*.hobi[0]* ballet *.hobi[1]* renang *.hobi[2]* taekwondo	3.4
biodataMahasiswa [3]	Elnathan	Yogyakarta	13	L	*.hobi[0]* renang *.hobi[1]* main *.hobi[2]* youtube	3.4
biodataMahasiswa [4]	Niken	Magelang	35	P	*.hobi[0]* jualan *.hobi[1]* menanam *.hobi[2]* arisan	2.5

biodataMahasiswa [5]	Liwin	Palangkaraya	35	P	***.hobi[0]*** memasak ***.hobi[1]*** menanam ***.hobi[2]*** berkebun	2.7
biodataMahasiswa [6]	Satrio	Semarang	26	L	***.hobi[0]*** karate ***.hobi[1]*** band ***.hobi[2]*** driving	3.1
biodataMahasiswa [7]	Dion	Bantul	22	L	***.hobi[0]*** makelaran ***.hobi[1]*** berdagang ***.hobi[2]*** karaoke	2.9
biodataMahasiswa [8]	Fifin	Purwokerto	32	P	***.hobi[0]*** mengajar ***.hobi[1]*** menanam ***.hobi[2]*** bernyanyi	3.1
biodataMahasiswa [9]	Hermon	Banjarmasin	26	L	***.hobi[0]*** bertukang ***.hobi[1]*** menggambar ***.hobi[2]*** bertualang	3.3

Algoritma Penambahan data di depan

Penambahan data di bagian depan dilakukan dengan menciptakan ruang kosong pada larik paling atas (larik ke-0) dan memasukkan data baru pada ruang kosong tersebut.

Proses ini harus didahului dengan proses penggeseran data secara berturut-turut mulai dari data ke-9 (terakhir) sampai dengan data ke-0 sebanyak satu langkah ke bawah sehingga akan menciptakan ruang kosong pada larik bagian ke-0, sementara data yang bergeser akan menempati larik ke-1 hingga larik ke-10. Setelah proses penggeseran data selesai nilai N (banyaknya data) harus tambah dengan 1.

	.nama	***.alamat***	***.umur***	***.jekel***	***.hobi [ ]***	***.IPK***
biodataMahasiswa [0]					***.hobi[0]*** ***.hobi[1]*** ***.hobi[2]***	
biodataMahasiswa [1]	AgungBP	Jakarta	28	L	***.hobi[0]*** musik ***.hobi[1]*** mancing ***.hobi[2]*** touring	3.5
biodataMahasiswa [2]	Rulieta	KualaKapuas	17	P	***.hobi[0]*** tari ***.hobi[1]*** rajut ***.hobi[2]*** jahit	3
biodataMahasiswa [3]	Kayra	Yogyakarta	15	P	***.hobi[0]*** ballet ***.hobi[1]*** renang ***.hobi[2]*** taekwondo	3.4
biodataMahasiswa [4]	Elnathan	Yogyakarta	13	L	***.hobi[0]*** renang ***.hobi[1]*** main ***.hobi[2]*** youtube	3.4
biodataMahasiswa [5]	Niken	Magelang	35	P	***.hobi[0]*** jualan ***.hobi[1]*** menanam ***.hobi[2]*** arisan	2.5
biodataMahasiswa [6]	Liwin	Palangkaraya	35	P	***.hobi[0]*** memasak ***.hobi[1]*** menanam ***.hobi[2]*** berkebun	2.7
biodataMahasiswa [7]	Satrio	Semarang	26	L	***.hobi[0]*** karate ***.hobi[1]*** band ***.hobi[2]*** driving	3.1
biodataMahasiswa [8]	Dion	Bantul	22	L	***.hobi[0]*** makelaran ***.hobi[1]*** berdagang ***.hobi[2]*** karaoke	2.9
biodataMahasiswa [9]	Fifin	Purwokerto	32	P	***.hobi[0]*** mengajar ***.hobi[1]*** menanam ***.hobi[2]*** bernyanyi	3.1
biodataMahasiswa [10]	Hermon	Banjarmasin	26	L	***.hobi[0]*** bertukang ***.hobi[1]*** menggambar ***.hobi[2]*** bertualang	3.3

ruang kosong pada larik ke-0, yg terbentuk setelah semua data digeser turun ke bawah

kondisi data setelah mengalami pergeseran turun 1 langkah ke bawah

Untuk mengimplementasikan proses penambahan data baru di bagian depan anda dapat memperhatikan program 3.1 berikut ini.

```
//-------------------------------------------------
//---     Fungsi untuk Menambah Data Di Depan    ---
//-------------------------------------------------
public static void tambahDataDiDepan(formatBiodata biodataMahasiswa[])
{
    //bagian membuat record sementara untuk menampung data baru-------------
    formatBiodata biodataMahasiswaBaru = new formatBiodata();

    //bagian entri data baru ke penyimpan sementara-----------------------
    Scanner masukan = new Scanner(System.in);
    int bacaTombol=0;
    System.out.print("Silakan masukkan nama anda : ");
    biodataMahasiswaBaru.nama = masukan.next();
    System.out.print("Silakan masukkan alamat anda : ");
    biodataMahasiswaBaru.alamat = masukan.next();
    System.out.print("Silakan masukkan umur anda : ");
    biodataMahasiswaBaru.umur = masukan.nextInt();


    System.out.print("Silakan masukkan Jenis Kelamin anda : ");
        try
        {   bacaTombol = System.in.read();
        }
        catch(java.io.IOException e)
        {
        }
    biodataMahasiswaBaru.jekel = (char)bacaTombol;

    System.out.println("Silakan masukkan hobi (maks 3) : ");
    System.out.print("hobi ke-0 : ");
    biodataMahasiswaBaru.hobi[0] = masukan.next();
    System.out.print("hobi ke-1 : ");
    biodataMahasiswaBaru.hobi[1] = masukan.next();
    System.out.print("hobi ke-2 : ");
    biodataMahasiswaBaru.hobi[2] = masukan.next();

    System.out.print("Silakan masukkan IPK anda : ");
    biodataMahasiswaBaru.ipk = masukan.nextFloat();

    //bagian menggeser isi larik mulai dari Belakang s/d 0 selangkah ke bawah
    for (int i=N-1;   i>= 0;  i--)
    {   biodataMahasiswa[i+1] = biodataMahasiswa[i];
    }

    //bagian memindahkan data baru ke larik ke-0----------------------
    biodataMahasiswa[0] = biodataMahasiswaBaru;

    //memperbaharui banyaknya data (N), banyaknya data bertambah satu------
    N++;
}
```

Program 3.1

Algoritma Penambahan data di tengah

Penambahan data di bagian tengah dilakukan dengan menciptakan ruang kosong pada larik ke-T di mana T adalah posisi target kemudian memasukkan data baru pada ruang kosong tersebut.

Proses ini harus didahului dengan proses penggeseran data secara berturut-turut mulai dari data ke-9 (terakhir) sampai dengan data ke-T sebanyak satu langkah ke bawah sehingga akan menciptakan ruang kosong pada larik ke-T, sementara data yang

bergeser akan menempati larik ke-T+1 hingga larik ke-10. Setelah proses penggeseran data selesai nilai N (banyaknya data) harus tambah dengan 1.

	.nama	.alamat	.umur	.jekel	.hobi [ ]	.IPK
biodataMahasiswa [0]	AgungBP	Jakarta	28	L	.hobi[0] musik .hobi[1] mancing .hobi[2] touring	3.5
biodataMahasiswa [1]	Rulieta	KualaKapuas	17	P	.hobi[0] tari .hobi[1] rajut .hobi[2] jahit	3
biodataMahasiswa [2]	Kayra	Yogyakarta	15	P	.hobi[0] ballet .hobi[1] renang .hobi[2] taekwondo	3.4
biodataMahasiswa [3]	Elnathan	Yogyakarta	13	L	.hobi[0] renang .hobi[1] main .hobi[2] youtube	3.4
biodataMahasiswa [4]					.hobi[0] .hobi[1] .hobi[2]	
biodataMahasiswa [5]	Niken	Magelang	35	P	.hobi[0] jualan .hobi[1] menanam .hobi[2] arisan	2.5
biodataMahasiswa [6]	Liwin	Palangkaraya	35	P	.hobi[0] memasak .hobi[1] menanam .hobi[2] berkebun	2.7
biodataMahasiswa [7]	Satrio	Semarang	26	L	.hobi[0] karate .hobi[1] band .hobi[2] driving	3.1
biodataMahasiswa [8]	Dion	Bantul	22	L	.hobi[0] makelaran .hobi[1] berdagang .hobi[2] karaoke	2.9
biodataMahasiswa [9]	Fifin	Purwokerto	32	P	.hobi[0] mengajar .hobi[1] menanam .hobi[2] bernyanyi	3.1
biodataMahasiswa [10]	Hermon	Banjarmasin	26	L	.hobi[0] bertukang .hobi[1] menggambar .hobi[2] bertualang	3.3

kondisi data yang tetap, tidak mengalami pergeseran

ruang kosong pada larik ke-T, yg terbentuk setelah sebagian data digeser turun ke bawah

kondisi data setelah mengalami pergeseran turun 1 langkah ke bawah

Untuk mengimplementasikan proses penambahan data baru di bagian tengah anda dapat memperhatikan program 3.2 berikut ini.

```
//-------------------------------------------------
//---     Fungsi untuk Menambah Data Di Tengah    ---
//-------------------------------------------------
public static void tambahDataDiTengah(formatBiodata biodataMahasiswa[])
{
    //bagian membuat record sementara untuk menampung data baru-----------
    formatBiodata biodataMahasiswaBaru = new formatBiodata();

    //bagian entri data baru ke penyimpan sementara-----------------------
    Scanner masukan = new Scanner(System.in);
    int bacaTombol=0;
    System.out.print("Silakan masukkan nama anda : ");
    biodataMahasiswaBaru.nama = masukan.next();
    System.out.print("Silakan masukkan alamat anda : ");
    biodataMahasiswaBaru.alamat = masukan.next();
    System.out.print("Silakan masukkan umur anda : ");
    biodataMahasiswaBaru.umur = masukan.nextInt();
    System.out.print("Silakan masukkan Jenis Kelamin anda : ");
        try
        {   bacaTombol = System.in.read();
        }
        catch(java.io.IOException e)
        {
```

```
            }
        biodataMahasiswaBaru.jekel = (char)bacaTombol;

        System.out.println("Silakan masukkan hobi (maks 3) : ");
        System.out.print("hobi ke-0 : ");
        biodataMahasiswaBaru.hobi[0] = masukan.next();
        System.out.print("hobi ke-1 : ");
        biodataMahasiswaBaru.hobi[1] = masukan.next();
        System.out.print("hobi ke-2 : ");
        biodataMahasiswaBaru.hobi[2] = masukan.next();

        System.out.print("Silakan masukkan IPK anda : ");
        biodataMahasiswaBaru.ipk = masukan.nextFloat();

        //bagian menentukan posisi target T ----------------------------------
        int T;
        System.out.print("Pada posisi ke berapa data akan dimasukkan ? : ");
        T = masukan.nextInt();

        //bagian menggeser isi larik mulai dari Belakang s/d T selangkah ke belakang
        for (int i=N-1;   i>= T;  i--)
        {   biodataMahasiswa[i+1] = biodataMahasiswa[i];
        }


        //bagian memindahkan data baru ke larik ke-T----------------------
        biodataMahasiswa[T] = biodataMahasiswaBaru;

        //memperbaharui banyaknya data (N), banyaknya data bertambah satu-------
        N++;
    }
```

Program 3.2

Algoritma Penambahan data di belakang

Penambahan data di belakang dilakukan dengan memasukkan data yang baru pada larik ke-N di mana N adalah posisi paling akhir. Pada proses menambah data di belakang, kita tidak perlu melakukan penggeseran data yang telah ada dalam larik melainkan cukup dengan menaikkan nilai N nya saja (banyaknya data) dengan 1.

	.nama	.alamat	.umur	.jekel	.hobi [ ]		.IPK
biodataMahasiswa [0]	AgungBP	Jakarta	28	L	.hobi[0]	musik	3.5
					.hobi[1]	mancing	
					.hobi[2]	touring	
biodataMahasiswa [1]	Rulieta	KualaKapuas	17	P	.hobi[0]	tari	3
					.hobi[1]	rajut	
					.hobi[2]	jahit	
biodataMahasiswa [2]	Kayra	Yogyakarta	15	P	.hobi[0]	ballet	3.4
					.hobi[1]	renang	
					.hobi[2]	taekwondo	
biodataMahasiswa [3]	Elnathan	Yogyakarta	13	L	.hobi[0]	renang	3.4
					.hobi[1]	main	
					.hobi[2]	youtube	
biodataMahasiswa [4]	Niken	Magelang	35	P	.hobi[0]	jualan	2.5
					.hobi[1]	menanam	
					.hobi[2]	arisan	
biodataMahasiswa [5]	Liwin	Palangkaraya	35	P	.hobi[0]	memasak	2.7
					.hobi[1]	menanam	
					.hobi[2]	berkebun	
biodataMahasiswa [6]	Satrio	Semarang	26	L	.hobi[0]	karate	3.1
					.hobi[1]	band	
					.hobi[2]	driving	
biodataMahasiswa [7]	Dion	Bantul	22	L	.hobi[0]	makelaran	2.9
					.hobi[1]	berdagang	
					.hobi[2]	karaoke	
biodataMahasiswa [8]	Fifin	Purwokerto	32	P	.hobi[0]	mengajar	3.1
					.hobi[1]	menanam	
					.hobi[2]	bernyanyi	
biodataMahasiswa [9]	Hermon	Banjarmasin	26	L	.hobi[0]	bertukang	3.3
					.hobi[1]	menggambar	
					.hobi[2]	bertualang	
biodataMahasiswa [10]					.hobi[0]		
					.hobi[1]		
					.hobi[2]		

kondisi data yang tetap, tidak mengalami pergeseran

ruang kosong di larik ke-N, belum pernah ditempati oleh siapapun

Untuk mengimplementasikan proses penambahan data baru di bagian belakang anda dapat memperhatikan program 3.3 berikut ini.

```
//-------------------------------------------------
//---  Fungsi untuk Menambah Data Di Belakang    ---
//-------------------------------------------------
public static void tambahDataDiBelakang(formatBiodata biodataMahasiswa[])
{
    //bagian membuat record sementara untuk menampung data baru-------------
    formatBiodata biodataMahasiswaBaru = new formatBiodata();

    //bagian entri data baru ke penyimpan sementara-----------------------
    Scanner masukan = new Scanner(System.in);
    int bacaTombol=0;
    System.out.print("Silakan masukkan nama anda : ");
    biodataMahasiswaBaru.nama = masukan.next();
    System.out.print("Silakan masukkan alamat anda : ");
    biodataMahasiswaBaru.alamat = masukan.next();
    System.out.print("Silakan masukkan umur anda : ");
    biodataMahasiswaBaru.umur = masukan.nextInt();
    System.out.print("Silakan masukkan Jenis Kelamin anda : ");
        try
        {   bacaTombol = System.in.read();
        }
        catch(java.io.IOException e)
        {
        }
    biodataMahasiswaBaru.jekel = (char)bacaTombol;

    System.out.println("Silakan masukkan hobi (maks 3) : ");
    System.out.print("hobi ke-0 : ");
    biodataMahasiswaBaru.hobi[0] = masukan.next();
    System.out.print("hobi ke-1 : ");
    biodataMahasiswaBaru.hobi[1] = masukan.next();
```

```
        System.out.print("hobi ke-2 : ");
        biodataMahasiswaBaru.hobi[2] = masukan.next();

        System.out.print("Silakan masukkan IPK anda : ");
        biodataMahasiswaBaru.ipk = masukan.nextFloat();

        //bagian memindahkan data baru ke larik ke-N-----------------------
        biodataMahasiswa[N] = biodataMahasiswaBaru;

        //memperbaharui banyaknya data (N), banyaknya data bertambah satu----
        N++;
    }
```

Program 3.3

Menghapus Data

Menghapus data dari dalam larik dapat dilakukan di bagian depan, bagian tengah, dan bagian belakang.

Algoritma Menghapus data di depan

Menghapus data di bagian depan dilakukan dengan cara menimpa larik paling depan (larik ke-0) dengan larik ke-1, dilanjutkan dengan menimpa larik ke-1 dengan larik ke-2, ke-2 dengan ke-3, dan seterusnya. Proses ini sama dengan melakukan penggeseran secara berturut-turut data dimulai ke-0 sampai dengan data terakhir sebanyak satu langkah ke depan. Setelah proses penggeseran selesai nilai N (banyaknya data) akan dikurangi 1.

	.nama	.alamat	.umur	.jekel	.hobi [ ]	.IPK
biodataMahasiswa [0]	Ending	Banjarbaru	15	L	.hobi[0] desain .hobi[1] hiking .hobi[2] driving	2.5
biodataMahasiswa [1]	AgungBP	Jakarta	28	L	.hobi[0] musik .hobi[1] mancing .hobi[2] touring	3.5
biodataMahasiswa [2]	Rulieta	KualaKapuas	17	P	.hobi[0] tari .hobi[1] rajut .hobi[2] jahit	3
biodataMahasiswa [3]	Kayra	Yogyakarta	15	P	.hobi[0] ballet .hobi[1] renang .hobi[2] taekwondo	3.4
biodataMahasiswa [4]	Elnathan	Yogyakarta	13	L	.hobi[0] renang .hobi[1] main .hobi[2] youtube	3.4
biodataMahasiswa [5]	Niken	Magelang	35	P	.hobi[0] jualan .hobi[1] menanam .hobi[2] arisan	2.5
biodataMahasiswa [6]	Liwin	Palangkaraya	35	P	.hobi[0] memasak .hobi[1] menanam .hobi[2] berkebun	2.7
biodataMahasiswa [7]	Satrio	Semarang	26	L	.hobi[0] karate .hobi[1] band .hobi[2] driving	3.1
biodataMahasiswa [8]	Dion	Bantul	22	L	.hobi[0] makelaran .hobi[1] berdagang .hobi[2] karaoke	2.9
biodataMahasiswa [9]	Fifin	Purwokerto	32	P	.hobi[0] mengajar .hobi[1] menanam .hobi[2] bernyanyi	3.1
biodataMahasiswa [10]	Hermon	Banjarmasin	26	L	.hobi[0] bertukang .hobi[1] menggambar .hobi[2] bertualang	3.3

data yang akan dihapus pada larik ke-0 (akan ditimpa oleh data di bawahnya)

data-data yang harus digeser naik 1 langkah ke atas

Untuk mengimplementasikan proses penghapusan data di depan anda dapat memperhatikan program 3.4 berikut ini.

```
//-------------------------------------------------
//---     Fungsi untuk Menghapus Data Di Depan     ---
//-------------------------------------------------
public static void hapusDataDiDepan(formatBiodata biodataMahasiswa[])
{
    //bagian menggeser isi larik mulai dari 0 - Belakang selangkah ke depan
    for (int i=0;   i<= N-2;  i++)
    {   biodataMahasiswa[i] = biodataMahasiswa[i+1];
    }
    System.out.println("Proses menghapus data ke-0 selesai.");

    //memperbaharui banyaknya data (N), banyaknya data berkurang satu-------
    N--;
}
```

Program 3.4

Algoritma Menghapus data di tengah

Menghapus data di bagian tengah dilakukan dengan menentukan posisi target data yang ingin dihapus (T). Kemudian data yang akan dihapus pada larik ke-T ditimpa dengan larik ke-T+1, dilanjutkan dengan menimpa larik ke-T+1 dengan larik ke-T+2, larik ke-T+2 dengan ke-T+3, dan seterusnya. Proses ini sama dengan menggeser data mulai dari data ke-T+1 sampai dengan data terakhir sebanyak satu langkah ke atas.

	.nama	.alamat	.umur	.jekel	.hobi [ ]	.IPK
biodataMahasiswa [0]	AgungBP	Jakarta	28	L	.hobi[0] musik .hobi[1] mancing .hobi[2] touring	3.5
biodataMahasiswa [1]	Rulieta	KualaKapuas	17	P	.hobi[0] tari .hobi[1] rajut .hobi[2] jahit	3
biodataMahasiswa [2]	Kayra	Yogyakarta	15	P	.hobi[0] ballet .hobi[1] renang .hobi[2] taekwondo	3.4
biodataMahasiswa [3]	Elnathan	Yogyakarta	13	L	.hobi[0] renang .hobi[1] main .hobi[2] youtube	3.4
biodataMahasiswa [4]	***Ending***	***Banjarbaru***	***15***	***L***	.hobi[0] ***desain*** .hobi[1] ***hiking*** .hobi[2] ***driving***	***2.5***
biodataMahasiswa [5]	Niken	Magelang	35	P	.hobi[0] jualan .hobi[1] menanam .hobi[2] arisan	2.5
biodataMahasiswa [6]	Liwin	Palangkaraya	35	P	.hobi[0] memasak .hobi[1] menanam .hobi[2] berkebun	2.7
biodataMahasiswa [7]	Satrio	Semarang	26	L	.hobi[0] karate .hobi[1] band .hobi[2] driving	3.1
biodataMahasiswa [8]	Dion	Bantul	22	L	.hobi[0] makelaran .hobi[1] berdagang .hobi[2] karaoke	2.9
biodataMahasiswa [9]	Fifin	Purwokerto	32	P	.hobi[0] mengajar .hobi[1] menanam .hobi[2] bernyanyi	3.1
biodataMahasiswa [10]	Hermon	Banjarmasin	26	L	.hobi[0] bertukang .hobi[1] menggambar .hobi[2] bertualang	3.3

kondisi data yang tetap, tidak mengalami pergeseran

data yang akan dihapus pada larik ke-T (akan ditimpa oleh data di bawahnya)

data-data yang harus digeser naik 1 langkah ke atas

Untuk mengimplementasikan proses penghapusan data di tengah anda dapat memperhatikan program 3.5 berikut ini.

```
//-------------------------------------------------
//---    Fungsi untuk Menghapus Data Di Tengah   ---
//-------------------------------------------------
public static void hapusDataDiTengah(formatBiodata biodataMahasiswa[])
{
    //bagian menentukan posisi target T -------------------------------------
    Scanner masukan = new Scanner(System.in);
    int T;
    System.out.print("Tuliskan posisi data yang akan dibapus : ");
    T = masukan.nextInt();

    //bagian menggeser isi larik mulai dari T - Belakang selangkah ke depan
    for (int i=T;   i<= N-2;  i++)
    {   biodataMahasiswa[i] = biodataMahasiswa[i+1];
    }
    System.out.println("Proses menghapus data ke-" + T + " selesai.");

    //memperbaharui banyaknya data (N), banyaknya data berkurang satu-------
    N--;
}
```

Program 3.5

Algoritma Menghapus data di belakang

Menghapus data di bagian belakang dilakukan tidak perlu ada penggeseran data. Proses menghapus data di bagian belakang hanya dilakukan dengan memotong data terakhir dengan cara mengurangkan N dengan 1 (N--).

	.nama	.alamat	.umur	.jekel	.hobi [ ]	.IPK
biodataMahasiswa [0]	AgungBP	Jakarta	28	L	.hobi[0] musik .hobi[1] mancing .hobi[2] touring	3.5
biodataMahasiswa [1]	Rulieta	KualaKapuas	17	P	.hobi[0] tari .hobi[1] rajut .hobi[2] jahit	3
biodataMahasiswa [2]	Kayra	Yogyakarta	15	P	.hobi[0] ballet .hobi[1] renang .hobi[2] taekwondo	3.4
biodataMahasiswa [3]	Elnathan	Yogyakarta	13	L	.hobi[0] renang .hobi[1] main .hobi[2] youtube	3.4
biodataMahasiswa [4]	Niken	Magelang	35	P	.hobi[0] jualan .hobi[1] menanam .hobi[2] arisan	2.5
biodataMahasiswa [5]	Liwin	Palangkaraya	35	P	.hobi[0] memasak .hobi[1] menanam .hobi[2] berkebun	2.7
biodataMahasiswa [6]	Satrio	Semarang	26	L	.hobi[0] karate .hobi[1] band .hobi[2] driving	3.1
biodataMahasiswa [7]	Dion	Bantul	22	L	.hobi[0] makelaran .hobi[1] berdagang .hobi[2] karaoke	2.9
biodataMahasiswa [8]	Fifin	Purwokerto	32	P	.hobi[0] mengajar .hobi[1] menanam .hobi[2] bernyanyi	3.1
biodataMahasiswa [9]	Hermon	Banjarmasin	26	L	.hobi[0] bertukang .hobi[1] menggambar .hobi[2] bertualang	3.3
biodataMahasiswa [10]	***Ending***	***Banjarbaru***	***15***	***L***	.hobi[0] ***desain*** .hobi[1] ***hiking*** .hobi[2] ***driving***	***2.5***

kondisi data yang tetap, tidak mengalami pergeseran

data yang akan dihapus pada larik ke-N (dgn cara dipotong)

Untuk mengimplementasikan proses penghapusan data di belakang anda dapat memperhatikan program 3.6 berikut ini.

```
    //-------------------------------------------------
    //---   Fungsi untuk Menghapus Data Di Belakang  ---
    //-------------------------------------------------
    public static void hapusDataDiBelakang(formatBiodata biodataMahasiswa[])
    {
        System.out.println("Proses menghapus data paling akhir selesai.");

        //memperbaharui banyaknya data (N), banyaknya data berkurang satu-------
        N--;
    }
```

Program 3.6

Pada program 3.7 di bawah ini anda dapat melihat potongan/cuplikan program yang pernah anda tulis pada percobaan-percobaan sebelumnya. Selanjutnya anda dapat menambahkan fungsi-fungsi untuk menambah data dan juga menghapus data.

```
import java.util.Scanner;
class formatBiodata
{ //bagian deklarasi struktur record ----------------------------------
  String nama;
  String alamat;
  int    umur;
  char   jekel;
  String hobi[] = new String[3];
  float  ipk;
}

class menambahData
{ public static int N=0;

  //-------------------------------------------------
  //--- Fungsi untuk mengentri data ke dalam Larik ---
  //-------------------------------------------------
  public static void ngentriData(formatBiodata biodataMahasiswa[])
  { ....;
    ....;
  }

  //-------------------------------------------------
  //--- Fungsi untuk...                            ---
  //-------------------------------------------------
  public static void ......() //←fungsi tambah/hapus data di depan/tengah/belakang
  { ....;
    ....;
  }

  //-------------------------------------------------
  //---        Fungsi untuk Menampilkan Data       ---
  //-------------------------------------------------
  public static void tampilkanData(formatBiodata biodataMahasiswa[])
  {
    //bagian menampilkan isi struktur Larik -------
    System.out.println("-------------------------------------------------------");
    System.out.println("NAMA  ALAMAT UMUR  JEKEL  HOBI[0] HOBI[1] HOBI[2]   IPK");
    System.out.println("-------------------------------------------------------");
    for (int i=0;  i<=N-1;  i++)
    { System.out.print  (i +".");
      System.out.print  (biodataMahasiswa[i].nama + "\t");
      System.out.print  (biodataMahasiswa[i].alamat + "\t");
      System.out.print  (biodataMahasiswa[i].umur + "\t");
```

```
      System.out.print  (biodataMahasiswa[i].jekel + "\t");
      System.out.print  (biodataMahasiswa[i].hobi[0] + "\t");
      System.out.print  (biodataMahasiswa[i].hobi[1] + "\t");
      System.out.print  (biodataMahasiswa[i].hobi[2] + "\t");
      System.out.println(biodataMahasiswa[i].ipk);
    }
    System.out.println("-------------------------------------------------------");
  }

  //-----------------------------------------------
  //---              Program Utama              ---
  //-----------------------------------------------
  public static void main(String[] args)
  { //bagian deklarasi record berbasis LARIK -----------------------
    formatBiodata    biodataMahasiswa[] = new formatBiodata[10];
    biodataMahasiswa[0]  = new formatBiodata();
    biodataMahasiswa[1]  = new formatBiodata();
    biodataMahasiswa[2]  = new formatBiodata();
    biodataMahasiswa[3]  = new formatBiodata();
    ....;
    biodataMahasiswa[8] = new formatBiodata();
    biodataMahasiswa[9] = new formatBiodata();
    ....;
    ....;

    ngentriData(biodataMahasiswa);
    tampilkanData(biodataMahasiswa);
    ....; //← lakukan pemanggilan fungsi tambah dan hapus (depan/tengah/belakang)

    tampilkanData(biodataMahasiswa);
  }
}
```

Program 3.7

PRAKTIK

1. Praktek 1

Modifikasilah program yang telah anda miliki pada percobaan-percobaan sebelumnya, selanjutnya disebut master program (perhatikan program 3.7), dengan menambahkan potongan program 3.1.
Eksekusi dan ujilah program anda dengan menambahkan sebuah data baru di depan. Bagaimanakah hasilnya? Catatlah kondisi larik sebelum dilakukan penambahan data maupun setelah dilakukan penambahan data. Bandingkanlah hasilnya.

2. Prakter 2

Kembangkanlah lagi master program anda dengan menambahkan potongan program 3.2.

Eksekusi dan ujilah program anda dengan menambahkan sebuah data baru di tengah. Bagaimanakah hasilnya? Catatlah kondisi larik sebelum dilakukan penambahan data maupun setelah dilakukan penambahan data. Bandingkanlah hasilnya.

3. Praktek 3

Kembangkanlah lagi master program anda dengan menambahkan potongan program 3.3.
Eksekusi dan ujilah program dengan menambahkan sebuah data baru di belakang. Bagaimanakah hasilnya? Catatlah kondisi larik sebelum dilakukan penambahan data maupun setelah dilakukan penambahan data. Bandingkanlah hasilnya

4. Praktek 4

Kembangkanlah lagi master program anda dengan menambahkan potongan program 3.4, 3.5 dan 3.6.
Eksekusi dan ujilah program anda untuk menghapus sebuah data di depan/ tengah dan belakang larik.
Bagaimanakah hasilnya? Catatlah kondisi larik sebelum dilakukan penghapusan data maupun setelah dilakukan penghapusan. Bandingkanlah hasilnya.

LATIHAN

1. Kembangkan program yang telah anda buat, supaya dapat digunakan untuk menukarkan data. Contohnya Data pada record ke-4 ditukar dengan data pada record ke-7.

TUGAS

1. Buatlah sebuah fungsi untuk mengedit data pada record yang dipilih oleh user. Contohnya user ingin mengedit data yang ada pada record ke -4, dari yang semula bernama Niken alamat Magelang, jenis kelamin P, menjadi Sapto, alamat Magelang jenis kelamin L..

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 24-35, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 4
PENGELOLAAN DATA PADA ARRAY/ LARIK: PENCARIAN DATA (SEARCHING)

CAPAIANPEMBELAJARAN

Mahasiswa dapat melakukan pencarian terhadap suatu data yang terdapat di dalam larik

.

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Searching (Pencarian)

Searching (pencarian) adalah suatu metode pencarian sebuah data dalam sekumpulan besar data. Jika data yang dicari ditemukan maka program harus dapat memberikan informasi bahwa data yang dicari ditemukan, namun jika tidak berhasil ditemukan, program harus memberikan informasi bahwa data tersebut tidak ada didalam kumpulan data yang bersangkutan.

Ada beberapa proses pengelolaan data yang sangat bergantung pada proses pencarian ini antara lain proses pengeditan data dan proses penghapusan data berdasarkan kata kunci. Kedua proses ini baru dapat dilakukan apabila proses pencarian data berhasil. Pengeditan data baru dapat dilakukan apabila data yang hendak diedit ketemu. Demikian juga proses menghapus data baru dapat dilakukan jika data yang hendak dihapus ketemu.

Ada beberapa metode pencarian data yaitu :

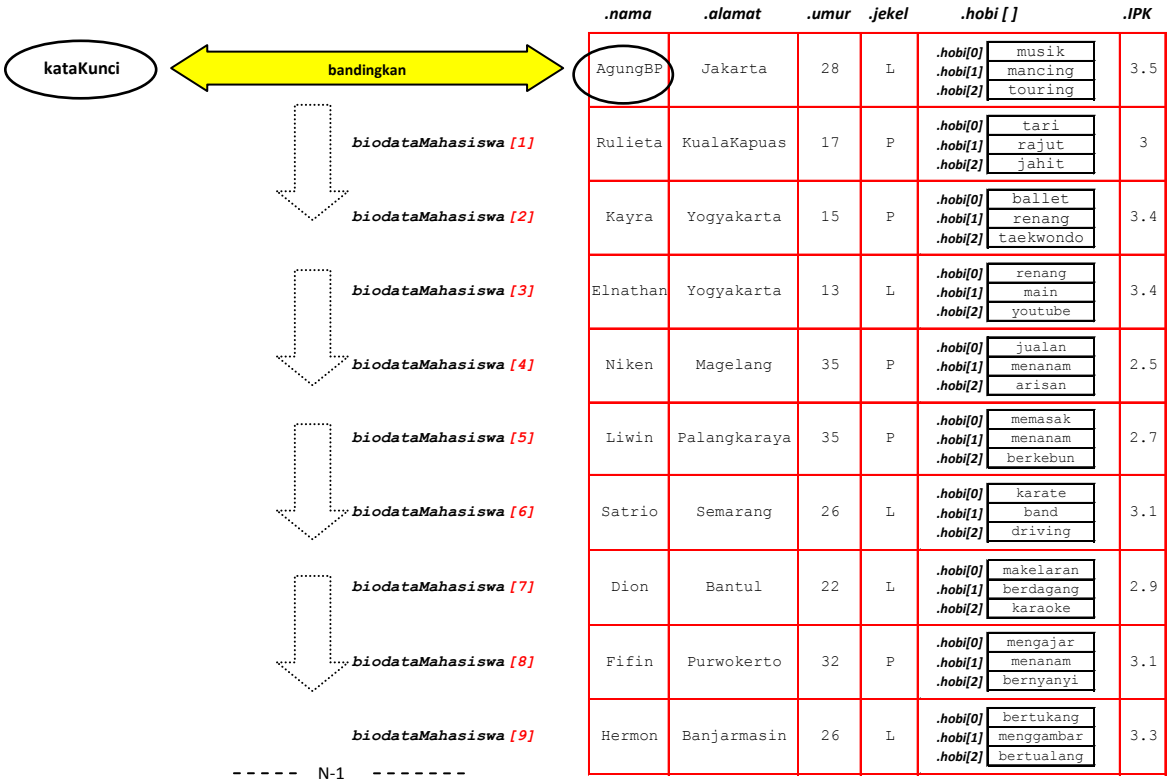
a. **Linear Search** (Pencarian Linear) atau **Sequential Search** (Pencarian Sekuensial)
b. **Binary Search** (Pencarian Biner)

.

A.Algoritma Linear Search (Pencarian Linier)

Linier Search atau **Sequential Search** adalah metode pencarian yang membandingkan kata kunci (data yang dicari) dengan setiap data yang ada dalam larik,

mulai dari data pertama hingga data terakhir**.** Pencarian linear dilakukan dengan mencocokkan kata kunci dengan data yang ada di dalam larik satu-persatu mulai dari data pertama (larik ke-0) hingga data terakhir (larik ke-N-1).



Gambar 4.1 Skema Pencarian Sekuensial

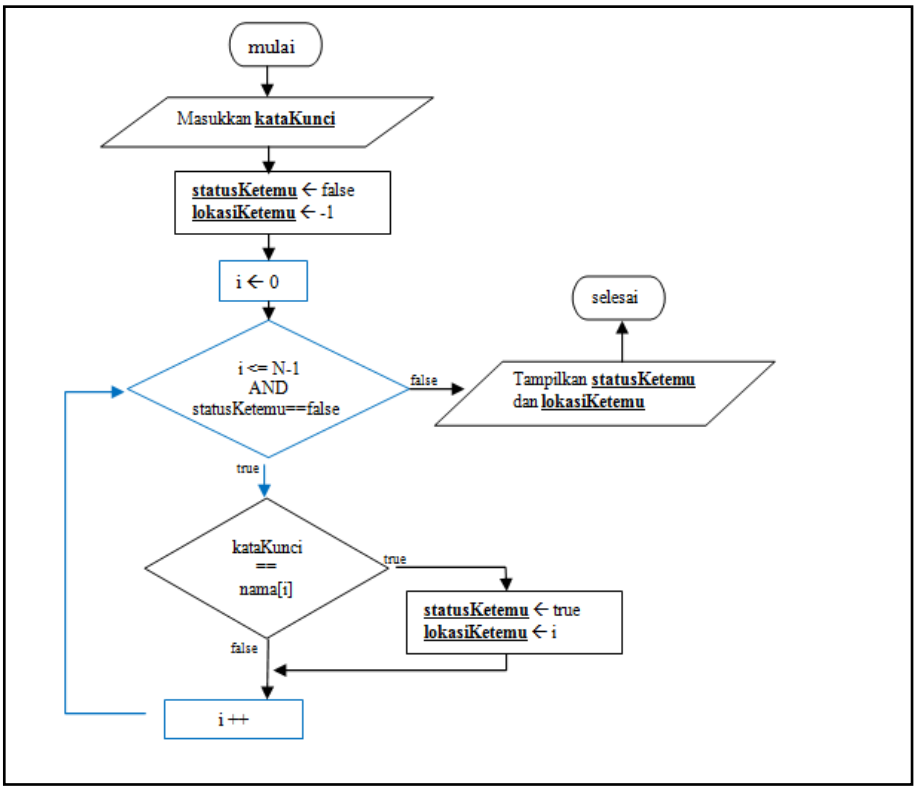
Proses dimulai dengan mencocokkan data larik ke-0 dengan kata kunci pencarian. Jika kata kunci tidak sesuai dengan data larik ke-0 proses pencarian dilanjutkan dengan mencocokkan kata kunci dengan data larik ke-1. Jika masih tidak sesuai proses dilanjutkan kembali degan mencocokkan kata kunci dengan data larik ke-2, ke-3 demikian seterusnya hingga hingga ditemukan data yang cocok.

Apabila dalam proses mencocokkan satu-persatu data ditemukan ada data yang sama dengan kata kunci pencarian maka posisi data tersebut dicatat dan proses dihentikan kemudian diberikan informasi bahwa data telah ditemukan pada posisi tersebut.

Lalu akan timbul pertanyaan jika proses dihentikan bagaimana jika di dalam larik terdapat data yang lainnya yang sama dengan kata kunci (lebih dari satu data yang sesuai)? Tidak ada aturan yang mengharuskan proses dihentikan atau diteruskan namun dalam praktikum ini kita asumsikan bahwa tidak ada data yang sama di dalam larik kita.

Pencarian linear bisa saja menghasilkan kemungkinan tidak ditemukan sama sekali data di dalam larik yang sesuai dengan kata kunci pencarian, bahkan setelah penelusuran data mencapai larik yang terakhir.

Langkah Pencarian Linear dalam praktikum ini mengikuti logika yang tergambar dalam flowchart gambar 4.2 berikut ini.



Gambar 4.2

Untuk membuat program pencarian linear berdasarkan flowchart di atas, lihatlah script program 4.1 dan 4.2 di bawah ini. Program 4.1 ditulis menggunakan perulangan While, sementara program 4.2 ditulis menggunakan perulangan For

```
//-------------------------------------------------------------------
//--- Fungsi untuk Mencari Data Secara Linear Search (Loop: While)---
//-------------------------------------------------------------------
public static void cariDataLinear (formatBiodata biodataMahasiswa[])
{
  Scanner masukan = new Scanner(System.in);

  //bagian   memasukkan   kata   kunci   ----------------------------------------
  System.out.print("Silakan masukkan kataKunci data yang anda cari :");
  String kataKunci = masukan.next();

  boolean statusKetemu = false;
  int lokasiKetemu = -1;

  //bagian mencari data satu persatu urut dari larik terdepan
  int i = 0;
  while ((i<=N-1) && (statusKetemu==false))
  {
    //mencocokkan biodataMahasiswa[i].nama == kataKunci
    if (kataKunci.equals(biodataMahasiswa[i].nama))
    {
      statusKetemu = true;
      lokasiKetemu = i;
    }
    i++;
```

```
  }
  System.out.println("Status   Ketemu:   "+statusKetemu+"   di   posisi   ke   "   +
lokasiKetemu);
}
```

Program 4.1

```
    //-----------------------------------------------------------------
    //--- Fungsi untuk Mencari Data Secara Linear Search (Loop: For) ---
    //-----------------------------------------------------------------
    public static void mencariDataLinear(formatBiodata biodataMahasiswa[])
    {   String kataKunci;
        int lokasi=-1;
        boolean statusKetemu=false;

        //bagian memasukkan kata kunci ----------------------------------------
        Scanner masukan = new Scanner(System.in);
        System.out.print("Masukkan kata kunci pencarian : ");
        kataKunci = masukan.next();

        //bagian mencari data satu persatu urut dari larik terdepan
        for (int i=0;   i<= N-1;  i++)
        {
            //mencocokkan biodataMahasiswa[i].nama == kataKunci
            if (biodataMahasiswa[i].nama.equals(kataKunci))
            {   statusKetemu = true;
                lokasi = i;
                break;
            }
        }

        if (statusKetemu == true)
        {   System.out.println("Data yang anda cari KETEMU di larik ke :"+ lokasi);
        }
        else
        {   System.out.println("maap, nama yang anda cari tidak ditemukan");
        }
    }
```

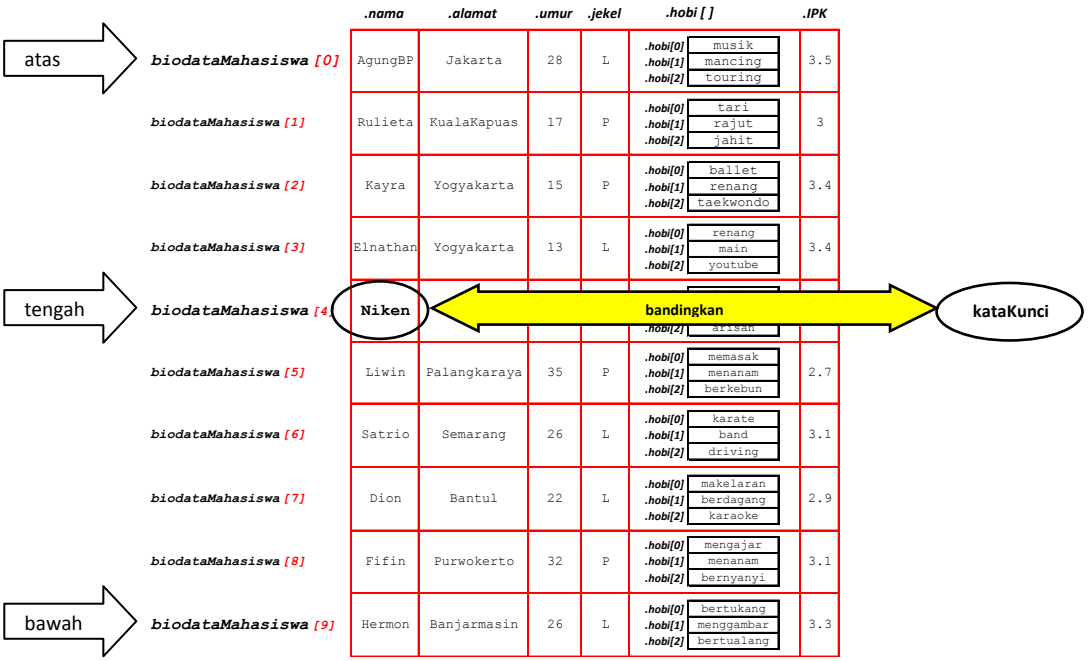
Program 4.2

B. Binary Searh (Pencarian Biner)

Binary Search adalah pencarian terhadap data yang sudah terurut. Data kunci dibandingkan dengan target yaitu data yang berada di tengah larik (data pivot). Jika kata kunci sama dengan target maka data ditemukan dan pencarian dihentikan.

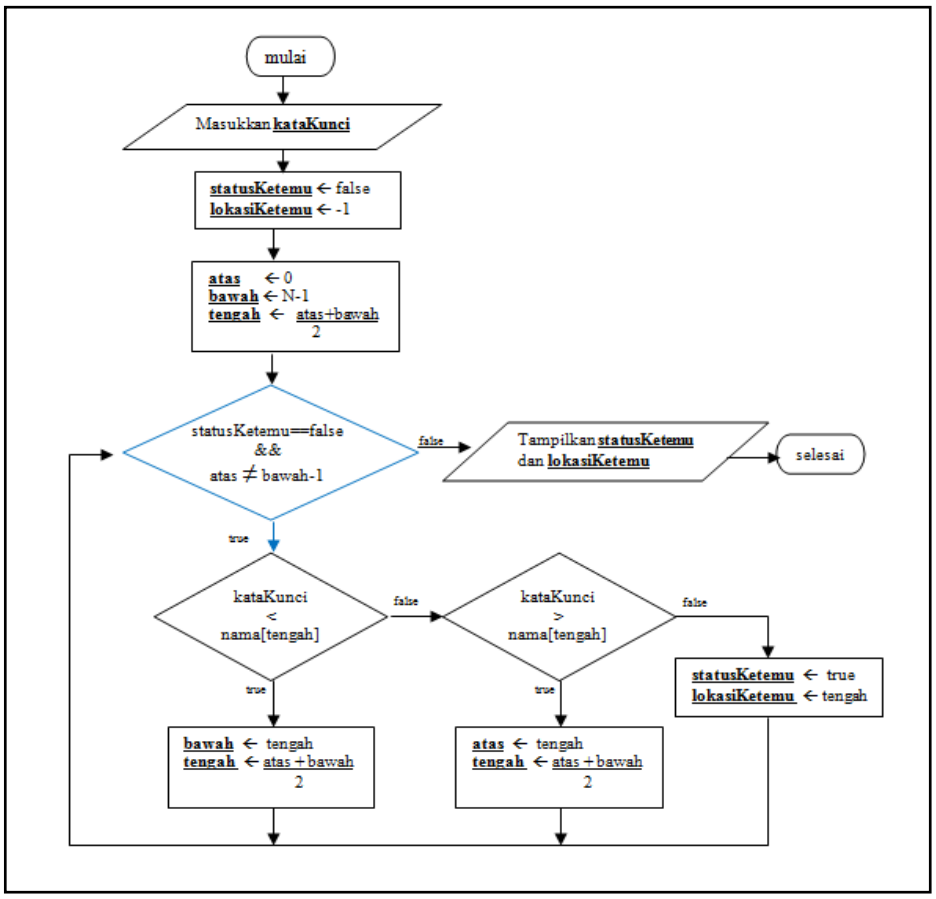
Jika kata kunci ternyata lebih besar daripada data di tengah berarti diprediksi bahwa target berada di potongan larik sebelah kanan sehingga fokus pecarian akan dilakukan disebelah kanan dari data tengah dengan cara mengubah tengah menjadi kiri kemudian nilai tengah dicari kembali. Hal ini berlaku juga sebaliknya jika kata kunci ternyata lebih kecil dari data tengah. Jika belum ditemukan maka data pada sisi pencarian akan dibagi dua kembali, dibandingkan lagi, demikian seterusnya sampai data ditemukan/ tidak ditemukan.

Kelemahan dari metode Binary search ini adalah untuk dapat melakukan pengurutan data **harus dalam keadaan urut**. Pada data yang belum urut pencarian biner tidak dapat dilakukan.



Gambar 4.3 Skema Pencarian Biner

Langkah Pencarian Biner dilakukan dengan mengikuti logika yang tergambar dalam flowchart gambar 4.3 berikut ini.



Gambar 4.4

Program 4.3 berikut menyajikan fungsi program untuk mencari data di dalam larik secara biner.

```
//-----------------------------------------------------
//--- Fungsi untuk Mencari Data Secara Binary Search ---
//-----------------------------------------------------
public static void mencariDataBiner(formatBiodata biodataMahasiswa[])
{   String kataKunci;
    int lokasi=-1;
    boolean statusKetemu=false;

    //bagian memasukkan kata kunci ----------------------------------------
    Scanner masukan = new Scanner(System.in);
    System.out.print("Masukkan kata kunci pencarian : ");
    kataKunci = masukan.next();

    //bagian mencari data satu persatu urut dari larik terdepan
    int atas,bawah,tengah;
    atas = 0;
    bawah = N-1;
    tengah = (atas + bawah) / 2;
    while ((statusKetemu == false) && (bawah-atas!=1))
    {   System.out.println(biodataMahasiswa[tengah].nama+" <---> "+kataKunci );

        //jika kataKunci < biodataMahasiswa[tengah].nama)
        if  (kataKunci.compareTo(biodataMahasiswa[tengah].nama) < 0)
        {   bawah = tengah;
            tengah = (atas + bawah) / 2;
        }
```

```
            //jika kataKunci > biodataMahasiswa[tengah].nama)
            else if  (kataKunci.compareTo(biodataMahasiswa[tengah].nama) > 0)
            {   atas = tengah;
                tengah = (atas + bawah) / 2;
            }
            else
            {   statusKetemu = true;
                lokasi = tengah;
            }
        }
        if (statusKetemu == true)
        {   System.out.println("Data yang anda cari KETEMU di larik ke :"+ lokasi);
        }
        else
        {   System.out.println("maap, nama yang anda cari tidak ditemukan ");
        }
    }
```

Program 4.3

PRAKTIK

1. Praktek 1

Tambahkan program 4.1 atau 4.2 ke dalam master program yang telah anda buat pada praktikum yang lalu. Eksekusi master program anda untuk melakukan pencarian data secara linear.

Ujilah program dengan mencari data yang memang ada di dalam larik (akan mengasilkan informasi bahwa data berhasil ditemukan), maupun untuk menari data yang memang tidak ada di dalam larik (akan menghasilkan informasi bahwa data yang dicari tidak ditemukan). Bagaimana hasilnya? Catat dan simpulkan dalam laporan anda.

2. Praktek 2

Tambahkan program 4.3 ke dalam master program yang telah anda. Eksekusi master program anda untuk melakukan pencarian data secara biner.

Pengujian pertama dilakukan terhadap data larik yang tidak dalam kondisi urut. Dapatkan pencarian biner dilakukan?

Pengujian kedua dilakukan terhadap data larik yang urut (data dientri ke dalam larik dalam keadaan urut). Ujilah program dengan mencari data yang memang ada di dalam larik (akan mengasilkan informasi bahwa data berhasil ditemukan), maupun untuk menari data yang memang tidak ada di dalam larik (akan menghasilkan informasi bahwa data yang dicari tidak ditemukan). Bagaimana hasilnya? Catat dan simpulkan dalam laporan anda.

LATIHAN

1. Buatlah sebuah fungsi untuk menampilkan data dengan syarat tertentu, Contohnya Tampilkan data yang mahasiswa yang berjenis kelamin L saja.

TUGAS

1. Buatlah suatu fungsi untuk menghapus data di mana data yang akan dihapus harus dicari dahulu. Apabila data ditemukan (bisa saja data ditemukan pada larik bagian depan, tengah ataupun belakang) maka data tersebut langsung dihapus. Apabila data tidak ditemukan maka proses menghapus tidak jadi dilakukan.

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 36-42, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 5
PENGELOLAAN DATA PADA ARRAY/ LARIK: PENGURUTAN (SORTING)

CAPAIANPEMBELAJARAN

Mahasiswa dapat melakukan pengurutan terhadap suatu data yang terdapat di dalam larik

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Sorting (Pengurutan)

Sorting (pengurutan) merupakan konsep/ algoritma yang sering digunakan. Jika berhubungan dengan jumlah data yang besar, data akan mudah dikelola jika sudah terurut. Algoritma pengurutan melibatkan beberapa konsep-konsep dasar pemograman, yaitu:

- Urutan/Sequence. Kaidah pemrograman yang menyatakan bahwa perintah-perintah dalam program komputer akan dieksekusi menurut urutannya dari atas ke bawah
- Seleksi/Selection. Perintah-perintah dalam program komputer akan dieksekusi berdasarkan nilai kebenaran boolean tertentu.
- Perulangan/Loop. Sejumlah perintah dalam program komputer yang akan dieksekusi beberapa kali berdasarkan nilai kebenaran boolean-nya.

Sorting dapat dibuat dengan dua model, yatu Ascending dan Descending. Ascending yaitu melakukan pengurutan dari data yang terkecil ke data yang lebih besar. Contoh : 1, 2, 3, . . . ,10.
Descending yaitu melakukan pengurutan dari data yang terbesar ke data yang lebih kecil. Contoh: 10, 9, 8, . . . , 1.

Ada beberapa metode sorting, misalnya :

- Bubble Sort
- Selection Sort
- Insertion Sort
- Quick Sort
- MergeSort
- dan lainnya

Pada praktikum ini kita akan mempelajari teknik sorting **bubble sort, selection Sort** dan **Insertion Sort**.

a. Algoritma Bubble Sort

Algoritma bubble sort akan membandingkan sebuah elemen dengan elemen di sebelah kanannya. Jika elemen yang dibaca lebih besar dari elemen di sebelah kanannya maka kedua elemen tersebut harus ditukar (untuk kasus Ascending). Proses ”banding-tukar” ini akan dilakukan mulai dari elemen pertama (data paling kiri) hingga elemen terakhir (data paling kanan) sebagai satu buah siklus. Pada akhir siklus yang pertama akan diperoleh hasil berupa kondisi di mana data yang paling besar berada di posisi paling kanan.

Pada siklus berikutnya, proses ”banding-tukar” akan dilakukan lakukan lagi untuk mendapatkan data terbesar kedua untuk diletakkan di posisi kedua dari kanan. Kemudian siklus akan diulang kembali untuk ketiga kalinya, keempat kalinya, kelima kalinya demikian seterusnya hingga diperoleh data terbesar ketiga, keempat, kelima, dan seterusnya. Keberlangsungan siklus baru akan berhenti apabila seluruh data terurutkan.

Agar mudah dipahami akan disimulasikan contoh pengurutan data berikut ini.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
DATA AWAL	Rulieta	Hermon	Satrio	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Dengan metode bubblesort, dapat kita lihat proses pengurutan sebagaimana tersaji pada gambar berikut ini.

Siklus 1 (pada saat nilai *j=0*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**Rulieta**	**Hermon**	Satrio	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion
	[i] > [i+1], tukar									
Langkah 2 : Pada saat *i=1*	Hermon	**Rulieta**	**Satrio**	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion
		[i] < [i+1], tidak tukar								
Langkah 3 : Pada saat *i=2*	Hermon	Rulieta	**Satrio**	**AgungBP**	Niken	Fifin	Liwin	Kayra	Elnathan	Dion
			[i] > [i+1], tukar							
Langkah 4 : Pada saat *i=3*	Hermon	Rulieta	AgungBP	**Satrio**	**Niken**	Fifin	Liwin	Kayra	Elnathan	Dion
				[i] > [i+1], tukar						
Langkah 5 : Pada saat *i=4*	Hermon	Rulieta	AgungBP	Niken	**Satrio**	**Fifin**	Liwin	Kayra	Elnathan	Dion
					[i] > [i+1], tukar					
Langkah 6 : Pada saat *i=5*	Hermon	Rulieta	AgungBP	Niken	Fifin	**Satrio**	**Liwin**	Kayra	Elnathan	Dion
						[i] > [i+1], tukar				
Langkah 7 : Pada saat *i=6*	Hermon	Rulieta	AgungBP	Niken	Fifin	Liwin	**Satrio**	**Kayra**	Elnathan	Dion
							[i] > [i+1], tukar			
Langkah 8 : Pada saat *i=7*	Hermon	Rulieta	AgungBP	Niken	Fifin	Liwin	Kayra	**Satrio**	**Elnathan**	Dion
								[i] > [i+1], tukar		
Langkah 9 : Pada saat *i=8*	Hermon	Rulieta	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	**Satrio**	**Dion**
									[i] > [i+1], tukar	
Hasil :	Hermon	Rulieta	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion	**Satrio**

Siklus 2 (pada saat nilai *j=1*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**Hermon**	**Rulieta**	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion	*Satrio*
	[i] < [i+1], tidak tukar									
Langkah 2 : Pada saat *i=1*	Hermon	**Rulieta**	**AgungBP**	Niken	Fifin	Liwin	Kayra	Elnathan	Dion	*Satrio*
		[i] > [i+1], tukar								
Langkah 3 : Pada saat *i=2*	Hermon	AgungBP	**Rulieta**	**Niken**	Fifin	Liwin	Kayra	Elnathan	Dion	*Satrio*
			[i] > [i+1], tukar							
Langkah 4 : Pada saat *i=3*	Hermon	AgungBP	Niken	**Rulieta**	**Fifin**	Liwin	Kayra	Elnathan	Dion	*Satrio*
				[i] > [i+1], tukar						
Langkah 5 : Pada saat *i=4*	Hermon	AgungBP	Niken	Fifin	**Rulieta**	**Liwin**	Kayra	Elnathan	Dion	*Satrio*
					[i] > [i+1], tukar					
Langkah 6 : Pada saat *i=5*	Hermon	AgungBP	Niken	Fifin	Liwin	**Rulieta**	**Kayra**	Elnathan	Dion	*Satrio*
						[i] > [i+1], tukar				
Langkah 7 : Pada saat *i=6*	Hermon	AgungBP	Niken	Fifin	Liwin	Kayra	**Rulieta**	**Elnathan**	Dion	*Satrio*
							[i] > [i+1], tukar			
Langkah 8 : Pada saat *i=7*	Hermon	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	**Rulieta**	**Dion**	*Satrio*
								[i] > [i+1], tukar		
Hasil :	Hermon	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion	**Rulieta**	*Satrio*

Siklus 3 (pada saat nilai *j=2*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**Hermon**	**AgungBP**	Niken	Fifin	Liwin	Kayra	Elnathan	Dion	*Rulieta*	*Satrio*
		*[i] > [i+1], **tukar***								
Langkah 2 : Pada saat *i=1*	AgungBP	**Hermon**	**Niken**	Fifin	Liwin	Kayra	Elnathan	Dion	*Rulieta*	*Satrio*
			*[i] < [i+1], tidak **tukar***							
Langkah 3 : Pada saat *i=2*	AgungBP	Hermon	**Niken**	**Fifin**	Liwin	Kayra	Elnathan	Dion	*Rulieta*	*Satrio*
				*[i] > [i+1], **tukar***						
Langkah 4 : Pada saat *i=3*	AgungBP	Hermon	Fifin	**Niken**	**Liwin**	Kayra	Elnathan	Dion	*Rulieta*	*Satrio*
					*[i] > [i+1], **tukar***					
Langkah 5 : Pada saat *i=4*	AgungBP	Hermon	Fifin	Liwin	**Niken**	**Kayra**	Elnathan	Dion	*Rulieta*	*Satrio*
						*[i] > [i+1], **tukar***				
Langkah 6 : Pada saat *i=5*	AgungBP	Hermon	Fifin	Liwin	Kayra	**Niken**	**Elnathan**	Dion	*Rulieta*	*Satrio*
							*[i] > [i+1], **tukar***			
Langkah 7 : Pada saat *i=6*	AgungBP	Hermon	Fifin	Liwin	Kayra	Elnathan	**Niken**	**Dion**	*Rulieta*	*Satrio*
								*[i] > [i+1], **tukar***		
Hasil :	AgungBP	Hermon	Fifin	Liwin	Kayra	Elnathan	Dion	**Niken**	*Rulieta*	*Satrio*

Siklus 4 (pada saat nilai *j=3*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**AgungBP**	**Hermon**	Fifin	Liwin	Kayra	Elnathan	Dion	*Niken*	*Rulieta*	*Satrio*
		*[i] < [i+1], tidak **tukar***								
Langkah 2 : Pada saat *i=1*	AgungBP	**Hermon**	**Fifin**	Liwin	Kayra	Elnathan	Dion	*Niken*	*Rulieta*	*Satrio*
			*[i] > [i+1], **tukar***							
Langkah 3 : Pada saat *i=2*	AgungBP	Fifin	**Hermon**	**Liwin**	Kayra	Elnathan	Dion	*Niken*	*Rulieta*	*Satrio*
				*[i] < [i+1], tidak **tukar***						
Langkah 4 : Pada saat *i=3*	AgungBP	Fifin	Hermon	**Liwin**	**Kayra**	Elnathan	Dion	*Niken*	*Rulieta*	*Satrio*
					*[i] > [i+1], **tukar***					
Langkah 5 : Pada saat *i=4*	AgungBP	Fifin	Hermon	Kayra	**Liwin**	**Elnathan**	Dion	*Niken*	*Rulieta*	*Satrio*
						*[i] > [i+1], **tukar***				
Langkah 6 : Pada saat *i=5*	AgungBP	Fifin	Hermon	Kayra	Elnathan	**Liwin**	**Dion**	*Niken*	*Rulieta*	*Satrio*
							*[i] > [i+1], **tukar***			
Hasil :	AgungBP	Fifin	Hermon	Kayra	Elnathan	Dion	**Liwin**	*Niken*	*Rulieta*	*Satrio*

Siklus 5 (pada saat nilai *j=3*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**AgungBP**	**Fifin**	Hermon	Kayra	Elnathan	Dion	*Liwin*	*Niken*	*Rulieta*	*Satrio*
		*[i] < [i+1], tidak **tukar***								
Langkah 2 : Pada saat *i=1*	AgungBP	**Fifin**	**Hermon**	Kayra	Elnathan	Dion	*Liwin*	*Niken*	*Rulieta*	*Satrio*
			*[i] < [i+1], tidak **tukar***							
Langkah 3 : Pada saat *i=2*	AgungBP	Fifin	**Hermon**	**Kayra**	Elnathan	Dion	*Liwin*	*Niken*	*Rulieta*	*Satrio*
				*[i] < [i+1], tidak **tukar***						
Langkah 4 : Pada saat *i=3*	AgungBP	Fifin	Hermon	**Kayra**	**Elnathan**	Dion	*Liwin*	*Niken*	*Rulieta*	*Satrio*
					*[i] > [i+1], **tukar***					
Langkah 5 : Pada saat *i=4*	AgungBP	Fifin	Hermon	Elnathan	**Kayra**	**Dion**	*Liwin*	*Niken*	*Rulieta*	*Satrio*
						*[i] > [i+1], **tukar***				
Hasil :	AgungBP	Fifin	Hermon	Elnathan	Dion	**Kayra**	*Liwin*	*Niken*	*Rulieta*	*Satrio*

Siklus 6 (pada saat nilai *j=5*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**AgungBP**	**Fifin**	Hermon	Elnathan	Dion	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
		[i] < [i+1], tidak ***tukar***								
Langkah 2 : Pada saat *i=1*	AgungBP	**Fifin**	**Hermon**	Elnathan	Dion	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
			[i] < [i+1], tidak ***tukar***							
Langkah 3 : Pada saat *i=2*	AgungBP	Fifin	**Hermon**	**Elnathan**	Dion	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
				[i] > [i+1], ***tukar***						
Langkah 4 : Pada saat *i=3*	AgungBP	Fifin	Elnathan	**Hermon**	**Dion**	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
					[i] > [i+1], ***tukar***					
Hasil :	AgungBP	Fifin	Elnathan	Dion	**Hermon**	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*

Siklus 7 (pada saat nilai *j=6*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**AgungBP**	**Fifin**	Elnathan	Dion	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
		[i] < [i+1], tidak ***tukar***								
Langkah 2 : Pada saat *i=1*	AgungBP	**Fifin**	**Elnathan**	Dion	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
			[i] > [i+1], ***tukar***							
Langkah 3 : Pada saat *i=2*	AgungBP	Elnathan	**Fifin**	**Dion**	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
				[i] > [i+1], ***tukar***						
Hasil :	AgungBP	Elnathan	Dion	**Fifin**	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*

Siklus 8 (pada saat nilai *j=7*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**AgungBP**	**Elnathan**	Dion	*Fifin*	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
		[i] < [i+1], tidak ***tukar***								
Langkah 2 : Pada saat *i=1*	AgungBP	**Elnathan**	**Dion**	*Fifin*	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
			[i] > [i+1], ***tukar***							
Hasil :	AgungBP	Dion	**Elnathan**	*Fifin*	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*

Siklus 9 (pada saat nilai *j=8*)

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Langkah 1 : Pada saat *i=0*	**AgungBP**	**Dion**	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*
		[i] < [i+1], tidak ***tukar***								
Hasil :	AgungBP	**Dion**	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*

Hasil Akhir

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	*Liwin*	*Niken*	*Rulieta*	*Satrio*

Untuk mengimplementasikan teknik bubblesort, anda dapat memperhatikan program 5.1 berikut ini.

```
//-------------------------------------------------------
//---  Fungsi untuk Mengurutkan Data (BubbleSort)   ---
//-------------------------------------------------------
public static void mengurutkanDataBubble(formatBiodata
biodataMahasiswa[])
{
    formatBiodata biodataSementara = new formatBiodata();
    int indeksTerakhir = N-1;
    for (int j=0;j<=indeksTerakhir - 1; j++)
    {
```

```
        for (int i=0; i<=indeksTerakhir -1 -j;  i++)
        {
            // perintah dibawah ini identik dengan if (nama[i]>nama[i+1])
            if (biodataMahasiswa[i].nama.compareTo
                                (biodataMahasiswa[i+1].nama) > 0)
            {   biodataSementara = biodataMahasiswa[i];
                biodataMahasiswa[i]   = biodataMahasiswa[i+1];
                biodataMahasiswa[i+1] = biodataSementara;
            }
        }
    }
}
```

b. Algoritma Selection Sort

Algoritma selection sort akan membagi larik menjadi dua bagian. Bagian kiri akan menjadi data yang sudah urut sedangkan bagian kanan adalah bagian yang masih acak.

Proses pengurutan dilakukan dengan cara mengambil sebuah elemen kunci, kemudian membandingkan dengan elemen terkecil dari larik bagian kanan. Jika elemen kunci lebih besar dari elemen terkecil dari larik bagian kanan tersebut, tukarkan kedua elemen tersebut.

Proses dilanjutkan dengan mengambil sebuah elemen kunci yang baru, kemudian membandingkan lagi elemen kunci tersebut dengan elemen terkecil dari larik bagian kanan yang masih tersisa. Jika elemen kunci lebih besar dari elemen terkecil dari larik bagian kanan tersebut, lakukan kembali pertukarkan kedua elemen seperti langkah sebelumnya. Proses seperti ini harus diulang terus menerus hingga larik bagian kiri penuh dan larik bagian kanan habis.

Agar mudah dipahami akan disimulasikan contoh pengurutan data berikut ini.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
DATA AWAL	Rulieta	Hermon	Satrio	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Dengan metode selection sort, dapat kita lihat proses pengurutan sebagaimana tersaji pada gambar berikut ini.

LANGKAH 1

Rulieta	Hermon	Satrio	**AgungBP**	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

[0] <---bandingkan---> (terkecil)

Rulieta > AgungBP, tukarkan.

AgungBP	Hermon	Satrio	**Rulieta**	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

LANGKAH 2

AgungBP	**Hermon**	Satrio	Rulieta	Niken	Fifin	Liwin	Kayra	Elnathan	**Dion**

[1] <----------bandingkan----------> (terkecil)

Hermon > Dion, tukarkan.

AgungBP	**Dion**	Satrio	Rulieta	Niken	Fifin	Liwin	Kayra	Elnathan	**Hermon**

LANGKAH 3

AgungBP	*Dion*	**Satrio**	Rulieta	Niken	Fifin	Liwin	Kayra	**Elnathan**	Hermon

[2] <----------bandingkan----------> (terkecil)

Satrio > Elnathan, tukarkan.

AgungBP	*Dion*	**Elnathan**	Rulieta	Niken	Fifin	Liwin	Kayra	**Satrio**	Hermon

LANGKAH 4

AgungBP	*Dion*	*Elnathan*	**Rulieta**	Niken	**Fifin**	Liwin	Kayra	Satrio	Hermon

[3] <--------> (terkecil)

Rulieta > Fifin, tukarkan.

AgungBP	*Dion*	*Elnathan*	**Fifin**	Niken	**Rulieta**	Liwin	Kayra	Satrio	Hermon

LANGKAH 5

AgungBP	*Dion*	*Elnathan*	*Fifin*	**Niken**	Rulieta	Liwin	Kayra	Satrio	**Hermon**

[4] <----------bandingkan----------> (terkecil)

Niken > Hermon, tukarkan.

AgungBP	*Dion*	*Elnathan*	*Fifin*	**Hermon**	Rulieta	Liwin	Kayra	Satrio	**Niken**

LANGKAH 6

AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	**Rulieta**	Liwin	**Kayra**	Satrio	Niken

[4] <--------> (terkecil)

Rulieta > Kayra, tukarkan.

AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	**Kayra**	Liwin	**Rulieta**	Satrio	Niken

LANGKAH 7

AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	**Liwin**	Rulieta	Satrio	**Niken**

[5] <----------> (terkecil)

Liwin < Niken, TIDAK ditukar.

AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	**Liwin**	Rulieta	Satrio	**Niken**

LANGKAH 8

AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	*Liwin*	**Rulieta**	Satrio	**Niken**

[6] <---------> (terkecil)

Rulieta > Niken, tukarkan.

AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	*Liwin*	**Niken**	Satrio	**Rulieta**

LANGKAH 9

AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	*Liwin*	*Niken*	**Satrio**	**Rulieta**

[6] (terkecil)

Satrio > Rulieta, tukari

AgungBP	*Dion*	*Elnathan*	*Fifin*	*Hermon*	*Kayra*	*Liwin*	*Niken*	**Rulieta**	**Satrio**

HASIL AKHIR

Untuk mengimplementasikan teknik selection sort, anda dapat memperhatikan program 5.2 berikut ini.

```
//-------------------------------------------------------
//---  Fungsi untuk Mengurutkan Data (Selection)    ---
//-------------------------------------------------------
public static void mengurutkanDataSelection(formatBiodata
biodataMahasiswa[])
{
   formatBiodata biodataSementara = new formatBiodata();
   String teksTerkecil = "";
   int lokasi=0;

   //bagian mengurutkan dengan teknik selection
   for (int i=0;   i<=N-2;  i++)
   {
      //data pertama yang dibaca dianggap data terkecil
      teksTerkecil = "zzzzzzz";

      //menentukan bilangan terkecil mulai larik ke i+1 sampai N-1
      for (int S=i+1;  S<=N-1;  S++)
      {
         if (biodataMahasiswa[S].nama.compareTo(teksTerkecil)<0)
         {  //jika data[S] adlh bilangan terkecil, simpan di
teksTerkecil
            teksTerkecil = biodataMahasiswa[S].nama;
            //mencatat posisi dimana data terkecil ada
            lokasi = S;
         }
      }
      //membandingkan data[lokasi] yang adalah data terkecil,
      // versus data[i] yang adalah 'diagonal ke-i'
      if (biodataMahasiswa[i].nama.compareTo
                              (biodataMahasiswa[lokasi].nama)>0)
      {
         //tukar posisi
         {  biodataSementara = biodataMahasiswa[i];
            biodataMahasiswa[i] = biodataMahasiswa[lokasi];
            biodataMahasiswa[lokasi] = biodataSementara;
         }
      }
   }
}
```

c. Algoritma Insertion Sort

Algoritma Insertion sort akan membagi larik menjadi dua bagian. Bagian kiri akan menjadi data yang sudah urut sedangkan bagian kanan adalah bagian yang masih acak.

Proses pengurutan dilakukan dengan cara mengambil sebuah elemen kunci, kemudian meletakkan di tempat yang sesuai pada bagian kiri larik. Proses pencarian tempat yang sesuai dilakukan dengan membandingkan elemen kunci tersebut dengan setiap elemen yang ada di bagian kiri larik, dimulai dari kanan. Apabila ditemukan tempat yang sesuai maka elemen kunci tersebut akan disisipkan di lokasi yang dimaksud. Proses dilanjutkan dengan mengambil sebuah elemen kunci yang baru yang

ada di sebelah kanan elemen kunci yang sebelumnya, kemudian dilakukan kembali proses di atas, hingga semua data pada bagian kanan larik habis.

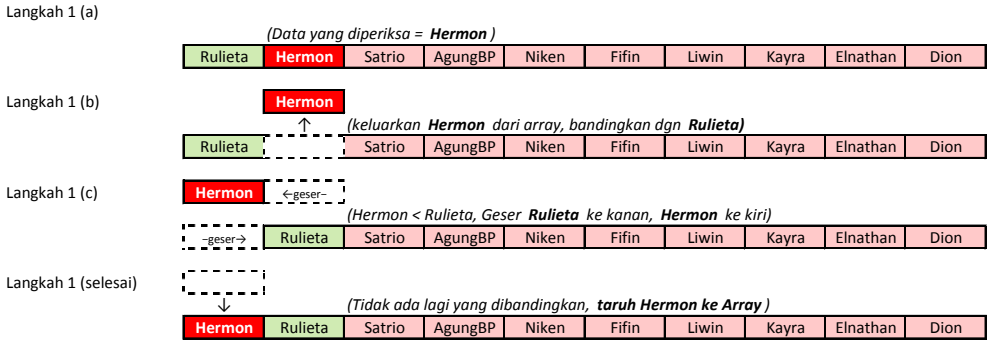
Agar mudah dipahami akan disimulasikan contoh pengurutan data berikut ini.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
DATA AWAL	Rulieta	Hermon	Satrio	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Dengan metode insertion sort, dapat kita lihat proses pengurutan sebagaimana tersaji pada gambar berikut ini.

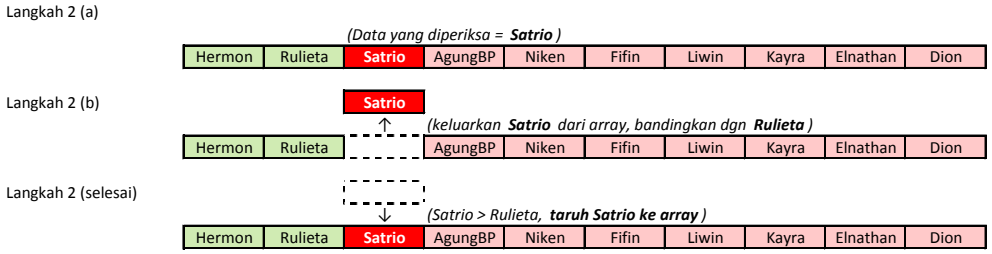
LANGKAH 1

(memeriksa orang ke-2)



LANGKAH 2

(memeriksa orang ke-3)



LANGKAH 3

(memeriksa orang ke-4)

Langkah 3 (a)

*(Data yang diperiksa = **AgungBP**)*

Hermon	Rulieta	Satrio	AgungBP	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 3 (b)

AgungBP

↑ *(keluarkan **AgungBP** dari array, bandingkan dgn **Satrio**)*

Hermon	Rulieta	Satrio		Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 3 (c)

AgungBP ←geser–

*(AgungBP < Satrio, Geser **Satrio** ke kanan, **AgungBP** ke kiri)*

Hermon	Rulieta	–geser→	Satrio	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 3 (d)

AgungBP ←geser–

*(AgungBP < Rulieta, Geser **Rulieta** ke kanan, **AgungBP** ke kiri)*

Hermon	–geser→	Rulieta	Satrio	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 3 (e)

AgungBP ←geser–

*(AgungBP < Hermon, Geser **Hermon** ke kanan, **AgungBP** ke kiri)*

–geser→	Hermon	Rulieta	Satrio	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 3 (selesai)

↓ *(Tidak ada lagi yang dibandingkan, taruh AgungBP ke Array)*

AgungBP	Hermon	Rulieta	Satrio	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

LANGKAH 4

(memeriksa orang ke-5)

Langkah 4 (a)

*(Data yang diperiksa = **Niken**)*

AgungBP	Hermon	Rulieta	Satrio	Niken	Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 4 (b)

Niken

↑ *(keluarkan **Niken** dari array, bandingkan dgn **Satrio**)*

AgungBP	Hermon	Rulieta	Satrio		Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 4 (c)

Niken ←geser–

*(Niken < Satrio, Geser **Satrio** ke kanan, **Niken** ke kiri)*

AgungBP	Hermon	Rulieta	–geser→	Satrio	Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 4 (d)

Niken ←geser–

*(Niken < Rulieta Geser **Rulieta** ke kanan, **Niken** ke kiri)*

AgungBP	Hermon	–geser→	Rulieta	Satrio	Fifin	Liwin	Kayra	Elnathan	Dion

Langkah 4 (selesai)

↓ *(Niken > Hermon, **taruh Niken ke array**)*

AgungBP	Hermon	Niken	Rulieta	Satrio	Fifin	Liwin	Kayra	Elnathan	Dion

LANGKAH 5

(memeriksa orang ke-6)

Langkah 5 (a)

*(Data yang diperiksa = **Fifin**)*

AgungBP	Hermon	Niken	Rulieta	Satrio	**Fifin**	Liwin	Kayra	Elnathan	Dion

Langkah 5 (b)

Fifin

*(keluarkan **Fifin** dari array, bandingkan dgn **Satrio**)* ↑

AgungBP	Hermon	Niken	Rulieta	Satrio		Liwin	Kayra	Elnathan	Dion

Langkah 5 (c)

Fifin ←geser–

*(Fifin < Satrio, Geser **Satrio** ke kanan, **Fifin** ke kiri)*

AgungBP	Hermon	Niken	Rulieta	–geser→	Satrio	Liwin	Kayra	Elnathan	Dion

Langkah 5 (d)

Fifin ←geser–

*(Fifin < Rulieta, Geser **Rulieta** ke kanan, **Fifin** ke kiri)*

AgungBP	Hermon	Niken	–geser→	Rulieta	Satrio	Liwin	Kayra	Elnathan	Dion

Langkah 5 (e)

Fifin ←geser–

*(Fifin < Niken, Geser **Niken** ke kanan, **Fifin** ke kiri)*

AgungBP	Hermon	–geser→	Niken	Rulieta	Satrio	Liwin	Kayra	Elnathan	Dion

Langkah 5 (f)

Fifin ←geser–

*(Fifin < Hermon, Geser **Hermon** ke kanan, **Fifin** ke kiri)*

AgungBP	–geser→	Hermon	Niken	Rulieta	Satrio	Liwin	Kayra	Elnathan	Dion

Langkah 5 (selesai)

↓ *(Fifin > AgungBP, **taruh Fifin ke array**)*

AgungBP	**Fifin**	Hermon	Niken	Rulieta	Satrio	Liwin	Kayra	Elnathan	Dion

LANGKAH 6

(memeriksa orang ke-7)

Langkah 6 (a)

*(Data yang diperiksa = **Liwin**)*

AgungBP	Fifin	Hermon	Niken	Rulieta	Satrio	**Liwin**	Kayra	Elnathan	Dion

Langkah 6 (b)

Liwin

*(keluarkan **Liwin** dari array, bandingkan dgn **Satrio**)* ↑

AgungBP	Fifin	Hermon	Niken	Rulieta	Satrio		Kayra	Elnathan	Dion

Langkah 6 (c)

Liwin ←geser–

*(Liwin < Satrio, Geser **Satrio** ke kanan, **Liwin** ke kiri)*

AgungBP	Fifin	Hermon	Niken	Rulieta	–geser→	Satrio	Kayra	Elnathan	Dion

Langkah 6 (d)

Liwin ←geser–

*(Liwin < Rulieta, Geser **Rulieta** ke kanan, **Liwin** ke kiri)*

AgungBP	Fifin	Hermon	Niken	–geser→	Rulieta	Satrio	Kayra	Elnathan	Dion

Langkah 6 (e)

Liwin ←geser–

*(Liwin < Niken, Geser **Niken** ke kanan, **Liwin** ke kiri)*

AgungBP	Fifin	Hermon	–geser→	Niken	Rulieta	Satrio	Kayra	Elnathan	Dion

Langkah 6 (selesai)

↓ *(Liwin < Hermon **taruh Liwin ke array**)*

AgungBP	Fifin	Hermon	**Liwin**	Niken	Rulieta	Satrio	Kayra	Elnathan	Dion

LANGKAH 7

(memeriksa orang ke-8)

Langkah 7 (a)

*(Data yang diperiksa = **Kayra**)*

AgungBP	Fifin	Hermon	Liwin	Niken	Rulieta	Satrio	**Kayra**	Elnathan	Dion

Langkah 7 (b)

Kayra

*(keluarkan **Kayra** dari array, bandingkan dgn **Satrio**)* ↑

AgungBP	Fifin	Hermon	Liwin	Niken	Rulieta	Satrio		Elnathan	Dion

Langkah 7 (c)

Kayra ←geser–

*(Kayra < Satrio, Geser **Satrio** ke kanan, **Kayra** ke kiri)*

AgungBP	Fifin	Hermon	Liwin	Niken	Rulieta	–geser→	Satrio	Elnathan	Dion

Langkah 7 (d)

Kayra ←geser–

*(Kayra < Rulieta, Geser **Rulieta** ke kanan, **kayra** ke kiri)*

AgungBP	Fifin	Hermon	Liwin	Niken	–geser→	Rulieta	Satrio	Elnathan	Dion

Langkah 7 (e)

Kayra ←geser–

*(Kayra < Niken, Geser **Niken** ke kanan, **Kayra** ke kiri)*

AgungBP	Fifin	Hermon	Liwin	–geser→	Niken	Rulieta	Satrio	Elnathan	Dion

Langkah 7 (f)

Kayra ←geser–

*(Kayra < Liwin, Geser **Liwin** ke kanan, **Kayra** ke kiri)*

AgungBP	Fifin	Hermon	–geser→	Liwin	Niken	Rulieta	Satrio	Elnathan	Dion

Langkah 7 (selesai)

↓ *(Kayra > Hermon **taruh Kayra ke array**)*

AgungBP	Fifin	Hermon	**Kayra**	Liwin	Niken	Rulieta	Satrio	Elnathan	Dion

LANGKAH 8

(memeriksa orang ke-9)

Langkah 8 (a)

*(Data yang diperiksa = **Elnathan**)*

AgungBP	Fifin	Hermon	Kayra	Liwin	Niken	Rulieta	Satrio	**Elnathan**	Dion

Langkah 8 (b)

Elnathan

*(keluarkan **Elnathan** dari array, bandingkan dgn **Satrio**)* ↑

AgungBP	Fifin	Hermon	Kayra	Liwin	Niken	Rulieta	Satrio		Dion

Langkah 8 (c)

Elnathan ←geser–

*(Elnathan < Satrio, Geser **Satrio** ke kanan, **Elnathan** ke kiri)*

AgungBP	Fifin	Hermon	Kayra	Liwin	Niken	Rulieta	–geser→	Satrio	Dion

Langkah 8 (d)

Elnathan ←geser–

*(Elnathan < Rulieta, Geser **Rulieta** ke kanan, **Elnathan** ke kiri)*

AgungBP	Fifin	Hermon	Kayra	Liwin	Niken	–geser→	Rulieta	Satrio	Dion

Langkah 8 (e)

Elnathan ←geser–

*(Elnathan < Niken, Geser **Niken** ke kanan, **Elnathan** ke kiri)*

AgungBP	Fifin	Hermon	Kayra	Liwin	–geser→	Niken	Rulieta	Satrio	Dion

Langkah 8 (f)

Elnathan ←geser–

*(Elnathan < Liwin, Geser **Liwin** ke kanan, **Elnathan** ke kiri)*

AgungBP	Fifin	Hermon	Kayra	–geser→	Liwin	Niken	Rulieta	Satrio	Dion

Langkah 8 (g)

Elnathan ←geser–

*(Elnathan < Kayra, Geser **Kayra** ke kanan, **Elnathan** ke kiri)*

AgungBP	Fifin	Hermon	–geser→	Kayra	Liwin	Niken	Rulieta	Satrio	Dion

Langkah 8 (h)

Elnathan ←geser–

*(Elnathan < Hermon, Geser **Hermon** ke kanan, **Elnathan** ke kiri)*

AgungBP	Fifin	–geser→	Hermon	Kayra	Liwin	Niken	Rulieta	Satrio	Dion

Langkah 8 (i)

Elnathan ←geser–

*(Elnathan < Fifin, Geser **Fifin** ke kanan, **Elnathan** ke kiri)*

AgungBP	–geser→	Fifin	Hermon	Kayra	Liwin	Niken	Rulieta	Satrio	Dion

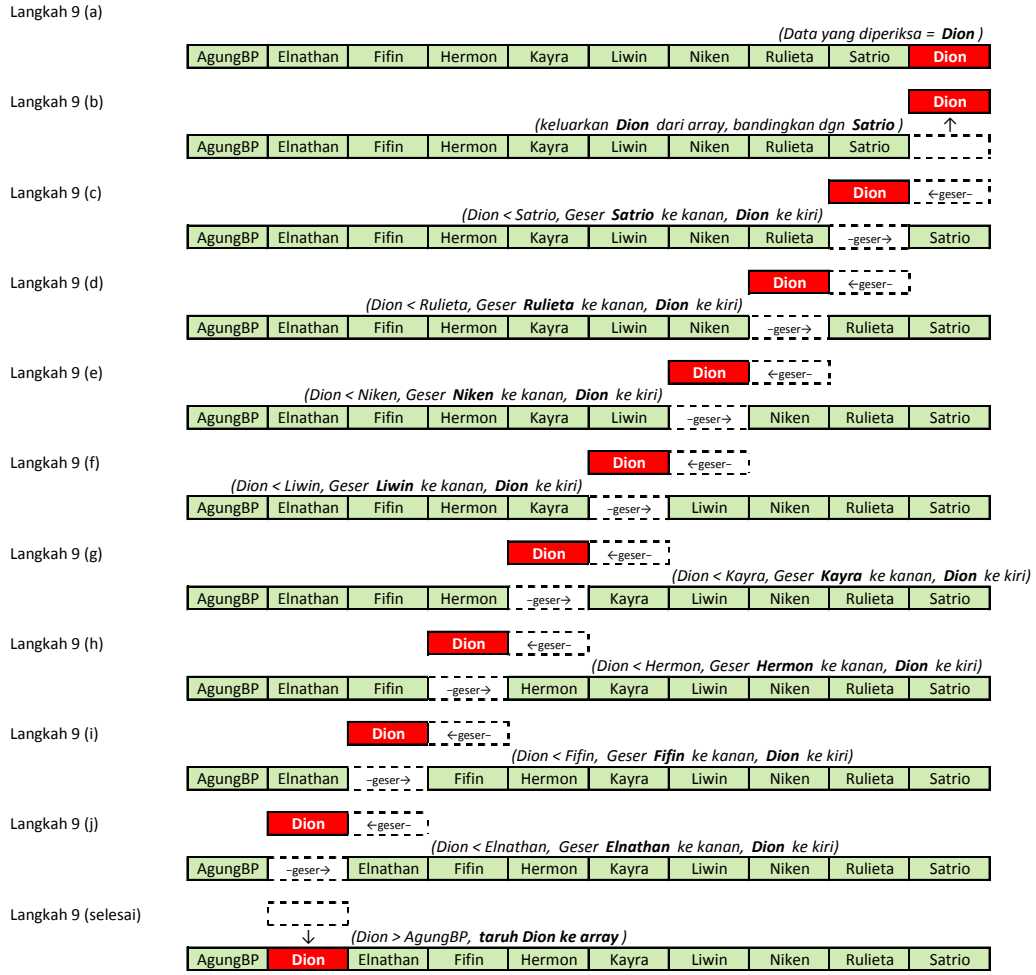
Langkah 8 (selesai)

↓ *(Elnathan > AgungBP, **taruh Elnathan ke array**)*

AgungBP	**Elnathan**	Fifin	Hermon	Kayra	Liwin	Niken	Rulieta	Satrio	Dion

LANGKAH 9

(memeriksa orang ke-10)



HASIL AKHIR

HASIL AKHIR	AgungBP	Dion	Elnathan	Fifin	Hermon	Kayra	Liwin	Niken	Rulieta	Satrio

Perhatikan program 5.3 berikut.

```
//-------------------------------------------------------
//---  Fungsi untuk Mengurutkan Data (Insertion)     ---
//-------------------------------------------------------
public static void mengurutkanDataInsertion(formatBiodata biodataMahasiswa[])
{
    formatBiodata biodataSementara = new formatBiodata();

    //untuk menentukan awal dari data sisi kanan (sisi yg masih berantakan)
    int awal;
    //untuk mencari posisi yg tepat pada sisi kiri (sisi yg sudah berurutan)
    int cari;
    awal = 1;
    while (awal <=N-1)
    {
        biodataSementara = biodataMahasiswa[awal];
        cari = awal-1;
```

```
        //cari akan bergerak dari kanan (awal-1) ke kiri
        while ( cari >= 0)
        {
            //( biodataMahasiswa[cari].nama > biodataSementara.nama )
            if (biodataMahasiswa[cari].nama.compareTo(biodataSementara.nama)>0)
            {
                biodataMahasiswa[cari+1] = biodataMahasiswa[cari];
                biodataMahasiswa[cari] = biodataSementara;
                cari--; //cari digeser kekiri 1 langkah
            }
            else
            {   biodataMahasiswa[cari+1] = biodataSementara;
                // perintah ini untuk keluar dari loop while
                cari=-1;
            }
        }
        awal++;
    }
}
```

Program 5.3

PRAKTIK

1. Praktek 1

Tambahkan program 5.1 ke dalam master program yang telah anda buat pada praktikum yang lalu. Eksekusi master program anda untuk melakukan pengurutan data secara ***Bubblesort***.

Ujilah program dengan mengentri data secara acak (nama mahasiswa dientri tidak dalam keadaan urut) kemudian lakukan pengurutan terhadap data tersebut, kemudian tampilkan data kembali. Bagaimana hasilnya? Catat dan simpulkan dalam laporan anda.

2. Praktek 2

Tambahkan program 5.2 ke dalam master program anda. Eksekusi master program anda untuk melakukan pengurutan data secara ***SelecionSort***

Ujilah program dengan mengentri data secara acak (nama mahasiswa dientri tidak dalam keadaan urut) kemudian lakukan pengurutan terhadap data tersebut, kemudian tampilkan data kembali. Bagaimana hasilnya? Catat dan simpulkan dalam laporan anda

3. Praktek 3

Tambahkan program 5.3 ke dalam master program yang telah anda buat pada praktikum yang lalu. Eksekusi master program anda untuk melakukan pengurutan data secara ***InsertionSort***.

Ujilah program dengan mengentri data secara acak (nama mahasiswa dientri tidak dalam keadaan urut) kemudian lakukan pengurutan terhadap data tersebut, kemudian tampilkan data kembali. Bagaimana hasilnya? Catat dan simpulkan dalam laporan anda

4. Praktek 4

Sekarang pikirkan, kemudian modifikasilah ketiga program sorting di atas supaya nama-nama dalam larik menjadi urut turun (descending).

LATIHAN

1. Modifikasilah program 5.1 dan program 5.2 agar dapat mengurutkan data di dalam larik ***berdasarkan IPK***
2. .....

TUGAS

1. Buatlah sebuah program Sorting dengan metode insertion dalam java menggunakan dengan menu :
   - insert,
   - View,
   - Sort,
   - Exit

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 43-57, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 6
PEMANFAATAN ARRAY/LARIK SEBAGAI STACK (TUMPUKAN) DAN QUEUE (ANTRIAN)

CAPAIANPEMBELAJARAN

Mahasiswa dapat mengimplementasikan tumpukan dan antrian untuk berbagai keperluan dengan menggunakan bahasa pemrograman Java

KEBUTUHAN ALAT/BAHAN/SOFTWARE

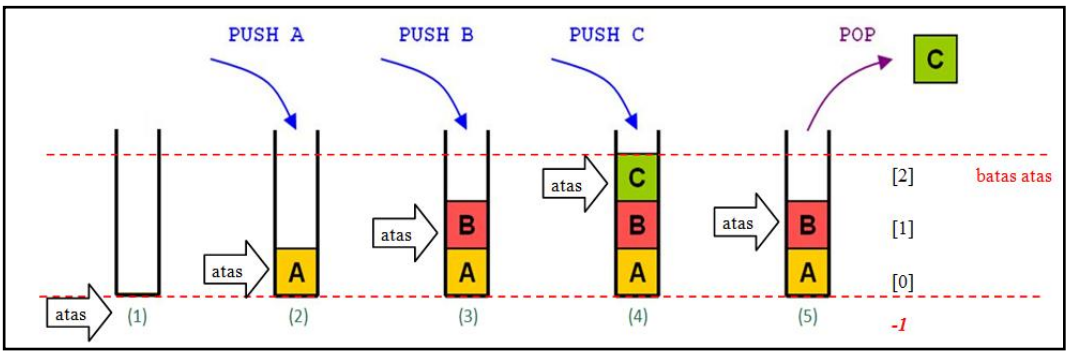
1. TextPad

DASAR TEORI

Stack (Tumpukan)

Tumpukan atau stack adalah teknik peletakan data dimana seolah-olah data yang satu diletakkan 'diatas' data yang lain. Dalam tumpukan semua penyisipan dan penghapusan data dilakukan hanya melalui satu pintu yang disebut top (puncak) tumpukan. Tumpukan dapat dibayangkan seperti tumpukan buku dimana hanya buku teratas yang dapat diperoleh kembali dengan satu langkah. Buku-buku yang terdapat dibawah hanya dapat diambil setelah buku yang berada diatasnya diambil (dikeluarkan) terlebih dahulu. Dengan bentuk yang demikian tumpukan dapat dikatakan sebagai struktur data yang bersifat **LIFO (Last In First Out**).

Dalam tumpukan terdapat 2 buah operasi data yaitu (1) operasi **push** dan (2) operasi **pop**. Operasi **push** adalah operasi untuk menambahkan sebuah data ke dalam tumpukan, sedangkan operasi **pop** adalah operasi untuk mengambil (menghapus) data dari dalam tumpukan.



Gambar 7.1 Ilustrasi Tumpukan

Dalam implementasinya tumpukan selalu memiliki kapasitas yaitu batas maksimum di mana tumpukan mampu menampung data.

Ada 2 kondisi yang menjadi batasan sebuah tumpukan yaitu :

1. Kondisi **Tumpukan penuh**.
   Kondisi tumpukan penuh adalah kondisi di mana nilai atas = nilai batasAtas. Apabila kondisi ini terjadi maka tumpukan tidak lagi dapat menerima pemasukan data baru (push data baru)

2. Kondisi **Tumpukan kosong**.
   Kondisi tumpukan kosong adalah kondisi di mana nilai atas = -1. Apabila kondisi ini terjadi maka kita tidak lagi dapat melakukan pengambilan/penghapusan data (pop data) dari tumpukan

Program tumpukan dapat dilihat pada program 7.1 berikut ini.

```
public class program_tumpukan
{   public static int N = 5;
    public static int atas = -1;

    public static void PUSH (String tumpukan[], String data)
    {   if (atas == N-1) //jika tumpukan penuh
        {   System.out.println("maap, tumpukan penuh, PUSH " + data
                        + "  tidak dapat dilakukan");
        }
        else //jika tumpukan tidak penuh
        {   atas = atas + 1;
            tumpukan[atas] = data;
            System.out.println("PUSH " + data + " berhasil..");
        }
    }
    public static String POP (String tumpukan[])
    {   String hasil;
        if (atas < 0 ) //jika tumpukan kosong
        {   hasil = "TUMPUKAN KOSONG, POP GAGAL DILAKUKAN";
        }
        else //jika tumpukan tidak kosong
        {   hasil = tumpukan[atas];
            atas = atas - 1;
        }
        return (hasil);
    }
    public static void lihatTumpukan(String tumpukan[])
```

```
    {   System.out.println("");
        System.out.println("--TUMPUKAN:-");
        for (int i=atas;  i>=0;  i--)
        {   System.out.println(tumpukan[i]);
        }
        System.out.println("--akhir tumpukan-");
        System.out.println("");
    }

    public static void main (String[] args)
    {   String tumpukan[] = new String[10];
        .............;
        .............;
        .............;
        .............;
    }
}
```

Program 7.1

Queue (Antrian)

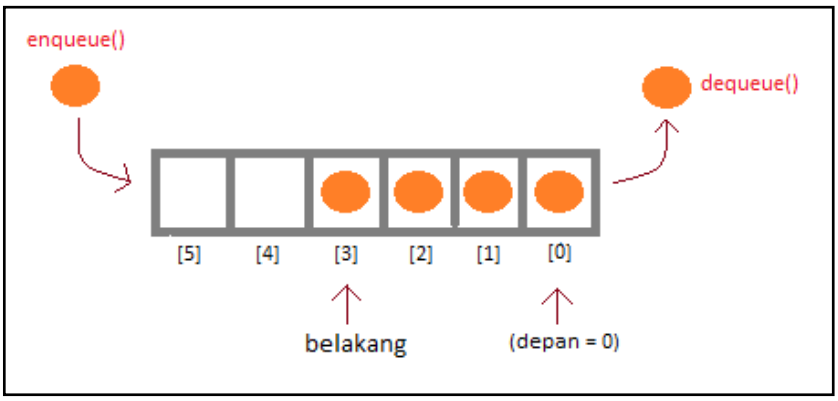
Antrian atau Queue adalah teknik peletakan data dimana seolah-olah data yang satu diletakkan 'dibelakang' data yang lain. Dalam antrian semua penyisipan dan penghapusan data dilakukan melalui dua pintu yaitu belakang dan depan. Pintu belakang digunakan sebagai jalan masuk data, sedangkan pintu depan digunakan sebagai jalan keluar data.

Antrian dapat dibayangkan seperti antrian para perima zakat. Orang yang ingin mendapatkan zakat harus mengantri dengan cara masuk melalui sisi belakang. Apabila seorang penerima telah terlayani maka dia harus meninggalkan antrian melalui sisi depan.

Dengan bentuk yang demikian antrian dapat dikatakan sebagai struktur data yang bersifat **FIFO (First In First Out**) artinya pengantri yang datang paling awal akan keluar paling dahulu.

Dalam antrian terdapat 2 buah operasi data yaitu (1) operasi **enQueue** dan (2) operasi **deQueue**. Operasi **enQueue** adalah operasi untuk menambahkan sebuah data ke dalam antrian dan dilakukan pada pintu belakang, sedangkan operasi **deQueue** adalah operasi untuk mengambil (menghapus) data dari dalam antrian dan dilakukan pada pintu depan.

Dalam implementasinya, sebuah antrian pasti memiliki kapasitas atau batas daya tampung antrian. Dalam ilustrasi di atas, kapasitas antrian biasanya disesuaikan dengan banyaknya zakat yang diberikan. Apabila hanya tersedia zakat untuk 10 orang maka kapasitas antrian hanya akan menampung 10 orang, sehingga orang ke-11, ke-12, dan seterusnya pasti akan ditolak untuk masuk ke dalam antrian.



Gambar 7.2 Ilustrasi Antrian

Ada 2 kondisi yang menjadi batasan sebuah antrian yaitu :

1. Kondisi **Antrian penuh**.
   Kondisi antrian penuh adalah kondisi di mana nilai `belakang` = nilai `batasAntrian`. Apabila kondisi ini terjadi maka antrian tidak lagi dapat menerima pemasukan data baru (enqueue data baru)

2. Kondisi **Antrian kosong**.
   Kondisi antrian kosong adalah kondisi di mana nilai `belakang` = -1. Apabila kondisi ini terjadi maka kita tidak lagi dapat melakukan pengambilan/penghapusan data (dequeue data) dari antrian

Program antrian dapat dilihat pada program 7.2 berikut ini.

```
public class program_antrian
{   public static int N = 5;
    public static int belakang = -1;

    public static void ENQUEUE (String antrian[], String data)
    {   if (belakang == N-1) //jika antrian penuh
        {   System.out.println("maap, antrian penuh, ENQUEUE "
                        + data + "  tidak dapat dilakukan");
        }
        else //jika antrian tidak penuh
        {   belakang = belakang + 1;
            antrian[belakang] = data;
            System.out.println("ENQUEUE " + data + " berhasil..");
        }
    }
    public static String DEQUEUE (String antrian[])
    {   String hasil;
        if (belakang < 0 ) //jika antrian kosong
        {   hasil = "ANTRIAN KOSONG, DEQUEUE GAGAL DILAKUKAN";
        }
        else //jika antrian tidak kosong
        {   hasil = antrian[0];
            //----menggeser data kedua dst, maju selangkah ke depan
            for (int i=0; i<=belakang-1;  i++)
            {   antrian[i] = antrian[i+1];
            }
            belakang = belakang - 1;
        }
```

```
        return (hasil);
    }

    public static void lihatAntrian(String antrian[])
    {   System.out.println("-----------------------------------------");
        System.out.print("ISI ANTRIAN : (depan)");
        for (int i=0;  i<=belakang;  i++)
        {   System.out.print("  " + antrian[i]);
        }
        System.out.println("  (belakang)");
        System.out.println("-----------------------------------------");
    }
    public static void main (String[] args)
    {   String antrian[] = new String[5];
        .............;
        .............;
        .............;
        .............;
    }
}
```

Program 7.2.

PRAKTIK

1. Praktik 1

Tuliskan program 7.1 menggunakan TextPad. Tambahkan pada bagian program utama operasi "push" berturut-turut :

```
push (tumpukan, "Buku A");
push (tumpukan, "Buku B");
push (tumpukan, "Buku C");
push (tumpukan, "Buku D");
bacaTumpukan(tumpukan);
```

Sekarang jalankan program diatas, dan amati yang terjadi. Buku apa sajakah yang dapat di *push* ke dalam tumpukan? Apakah buku A, B, C, D, keempatnya dapat dipush ke dalam tumupukan?
Amatilah posisi "atas". Menurut anda, setelah dipush nya keempat buku di ABCD, berada di posisi manakah "atas"? Catatlah dalam laporan anda.

2. Praktik 2

Sekarang, lakukan lagi proses *push* untuk sederet data "buku" berikut ini ke dalam tumpukan:

```
push (tumpukan, "Buku E");
push (tumpukan, "Buku F");
push (tumpukan, "Buku G");
bacaTumpukan(tumpukan);
```

Sekarang jalankan program diatas, dan amati yang terjadi. “Buku” apa sajakah yang dapat di *push* ke dalam tumpukan? Adakah “Buku” yang gagal di *push* ke dalam tumpukan? Apa sebabnya? Mengapa bisa demikian? Menurut anda bagian manakah dari program di atas yang menyebabkannya? Jelaskanlah dalam laporan anda.

3. Praktik 3

Sekarang, lakukan *pop* dari tumpukan dengan cara menambahkan perintah berikut ke **bagian akhir** dari program utama .

```
System.out.println("POP: " + pop(tumpukan));
bacaTumpukan(tumpukan);
```

“Buku” apakah yang ter-*pop* ? Mengapa bisa demikian?

Sekarang amatilah kondisi tumpukan saat ini. Adakah yang berkurang? Mengapa hal itu bisa terjadi? Menurut anda, bagian manakah dari program di atas yang menyebabkan itu terjadi? Bagaimanakah juga dengan kondisi “atas” saat ini? Jelaskan pada laporan anda.

4. Praktik 4

Sekarang, lakukan lagi proses *pop* dari tumpukan sebanyak 2 kali berturut-turut :

```
System.out.println("POP: " + pop(tumpukan));
System.out.println("POP: " + pop(tumpukan));
bacaTumpukan(tumpukan);
```

Catatlah “Buku” apakah saja yang ter-‘*pop’* ?

Sekarang amatilah kondisi tumpukan saat ini. Adakah perubahan yang terjadi pada kondisi tumpukan saat ini dibanding dengan pada percobaan sebelumnya? Mengapa hal itu bisa terjadi? Bagaimanakah juga dengan kondisi “atas” saat ini? Jelaskan pada laporan anda.

5. Praktik 5

Sekarang, lakukan lagi proses *pop* dari tumpukan sebanyak 3 kali berturut-turut :

```
System.out.println("POP: " + pop(tumpukan));
System.out.println("POP: " + pop(tumpukan));
System.out.println("POP: " + pop(tumpukan));
System.out.println("POP: " + pop(tumpukan));
bacaTumpukan(tumpukan);
```

Adakah proses *pop* yang tidak dapat (gagal) dilakukan? Mengapa bisa demikian? Jelaskan pada laporan anda.

6. Praktik 6

Tuliskan program 7.2 menggunakan TextPad. Tambahkan pada bagian program utama operasi “enQueue” berturut-turut :

```
enQueue (antrian, "Mobil A");
enQueue (antrian, "Mobil B");
enQueue (antrian, "Mobil C");
bacaAntrian(antrian);
```

Sekarang jalankan program diatas, dan amati yang terjadi. “Mobil” apa sajakah yang dapat di *enQueue* ke dalam antrian? Lalu bagaimanakah dengan ‘belakang’? Berada di posisi manakah ‘belakang’ saat ini? Catatlah dalam laporan anda

7. Praktik 7

Sekarang, lakukan lagi proses *enQueue* sederet data “Mobil” berikut ini ke dalam antrian:

```
enQueue (antrian, "Mobil D");
enQueue (antrian, "Mobil E");
enQueue (antrian, "Mobil F");
bacaAntrian(antrian);
```

Sekarang jalankan program diatas, dan amati yang terjadi. Bagaimanakah kondisi antrian saat ini? “Mobil” apa sajakah yang ada dalam antrian? Adakah “Mobil” yang gagal di *enQueue* ke dalam antrian? Apa sebabnya? Mengapa bisa demikian? Menurut anda bagian manakah dari program di atas yang menyebabkannya? Jelaskanlah dalam laporan anda.

Lalu bagaimana dengan posisi “belakang” saat ini? Adakah perbedaan posisi “belakang” saat ini jika dibandingkan dengan posisi “belakang” pada pelaksanaan praktikum ke-6. Catatlah dalam laporan anda.

8. Praktik 8

Sekarang, lakukan *deQueue* dari antrian dengan cara menambahkan perintah berikut ke bagian akhir dari program utama .

```
System.out.println("deQueue: " + deQueue(antrian));
bacaAntrian(antrian);
```

“Mobil” apakah yang ter-*deQueue* ? Mengapa bisa demikian?

Sekarang amatilah kondisi antrian saat ini. Adakah perubahan yang terjadi pada kondisi antrian saat ini dibanding dengan pada percobaan sebelumnya? Apakah setiap data “Mobil B,C,D,..dst” masih berada di posisinya yang sama? Mengapa hal itu bisa terjadi?

Menurut anda, bagian manakah dari program di atas yang menyebabkan itu terjadi? Bagaimanakah juga dengan kondisi ekor saat ini? Jelaskan pada laporan anda.

9. Praktik 9

Sekarang, lakukan lagi proses *deQueue* dari antrian sebanyak 3 kali berturut-turut :

```
System.out.println("deQueue: " + deQueue(antrian));
System.out.println("deQueue: " + deQueue(antrian));
System.out.println("deQueue: " + deQueue(antrian));
bacaAntrian(antrian);
```

Catatlah "Mobil" apakah saja yang ter-*deQueue* ?

Sekarang amatilah kondisi antrian saat ini. Adakah perubahan yang terjadi pada kondisi antrian saat ini dibanding dengan pada percobaan sebelumnya? Apakah setiap data "Mobil B,C,D,..dst" masih berada di posisinya yang sama? Mengapa hal itu bisa terjadi? Bagaimanakah juga dengan kondisi ekor saat ini? Jelaskan pada laporan anda.

10. Praktik 10

Sekarang, lakukan lagi proses *deQueue* dari antrian sebanyak 2 kali berturut-turut.

```
System.out.println("deQueue: " + deQueue(antrian));
System.out.println("deQueue: " + deQueue(antrian));
bacaAntrian(antrian);
```

Adakah proses *deQueue* yang tidak dapat dilakukan? Mengapa bisa demikian? Jelaskan pada laporan anda.

LATIHAN

1. .....
2. .....

TUGAS

1. ……
2. …..

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 58-65, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 7
POINTER DALAM JAVA

CAPAIANPEMBELAJARAN

Mahasiswa dapat mengekploitasi variabel pointer di dalam java

KEBUTUHAN ALAT/BAHAN/SOFTWARE

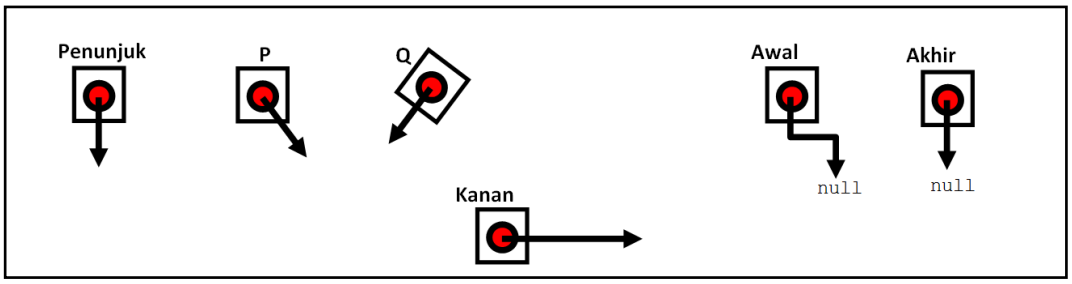
1. TextPad

DASAR TEORI

A. PENGENALAN POINTER

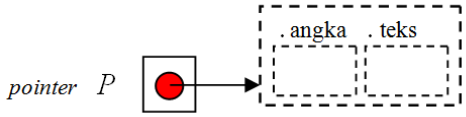
Pointer (Penunjuk) adalah sebuah variabel bertipe khusus yang dapat menunjuk (menyimpan alamat) sebuah variabel/ heap/ obyek. Tipe data dari variabel pointer (selanjutnya disebut pointer) tidak bertipe primitif sebagaimana int, char, boolean melainkan bertipe data buatan sehingga harus dideklarasikan terlebih dahulu menggunakan sebuah ***class*** *(pelajari kembali modul 2 bagian "Struktur Penyimpan Terstruktur berbasis Record/Rekaman")*.

Pointer dapat diberi nama apapun yang menggambarkan untuk apa pointer tersebut diciptakan. Contoh :



Lalu bagaimana sebuah pointer diciptakan? Untuk lebih jelasnya perhatikanlah program di bawah ini.

```
class tipePointer
{     int angka;
      string teks;
}

tipePointer  P;
```



Pada script di atas, pointer P adalah sebuah pointer yang dapat menyimpan sebuah alamat dari sebuah variabel obyek (HEAP) yang bertipe 'tipePointer'. Di dalam heap tersebut nantinya terdapat dua buah anggota yaitu `angka` dan `teks` (tetapi heap itu sediri sat ini belum diciptakan). Dalam pengertian yang lebih sederhana, kata "menyimpan" alamat sebuah heap dapat diartikan bahwa pointer P mampu menunjuk ke sebuah heap yang berisi variabel angka dan teks.

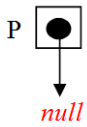
Ada tiga karakteristik dari sebuah pointer yaitu :

a. Pointer dapat menunjuk ke NULL
b. Pointer dapat menunjuk ke sebuah HEAP
c. Pointer dapat saling mengcopy (alamat) heap yang sedang ditunjuk

a. Pointer dapat menunjuk ke NULL

Null adalah sebuah tempat kosong di memori yang tidak memiliki nilai. Null dapat diumpamakan seperti ground pada sebuah rangkaian elektronika. Pointer dapat diarahkan untuk menunjuk ke null tersebut seperti gambar di bawah ini.

```
tipePointer P;

P = null
```

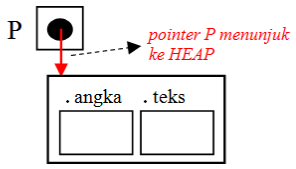


b. Pointer dapat menunjuk ke sebuah HEAP baru.

Pointer dapat menunjuk ke sebuah heap baru. Heap adalah sebuah variabel tidak bernama yang dapat menyimpan sebuah nilai sebagaimana sebuah variabel pada umumnya. Dalam konsep OOP heap dikenal dengan nama **obyek**. Heap juga sering disebut **Node /Simpul/ List/ Senarai/ Record**.

Agar pointer dapat menunjuk ke sebuah heap, heap harus diciptakan terlebih dahulu menggunakan perintah ***new()*** baru kemudian sebuah pointer dapat diarahkan untuk menunjuk ke heap tersebut. Perintah yang digunakan agar pointer P menunjuk ke heap yang baru tersebut adalah "**P =**".

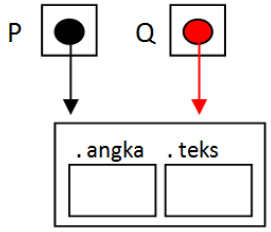
```
tipePointer P;

P = new tipePointer();
```



c. Pointer dapat saling mengcopykan alamat heap yang sedang ditunjuk

Pointer dapat saling mengcopykan sebuah alamat heap yang sedang ditunjuknya. Apabila alamat sebuah heap dicopykan dari satu pointer (misalnya P) ke pointer yang lain (misalnya Q) maka itu berarti pointer yang menerima copynya (Q) diarahkan untuk menunjuk ke heap yang sedang ditunjuk oleh pointer pemberi (P).

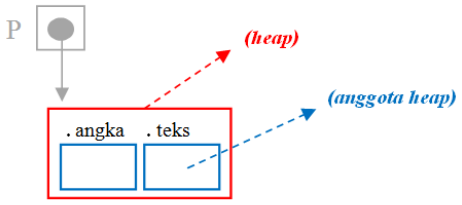
```
tipePointer P;
P = new tipePointer;

tipePointer Q;

Q = P;  ← Q diarahkan menunjuk ke heap yang ditunjuk oleh P
```



B. PENGENALAN HEAP

Heap, disebut juga obyek/ simpul/ node, adalah sebuah variabel tak bernama yang dialokasikan secara khusus dalam memori yang dapat menyimpan banyak variabel. Heap diciptakan dengan perintah ***new***. Perhatikan program berikut.

```
class tipePointer
{     int angka;
      string teks;
}

tipePointer  P;

P = new tipePointer()   ← perintah untuk menciptakan heap
```



Pada program di atas, perintah '`new tipePointer()`' digunakan untuk menciptakan sebuah heap, sedangkan perintah '`P =`' digunakan untuk mengarahkan pointer P untuk menunjuk ke heap yang baru saja diciptakan.

Tipe data dari sebuah heap tergantung dari tipe kelas di mana heap tersebut dibentuk. Pada program di atas, heap diciptakan bertipe “tipePointer” yang memiliki 2 variabel anggota yaitu `angka` (bertipe int) dan `teks` (bertipe String). Oleh karena heap diciptakan dalam tipe data “tipePointer” maka heap tersebut secara otomatis juga memiliki 2 variabel, angka dan teks.

Sebagaimana variabel biasa bertipe primitif yang digunakan untuk menyimpan nilai, heap juga dapat menyimpan sebuah nilai.

Anggota heap yang bernama *angka* dan *teks* dapat diakses, baik untuk ditampilkan ke layar menggunakan `System.out.print()`, maupun untuk dimanipulasi nilainya penggunakan penugasan variabel (assigment). Format yang digunakan untuk mengakses maupun memanipulasi anggota sebuah heap adalah :

[nama pointer] ● [nama variabel anggota]

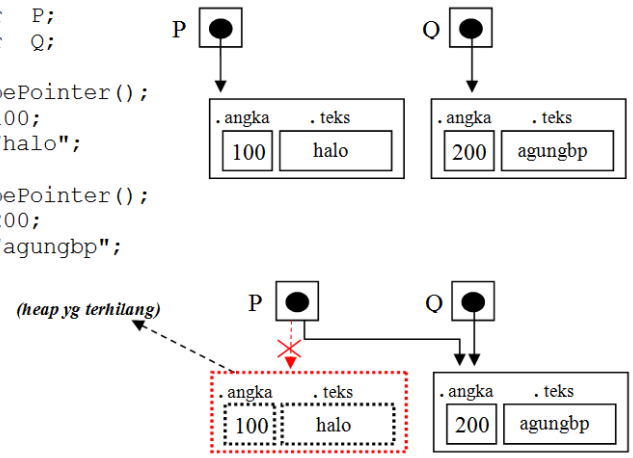
Sebagai contoh perhatikan program di bawah ini.

```
P.angka = 20;
P.teks = “agungbp”;
System.out.println (P.angka);
System.out.println (P.teks);
```

Berbeda dengan bahasa C++, heap dalam bahasa java dapat diciptakan namun tidak disediakan perintah untuk menghapusnya. Hal ini karena compiler java secara omatis akan memusnahkan heap saat program ataupun subprogram selesai dieksekusi.

Dalam prakteknya, sebuah heap dapat juga “terhilang” di dalam memori yaitu jika heap tersebut ditinggalkan oleh pointer yang menunjuknya. Yang dimaksud terhilang di atas adalah sekalipun heap tersebut masih ada di dalam memori, heap tidak dapat diakses lagi karena tidak ada pointer yang menjadi sarana untuk mengaksesnya. Sebagai contoh perhatikan program berikut ini.

```
class tipePointer
{     int angka;
      string teks;
}

tipePointer  P;
tipePointer  Q;

P = new tipePointer();
P.angka = 100;
P.teks  = "halo";

Q = new tipePointer();
Q.angka = 200;
Q.teks  = "agungbp";

P = Q;
```

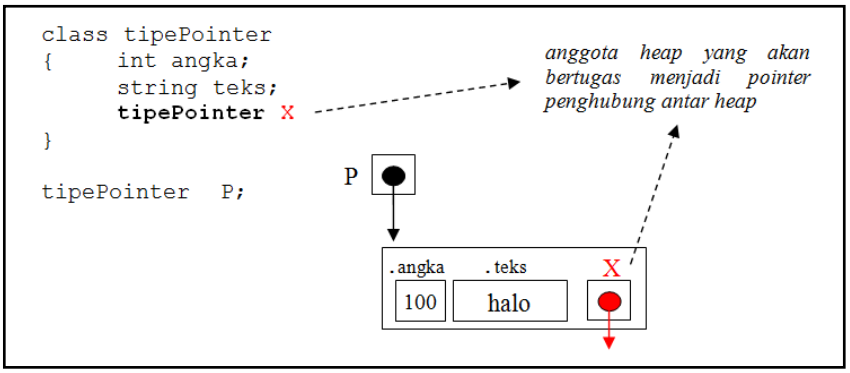


Pada program di atas pointer P menunjuk ke heap yang berisi angka=*100* dan teks=*"halo"*, sedangkan pointer Q menunjuk ke heap yang berisi angka=*200* dan teks=*"agungbp"*. Pada kondisi ini apabila kemudian dilakukan perintah `P = Q;` maka `P` yang tadinya menunjuk ke heap yang berisi *100* dan *"halo"* akan berubah arah menunjuk ke heap yang berisi *200* dan *"agungbp"*. Hal ini akan mengakibatkan heap yang berisi angka=*100* dan teks=*"halo"* terhilang dan tidak dapat diselamatkan lagi.

Di sisi lain, sebuah heap dapat ditunjuk oleh lebih dari satu pointer. Pada kasus di atas, setelah penunjukan P berpindah dari heap yang berisi angka=*100* dan teks=*"halo"* menjadi menunjuk ke heap yang berisi angka=*200* dan teks=*"agungbp"*, maka heap yang berisi angka=*200* dan teks=*"agungbp"* memiliki 2 buah pointer yang menunjuknya yaitu P dan Q. Dalam kondisi seperti ini heap yang berisi angka=*200* dan teks=*"agungbp"* dapat diakses melalui 2 cara yaitu `P.angka` ataupun `Q.angka`, keduanya akan merujuk pada sebuah nilai yang sama. Demikian juga `P.teks` dan `Q.teks` keduanya juga merujuk pada nilai yang sama.

C. HEAP YANG SALING TERHUBUNG (LINKEDLIST / SENARAI BERANTAI)

Heap dapat juga dibuat untuk saling terhubung satu sama lain. Untuk dapat melakukan hal ini kita harus menambahkan sebuah variabel baru sebagai anggota heap yang harus bertipe pointer. Perhatikan contoh berikut.

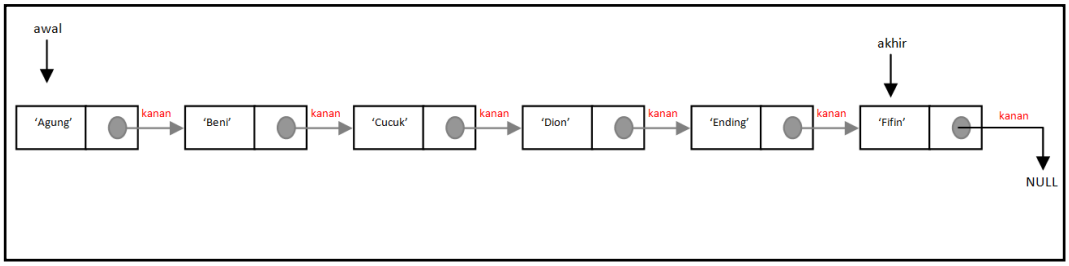


Dari struktur heap di atas tampak bahwa ada sebuah pointer tambahan X yang merupakan anggota kelas "`tipePointer`" yang dideklarasikan juga menggunakan tipe data "`tipePointer`" yang adalah dirinya sendiri. Karena dideklarasikan menggunakan tipe data "`tipePointer`" sebagaimana pointer `P`, maka pointer `X` juga memiliki kemampuan dan karakteristik yang persis sama dengan pointer `P`. Pointer `X` inilah yang nantinya dimanfaatkan untuk menjadi pointer penghubung (pointer pengait) antara heap tempat dia berada dengan heap yang lain. Kosep heap terhubung inilah yang nantinya menjadi cikal bakal adanya **Linkedlist (senarai berantai).**

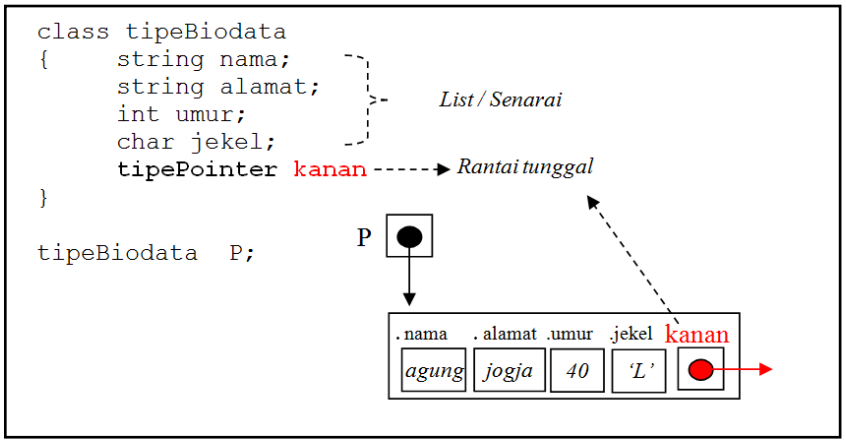
Linked list (daftar terhubung) atau disebut juga *Senarai Berantai,* adalah pengalokasian memori secara dinamis untuk dapat digunakan untuk menyimpan data (list). Mengapa disebut dinamis karena alokasi memori ini dilakukan dengan membuat simpul (*heap)* di memori. Jika pada larik/ array kita hanya dapat menyimpan data dalam jumlah tertentu dan tidak dapat diubah (statis), pada *linked list* kita dapat menyimpan data secara lebih dinamis karena pengalokasian simpul di memori baru akan dilakukan pada saat datanya datang.

D. PENGENALAN SINGLE LINKED LIST (SENARAI BERANTAI TUNGGAL)

Single Linked List adalah kumpulan heap/ obyek/ simpul/ node yang saling terhubung satu sama lain (*linked*) yang dimanfaatkan untuk menyimpan sederet data (*list*) dimana pada setiap heap yang ada terdapat **satu buah pointer anggota (*single*)** yang bertugas sebagai pointer pengait yang digunakan untuk mengkaitkan diri dengan simpul sejenis yang ada di sebelah kanannya. Kebanyakan orang menyingkat Single Linked List (Senarai Berantai Tunggal) hanya dengan sebutan Linked List (Senarai Berantai).



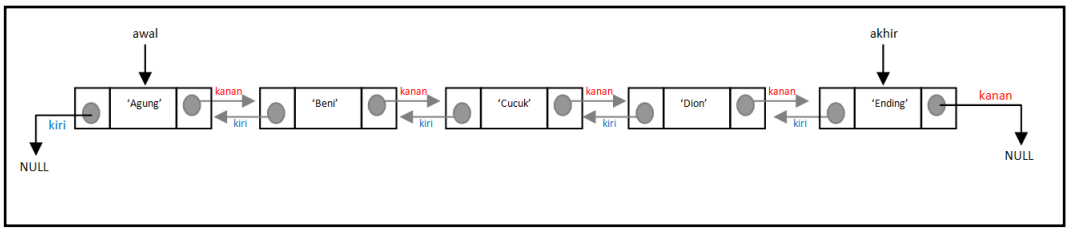
Lalu bagaimana Single Linkedlist diciptakan? Untuk lebih jelasnya perhatikanlah program di bawah ini.



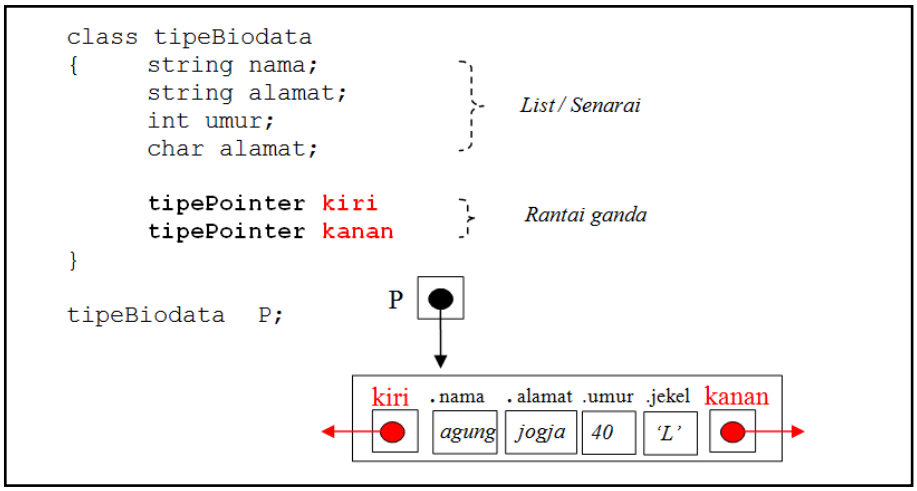
Dari struktur heap di atas terlihat ada 2 kelompok variabel. Pertama adalah List/Senarai yang berisi kumpulan biodata (`nama`, `alamat`, `umur` dan `jenis kelamin`), dan yang kedua adalah sebuah rantai penghubung `kanan` yang nantinya akan bertugas untuk merangkaikan setiap heap dengan heap yang lain. Pada single linkedlist rantai ini berjumlah tunggal yaitu `kanan`.

E. PENGENALAN DOUBLE LINKED LIST (SENARAI BERANTAI GANDA)

Double Linked List adalah kumpulan heap/ obyek/ simpul/ node yang saling terhubung satu sama lain (*linked*) yang dimanfaatkan untuk menyimpan sederet data (*list*) dimana pada setiap heap yang ada terdapat **dua buah pointer anggota (*double*)** yang bertugas sebagai pointer pengait yang digunakan untuk mengkaitkan diri dengan simpul sejenis yang ada di sebelah kiri dan kanannya.



Lalu bagaimana Double Linkedlist diciptakan? Untuk lebih jelasnya perhatikanlah program di bawah ini.



Dari struktur heap di atas terlihat ada 2 kelompok variabel. Pertama adalah List/Senarai yang berisi kumpulan biodata (`nama`, `alamat`, `umur` dan `jenis kelamin`), dan yang kedua adalah dua buah rantai penghubung yaitu `kiri` dan `kanan` yang nantinya akan bertugas untuk merangkaikan setiap heap dengan heap yang lain. Pointer `kiri` nantinya akan bertugas menghubungkan heap dengan heap tetangga sebelah kiri sementara pointer `kanan` akan bertugas menghubungkan heap dengan tetangga heap sebelah kanan.

Pembahasan lebih lanjut tentang Double Linked List dapat dilihat pada bab 10

F. MEMANIPULASI ANGGOTA HEAP MILIK POINTER TETANGGA

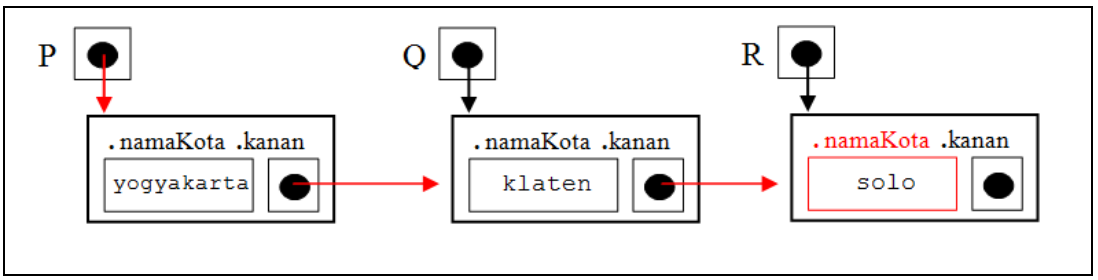
Pada heap-heap yang telah terhubung dalam sebuah linkedlist, sebuah elemen heap dapat diakses dari pointer lain yang bahkan tidak sedang menunjuk heap tersebut. Sebagai contoh perhatikan program di bawah ini.

```
import java.util.Scanner;
class tipePointer
{  String namaKota;
   tipePointer kanan;
}
class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P;
      P = new tipePointer();
```

```
      P.namaKota  = "Yogyakarta";

      tipePointer Q;
      Q = new tipePointer();
      Q.namaKota  = "Klaten";

      tipePointer R;
      R = new tipePointer();
      R.namaKota  = "Solo";

      P.kanan = Q;
      Q.kanan = R;
      R.kanan = null;
      System.out.println("Nilai elemen namaKota pointer P,Q dan R adalah :");
      System.out.println("-----------------------------------------------");
      System.out.println("Nilai P.namaKota adalah = " + P.namaKota);
      System.out.println("Nilai Q.namaKota adalah = " + Q.namaKota);
      System.out.println("Nilai R.namaKota adalah = " + R.namaKota);

      P.kanan.kanan.namaKota = "Surakarta"; <--mengakses R.namaKota dari pointer P

      System.out.println("Nilai elemen namaKota pointer P,Q dan R adalah :");
      System.out.println("-----------------------------------------------");
      System.out.println("Nilai P.namaKota adalah = " + P.namaKota);
      System.out.println("Nilai Q.namaKota adalah = " + Q.namaKota);
      System.out.println("Nilai R.namaKota adalah = " + R.namaKota);
  }
}

Hasil Eksekusi :
   Nilai elemen namaKota pointer P,Q dan R adalah :
   -----------------------------------------------
   Nilai P.namaKota adalah = Yogyakarta
   Nilai Q.namaKota adalah = Klaten
   Nilai R.namaKota adalah = Solo
   Nilai elemen namaKota pointer P,Q dan R adalah :
   -----------------------------------------------
   Nilai P.namaKota adalah = Yogyakarta
   Nilai Q.namaKota adalah = Klaten
   Nilai R.namaKota adalah = Surakarta
   Press any key to continue . . .
```

Pada program di atas terlihat ada upaya untuk mengganti elemen "*Solo*" yang sedang ditunjuk oleh pointer `R` menjadi "*Surakarta*" namun proses penggantiannya dilakukan melalui pointer `P` (perhatikan instruksi `P.kanan.kanan.namaKota = "Surakarta";`) Hal ini mengakibatkan nilai elemen `R.namaKota` akan berubah dari yang semula menampilkan "*Solo*" berubah menjadi "*Surakarta*".

Perhatikan ilustrasi berikut ini.



Pada gambar di atas, selain melalui pointer `R`, elemen `R.namaKota` yang berisi teks "*Solo*" dapat juga diakses melalui pointer "`P.kanan.kanan.namaKota`".

PRAKTIK

1. Praktek 1

Tuliskan program berikut ini menggunakan TextPad.

```
import java.util.Scanner;

class tipePointer
{  int angka;
   string teks;
}

class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P;
      P = null;
      System.out.println("Nilai P adalah = " + P);
  }
}
```

Running program di atas dan lihat apa hasil yang terjadi? Apakah nilai pointer P dapat dilihat?

Sekarang modifikasilah program di atas menjadi seperti berikut ini.

```
import java.util.Scanner;
class tipePointer
{  int angka;
   string teks;
}

class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P;
      P = null;
      if (P == null)
          System.out.println("Pointer P menunjuk ke Null");
      else
          System.out.println("Pointer P mengarah ke tempat lain");
  }
}
```

Sekarang runninglah program di atas dan lihat apa hasil yang terjadi? Bagaimana cara pointer P bisa menunjuk ke Null? Instruksi apa yang menyebabkannya pointer P menunjuk ke null? Apakah null dapat dilihat? Jelaskan dalam laporan anda.

2. Praktek 2

Tuliskan program berikut ini.

```
import java.util.Scanner;
class tipePointer
{  int angka;
   string teks;
}

class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P;
      P = new tipePointer();
      P.angka = 100;
      P.teks  = "Halo";

      tipePointer Q = new tipePointer();
      Q.angka = 200;
      Q.teks  = "Akakom";

      System.out.println("Nilai elemen P dan Q adalah :");
      System.out.println("----------------------------");
      System.out.println("Nilai P.angka adalah = " + P.angka);
      System.out.println("Nilai P.teks adalah = " + P.teks);
      System.out.println("Nilai Q.angka adalah = " + Q.angka);
      System.out.println("Nilai Q.teks adalah = " + Q.teks);
  }
}
```

Sekarang running lah program di atas dan lihat apa hasil yang terjadi? Berapa masing-masing nilai elemen angka dan teks, baik yang ditunjuk oleh pointer P maupun pointer Q? Jelaskan dalam laporan anda.

Sekarang modifikasilah program di atas dengan menambahkan pointer R seperti di bawah ini.

```
import java.util.Scanner;
class tipePointer
{  int angka;
   string teks;
}
class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P;
      P = new tipePointer();
      P.angka = 100;
      P.teks  = "Halo";

      tipePointer Q = new tipePointer();
      Q.angka = 200;
      Q.teks  = "Akakom";

      System.out.println("Nilai elemen P dan Q adalah :");
      System.out.println("----------------------------");
      System.out.println("Nilai P.angka adalah = " + P.angka);
      System.out.println("Nilai P.teks adalah = " + P.teks);
      System.out.println("Nilai Q.angka adalah = " + Q.angka);
      System.out.println("Nilai Q.teks adalah = " + Q.teks);


      tipePointer R;
      R = P;
```

```
      System.out.println("Nilai elemen R adalah :");
      System.out.println("------------------------");
      System.out.println("Nilai R.angka adalah = " + R.angka);
      System.out.println("Nilai R.teks adalah = " + R.teks);

      R = Q;
      System.out.println("Nilai elemen R saat ini adalah :");
      System.out.println("-------------------------------");
      System.out.println("Nilai R.angka adalah = " + R.angka);
      System.out.println("Nilai R.teks adalah = " + R.teks);
   }
}
```

Sekarang running lah program di atas dan lihat apa hasil yang terjadi? Berapa nilai elemen angka dan teks yang ditunjuk oleh pointer R? Bagaimana bisa nilai R dapat bisa menghasilkan nilai yang berbeda pada saat elemen angka dan teks dari R ditampilkan untuk kedua kalinya? Jelaskan dalam laporan anda.

```
import java.util.Scanner;
class tipePointer
{  int angka;
   string teks;
}

class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P = new tipePointer();
      P.angka = 100;
      P.teks  = "Halo";

      tipePointer Q = new tipePointer();
      Q.angka = 200;
      Q.teks  = "Akakom";

      System.out.println("Nilai P dan Q sebelum pointer dimanipulasi :");
      System.out.println("--------------------------------------------");
      System.out.println("Nilai P.angka adalah = " + P.angka);
      System.out.println("Nilai P.teks adalah = " + P.teks);

      System.out.println("Nilai Q.angka adalah = " + Q.angka);
      System.out.println("Nilai Q.teks adalah = " + Q.teks);

      tipePointer R;
      R = P;
      P = Q;
      Q = R;
      System.out.println("Nilai P dan Q setelah pointer dimanipulasi :");
      System.out.println("--------------------------------------------");
      System.out.println("Nilai P.angka adalah = " + P.angka);
      System.out.println("Nilai P.teks adalah = " + P.teks);

      System.out.println("Nilai Q.angka adalah = " + Q.angka);
      System.out.println("Nilai Q.teks adalah = " + Q.teks);
   }
}
```

Sekarang running lah program di atas dan lihat apa hasil yang terjadi? Berapa nilai masing-masing elemen angka dan teks, baik yang ditunjuk oleh pointer P maupun pointer Q sebelum pointer dimanipulasi maupun setelah dimanipulasi? Bagaimana itu bisa terjadi? Jelaskan dalam laporan anda.

3. Praktek 3

Tuliskan program berikut ini.

```
import java.util.Scanner;
class tipePointer
{  string namaKota;
   tipePointer kanan;
}

class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P;
      P = new tipePointer();
      P.namaKota  = "Yogyakarta";

      tipePointer Q;
      Q = new tipePointer();
      Q.namaKota  = "Klaten";

      tipePointer R;
      R = new tipePointer();
      R.namaKota  = "Solo";

      tipePointer S;
      S = new tipePointer();
      S.namaKota  = "Sragen";

      tipePointer T;
      T = new tipePointer();
      T.namaKota  = "Ngawi";

      System.out.println("Nilai P,Q,R,S,T adalah :");
      System.out.println("------------------------");
      System.out.println("Nilai P.namaKota adalah = " + P.namaKota);
      System.out.println("Nilai Q.namaKota adalah = " + Q.namaKota);
      System.out.println("Nilai R.namaKota adalah = " + R.namaKota);
      System.out.println("Nilai S.namaKota adalah = " + S.namaKota);
      System.out.println("Nilai T.namaKota adalah = " + T.namaKota);

      P.kanan = Q;
      Q.kanan = R;
      R.kanan = S;
      S.kanan = T;
      T.kanan = null;

      System.out.println("Nilai-nilai yang dapat diakses dari pointer P adalah :");
      System.out.println("-----------------------------------------------------");
      System.out.println(P.namaKota);
      System.out.println(P.kanan.namaKota);
      System.out.println(P.kanan.kanan.namaKota);
      System.out.println(P.kanan.kanan.kanan.namaKota);
      System.out.println(P.kanan.kanan.kanan.kanan.namaKota);
   }
}
```

Sekarang running lah program di atas dan lihat apa hasil yang terjadi? Berapa nilai masing-masing elemen namaKota untuk P, Q, R, S, T dengan dilakukannya deretan perintah :

```
System.out.println("Nilai P.namaKota adalah = " + P.namaKota);
System.out.println("Nilai Q.namaKota adalah = " + Q.namaKota);
System.out.println("Nilai R.namaKota adalah = " + R.namaKota);
System.out.println("Nilai S.namaKota adalah = " + S.namaKota);
System.out.println("Nilai T.namaKota adalah = " + T.namaKota);
```

Apa pula hasil yang diperoleh dengan dilakukannya deretan perintah :

```
System.out.println(P.namaKota);
System.out.println(P.kanan.namaKota);
System.out.println(P.kanan.kanan.namaKota);
```

```
System.out.println(P.kanan.kanan.kanan.namaKota);
System.out.println(P.kanan.kanan.kanan.kanan.namaKota);
```

Apa yang dapat anda simpulkan dari percobaan di atas?

Sekarang tambahkan perintah berikut ini di bagian terakhir dari program anda.

```
System.out.println(R.namaKota);
P.kanan.kanan.namaKota = "Surakarta");
System.out.println(R.namaKota);
```

Apa yang terjadi pada elemen R.namaKota ? Bagaimana nilai R.namaKota dapat berubah? Jelaskan dalam laporan anda.

Sekarang modifikasilah juga sehingga menjadi seperti program berikut ini.

```
import java.util.Scanner;

class tipePointer
{  string namaKota;
   tipePointer kanan;
}

class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P;
      P = new tipePointer();
      P.namaKota  = "Yogyakarta";

      tipePointer Q;
      Q = new tipePointer();
      Q.namaKota  = "Klaten";

      tipePointer R;
      R = new tipePointer();
      R.namaKota  = "Solo";

      tipePointer S;
      S = new tipePointer();
      S.namaKota  = "Sragen";

      tipePointer T;
      T = new tipePointer();
      T.namaKota  = "Ngawi";

      System.out.println("Elemen namaKota untuk pointer P,Q,R,S,T adalah :");
      System.out.println("-------------------------------------------------");
      System.out.println("Nilai P.namaKota adalah = " + P.namaKota);
      System.out.println("Nilai Q.namaKota adalah = " + Q.namaKota);
      System.out.println("Nilai R.namaKota adalah = " + R.namaKota);
      System.out.println("Nilai S.namaKota adalah = " + S.namaKota);
      System.out.println("Nilai T.namaKota adalah = " + T.namaKota);

      P.kanan = Q;
      Q.kanan = R;
      R.kanan = S;
      S.kanan = T;
      T.kanan = null;

      System.out.println("Elemen namaKota untuk pointer P,Q,R,S,T adalah :");
      System.out.println("-------------------------------------------------");
      System.out.println("Nilai P.namaKota adalah = " + P.namaKota);
      System.out.println("Nilai Q.namaKota adalah = " + P.kanan.namaKota);
      System.out.println("Nilai R.namaKota adalah = " + P.kanan.kanan.namaKota);
```

```
      System.out.println("Nilai       S.namaKota       adalah       =       "       +
P.kanan.kanan.kanan.namaKota);
   System.out.print("Nilai             T.namaKota             adalah=             "+
P.kanan.kanan.kanan.kanan.namaKota);

      tipePointer BANTU;
      BANTU = P;
      while (BANTU!=null)
      {  System.out.println("Nilai BANTU.namaKota adalah = " + BANTU.namaKota);
         BANTU = BANTU.kanan;
      }
  }
}
```

Sekarang running lah program di atas dan lihat apa yang terjadi dengan BANTU.kanan ketika ditampilkan? Berapa kalikah instruksi `System.out.println("Nilai BANTU.namaKota adalah = " + BANTU.namaKota);` dieksekusi oleh compiler? Selalu samakah hasilnya? Bagaimana ini bisa terjadi? Jelaskan dalam laporan anda.

Sekarang modifikasilah juga sehingga dengan menambahkan elemen "*kiri*" menjadi seperti program berikut ini.

```
import java.util.Scanner;

class tipePointer
{  string namaKota;
   tipePointer kiri;
   tipePointer kanan;
}

class belajarPointer
{  public static void main(String[] args)
   {
      tipePointer P;
      P = new tipePointer();
      P.namaKota  = "Yogyakarta";

      tipePointer Q;
      Q = new tipePointer();
      Q.namaKota  = "Klaten";

      tipePointer R;
      R = new tipePointer();
      R.namaKota  = "Solo";

      tipePointer S;
      S = new tipePointer();
      S.namaKota  = "Sragen";

      tipePointer T;
      T = new tipePointer();
      T.namaKota  = "Ngawi";

      System.out.println("Elemen namaKota untuk pointer P,Q,R,S,T adalah :");
      System.out.println("-------------------------------------------------");
      System.out.println("Nilai P.namaKota adalah = " + P.namaKota);
      System.out.println("Nilai Q.namaKota adalah = " + Q.namaKota);
      System.out.println("Nilai R.namaKota adalah = " + R.namaKota);
      System.out.println("Nilai S.namaKota adalah = " + S.namaKota);
      System.out.println("Nilai T.namaKota adalah = " + T.namaKota);

      P.kanan = Q;
      Q.kanan = R;
      R.kanan = S;
      S.kanan = T;
```

```
        T.kanan = null;

        P.kiri = null;
        Q.kiri = P;
        R.kiri = R;
        S.kiri = R;
        T.kiri = S;

        System.out.println("Elemen namaKota untuk pointer P adalah :");
        System.out.println("----------------------------------------");
        System.out.println(P.namaKota);
        System.out.println(P.kanan.kiri.namaKota);
        System.out.println(P.kanan.kanan.kiri.kiri.namaKota);

        tipePointer BANTU;
        BANTU = T;
        while (BANTU!=null)
        {  System.out.println("Nilai BANTU.namaKota adalah = " + BANTU.namaKota);
           BANTU = BANTU.kiri;
        }
    }
}
```

Sekarang running lah program di atas dan lihat apa yang terjadi dengan BANTU.kanan ketika ditampilkan? Berapa kalikah instruksi `System.out.println("Nilai BANTU.namaKota adalah = " + BANTU.namaKota);` dieksekusi oleh compiler? Selalu samakah hasilnya? Bagaimana ini bisa terjadi? Jelaskan dalam laporan anda.

LATIHAN

1. .....
2. .....

TUGAS

1. .....
2. .....

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 66-80, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 8
SINGLE LINKEDLIST (SENARAI BERANTAI TUNGGAL)

CAPAIANPEMBELAJARAN

Mahasiswa dapat mengimplementasikan penggunaan Single Linked List

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Untuk mempelajari bagaimana sebuah linkedlist terbentuk anda dapat mempelajari kembali pelajaran pada modul 7 mengenai Single Linkedlist.

Linked list (daftar yang terhubung) atau disebut juga *Senarai Berantai,* adalah pengalokasian memori secara dinamis agar dapat digunakan untuk menyimpan deretan data (list). Mengapa disebut dinamis? Karena alokasi memori ini dilakukan dengan membuat simpul (*heap)* di memori. Jika pada larik/ array kita hanya dapat menyimpan data hanya dalam jumlah tertentu saja dan tidak dapat diubah (statis), maka pada *linked list* kita dapat menyimpan data secara lebih dinamis karena pengalokasian simpul baru di memori baru akan dilakukan pada saat diperlukan.

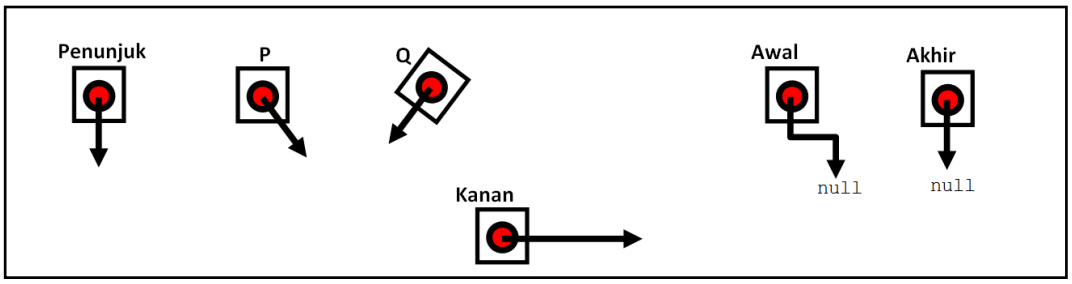
Single Linked List (Senarai Berantai Tunggal) adalah kumpulan heap/ obyek/ simpul/ node yang saling terhubung satu sama lain (*linked*) yang dimanfaatkan untuk menyimpan sederet data (*list*) dimana pada setiap heap yang ada terdapat **satu buah pointer anggota (*single*)** yang bertugas sebagai pointer pengait yang digunakan untuk mengkaitkan diri dengan simpul sejenis yang ada di sebelah kanannya. Kebanyakan orang menyingkat Single Linked List (Senarai Berantai Tunggal) hanya dengan sebutan Linked List (Senarai Berantai)

Di dalam konsep Single Linked List ada 3 unsur pendukung yang penting, yaitu :

1. **Penunjuk** (disebut juga dengan ***pointer***)
2. **Simpul** (disebut juga dengan ***list*** atau ***node*** *atau* ***heap***)
3. **Senarai Berantai Tunggal** atau Single Linked List.

<u>Penunjuk (pointer)</u>

Penunjuk atau pointer yang digunakan di sini sama dengan yang telah anda pelajari pada modul 7, Pada linkedlist, pointer akan dipergunakan untuk menunjuk sebuah simpul atau heap. Ilustrasi dari sebuah pointer dapat dilihat pada gambar 8.1 berikut.

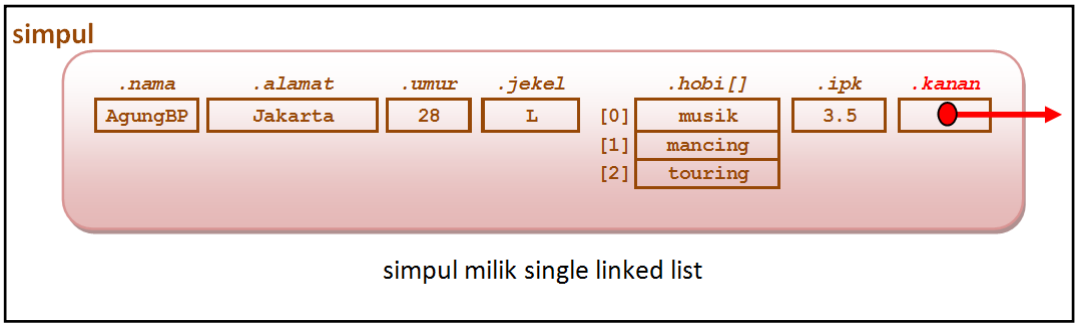


Gambar 8.1

Penunjuk dapat menunjuk ke sebuah sebuah simpul, ataupun ke sebuah tempat kosong (*null*).

Simpul (Node)

Simpul yang digunakan dalam bab ini adalah *simpul tunggal* yang diilustrasikan seperti gambar 8.2. *(Gambar 8.2 merupakan dimodifikasi dari Gambar 2.2 modul 2, dari yang sebelumnya bertipe /kelas "**biodataMahasiswa**" diganti menjadi tipe/ kelas "**simpul**", dengan tambahan anggota bernama "**kanan**")*



Gambar 8.2

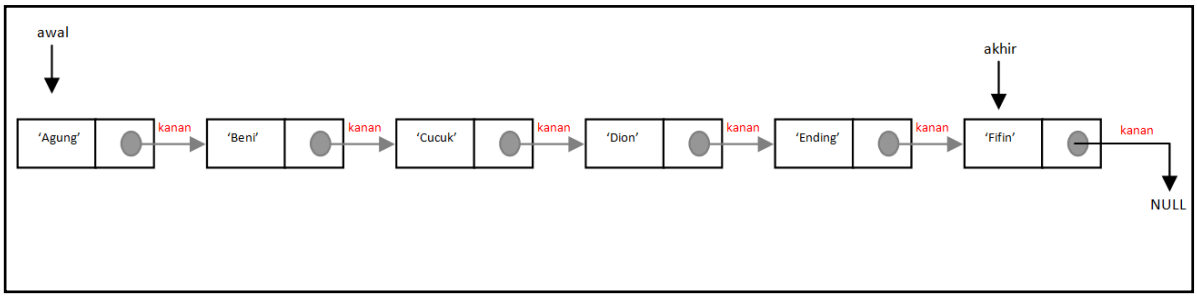
Simpul yang demikian disebut 'simpul tunggal' karena simpul ini hanya mempunyai sebuah penunjuk, yaitu 'kanan'. Artinya simpul ini hanya dapat menunjuk ke simpul yang serupa dengan dirinya yang letaknya disebelah kanannya.

Simpul serta penunjuknya dideklarasikan dengan cara :

```
class simpul
{   //bagian deklarasi struktur record ----------------------------
-
    String nama;
    String alamat;
    int    umur;
    char   jekel;
    String hobi[] = new String[3];
    float  ipk;
    simpul kanan;  //bagian yang ditambahkan dari program 2.2
}
```

Senarai Berantai Tunggal (Single Linked List)

Senarai Berantai Tunggal adalah kumpulan simpul-simpul yang terhubung satu dengan yang lain yang digambarkan sebagai berikut :



Gambar 8.3

Senarai Berantai Tunggal di deklarasikan dengan cara sebagai berikut :

```
class senarai
{ protected
        simpul awal;
        simpul akhir;
  public
        public static void inisialisasiSenarai()
        {   awal  = null;
            akhir = null;
        }
        public static void tambahDepan()
        {   ................        }
        public static void tambahBelakang()
        {   ................        }
        public static void tambahTengah()
        {   ................        }
        public static void cetakSenarai()
        {   ................        }
        public static hapus()
        {   ................        }
        public static void main()
        {   ................        }
}
```

Di dalam pengelolaan data menggunakan Single LinkedList terdapat beberapa operasi yang dapat dilakukan antara lain :

a. Operasi Tambah Data di Depan *(dipelajari pada Modul 8 ini)*
b. Operasi Tambah Data di Belakang *(dipelajari pada Modul 8 ini)*
c. Operasi Menampilkan (mencetak) semua Data *(dipelajari pada Modul 8 ini)*
d. Operasi Tambah Data di Tengah *(dipelajari pada Modul 9)*
e. Operasi Menghapus Data *(dipelajari pada Modul 9)*
f. Operasi Pencarian Data *(dipelajari pada Modul 11)*
g. Operasi Pengeditan Data *(dipelajari pada Modul 11)*
h. Operasi Pengurutan Data *(dipelajari pada Modul 11)*

Pada modul 8 ini kita akan mempelajari bersama operasi Tambah data di depan, belakang dan bagaimana menampilkan isi single linkedlist

PRAKTIK

1. Praktek 1

Tuliskan program berikut ini.

```
import java.util.Scanner;
class simpul
{  //bagian deklarasi struktur record ----------------------------------
   String nama;
   String alamat;
   int    umur;
   char   jekel;
   String hobi[] = new String[3];
   float  ipk;
   simpul kanan;
}

class singleLinkedList
{
   public static simpul awal;
   public static simpul akhir;

   public static void inisialisasiSenaraiKosong()
   {  awal  = null;
      akhir = null;
   }


   public static void tambahDepan()
   {
      //------------bagian entri data dari keyboard--------------
      String NAMA;
      String ALAMAT;
      int    UMUR;
      char   JEKEL;
      String HOBI[] = new String[3];
      float  IPK;
      Scanner masukan = new Scanner(System.in);
      int bacaTombol=0;

      System.out.println("TAMBAH DEPAN : ");
      System.out.print("Silakan masukkan nama anda : ");
      NAMA = masukan.nextLine();
      System.out.print("Silakan masukkan alamat anda : ");
      ALAMAT = masukan.nextLine();
      System.out.print("Silakan masukkan umur anda : ");
      UMUR = masukan.nextInt();
      System.out.print("Silakan masukkan Jenis Kelamin anda : ");
      try
      {  bacaTombol = System.in.read();
      }
      catch(java.io.IOException e)
      {
      }
      JEKEL = (char)bacaTombol;
      System.out.println("Silakan masukkan hobi (maks 3) : ");
      System.out.print("hobi ke-0 : ");
      HOBI[0] = masukan.next();
      System.out.print("hobi ke-1 : ");
      HOBI[1] = masukan.next();
      System.out.print("hobi ke-2 : ");
```

```
      HOBI[2] = masukan.next();
      System.out.print("Silakan masukkan IPK anda : ");
      IPK = masukan.nextFloat();

      //------------bagian menciptakan & mengisi simpul baru--------------
      simpul  baru;
      baru = new simpul();
      baru.nama    = NAMA;
      baru.alamat  = ALAMAT;
      baru.umur    = UMUR;
      baru.jekel   = JEKEL;
      baru.hobi[0] = HOBI[0];
      baru.hobi[1] = HOBI[1];
      baru.hobi[2] = HOBI[2];
      baru.ipk     = IPK;

      //---------bagian mencangkokkan simpul baru ke dalam simpul lama------------
      if (awal == null)       // jika senarai masih kosong
      {  awal  = baru;
         akhir = baru;
         baru.kanan = null;
      }
      else // jika senarai tidak kosong
      {  baru.kanan = awal;
         awal = baru;
      }
   }

   public static void tambahBelakang()
   {
      //------------bagian entri data dari keyboard--------------
      String NAMA;
      String ALAMAT;
      int    UMUR;
      char   JEKEL;
      String HOBI[] = new String[3];
      float  IPK;
      Scanner masukan = new Scanner(System.in);
      int bacaTombol=0;

      System.out.println("TAMBAH BELAKANG : ");
      System.out.print("Silakan masukkan nama anda : ");
      NAMA = masukan.nextLine();
      System.out.print("Silakan masukkan alamat anda : ");
      ALAMAT = masukan.nextLine();
      System.out.print("Silakan masukkan umur anda : ");
      UMUR = masukan.nextInt();
      System.out.print("Silakan masukkan Jenis Kelamin anda : ");
      try
      {  bacaTombol = System.in.read();
      }
      catch(java.io.IOException e)
      {
      }
      JEKEL = (char)bacaTombol;
      System.out.println("Silakan masukkan hobi (maks 3) : ");
      System.out.print("hobi ke-0 : ");
      HOBI[0] = masukan.next();
      System.out.print("hobi ke-1 : ");
      HOBI[1] = masukan.next();
      System.out.print("hobi ke-2 : ");
      HOBI[2] = masukan.next();
      System.out.print("Silakan masukkan IPK anda : ");
      IPK = masukan.nextFloat();

      //------------bagian menciptakan & mengisi simpul baru--------------
      simpul  baru;
      baru = new simpul();
      baru.nama    = NAMA;
      baru.alamat  = ALAMAT;
```

```
      baru.umur    = UMUR;
      baru.jekel   = JEKEL;
      baru.hobi[0] = HOBI[0];
      baru.hobi[1] = HOBI[1];
      baru.hobi[2] = HOBI[2];
      baru.ipk     = IPK;

      //----------bagian mencangkokkan simpul baru ke dalam simpul lama----------
      if (awal == null)        // jika senarai kosong
      {  awal  = baru;
         akhir = baru;
         baru.kanan = null;                    }
      else            // jika senarai tidak kosong
      {  akhir.kanan = baru;
         akhir = baru;
         baru.kanan = null;
      }
   }

   public static void cetakSenarai()
   {
      if (awal==null) // jika senarai masih kosong
         System.out.print("....MAAF SENARAI KOSONG....");
      else  // jika senarai tidak kosong
      {
         System.out.println("----------------------------------------------------");
         System.out.println("NO NAMA           ALAMAT       UMUR    JEKEL   IPK ");
         System.out.println("----------------------------------------------------");
         simpul bantu;
         bantu = awal;
         while (bantu != null)
         {
            System.out.print  (bantu.nama + "\t  ");
            System.out.print  (bantu.alamat + "\t  ");
            System.out.print  (bantu.umur + "\t  ");
            System.out.print  (bantu.jekel + "\t  ");
            System.out.print  (bantu.hobi[0] + "\t  ");
            System.out.print  (bantu.hobi[1] + "\t  ");
            System.out.print  (bantu.hobi[2] + "\t  ");
            System.out.println(bantu.ipk);
            bantu = bantu.kanan;
         }
         System.out.println("----------------------------------------------------");
      }
   }

   //---------bagian program utama-----------
   public static void main(String[] args)
   {
      inisialisasiSenaraiKosong();
      tambahDepan();
      tambahDepan();
      tambahDepan();
      tambahBelakang();
      tambahBelakang();
      tambahBelakang();
      cetakSenarai();
   }
}
```

Operasi-operasi pada Senarai Berantai Tunggal dapat dilakukan di sini.
(tambah depan, tengah, belakang, hapus, cetak, dll)

Program 8.1

Setelah anda tulis program di atas, tambahkanlah pada program utama (pada bagian yang dititik-titik) perintah-perintah berikut ini :

```
tambahDepan();
tambahDepan();
tambahDepan();
```

dengan mengisikan sembarang data mahasiswa. Catatlah urutan data yang anda entri. Kemudian tambahkanlah di bawahnya perintah untuk mencetak senarai berikut :

```
cetakSenarai();
```

Jalankanlah program dan amati apa yang terjadi pada hasil runningnya? Deretan *nama* siapa sajakah yang muncul lebih dahulu dan yang terkemudian? Bandingkan dengan urutan data yang anda entri sebelumnya. Samakah urutannya? Catatlah dan simpulkan dalam laporan anda.

2. Praktek 2

Sekarang, tambahkanlah perintah berikut ini dan letakkanlah tepat setelah perintah untuk mencetak senarai di atas.

```
tambahBelakang();
tambahBelakang();
tambahBelakang();
tambahBelakang();
```

lalu tambahkanlah lagi di bawahnya perintah untuk mencetak senarai sebagai berikut :

```
cetakSenarai();
```

kemudian jalankan program dan amati apa yang terjadi pada hasil runningnya? Deretan *nama* siapa sajakah yang muncul? Catatlah dan simpulkan dalam laporan anda.

LATIHAN

1. ……
2. …..

TUGAS

1. Dengan menggunakan data yang ada isikan, ilustrasikan/ Gambarkan proses menambah di depan dan menambah data di belakang. Lakukan hal tersebut pada kertas tersendiri dan ditulis tangan (bukan diketik/print).
2. …..

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 81-87, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 9
SINGLE LINKEDLIST (LANJUTAN)

CAPAIANPEMBELAJARAN

Mahasiswa dapat mengimplementasikan penggunaan Single Linked List

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Pada pertemuan terdahulu kita telah mempraktekkan penggunaan single linkedlist untuk mengelola data, khususnya untuk menginisialisasi linkedlist, menambah data di depan, menambah data di belakang, dan menampilkan isi linkedlist.

Pada pertemuan ini kita akan melanjutkan pengelolaan data dengan single linkedlist khususnya untuk menambah data di tengah dan menghapus data.

PRAKTIK

1. Praktek 1

Tuliskan program berikut ini menggunakan TextPad

```
import java.util.Scanner;
class simpul
{  //bagian deklarasi struktur record ----------------------------------
   String nama;
   String alamat;
   int    umur;
   char   jekel;
   String hobi[] = new String[3];
```

```
   float  ipk;
   simpul kanan;
}

class singleLinkedList
{
   public static simpul awal;
   public static simpul akhir;

   public static void inisialisasiSenaraiKosong()
   {  .......
   }

   public static void tambahDepan()
   {
      .........
   }

   public static void tambahBelakang()
   {
      ........
   }


   public static void cetakSenarai()
   {
      .........
   }

   public static int hitungJumlahSimpul()
   {  int N = 0;
      simpul bantu;
      bantu = awal;
      while (bantu!=null)
      {  N++;
         bantu = bantu.kanan;
      }
      return(N);
   }


   public static void tambahTengah()
   {
      Scanner masukan = new Scanner(System.in);
      System.out.println("Tentukan Lokasi Penambahan Data");
      int LOKASI = masukan.nextInt();

      int jumlahSimpulYangAda = hitungJumlahSimpul();
      if (LOKASI==1)
         System.out.println("Lakukan penambahan di depan");

      else if (LOKASI > jumlahSimpulYangAda)
         System.out.println("Lakukan penambahan di belakang");

      else
      {
         //------------bagian entri data dari keyboard--------------
         String NAMA;
         String ALAMAT;
         int    UMUR;
         char   JEKEL;
         String HOBI[] = new String[3];
         float  IPK;
         //Scanner masukan = new Scanner(System.in);
         int bacaTombol=0;

         System.out.println("TAMBAH TENGAH : ");
         System.out.print("Silakan masukkan nama anda : ");
         NAMA = masukan.nextLine();
         System.out.print("Silakan masukkan alamat anda : ");
```

```
        ALAMAT = masukan.nextLine();
        System.out.print("Silakan masukkan umur anda : ");
        UMUR = masukan.nextInt();
        System.out.print("Silakan masukkan Jenis Kelamin anda : ");
        try
        {  bacaTombol = System.in.read();
        }
        catch(java.io.IOException e)
        {
        }
        JEKEL = (char)bacaTombol;
        System.out.println("Silakan masukkan hobi (maks 3) : ");
        System.out.print("hobi ke-0 : ");
        HOBI[0] = masukan.next();
        System.out.print("hobi ke-1 : ");
        HOBI[1] = masukan.next();
        System.out.print("hobi ke-2 : ");
        HOBI[2] = masukan.next();
        System.out.print("Silakan masukkan IPK anda : ");
        IPK = masukan.nextFloat();

        //------------bagian menemukan posisi yang dikehendaki--------------
        simpul bantu;
        bantu = awal;
        int N = 1;
        while ((N<LOKASI-1) && (bantu!=akhir))
        {  bantu = bantu.kanan;
           N++;
        }

        //------------bagian menciptakan & mengisi simpul baru--------------
        simpul baru  = new simpul();
        baru.nama    = NAMA;
        baru.alamat  = ALAMAT;
        baru.umur    = UMUR;
        baru.jekel   = JEKEL;
        baru.hobi[0] = HOBI[0];
        baru.hobi[1] = HOBI[1];
        baru.hobi[2] = HOBI[2];
        baru.ipk     = IPK;

        //--------bagian mencangkokkan simpul baru ke dalam linkedlist lama------
        bantu.kanan = baru;
        baru.kanan  = bantu.kanan;
     }
   }

   //---------bagian program utama-----------
   public static void main(String[] args)
   {
      inisialisasiSenaraiKosong();
      tambahDepan();
      tambahDepan();
      tambahDepan();
      tambahDepan();
      tambahBelakang();
      tambahBelakang();
      tambahBelakang();
      tambahBelakang();
      tambahTengah();
      cetakSenarai();
  }
}
```

Lakukan penambahan ditengah disini.

Program 9.1

Setelah anda tulis program di atas, tambahkanlah pada program utama perintah untuk menambahkan didepan (4x) dan untuk menambahkan di belakang (4x) seperti berikut ini :

```
tambahDepan();
tambahDepan();
tambahDepan();
tambahDepan();
tambahBelakang();
tambahBelakang();
tambahBelakang();
tambahBelakang();
tambahTengah();
cetakSenarai();
```

Jalankanlah program dan amati apa yang terjadi pada hasil runningnya? Apakah data yang baru saja anda tambahkan di tengah linkedlist berhasil? Catatlah dan simpulkan dalam laporan anda.

2. Praktek 2

Sekarang kembangkan program anda seperti berikut.

```
import java.util.Scanner;
class simpul
{  //bagian deklarasi struktur record ----------------------------------
   String nama;
   String alamat;
   int    umur;
   char   jekel;
   String hobi[] = new String[3];
   float  ipk;
   simpul kanan;
}

class singleLinkedList
{
   public static simpul awal;
   public static simpul akhir;

   public static void inisialisasiSenaraiKosong()
   {  .......
   }

   public static void tambahDepan()
   {
      .........
   }

   public static void tambahBelakang()
   {
      ........
   }

   public static void cetakSenarai()
   {
      .........
   }

   public static int hitungJumlahSimpul()
   {  ...........
   }

   public static void tambahTengah()
   {
      ............
   }

   public static void hapus()
```

```
   {  if (awal == null) // jika senarai masih kosong
      {  System.out.println("senarai kosong, menghapus tidak dapat dilakukan");
      }
      else            // jika senarai tidak kosong
      {

         Scanner masukan = new Scanner(System.in);
         System.out.print("Silakan masukkan nama yang ingin dihapus : ");
         String NAMACARI = masukan.nextLine();

         if (awal == akhir) //jika hanya ada sebuah simpul
         {  if (awal.nama.equals(NAMACARI))
            {  System.out.println("menghapus "+NAMACARI+" dilakukan..");
               inisialisasiSenaraiKosong();
            }
            else
               System.out.println("data " +NAMACARI+" tidak ditemukan");
         }


         else if (awal.nama.equals(NAMACARI))//jika nama ditemukan di awal
         {  System.out.println("menghapus "+NAMACARI+" dilakukan..");
            awal = awal.kanan;
         }

         else
         {  simpul bantu;
            bantu = awal;
            while (bantu.kanan.nama.equals(NAMACARI)==false)
            {  bantu = bantu.kanan;
               if (bantu.kanan ==  null) break;
            }
            if ((bantu== akhir) && (akhir.nama.equals(NAMACARI)==false))
            {  System.out.println("data " +NAMACARI+" tidak ditemukan");
            }
            else if (akhir.nama.equals(NAMACARI))//jika nama ditemukan di akhir
            {  bantu.kanan = null;
               akhir = bantu;
            }
            else
            {  System.out.println("menghapus "+NAMACARI+" dilakukan..");
               bantu.kanan = bantu.kanan.kanan;
            }
         }
      }
   }


   //---------bagian program utama-----------
   public static void main(String[] args)
   {
      inisialisasiSenaraiKosong();
      tambahDepan();
      tambahDepan();
      tambahDepan();
      tambahDepan();
      tambahBelakang();
      tambahBelakang();
      tambahBelakang();
      tambahBelakang();
      tambahTengah();
      hapus();
      cetakSenarai();
   }
}
```

Lakukan penghapusan disini.

Program 9.2

Setelah anda tulis program di atas, tambahkanlah pada program utama perintah untuk menambahkan didepan (4x), untuk menambahkan di belakang (4x), dan menambahkan ditengah (1x) seperti berikut ini :

```
tambahDepan();
tambahDepan();
tambahDepan();
tambahDepan();
tambahBelakang();
tambahBelakang();
tambahBelakang();
tambahBelakang();
tambahTengah();
hapus();
cetakSenarai();
```

Jalankanlah program dan amati apa yang terjadi pada hasil runningnya? Apakah data yang baru saja anda hapus dari linkedlist berhasil? Catatlah dan simpulkan dalam laporan anda.

LATIHAN

1. ……
2. …..

TUGAS

1. Dengan menggunakan data yang ada isikan, ilustrasikanlah proses menambah data di tengah. Ilustrasikanlah juga proses menghapus data (hapus depan/ tengah/ belakang sangat tergantung oleh keadaan di manakah data anda ditemukan). Lakukah hal tersebut pada kertas tersendiri dan ditulis tangan (bukan diketik/print).
2.

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 88-93, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 10
DOUBLE LINKEDLIST (SENARAI BERANTAI GANDA)

CAPAIANPEMBELAJARAN

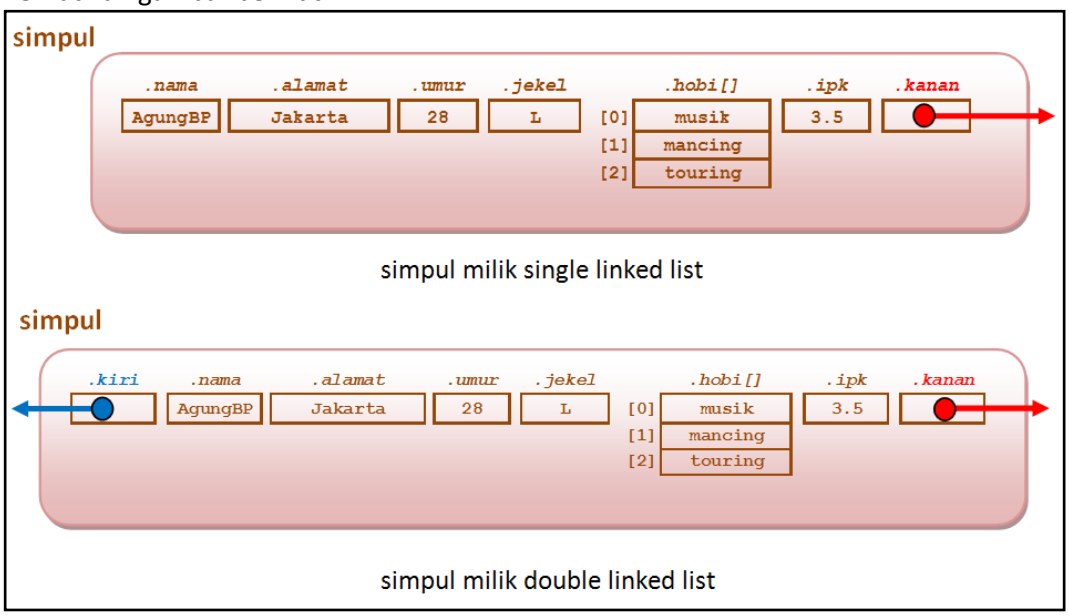
Mahasiswa dapat mengimplementasikan penggunaan Double Linked List

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Double Linked list atau disebut juga Senarai Berantai Ganda, hampir sama dengan *Single Linked List*. yaitu adalah pengalokasian memori secara dinamis. Bedanya adalah pada Double Linked List setiap simpul yang ada memiliki 2 buah penunjuk yang digunakan untuk mengkaitkan diri dengan simpul-simpul lain di kiri dan kanan. Perhatikah gambar berikut ini.



Gambar 10.1

Perbedaan lainnya adalah classnya (struktur)

Simpul milik Single Linkedlist :

```
class simpul
{   //bagian deklarasi struktur record -----------------------
    String nama;
    String alamat;
    int    umur;
    char   jekel;
    String hobi[] = new String[3];
    float  ipk;
    simpul kanan;
}
```

Simpul milik Double Linkedlist :

```
class simpul
{   //bagian deklarasi struktur record ----------------------
    String nama;
    String alamat;
    int    umur;
    char   jekel;
    String hobi[] = new String[3];
    float  ipk;
    simpul kiri;
    simpul kanan;
}
```

Oleh karena strukturnya kelasnya berbeda, maka metode dalam penanganan datanya pun juga ikut berubah seperti tambah depan, tengah belakang, hapus depan, tengah, belakang, dan lain-lain. Berikut ini akan disajikan programnya.

```
import java.util.Scanner;
class simpul
{  //bagian deklarasi struktur record ------------------------------
   String nama;
   String alamat;
   int    umur;
   char   jekel;
   String hobi[] = new String[3];
   float  ipk;
   simpul kiri;
   simpul kanan;
}

class senaraiGanda
{
   public static simpul awal;
   public static simpul akhir;

   public static void inisialisasiSenaraiKosong()
   {  awal  = null;
      akhir = null;
   }

   public static void tambahDepan()
   {  //------------bagian entri data dari keyboard--------------
      String NAMA;
      String ALAMAT;
      int    UMUR;
```

```
    char   JEKEL;
    String HOBI[] = new String[3];
    float  IPK;
    Scanner masukan = new Scanner(System.in);
    int bacaTombol=0;

    System.out.println("TAMBAH DEPAN : ");
    System.out.print("Silakan masukkan nama anda : ");
    NAMA = masukan.nextLine();
    System.out.print("Silakan masukkan alamat anda : ");
    ALAMAT = masukan.nextLine();
    System.out.print("Silakan masukkan umur anda : ");
    UMUR = masukan.nextInt();
    System.out.print("Silakan masukkan Jenis Kelamin anda : ");
    try
    {  bacaTombol = System.in.read();
    }
    catch(java.io.IOException e)
    {
    }
    JEKEL = (char)bacaTombol;
    System.out.println("Silakan masukkan hobi (maks 3) : ");
    System.out.print("hobi ke-0 : ");
    HOBI[0] = masukan.next();
    System.out.print("hobi ke-1 : ");
    HOBI[1] = masukan.next();
    System.out.print("hobi ke-2 : ");
    HOBI[2] = masukan.next();
    System.out.print("Silakan masukkan IPK anda : ");
    IPK = masukan.nextFloat();

    //------------bagian menciptakan & mengisi simpul baru--------------
    simpul  baru;
    baru = new simpul();
    baru.nama   = NAMA;
    baru.alamat = ALAMAT;
    baru.umur   = UMUR;
    baru.jekel  = JEKEL;
    baru.hobi[0] = HOBI[0];
    baru.hobi[1] = HOBI[1];
    baru.hobi[2] = HOBI[2];
    baru.ipk    = IPK;

    //--------bagian mencangkokkan simpul baru ke dalam simpul lama----------
    if (awal == null)       // jika senarai masih kosong
    {  awal  = baru;
       akhir = baru;
       baru.kiri = null;
       baru.kanan = null;
    }
    else // jika senarai tidak kosong
    {  baru.kanan = awal;
       awal.kiri  = baru;
       awal = baru;
       awal.kiri  = null;
    }
  }

  public static void tambahBelakang()
  {
    //------------bagian entri data dari keyboard--------------
    String NAMA;
    String ALAMAT;
    int    UMUR;
    char   JEKEL;
    String HOBI[] = new String[3];
    float  IPK;
    Scanner masukan = new Scanner(System.in);
    int bacaTombol=0;
```

```
    System.out.println("TAMBAH BELAKANG : ");
    System.out.print("Silakan masukkan nama anda : ");
    NAMA = masukan.nextLine();
    System.out.print("Silakan masukkan alamat anda : ");
    ALAMAT = masukan.nextLine();
    System.out.print("Silakan masukkan umur anda : ");
    UMUR = masukan.nextInt();
    System.out.print("Silakan masukkan Jenis Kelamin anda : ");
    try
    {  bacaTombol = System.in.read();
    }
    catch(java.io.IOException e)
    {
    }
    JEKEL = (char)bacaTombol;
    System.out.println("Silakan masukkan hobi (maks 3) : ");
    System.out.print("hobi ke-0 : ");
    HOBI[0] = masukan.next();
    System.out.print("hobi ke-1 : ");
    HOBI[1] = masukan.next();
    System.out.print("hobi ke-2 : ");
    HOBI[2] = masukan.next();
    System.out.print("Silakan masukkan IPK anda : ");
    IPK = masukan.nextFloat();

    //------------bagian menciptakan & mengisi simpul baru--------------
    simpul  baru;
    baru = new simpul();
    baru.nama   = NAMA;
    baru.alamat = ALAMAT;
    baru.umur   = UMUR;
    baru.jekel  = JEKEL;
    baru.hobi[0] = HOBI[0];
    baru.hobi[1] = HOBI[1];
    baru.hobi[2] = HOBI[2];
    baru.ipk    = IPK;

    //--------bagian mencangkokkan simpul baru ke dalam simpul lama---------
    if (awal == null)       // jika senarai kosong
    {  awal  = baru;
       akhir = baru;
       baru.kiri  = null;
       baru.kanan = null;
    }
    else            // jika senarai tidak kosong
    {  baru.kiri = akhir;
       akhir.kanan = baru;
       akhir = baru;
       akhir.kanan = null;
    }
  }

  public static int hitungJumlahSimpul()
  {  int N = 0;
     simpul bantu;
     bantu = awal;
     while (bantu!=null)
     {  N++;
        bantu = bantu.kanan;
     }
     return(N);
  }

  public static void tambahTengah()
  {
     //------------bagian menentukan lokasi target--------------
     Scanner masukan = new Scanner(System.in);
     System.out.println("Tentukan Lokasi Penambahan Data");
     int LOKASI = masukan.nextInt();
```

```
      int jumlahSimpulYangAda = hitungJumlahSimpul();
      if (LOKASI==1)
         System.out.println("Lakukan penambahan di depan");

      else if (LOKASI > jumlahSimpulYangAda)
         System.out.println("Lakukan penambahan di belakang");

      else
      {  //------------bagian entri data dari keyboard--------------
         String NAMA;
         String ALAMAT;
         int    UMUR;
         char   JEKEL;
         String HOBI[] = new String[3];
         float  IPK;
         //Scanner masukan = new Scanner(System.in);
         int bacaTombol=0;
         System.out.println("TAMBAH TENGAH : ");
         System.out.print("Silakan masukkan nama anda : ");
         NAMA = masukan.nextLine();
         System.out.print("Silakan masukkan alamat anda : ");
         ALAMAT = masukan.nextLine();
         System.out.print("Silakan masukkan umur anda : ");
         UMUR = masukan.nextInt();
         System.out.print("Silakan masukkan Jenis Kelamin anda : ");
         try
         {  bacaTombol = System.in.read();
         }
         catch(java.io.IOException e)
         {
         }
         JEKEL = (char)bacaTombol;
         System.out.println("Silakan masukkan hobi (maks 3) : ");
         System.out.print("hobi ke-0 : ");
         HOBI[0] = masukan.next();
         System.out.print("hobi ke-1 : ");
         HOBI[1] = masukan.next();
         System.out.print("hobi ke-2 : ");
         HOBI[2] = masukan.next();
         System.out.print("Silakan masukkan IPK anda : ");
         IPK = masukan.nextFloat();

         //------------bagian menemukan posisi yang dikehendaki-----------
         simpul bantu;
         bantu = awal;
         int i = 1;
         while ((i<LOKASI) && (bantu!=akhir))
         {  bantu = bantu.kanan;
            i++;
         }
         //------------bagian menciptakan & mengisi simpul baru-----------
         simpul baru = new simpul();
         baru.nama   = NAMA;
         baru.alamat = ALAMAT;
         baru.umur   = UMUR;
         baru.jekel  = JEKEL;
         baru.hobi[0] = HOBI[0];
         baru.hobi[1] = HOBI[1];
         baru.hobi[2] = HOBI[2];
         baru.ipk    = IPK;

         //------bagian mencangkokkan simpul baru ke dalam linkedlist lama------
         baru.kiri   = bantu.kiri;
         baru.kiri.kanan = baru;
         baru.kanan  = bantu;
         bantu.kiri = baru;
      }
   }

   public static void hapus()
```

```
{  if (awal == null) // jika senarai masih kosong
   {  System.out.println("senarai kosong, menghapus tidak dapat dilakukan");
   }
   else            // jika senarai tidak kosong
   {
      Scanner masukan = new Scanner(System.in);
      System.out.print("Silakan masukkan nama yang ingin dihapus : ");
      String NAMACARI = masukan.nextLine();

      if (awal == akhir) //jika hanya ada sebuah simpul
      {  if (awal.nama.equals(NAMACARI))
         {  System.out.println("menghapus "+NAMACARI+" dilakukan..");
            inisialisasiSenaraiKosong();
         }
         else
            System.out.println("data " +NAMACARI+" tidak ditemukan");
      }
      else if (awal.nama.equals(NAMACARI))//jika nama ditemukan di awal
      {  System.out.println("menghapus "+NAMACARI+" dilakukan..");
         awal = awal.kanan;
         awal.kiri = null;
      }
      else
      {  simpul bantu;
         bantu = awal.kanan;
         while (bantu.nama.equals(NAMACARI)==false)
         {  bantu = bantu.kanan;
            if (bantu.kanan ==  null) break;
         }
         if ((bantu == akhir) && (akhir.nama.equals(NAMACARI)==false))
         {  System.out.println("data " +NAMACARI+" tidak ditemukan");
         }
         else if (akhir.nama.equals(NAMACARI))//jika nama ditemukan di akhir
         {
            akhir = bantu.kiri;
            akhir.kanan = null;
         }
         else
         {  System.out.println("menghapus "+NAMACARI+" dilakukan..");
            bantu.kanan.kiri = bantu.kiri;
            bantu.kiri.kanan = bantu.kanan;
         }
      }
   }
}

public static void cetakSenaraiMaju()
{
   if (awal==null) // jika senarai masih kosong
      System.out.print("....MAAF SENARAI KOSONG....");
   else  // jika senarai tidak kosong
   {
      System.out.println("----------------------------------------------------");
      System.out.println("NO NAMA        ALAMAT          UMUR    JEKEL   IPK ");
      System.out.println("----------------------------------------------------");
      simpul bantu;
      bantu = awal;
      while (bantu != null)
      {
         System.out.print  (bantu.nama + "\t  ");
         System.out.print  (bantu.alamat + "\t  ");
         System.out.print  (bantu.umur + "\t  ");
         System.out.print  (bantu.jekel + "\t  ");
         System.out.print  (bantu.hobi[0] + "\t  ");
         System.out.print  (bantu.hobi[1] + "\t  ");
         System.out.print  (bantu.hobi[2] + "\t  ");
         System.out.println(bantu.ipk);
         bantu = bantu.kanan;
      }
      System.out.println("----------------------------------------------------");
```

```
      }
   }

   public static void cetakSenaraiMundur()
   {
      if (awal==null) // jika senarai masih kosong
         System.out.print("....MAAF SENARAI KOSONG....");
      else  // jika senarai tidak kosong
      {
         System.out.println("----------------------------------------------------");
         System.out.println("NO NAMA        ALAMAT          UMUR    JEKEL   IPK ");
         System.out.println("----------------------------------------------------");
         simpul bantu;
         bantu = akhir;
         while (bantu != null)
         {
            System.out.print  (bantu.nama + "\t  ");
            System.out.print  (bantu.alamat + "\t  ");
            System.out.print  (bantu.umur + "\t  ");
            System.out.print  (bantu.jekel + "\t  ");
            System.out.print  (bantu.hobi[0] + "\t  ");
            System.out.print  (bantu.hobi[1] + "\t  ");
            System.out.print  (bantu.hobi[2] + "\t  ");
            System.out.println(bantu.ipk);
            bantu = bantu.kiri;
         }
         System.out.println("----------------------------------------------------");
      }
   }

   public static void main(String[] args)
   {
      inisialisasiSenaraiKosong();
      tambahDepan()
      tambahDepan()
      tambahDepan()
      tambahBelakang()
      tambahBelakang()
      cetakSenaraiMaju()
      ..............
      ..............
      ..............
      ..............
      ..............
      ..............
      ..............
      ..............
      ..............
   }
}
```

Lakukan pemanggilan fungsi di sini

Program 10.1

PRAKTIK

1. Praktik 1

Tuliskan program 10.1 menggunakan TextPad kemudian lakukan penambahan data di depan, dibelakang dan di tengah. Lakukan juga menghapus data. Apakah berhasil? tunjukkan masing-masing hasil running outnya.

2. Praktik 2

Salah satu kelebihan double linkedlist adalah model linkedlist ini dapat mencetak baik secara maju maupun secara mundur. Tugas anda lakukan kedua kemampuan tersebut.

LATIHAN

1. ……
2. .....

TUGAS

1. Dengan menggunakan data yang ada isikan, ilustrasikanlah proses menambah data di depan, di tengah dan di belakang. Ilustrasikanlah juga proses menghapus data (hapus depan/ tengah/ belakang sangat tergantung oleh keadaan di manakah data anda ditemukan). Lakukah hal tersebut pada kertas tersendiri dan ditulis tangan (bukan diketik/print).

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 94-101, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 11
MENGURUTKAN DATA (SORTING) DAN PENCARIAN DATA (SEARCHING) PADA LINKEDLIST

CAPAIANPEMBELAJARAN

Mahasiswa dapat melakukan pengurutan terhadap suatu data yang terdapat di dalam linkedlist

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

A. SORTING PADA LINKEDLIST

Pada modul 5 kita yang lalu telah mempelajari bagaimana cara mengurutkan data yang tersimpan dalam sebuah sederet Larik/ array. Pada modul 11 ini kita akan mempelajari bagaimana melakukan cara mengurutkan data pada sebuah linkedlist.

Dalam prosesnya, pengurutan data dalam sebuah larik/array sangat mudah dilakukan. Hal ini karena pada struktur penyimpan larik terdapat indeks yang dapat membantu menandai data-data yang hendak dibandingkan/ ditukar.

Tidak seperti pada larik, pengurutan data dalam sebuah linkedlist lebih sulit dilakukan. Hal ini karena pada saat kita hendak membandingkan/ menukar data, tidak ada indeks yang dapat membantu untuk menandai data-data tersebut. Untuk itu diperlukan cara lain untuk menggantikan peran indeks tersebut.

Pada modul ini kita akan mempelajari bagaimana membuat program untuk mengurutkan data pada struktur penyimpan linkedlist. Metode pengurutan data yang akan kita gunakan adalah bubblesort, selection sort, dan insertion sort.

Sebelum kita mempelajari bagaimana program pengurutan pada linkedlist, perlu kita pahami terlebih dahulu bahwa di dalam banyak metode pengurutan data terdapat dua proses penting yang selalu ada, yaitu :

1. **membandingkan** dua buah data,
2. **menukar kedua data** tersebut jika diperlukan.

Pada proses membandingkan dua data, baik untuk struktur penyimpan array/larik maupun linkedlist, cara yang diterapkan cenderung sama, yaitu sama-sama harus ditemukan dahulu data yang dimaksud. (pada metode yang berbeda akan berbeda pula cara menemukannya).

Dalam proses menukar data, pada array/larik penukaran data hanya dapat dilakukan dengan menukar isi atau nilai (value) dari variabel yang menyimpan data-data tersebut. Sementara pada linkedlist penukaran data dapat dilakukan dengan 2 pendekatan :

1. Menukar isi variabelnya (yaitu dengan cara menukar isi heap namun posisi heap tidak berubah)
2. Menukar posisi heap (tukar heap, yaitu dengan merubah posisi heap/ simpul-simpul dalam linkedlist)

Pada modul ini akan kita pelajari bagaimana melakukan penukaran data dalam linkedlist dengan kedua teknik di atas.

Metode Sorting	**Single LinkedList**		**Double LinkedList**	
	Tukar Nilai	**Tukar Heap**	**Tukar Nilai**	**Tukar Heap**
Bubble Sort	*ada*	*ada*	*ada*	*ada*
Selection Sort	*ada*	-	*ada*	*ada*
Insertion Sort	-	*ada*	-	*ada*

a. Bubble Sort

Algoritma bubble sort akan membandingkan elemen yang saat ini dibaca dengan elemen yang berikutnya. Jika elemen yang saat ini dibaca lebih besar dari elemen berikutnya,maka tukarkan.

Berikut ini akan dibahas kedua pendekatan yang dimaksud untuk metode Bubble Sort.

Pendekatan pertama dimana proses penukaran data hanya dilakukan dengan cara menukar isi variabelnya saja. Program 11.1 berikut adalah fungsi pengurutan bubblesort **untuk single dan double linkedlist** dengan cara **menukar isi variabel**.

```
public static void tukarNilai( simpul X, simpul Y)
{   simpul sementara = new simpul();

    sementara.nama   = X.nama;
    sementara.alamat = X.alamat;
    sementara.umur   = X.umur;
    sementara.jekel  = X.jekel;
    sementara.ipk    = X.ipk;

    X.nama   = Y.nama;
    X.alamat = Y.alamat;
    X.umur   = Y.umur;
    X.jekel  = Y.jekel;
    X.ipk    = Y.ipk;
```

```
        Y.nama   = sementara.nama;
        Y.alamat = sementara.alamat;
        Y.umur   = sementara.umur;
        Y.jekel  = sementara.jekel;
        Y.ipk    = sementara.ipk;
    }


    //-------------------------------------------------
    // bisa untuk single LL dan double LL
    //-------------------------------------------------
    public static void mengurutkanDataBubble_TeknikTukarNilai()
    {
        int N = hitungJumlahSimpul();
        simpul A=null;
        simpul B=null;
        simpul berhenti = akhir.kanan;

        System.out.println ("Banyaknya simpul = " + hitungJumlahSimpul());
        for (int i=1;  i<= hitungJumlahSimpul()-1; i++)
        {
            A = awal;
            B = awal.kanan;
            int nomor = 1;
            while (B != berhenti)
            {   if (A.nama.compareTo(B.nama)>0)
                {
                   //tukarkan elemen dari simpul A dan elemen dari simpul B
                   tukarNilai(A,B);
                }
                A = A.kanan;
                B = B.kanan;
                nomor++;
            }
            berhenti = A;
        }
        System.out.println("===PROSES PENGURUTAN BUBBLE SELESAI======");
    }
```

Program 11.1. Fungsi Bubblesort untuk **single linkedlist dan double linkedlist** dengan **penukaran isi variabel**

Pendekatan kedua dimana proses penukaran data dilakukan dengan cara menukar posisi heapnya. Program 11.2 berikut adalah fungsi pengurutan bubblesort **untuk single linkedlist** dengan cara **menukar posisi heapnya**.

```
    public static void mengurutkanDataBubble_TeknikTukarHeap()
    {
        int N = hitungJumlahSimpul();
        simpul A=null;
        simpul B=null;
        simpul bantu=null;
        simpul berhenti = akhir.kanan;
        int nomor;

        System.out.println ("Banyaknya simpul = " + hitungJumlahSimpul());
        for (int i=1;  i<= hitungJumlahSimpul()-1; i++)
        //for (int i=1;  i<= 4; i++)
        {
            A = awal;
            B = awal.kanan;
```

```
            nomor = 1;

            //proses banding-tukar, khusus simpul pertama dgn sebelahnya
            if (A.nama.compareTo(B.nama)> 0)
            {
                A.kanan = B.kanan;
                B.kanan = A;
                awal = B;
            }

            //proses banding-tukar, simpul kedua dgn sebelahnya, dst
            nomor++;
            bantu = awal;
            while (bantu.kanan.kanan!=berhenti)
            {
                A = bantu.kanan;
                B = bantu.kanan.kanan;
                if (A.nama.compareTo(B.nama)>0)
                {
                    //tukarkan simpul A dan simpul B
                    A.kanan = B.kanan;
                    B.kanan = A;
                    bantu.kanan = B;
                    if (B==akhir) akhir = A;
                }
                bantu = bantu.kanan;
                nomor++;
            }
            berhenti = bantu.kanan;;
            System.out.println ("");
        }
        System.out.println("===PROSES PENGURUTAN BUBBLE SELESAI======");
    }
```

Program 11.2. Fungsi Bubblesort untuk **single linkedlist** dengan penukaran posisi heap

Pendekatan kedua dimana proses penukaran data dilakukan dengan cara menukar posisi heapnya. Program 11.3 berikut adalah fungsi pengurutan bubblesort **untuk double linkedlist** dengan cara **menukar posisi heapnya**.

```
    public static void mengurutkanDataBubble_TeknikTukarHeap()
    {
        int N = hitungJumlahSimpul();
        simpul bantu = awal;

        System.out.println ("Banyaknya simpul = " + hitungJumlahSimpul());

        for (int i=1;  i<= hitungJumlahSimpul(); i++)
        {
            //khusus menguji simpul pertama dgn sebelahnya
            if (awal.nama.compareTo(awal.kanan.nama)> 0)
            {
                bantu = awal.kanan;
                awal.kanan = bantu.kanan;
                bantu.kanan.kiri = awal;
                bantu.kanan = awal;
                bantu.kiri = null;
                awal.kiri = bantu;
                awal = bantu;
            }
```

```
            //khusus menguji simpul kedua dgn sebelahnya, simpul ketiga dgn
               sebelahnya,dst,
            bantu = awal;
            while (bantu.kanan != akhir)
            {   simpul A = bantu.kanan;
                simpul B = bantu.kanan.kanan;
                if (A.nama.compareTo(B.nama)>0)
                {
                    //tukarkan simpul A dan simpul B
                    A.kanan = B.kanan;
                    if (B!=akhir) A.kanan.kiri = A;

                    B.kanan.kiri = A;
                    B.kanan = A;
                    A.kiri = B;

                    bantu.kanan = B;
                    B.kiri = bantu;

                    if (B==akhir) akhir = A;
                }
                bantu = bantu.kanan;
            }
            System.out.println ("");
        }
        System.out.println("===PROSES PENGURUTAN BUBBLE SELESAI=====");
    }
```

Program 11.3. Fungsi Bubblesort untuk **double linkedlist** dengan penukaran posisi heap

b. Selection Sort (ada pada lampiran modul ini)
c. Insertion Sort (ada pada lampiran modul ini)

B. SEARCHING PADA LINKEDLIST

Pada modul 4 kita telah mempelajari topik tentang pencarian data dalam sebuah sederet Larik/ array. Pada modul 11 ini kita juga akan mempelajari bagaimana melakukan pencarian data yang tersimpan dalam sebuah linkedlist.

Pada searching linkedlist, pencarian yang dapat dilakukan adalah jenis pencarian Linear Search atau Sequential Search. Hal ini karena pencarian linear search dilakukan dengan menyisir data dari posisi data pertama hingga data terakhir. Sementara pencarian biner (binary search) cenderung sulit dilakukan karena metode bagi dua yang menjadi ciri khas metoden ini sulit diimplementasikan pada linkedlist.

Pada bagian ini akan kita pelajari bagaimana melakukan pencarian data dalam linkedlist metode linear search. Program 11.4 menjelaskan langkah-langkah yang dilakukan untuk pencarian linear menggunakan single linkedlist.

```
    public static void cariLinear()
    {
        if (awal==null) // jika senarai masih kosong
            System.out.print("....MAAF SENARAI KOSONG....");
        else  // jika senarai tidak kosong
        {
            Scanner masukan = new Scanner(System.in);
            System.out.print("Silakan masukkan nama yang anda cari : ");
            String NAMACARI = masukan.nextLine();
```

```
        boolean statusKetemu = false;
        int i = 0;
        int posisiKetemu=-1;

        simpul bantu;
        bantu = awal;
        while (bantu != null)
        {
            if (NAMACARI.equals(bantu.nama))
            {   statusKetemu = true;
                posisiKetemu = i;
            }
            bantu = bantu.kanan;
            i++;
        }
        System.out.println("Status Ketemu = "+statusKetemu +" di posisi
                ke "+posisiKetemu);
    }
}
```

Program 11.4. Fungsi Linear Search pada **single linkedlist**

PRAKTIK

1. Praktik 1

Tambahkan program 11.1 hingga 11.4 ke dalam master program yang telah anda buat pada praktikum yang lalu. Perhatikan mana progam yang digunakan untk single linklist dan mana yang untuk double linklist (jangan tertukar).

Eksekusi master program anda untuk melakukan pengurutan data secara ***Bubblesort.*** Ujilah program dengan mengentri data secara acak (nama mahasiswa dientri tidak dalam keadaan urut) kemudian lakukan pengurutan terhadap data tersebut, kemudian tampilkan data kembali. Bagaimana hasilnya? Catat dan simpulkan dalam laporan anda.

Eksekusi master program anda untuk melakukan pencarian data secara linear search. Ujilah juga dengan mengentri data secara acak kemudian lakukan pencarian sembarang terhadap data tersebut, kemudian tampilkan hasilnya.

Catat dan simpulkan percobaan anda hari ini dalam laporan anda

LATIHAN

1. .....
2. .....

TUGAS

1. ……
2.

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 102-107, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

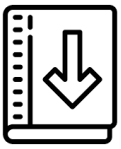
MODUL 12
COLLECTION

CAPAIANPEMBELAJARAN

Mahasiswa dapat mengimplementasikan penggunaan kelas collection

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad



DASAR TEORI

Collections secara umum memiliki makna adalah sebuah ***kumpulan.*** Dalam OOP collection dikenal sebagai suatu tempat atau wadah atau object yang dapat menyimpan object lainnya baik yang memiliki tipe data yang sama maupun tidak. Adapun beberapa contoh implementasi dari collection adalah Vector, Arraylist dan List.

<root interface> Collection
- a. <interface> Set
  - <implementing classes>
    - i. HashSet
    - ii. LinkedHashSet
    - iii. TreeSet
- b. <interface> List
  - <implementing classes>
    - i. ArrayList
    - ii. LinkedList
    - iii. Vector

Gambar 12.1. Implementasi Interface Collection

PRAKTIK

1. Praktik 1

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru** ?

```
import java.util.LinkedList;
public class PercobaanLinkedList
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      if (daftarMhs.isEmpty()==true)
      {  System.out.println("Linkedlist kosong");
      }
      else
      {  System.out.println("Linkedlist isi");
      }
   }

   public static void main(String args[])
   {
       new PercobaanLinkedList();
   }
}
```

2. Praktik 2

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru**?

```
import java.util.LinkedList;
public class PercobaanLinkedList
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      if (daftarMhs.isEmpty()==true)
      {  System.out.println("Linkedlist kosong");
      }
      else
      {  System.out.println("Linkedlist isi");
      }
```

```
      daftarMhs.add("Agungbp");
      daftarMhs.add("Bambang");


      if (daftarMhs.isEmpty()==true)
      {  System.out.println("Linkedlist kosong");
      }
      else
      {  System.out.println("Linkedlist isi");
      }
   }

   public static void main(String args[])
   {
       new PercobaanLinkedList();
   }
}
```

3. Praktik 3

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru**?

```
import java.util.LinkedList;

import java.util.LinkedList;
public class PercobaanLinkedList
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      daftarMhs.add("Agungbp");
      daftarMhs.add("Bambang");
      daftarMhs.add("Cucuk");
      daftarMhs.add("Dion");
      daftarMhs.add("Ending");
      daftarMhs.add("Fifin");
      daftarMhs.add("Gesit");
      daftarMhs.add("Heru");
      daftarMhs.add("Irma");
      daftarMhs.add("Janti");
      System.out.println(daftarMhs.get(0));
      System.out.println(daftarMhs.get(1));
      System.out.println(daftarMhs.get(2));
      System.out.println(daftarMhs.get(3));
      System.out.println(daftarMhs.get(4));
      System.out.println(daftarMhs.get(5));
      System.out.println(daftarMhs.get(6));
      System.out.println(daftarMhs.get(7));
      System.out.println(daftarMhs.get(8));
      System.out.println(daftarMhs.get(9));
      System.out.println("");
```

```
    }

    public static void main(String args[])
    {
        new PercobaanLinkedList();

    }
}
```

4. Praktik 4

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru**?

```
import java.util.LinkedList;
public class PercobaanLinkedList
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      daftarMhs.add("Agungbp");
      daftarMhs.add("Bambang");
      daftarMhs.add("Cucuk");
      daftarMhs.add("Dion");
      daftarMhs.add("Ending");
      daftarMhs.add("Fifin");
      daftarMhs.add("Gesit");
      daftarMhs.add("Heru");
      daftarMhs.add("Irma");
      daftarMhs.add("Janti");

      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");
   }

   public static void main(String args[])
   {
       new PercobaanLinkedList();
   }
}
```

5. Praktik 5

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru**?

```
import java.util.LinkedList;
public class PercobaanLinkedList
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      daftarMhs.add("Agungbp");
      daftarMhs.add("Bambang");
      daftarMhs.add("Cucuk");
      daftarMhs.add("Dion");
      daftarMhs.add("Ending");
      daftarMhs.add("Fifin");
      daftarMhs.add("Gesit");
      daftarMhs.add("Heru");
      daftarMhs.add("Irma");
      daftarMhs.add("Janti");

      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");


      daftarMhs.addFirst("Amir");
      daftarMhs.addLast("Zaenal");

      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");

      System.out.println("Mhs awal = " + daftarMhs.getFirst());
      System.out.println("Mhs akhir = " + daftarMhs.getLast());
   }

   public static void main(String args[])
   {
       new PercobaanLinkedList();
   }
}
```

6. Praktik 6

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru**?

```
import java.util.LinkedList;
public class PercobaanLinkedList
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      daftarMhs.add("Agungbp");
      daftarMhs.add("Bambang");
      daftarMhs.add("Cucuk");
      daftarMhs.add("Dion");
      daftarMhs.add("Ending");
      daftarMhs.add("Fifin");
      daftarMhs.add("Gesit");
      daftarMhs.add("Heru");
      daftarMhs.add("Irma");
      daftarMhs.add("Janti");
      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");

      daftarMhs.remove(5);
      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");

      daftarMhs.removeFirst();
      daftarMhs.removeLast();

      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");
   }

   public static void main(String args[])
   {
       new PercobaanLinkedList();
   }
}
```

7. Praktik 7

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru**?

```
import java.util.LinkedList;
public class PercobaanLinkedList
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      daftarMhs.add("Agungbp");
      daftarMhs.add("Bambang");
      daftarMhs.add("Cucuk");
      daftarMhs.add("Dion");
      daftarMhs.add("Ending");
      daftarMhs.add("Fifin");
      daftarMhs.add("Gesit");
      daftarMhs.add("Heru");
      daftarMhs.add("Irma");
      daftarMhs.add("Janti");
      System.out.println("");
      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");

      daftarMhs.set(0,"Parmin");

      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");
   }

   public static void main(String args[])
   {
       new PercobaanLinkedList();
   }
}
```

8. Praktik 8

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru**?

```
import java.util.LinkedList;
public class PercobaanLinkedList
```

```
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      daftarMhs.add("Agungbp");
      daftarMhs.add("Bambang");
      daftarMhs.add("Cucuk");
      daftarMhs.add("Dion");
      daftarMhs.add("Ending");
      daftarMhs.add("Fifin");
      daftarMhs.add("Gesit");
      daftarMhs.add("Heru");
      daftarMhs.add("Irma");
      daftarMhs.add("Janti");
      System.out.println("N = " + daftarMhs.size());
      for (int i=0;  i<= daftarMhs.size()-1;  i++)
      {
         System.out.println(i + "  " + daftarMhs.get(i));
       }
      System.out.println("");

      System.out.println(daftarMhs.contains("Heru"));
      System.out.println("Heru berada di posisi ke = " +
                                 daftarMhs.indexOf("Heru"));
      System.out.println("");
   }

   public static void main(String args[])
   {
       new PercobaanLinkedList();
   }
}
```

9. Praktik 9

Tuliskan program berikut ini dan eksekusilah. Kemudian jelaskan apa maksud dari perintah (method) yang ditandai **cetak tebal** dengan **warna biru**?

```
import java.util.LinkedList;
public class PercobaanLinkedList
{
   LinkedList daftarMhs = new LinkedList();

   PercobaanLinkedList()
   {
      daftarMhs.add("Agungbp");
      daftarMhs.add("Bambang");
      daftarMhs.add("Cucuk");
      daftarMhs.add("Dion");
      daftarMhs.add("Ending");
      daftarMhs.add("Fifin");
      daftarMhs.add("Gesit");
```

```
        daftarMhs.add("Heru");
        daftarMhs.add("Irma");
        daftarMhs.add("Janti");

        System.out.println("N = " + daftarMhs.size());
        for (int i=0;  i<= daftarMhs.size()-1;  i++)
        {
           System.out.println(i + "  " + daftarMhs.get(i));
         }
        System.out.println("");

        daftarMhs.clear();
        System.out.println("Clear selesai dijalankan... ");

        System.out.println("N = " + daftarMhs.size());
        for (int i=0;  i<= daftarMhs.size()-1;  i++)
        {
           System.out.println(i + "  " + daftarMhs.get(i));
         }
        System.out.println("");


        if (daftarMhs.isEmpty()==true)
        {  System.out.println("Linkedlist kosong");
        }
        else
        {  System.out.println("Linkedlist isi");
        }
    }

    public static void main(String args[])
    {
        new PercobaanLinkedList();
    }
}
```

LATIHAN

1. Latihan 1

   Tambahkan nilai dibawah pada list yang anda buat

   ```
   list.add(new Integer(10));
   ```

   Tambahkan juga untuk nilai Float dan Long

2. Latihan 2

Modifikasi program Pelaksanaan Praktikum 2 dengan menggunakan ArrayList

TUGAS

1. Apa perbedaan Vektor dan ArrayList pada packet java.util
2. Bagaimana perintah untuk menambah, menghapus dan menampilkan data pada List

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 108-116, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 13
POHON BINER

CAPAIANPEMBELAJARAN

1. Mahasiswa dapat mengimplementasikan penggunaan Simpul milik Double Linked List untuk pembuatan pohon

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Pohon (*tree*) adalah kumpulan akar (*root*) , cabang, dan simpul (*node*) yang saling terhubung secara hirarki

Pohon Biner (*binary tree*) adalah pohon dimana setiap simpulnya (node) hanya boleh memiliki **maksimal** 2 anak (dari cabang kiri, dan kanan)

Simpul (*node*) :

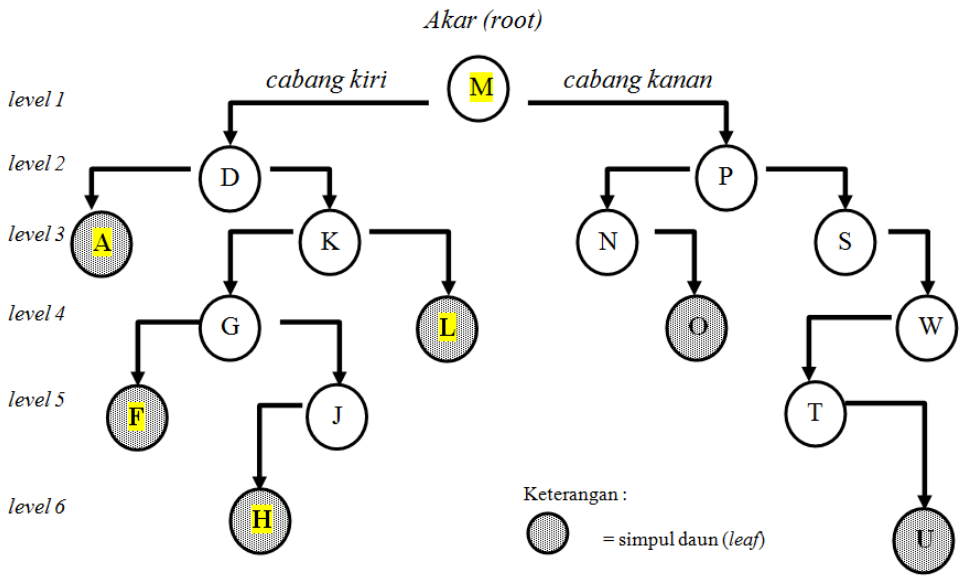
- simpul anak (*children*) hanya boleh punya 1 parent
- simpul orangtua (*parent*) hanya boleh punya **maksimal** 2 anak, namun boleh juga tidak punya anak

Simpul ***Akar*** adalah simpul yang tidak memiliki *parent*

Simpul ***Daun*** adalah simpul yang tidak memiliki anak (*children*)

Cabang di dalam pohon biner terdiri dari **cabang kiri**, **cabang kanan**

Level menunjukkan tingkat hirarki



Simpul yang digunakan untuk membentuk sebuah pohon sama dengan simpul yang digunakan pada Senarai Berantai Ganda, yaitu sebagai berikut :

```
class simpul
{   String elemen;
    simpul kiri;
    simpul kanan;
}
```

OPERASI PADA POHON

A. Menambah sebuah simpul ke dalam pohon

Algoritmanya :

A. Baca ***ElemenBaru***

B. Jika pohon masih kosong :
   → jadikan **ElemenBaru** sebagai akar, menuju langkah D.

C. Jika pohon tidak kosong :

   C.1. **PENUNJUK** = akar

   C.2. Baca **PENUNJUK**

   C.3. Jika **ElemenBaru** < **PENUNJUK**:

C3.a. jika **PENUNJUK** tidak punya anak di cabang kiri, jadikan **ElemenBaru** sebagai Anak di cabang kiri dari **PENUNJUK**. Menuju langkah D.

C3.b jika **PENUNJUK** punya anak di cabang kiri, jadikan anak cabang kiri sebagai **PENUNJUK**, ulangi langkah C.2

C.4. Jika **ElemenBaru** >= **PENUNJUK**:

C4.a. jika **PENUNJUK** tidak punya anak di cabang kanan, jadikan **ElemenBaru** sebagai Anak di cabang kanan dari **PENUNJUK**. Menuju langkah D.

C4.b. jika **PENUNJUK** punya anak di cabang kanan, jadikan anak cabang kanan sebagai **PENUNJUK**, ulangi langkah C.2

D. Selesai.

```
class pohon
{  public static simpul akar;

   public static void deklarasiPohon()
   {  akar = null;
   }
   public static simpul tambahSimpul(simpul Penunjuk, String ELEMEN)
   {  if (Penunjuk == null)
      {  simpul baru = new simpul();
         baru.elemen = ELEMEN;
         baru.kiri = null;
         baru.kanan = null;
         Penunjuk = baru;
         return(Penunjuk);
      }
      else
      {  if (ELEMEN.compareTo(Penunjuk.elemen) < 0 )
         {  Penunjuk.kiri  = tambahSimpul(Penunjuk.kiri, ELEMEN);
            return(Penunjuk);
         }
         else
         {  Penunjuk.kanan= tambahSimpul(Penunjuk.kanan, ELEMEN);
            return(Penunjuk);
         }
      }
   }
   ...............;
   ...............;

   public static void main(String[] args)
   {
      deklarasiPohon();
      ...............;
      ...............;
   }
}
```

B. Mencetak Isi Pohon

Mencetak isi pohon (kunjungan) dilakukan dengan cara rekursif yang terdiri dari :

1. **Preorder** adalah mencetak isi pohon dengan urutan :
    1. **Cetak** isi node yang dikunjungi
    2. kunjung Anak Cabang Kiri
    3. kunjungi Anak Cabang Kanan
2. **In Order** adalah mencetak isi pohon dengan urutan :
    1. kunjung Anak Cabang Kiri
    2. **Cetak** isi node yang dikunjungi
    3. kunjungi Anak Cabang Kanan
3. **Post Order** adalah mencetak isi pohon dengan urutan :
    1. kunjung Anak Cabang Kiri
    2. kunjungi Anak Cabang Kanan
    3. **Cetak** isi node yang dikunjungi

```
class pohon
{  ...............;
   ...............;
   public static void preOrder(simpul Penunjuk)
   {  if(Penunjuk != null)
      {  System.out.print(Penunjuk.elemen  + ",");
         preOrder(Penunjuk.kiri);
         preOrder(Penunjuk.kanan);
      }
   }
   public static void inOrder(simpul Penunjuk)
   {  if(Penunjuk != null)
      {  inOrder(Penunjuk.kiri);
         System.out.print(Penunjuk.elemen  + ",");
         inOrder(Penunjuk.kanan);
      }
   }
   public static void postOrder(simpul Penunjuk)
   {  if(Penunjuk != null)
      {  postOrder(Penunjuk.kiri);
         postOrder(Penunjuk.kanan);
         System.out.print(Penunjuk.elemen  + ",");
      }
   }
   ...............;
   ...............;
   public static void main(String[] args)
   {
      deklarasiPohon();
      ...............;
      ...............;
   }

}
```

PRAKTIK

1. Praktik 1

Tuliskan program berikut ini menggunakan TextPad

```
class simpul
{   .....
}
class pohon
{   .....
    .....
    void deklarasiPohon()
    .....
    .....
    simpul tambahSimpul(simpul Penunjuk, String ELEMEN)
    .....
    .....
    void preOrder(simpul Penunjuk)
    .....
    .....
    void inOrder(simpul Penunjuk)
    .....
    .....
    void postOrder(simpul Penunjuk)
    .....
    .....
}
class ProgramPohonBiner
{   public static void main(String[] args)
    {
        deklarasiPohon();
        ............
        ............
        ............
        ............
        ............
        ............

    }
}
```

Operasi-operasi pada pohon dapat dilakukan di sini.
(tambah simpul, cetak prOrder, inOrder dan postOrder)

Setelah anda tulis program di atas, tambahkanlah pada program utama perintah-perintah untuk menambah simpul berikut ini :

```
akar =tambahSimpul(akar,"M");
akar =tambahSimpul(akar,"P");
akar =tambahSimpul(akar,"D");
```

```
akar =tambahSimpul(akar,"A");
akar =tambahSimpul(akar,"S");
akar =tambahSimpul(akar,"K");
akar =tambahSimpul(akar,"N");

akar =tambahSimpul(akar,"G");
akar =tambahSimpul(akar,"O");
akar =tambahSimpul(akar,"L");
akar =tambahSimpul(akar,"W");

akar =tambahSimpul(akar,"F");
akar =tambahSimpul(akar,"J");
akar =tambahSimpul(akar,"T");

akar =tambahSimpul(akar,"H");
akar =tambahSimpul(akar,"U");
```

Di bawah perintah-perintah tersebut, tambahkan perintah untuk mencetak pohon biner secara preOrder berikut :

```
preOrder(akar);
```

Kemudian eksekusilah program di atas. Hasil apakah yang didapat? Catatlah dalam laporan anda dan jelaskan mengapa bisa demikian

2. Praktik 2

Lakukan juga eksperimen yang sama untuk cetak inOrder dan cetak postOrder berikut :

```
inOrder(akar);
```

Kemudian eksekusilah program di atas. Hasil apakah yang didapat? Catatlah dalam laporan anda dan jelaskan mengapa bisa demikian.

3. Praktik 3

Lakukan hal yang sama sebagaimana pelaksanaan praktikum 2 untuk cetak postOrder.

```
postOrder(akar);
```

Kemudian eksekusilah program di atas. Hasil apakah yang didapat? Catatlah dalam laporan anda dan jelaskan mengapa bisa demikian.

LATIHAN

1. …..
2. …..

TUGAS

1. Modifikasilah program pohon biner pada Modul 13 ini dari yang semula berelemen Karakter menjadi angka. Kemudian lakukan proses cetak PreOrder, InOrder dan PosOrder terhadapnya.
2. …..

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 117-123, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.

MODUL 14
HASHING PADA LARIK

CAPAIANPEMBELAJARAN

Mahasiswa dapat melakukan penempatan suatu data ke dalam larik menggunakan teknik hashing

KEBUTUHAN ALAT/BAHAN/SOFTWARE

1. TextPad

DASAR TEORI

Hashing

Hashing adalah teknik penempatan sebuah record pada larik dengan nomor indeks yang tidak standar menjadi standar [0], [1], [2], [3], .. dst. Hashing menempatkan sebuah record pada larik pada nomor indeks khusus yang merupakan hasil konversi dengan menggunakan **rumus khusus**.

Sebagai contoh, jika ada 8 orang mahasiswa dengan NIM masing-masing :

Tabel 14.1

NIM	NAMA	ALAMAT	UMUR	JEKEL	IPK
125410067	AgungBP	Jakarta	28	L	3.5
155410125	Rulieta	KualaKapuas	17	P	3.0
115410125	Kayra	Yogyakarta	15	P	3.4
115410066	Elnathan	Yogyakarta	13	L	2.4
165410136	Niken	Magelang	35	P	2.5
145410112	Liwin	Palangkaraya	35	P	2.7
155410133	Satrio	Semarang	26	L	3.1
155410143	Dion	Bantul	22	L	2.0

maka record-record tersebut akan ditempatkan pada larik dengan indeks sesuai dengan NIM yang dimilikinya yaitu :

Tabel 14.2

NIM	NAMA	ALAMAT	UMUR	JEKEL	IPK	ARRAY (NIM)
125410067	AgungBP	Jakarta	28	L	3.5	[125410067]
155410125	Rulieta	KualaKapuas	17	P	3.0	[155410125]
115410125	Kayra	Yogyakarta	15	P	3.4	[115410125]
115410066	Elnathan	Yogyakarta	13	L	2.4	[115410066]
165410136	Niken	Magelang	35	P	2.5	[165410136]
145410112	Liwin	Palangkaraya	35	P	2.7	[145410112]

```
155410133  Satrio     Semarang      26     L       3.1          [ 155410133 ]
155410143  Dion       Bantul        22     L       2.0          [ 155410143 ]
-------------------------------------------------------------   -------------
```

Faktanya untuk mendeklarasikan larik dengan 9 digit seperti di atas tentu memerlukan memori yang sangat besar (9 digit = 1 milyar). Itupun larik dengan kapasitas tersebut tidak akan digunakan seluruhnya karena sesungguhnya jumlah mahasiswa secara keseluruhan tidak banyak misalkan hanya berjumlah 1000 orang. Untuk itu diperlukan sebuah rumus yang mampu mengonversi NIM mahasiswa menjadi nomor larik antara 000 sampai dengan 999. Fungsi tersebut dinamakan ***fungsi hash.***

Ada beberapa metode dalam melakukan hashing yaitu

(1) **metode pembagian**,
(2) **metode midsquare**, dan
(3) **metode penjumlahan digit**.

Pada praktikum ini teknik hashing yang dibahas adalah dengan menggunakan **metode pembagian**.

Hashing dengan metode pembagian

Pada metode pembagian, nilai hash diperoleh dengan cara kunci yang akan di cari nilai hashnya (misalkan NIM) di 'modulus' dengan sebuah bilangan prima yang paling dekat dengan kapasitas array (N). Misalnya kapasitas array yang akan kita bangun sebesar 1000, maka bilangan prima terdekat 1000 yang akan digunakan sebagai pembagi NIM adalah 997. Dengan penghitungan indeks menggunakan modulus 997 maka penempatan data akan menjadi sebagai berikut :

Tabel 14.3

```
-------------------------------------------------------------   -------------
NIM        NAMA       ALAMAT        UMUR   JEKEL   IPK           ARRAY(modulus)
-------------------------------------------------------------   -------------

125410067  AgungBP    Jakarta       28     L       3.5              [ 428 ]
155410125  Rulieta    KualaKapuas   17     P       3.0              [ 756 ]
115410125  Kayra      Yogyakarta    15     P       3.4              [ 396 ]
115410066  Elnathan   Yogyakarta    13     L       2.4              [ 337 ]
165410136  Niken      Magelang      35     P       2.5              [ 857 ]
145410112  Liwin      Palangkaraya  35     P       2.7              [ 653 ]
155410133  Satrio     Semarang      26     L       3.1              [ 764 ]
155410143  Dion       Bantul        22     L       2.0              [ 774 ]
-------------------------------------------------------------   -------------
```

Program untuk hasing dengan metode pembagian dapat dilihat pada program 14.1 yaitu pada bagian fungsi **`hitungNilaiHash()`**.

Perhatikan program 14.1 berikut.

```
import java.util.Scanner;
class formatBiodata
{   //bagian deklarasi struktur record ----------------------------------
    int    nim;
    String nama;
    String alamat;
    int    umur;
```

```
    char   jekel;
    float  ipk;
}

class hashing
{   public static int N=0;

    public static int hitungNilaiHash(int nilaiAwal)
    {
        int hasil;
        hasil = nilaiAwal % 997;
        return (hasil);
    }

    public static void ngentriData(formatBiodata biodataMahasiswa[])
    {
        N = 1000;
        int NH;

        Scanner masukan = new Scanner(System.in);
        int bacaTombol=0;

        //bagian menentukan banyaknya data yang akan dientri --------------
        System.out.print("Berapa data yang akan dientri ? : ");
        int banyakEntri = masukan.nextInt();

        //bagian entri data baru ----------------------------------------------
        formatBiodata biodataMahasiswaBaru;
        for (int i=0;  i<=banyakEntri-1;  i++)
        {
            //bagian entri data baru ke penyimpan sementara --------------------
            biodataMahasiswaBaru = new formatBiodata();
            System.out.print("Silakan masukkan NIM anda : ");
            biodataMahasiswaBaru.nim = masukan.nextInt();
            System.out.print("Silakan masukkan nama anda : ");
            biodataMahasiswaBaru.nama = masukan.next();
            System.out.print("Silakan masukkan alamat anda : ");
            biodataMahasiswaBaru.alamat = masukan.next();
            System.out.print("Silakan masukkan umur anda : ");
            biodataMahasiswaBaru.umur = masukan.nextInt();
            System.out.print("Silakan masukkan Jenis Kelamin anda : ");
                try
                {       bacaTombol = System.in.read();
                }
                catch(java.io.IOException e)
                {
                }
            biodataMahasiswaBaru.jekel = (char)bacaTombol;
            System.out.print("Silakan masukkan IPK anda : ");
            biodataMahasiswaBaru.ipk = masukan.nextFloat();

            //bagian memindahkan data baru ke larik sesuai nilai Hashing -----
            NH = hitungNilaiHash(biodataMahasiswaBaru.nim);
            System.out.println  ("Biodata  "  +  biodataMahasiswaBaru.nama  +  "  akan
ditempatkan pada larik ke : " + NH);
            biodataMahasiswa[NH] = biodataMahasiswaBaru;
        }
    }



    public static void berhentiSebentar()
    {
        System.out.println ("");
        System.out.println ("Tekan tombol ENTER untuk melanjutkan...");
        Scanner masukan = new Scanner(System.in);
        int bacaTombol;
        do
        {   bacaTombol=0;
            try
            {   bacaTombol = System.in.read();
            }
            catch(java.io.IOException e)
            {
            }
```

```
        }
        while (bacaTombol != 13);  //tombol 13 adalah tombol enter
    }


    public static void tampilkanData(formatBiodata biodataMahasiswa[])
    {
        //bagian menampilkan isi struktur Larik -------------------------
        System.out.println("-------------------------------------------------------");
        System.out.println("NO NAMA           ALAMAT          UMUR    JEKEL   IPK  ");
        System.out.println("-------------------------------------------------------");
        for (int i=0;  i<=N-1;  i++)
        {   System.out.print  (i + "  ");
            System.out.print  (biodataMahasiswa[i].nama + "\t  ");
            System.out.print  (biodataMahasiswa[i].alamat + "\t  ");
            System.out.print  (biodataMahasiswa[i].umur + "\t  ");
            System.out.print  (biodataMahasiswa[i].jekel + "\t  ");
            System.out.println(biodataMahasiswa[i].ipk);
            if (i % 100 == 0)
                berhentiSebentar();
        }
        System.out.println("-------------------------------------------------------");
    }



    public static void main(String[] args)
    {
        //bagian deklarasi record berbasis LARIK -----------------------
        formatBiodata    biodataMahasiswa[] = new formatBiodata[1000];
        for (int i=0;  i<=999;  i++)
            biodataMahasiswa[i] = new formatBiodata();

        //pemanggilan fungsi-fungsi---------------------------
        ngentriData(biodataMahasiswa);
        tampilkanData(biodataMahasiswa);


  }
}
```

Program 14.1

PRAKTIK

1. Praktek 1

a. Tuliskan program 14.1 kemudian eksekusilah.
b. Saat eksekusi, entrilah kedelapan data pada tabel 14.1.
c. Lakukan pengamatan sebagai berikut: Diletakkan di manakah kedelapan data tersebut di dalam larik? *(Untuk menghemat saat mencetak, screenshootlah hanya bagian-bagian yang menampilkan kedelapan data tersebut, sementara larik-larik yang menunjukkan data kosong/null ditampilkan seperlunya saja)*.

2. Praktek 2

a. Eksekusilah kembali program 14.1.

b. Entrilah kedelapan data pada tabel 14.1 **ditambah dengan 2 buah berikut ini**.

Tabel 14.3

NIM	NAMA	ALAMAT	UMUR	JEKEL	IPK
155411140	Fifin	Purwokerto	32	P	3.1
155412137	Hermon	Banjarmasin	26	L	3.3

c. Screenshootlah bagian-bagian yang menampilkan kedelapan data tersebut, dan kedua data yang ditambahkan. Amati, di larik ke berapakah data 'Fifin' dan 'Hermon' ditempatkan? Bagaimana nasib data 'Dion'? Menurut anda mengapa hal ini terjadi?** Jelaskan dalam laporan resmi.

*(**terjadi **collision** atau **tabrakan** data)*

3. Praktek 3

a. Untuk mengatasi masalah yang terjadi pada pelaksanaan praktikum ke-2, modifikasilah fungsi `ngentriData()` dengan menambahkan instruksi (tercetak tebal) pada program sehingga menjadi seperti berikut.

```
public static void ngentriData(formatBiodata biodataMahasiswa[])
{

    N = 1000;
    int NH;

    Scanner masukan = new Scanner(System.in);
    int bacaTombol=0;

    //bagian menentukan banyaknya data yang akan dientri --------------
    System.out.print("Berapa data yang akan dientri ? : ");
    int banyakEntri = masukan.nextInt();

    //bagian membuat record sementara untuk menampung data baru--------
    formatBiodata biodataMahasiswaBaru;


    //bagian entri data baru ------------------------------------------
    for (int i=0;  i<=banyakEntri-1;  i++)
    {
        //bagian entri data baru ke penyimpan sementara ---------------
        biodataMahasiswaBaru = new formatBiodata();
        System.out.print("Silakan masukkan NIM anda : ");
        biodataMahasiswaBaru.nim = masukan.nextInt();
        System.out.print("Silakan masukkan nama anda : ");
        biodataMahasiswaBaru.nama = masukan.next();
        System.out.print("Silakan masukkan alamat anda : ");
        biodataMahasiswaBaru.alamat = masukan.next();
        System.out.print("Silakan masukkan umur anda : ");
        biodataMahasiswaBaru.umur = masukan.nextInt();
        System.out.print("Silakan masukkan Jenis Kelamin anda : ");
            try
            {       bacaTombol = System.in.read();
            }
            catch(java.io.IOException e)
            {
            }
        biodataMahasiswaBaru.jekel = (char)bacaTombol;
        System.out.print("Silakan masukkan IPK anda : ");
        biodataMahasiswaBaru.ipk = masukan.nextFloat();
```

```
            //bagian memindahkan data baru ke larik sesuai nilai Hashing---
            NH = hitungNilaiHash(biodataMahasiswaBaru.nim);

            //+++++++ MENGATASI COLLISION  ++++++++++++++++++++
            while (biodataMahasiswa[NH].nama != null)
            {   System.out.println("terjadi tabrakan pada NH=" + NH);
                NH++;
            }
            //++++++++++++++++++++++++++++++++++++++++++++++++++++

            System.out.println ("Biodata " + biodataMahasiswaBaru.nama + "
akan ditempatkan pada larik ke : " + NH);

            biodataMahasiswa[NH] = biodataMahasiswaBaru;
        }
    }
```

Program 14.2

b. Ulangi pelaksanaan praktikum 2 nomor a dan b.
c. Sekarang amati, di larik ke berapakah data Anton, Mantri dan Satrio ditempatkan? Apakah masih terjadi collision? *(jelaskan dalam laporan resmi)*

LATIHAN

1. Buatlah fungsi pencarian data di mana kunci pencarian yang dientri adalah NIM. Fungsi pencarian yang dimaksud bukanlah fungsi pencarian sekuensial yang mencari mulai dari data ke 0 hingga ke 1000, melainkan pencarian dengan ***menggunakan fungsi hash.***
2. .....

TUGAS

1. ....
2. ....

REFERENSI

Disadur dari Buku Ajar Struktur Data Menggunakan Java, Agung Budi Prasetyo, 2017, hal: 124-129, http://agungbudiprasetyo.atspace.com/buku/index.html, diakses pada 12:07 PM 8/06/2019.