BAB 10
PENGELOLAAN DATA PADA DOUBLE LINKEDLIST : PENAMBAHAN DATA DI DEPAN,TENGAH, BELAKANG, DAN PENGAPUSAN DATA

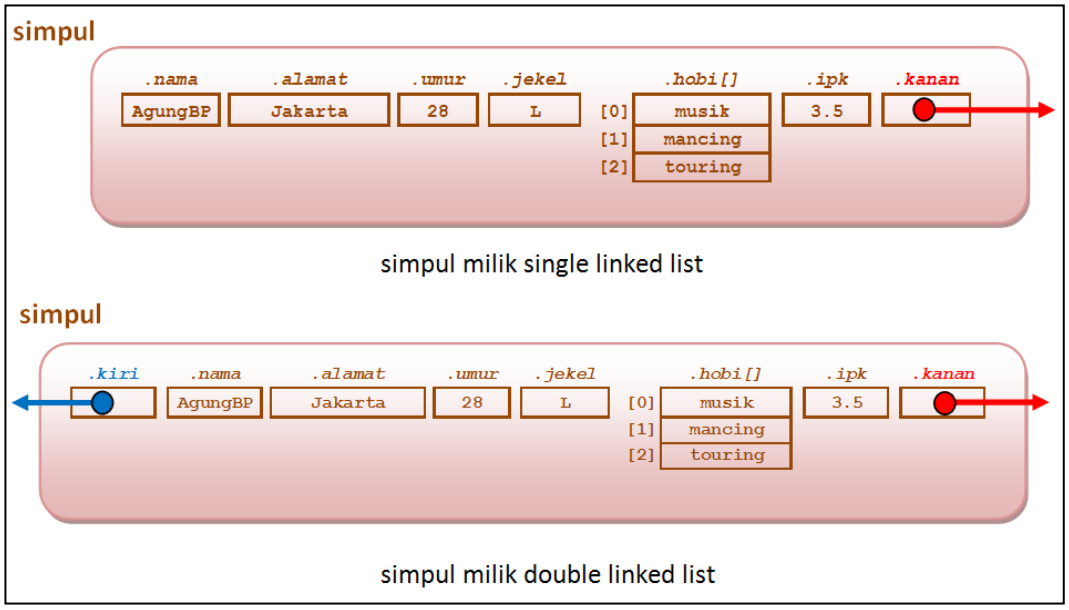
Tujuan :

Mahasiswa dapat mengimplementasikan penggunaan Double Linked List

Pada Bab sebelumnya kita telah mempelajari bagaimana pengelolaan data menggunakan media penyimpan Single Linkedlist, baik untuk menambah data di depan, tengah, belakang, menampilkan data, serta menghapus data baik di depan, tengah, dan belakang. Untuk selanjutnya pada bab 10 ini kita akan mempelajari juga mengelola data menggunakan media penyimpan Double Linkedlist, baik untuk menambah data, menampilkan data, serta menghapus data.

10.1. Double Linkedlist (Senarai Berantai Ganda)

Double Linkedlist atau disebut juga Senarai Berantai Ganda, hampir sama dengan *Single Linked List*. yaitu adalah pengalokasian memori secara dinamis untuk sebagai media untuk menyimpan dan mengelola . Bedanya adalah pada Double Linkedlist setiap simpul atau heap memiliki 2 buah penunjuk yang digunakan untuk mengkaitkan diri dengan simpul-simpul lain yaitu ***kiri*** dan ***kanan*** seperti ditunjukkan pada gambar 10.2 berikut ini.



Gambar 10.1

Perbedaan lainnya adalah pada deklarasi struktur classnya

Simpul milik Single Linkedlist :

```
class simpul
{
    //variabel-variabel penyimpan data  ----------
    String nama;
    String alamat;
    int    umur;
    char   jekel;
    String hobi[] = new String[3];
    float  ipk;

    //pointer penghubung ke heap berikutnya -------
    simpul kanan;
}
```

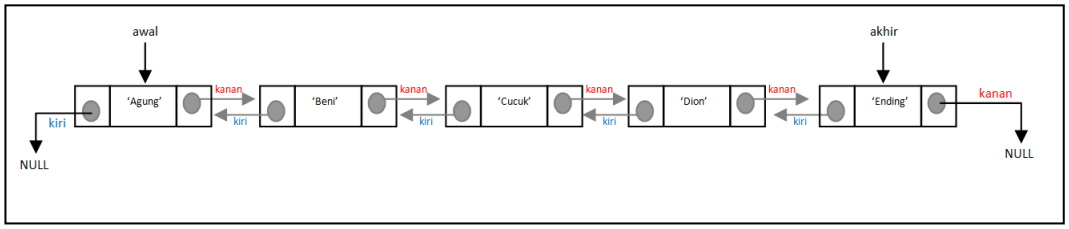
Simpul milik Double Linkedlist :

```
class simpul
{
    //variabel-variabel penyimpan data  ----------
    String nama;
    String alamat;
    int    umur;
    char   jekel;
    String hobi[] = new String[3];
    float  ipk;

    //pointer penghubung ke heap berikutnya -------
    simpul kiri;
    simpul kanan;
}
```

Oleh karena strukturnya kelasnya berbeda, maka metode dalam penanganan datanya pun juga berbeda seperti tambah depan, tengah belakang, hapus depan, tengah, belakang, dan lain-lain.

Secara umum Double Linkedlist memiliki bentuk seperti pada Gambar 10.2. Pada kedua ujung Double Linkedlist ini terdapat 2 buah pointer yang menjadi penunjuknya yaitu `Awal` yang menunjuk heap yang berada pada posisi paling depan, dan `Akhir` yang menunjuk heap yang berada pada posisi paling belakang. Sementara itu pada heap paling belakang dan paling depan masing-masing juga terkait dengan null. Adapun pointer Kanan dan Kiri yang ada pada masing-masing heap digunakan untuk saling menggandeng heap-heap yang berada di sebelahnya.

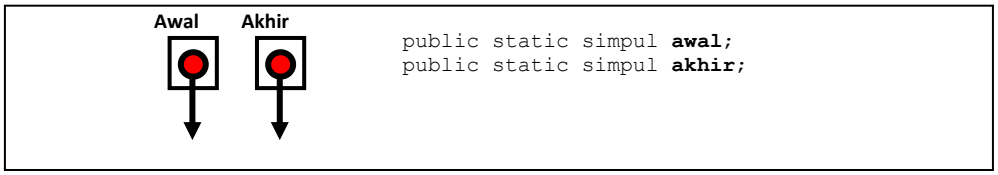


Gambar 10.2

10.2. Deklarasi Double Linkedlist

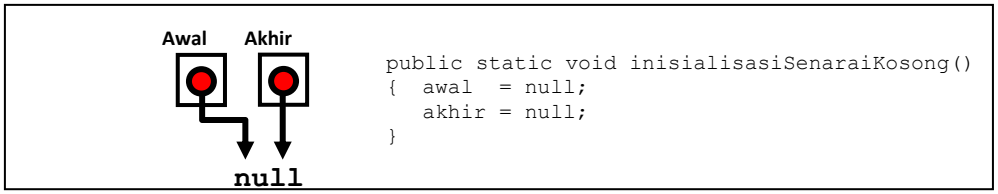
Dalam pembentukannya, Double Linkedlist di **deklarasikan** dan **diinisialisasi** dengan cara sebagai berikut :

Langkah 1: Menciptakan 2 buah pointer yaitu Awal dan Akhiir



Gambar 10.3

Langkah 2 : Mengarahkan agar pointer Awal dan Akhir menunjuk ke null. Hal ini akan menjadi penanda bahwa Double Linkedlist dalam kondisi kosong.



Gambar 10.4

Dalam script, deklarasi dan inisialisasi Double Linkedlist dituliskan sebagai berikut.

```
class senaraiGanda
{ protected
        simpul awal;
        simpul akhir;
  public
        public static void inisialisasiSenarai()
        {   awal  = null;
            akhir = null;
        }
        public static void tambahDepan()
        {   ..............          }
        public static void tambahBelakang()
        {   ..............          }
        public static int hitungJumlahSimpul()
        {   ..............          }
        public static void tambahTengah()
        {   ..............          }
        public static void cetakSenaraiMaju()
        {   ..............          }
        public static void cetakSenaraiMundur()
        {   ..............          }
        public static hapus()
        {   ..............          }
        public static void main()
        {   ..............          }
}
```

10.3. Operasi Tambah Data di Depan

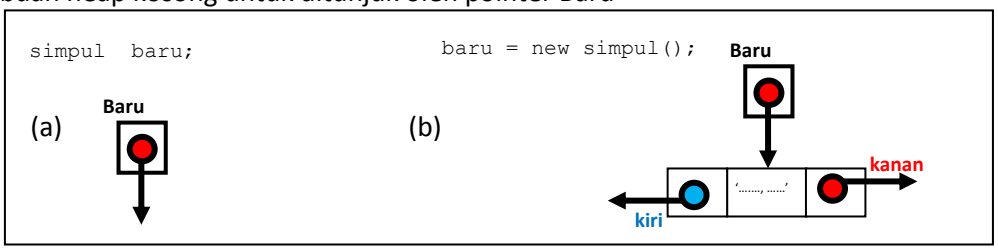
Operasi untuk menambah data baru pada Double Linkedlist di bagian depan dilakukan dengan langkah-langkah sebagai berikut.

Langkah 1 : Membuat variabel-variabel sementara dan menerima masukan data dari keyboard.

```
//------------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("masukkan nama: ");     NAMA = masukan.nextLine();
System.out.print("masukkan alamat: ");   ALAMAT = masukan.nextLine();
System.out.print("masukkan umur: ");     UMUR = masukan.nextInt();
System.out.print("masukkan Jenis Kelamin: ");
   try   {bacaTombol = System.in.read();}catch(java.io.IOException e){}
   JEKEL = (char)bacaTombol;
System.out.println("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("masukkan IPK: ");      IPK = masukan.nextFloat();
......
```

Langkah 2 : Membuat sebuah pointer bernama **Baru**, dilanjutkan dengan membuat sebuah heap kosong untuk ditunjuk oleh pointer Baru

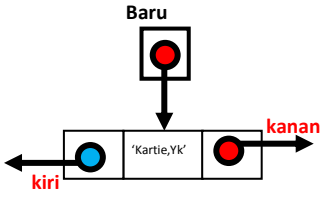


Gambar 10.5

Langkah 3 : Memindahkan isi variabel-variabel sementara ke dalam heap baru

```
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;
```

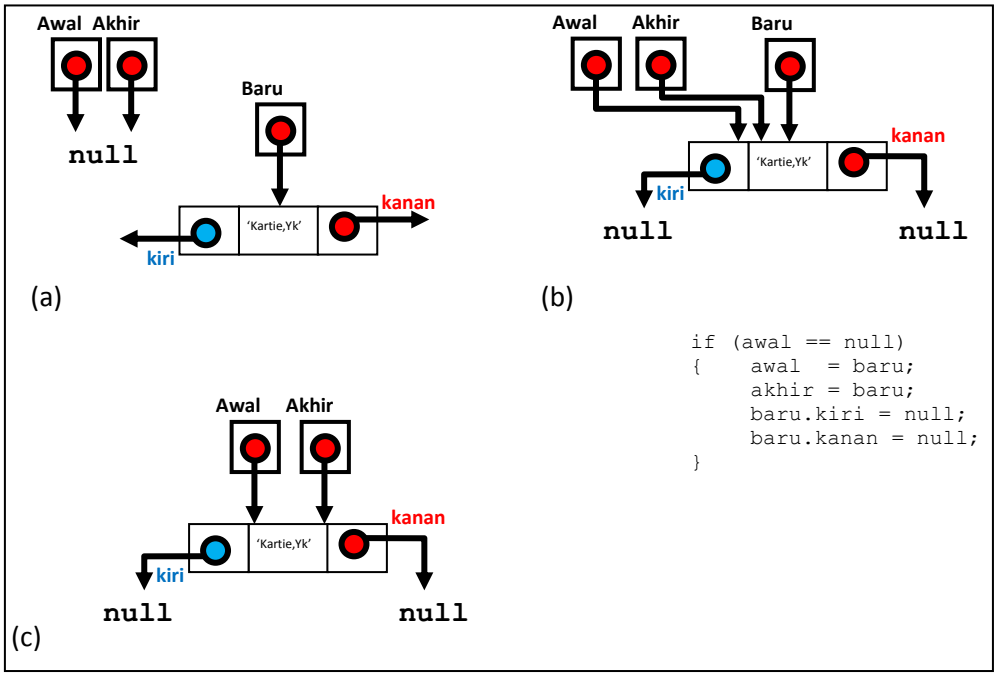
(pada gambar, tidak semua nilai diilustrasikan)



Gambar 10.6

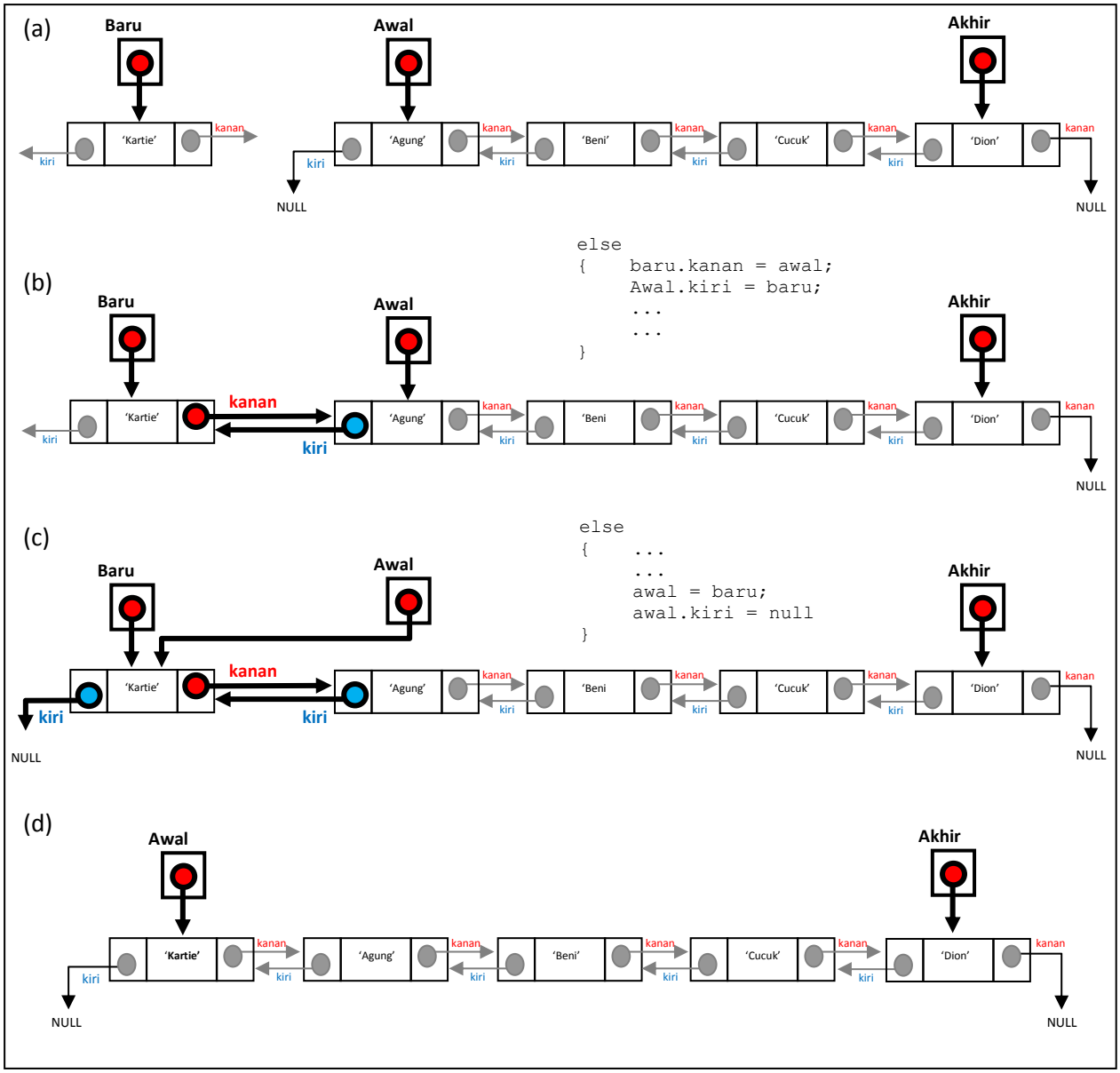
Langkah 4 :

Pada kondisi : Jika senarai masih kosong (ditandai dengan Awal atau Akhir menunjuk ke null), maka heap baru akan menjadi satu-satunya penyimpan data di dalam Double Linkedlist. Karena baru ada satu heap penyimpan data maka pointer Awal dan Akhir harus diarahkan menunjuk ke heap tersebut.



Gambar 10.7

Pada kondisi : Jika senarai tidak kosong (ditandai dengan Awal atau Akhir tidak menunjuk ke null) maka tempatkanlah heap baru sebagai awal dari Double Linkedlist dengan urutan cara sebagai berikut.



Gambar 10.8

10.4. Operasi Tambah Data di Belakang

Operasi untuk menambah data baru pada Double Linkedlist bagian belakang dilakukan dengan langkah-langkah sebagai berikut.

Langkah 1 : Membuat variabel-variabel sementara dan menerima masukan data dari keyboard.

```
//------------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("masukkan nama: ");     NAMA = masukan.nextLine();
System.out.print("masukkan alamat: ");   ALAMAT = masukan.nextLine();
System.out.print("masukkan umur: ");     UMUR = masukan.nextInt();
System.out.print("masukkan Jenis Kelamin: ");
   try   {bacaTombol = System.in.read();}catch(java.io.IOException e){}
   JEKEL = (char)bacaTombol;
System.out.println("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("masukkan IPK: ");      IPK = masukan.nextFloat();
......
```

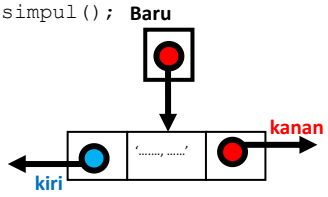
Langkah 2 : Membuat sebuah pointer bernama **Baru**, dilanjutkan dengan membuat sebuah heap kosong untuk ditunjuk oleh pointer Baru

```
simpul  baru;
```

(a) Baru

```
baru = new simpul();
```

(b) Baru

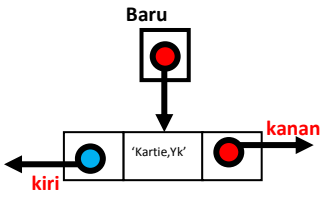


Gambar 10.9

Langkah 3 : Memindahkan isi variabel-variabel sementara ke dalam heap baru

```
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;
```

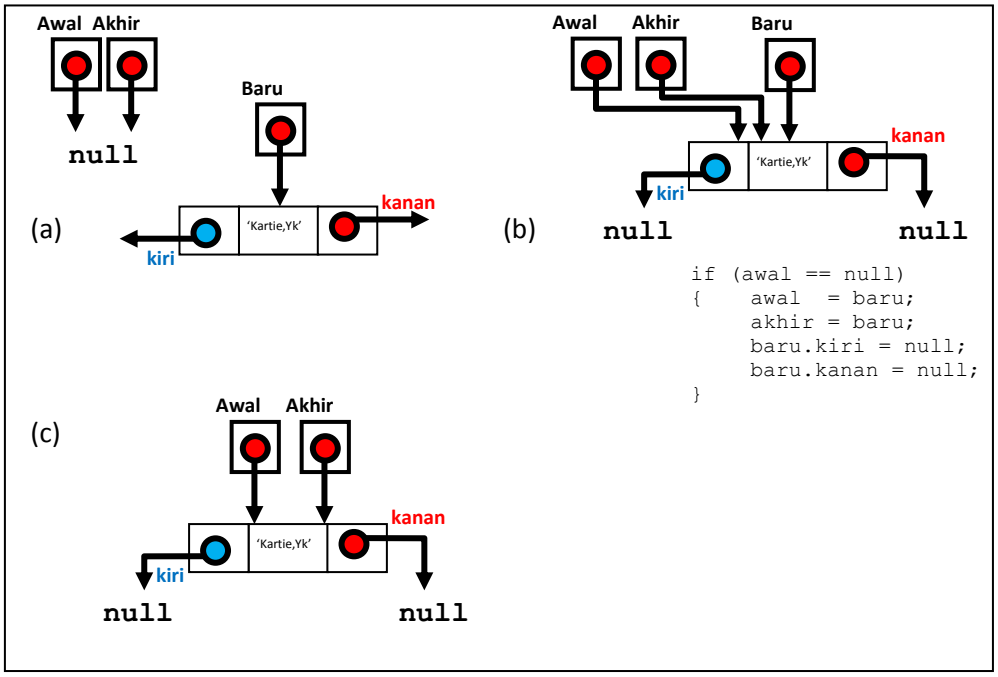
(pada gambar, tidak semua nilai diilustrasikan)



Gambar 10.10

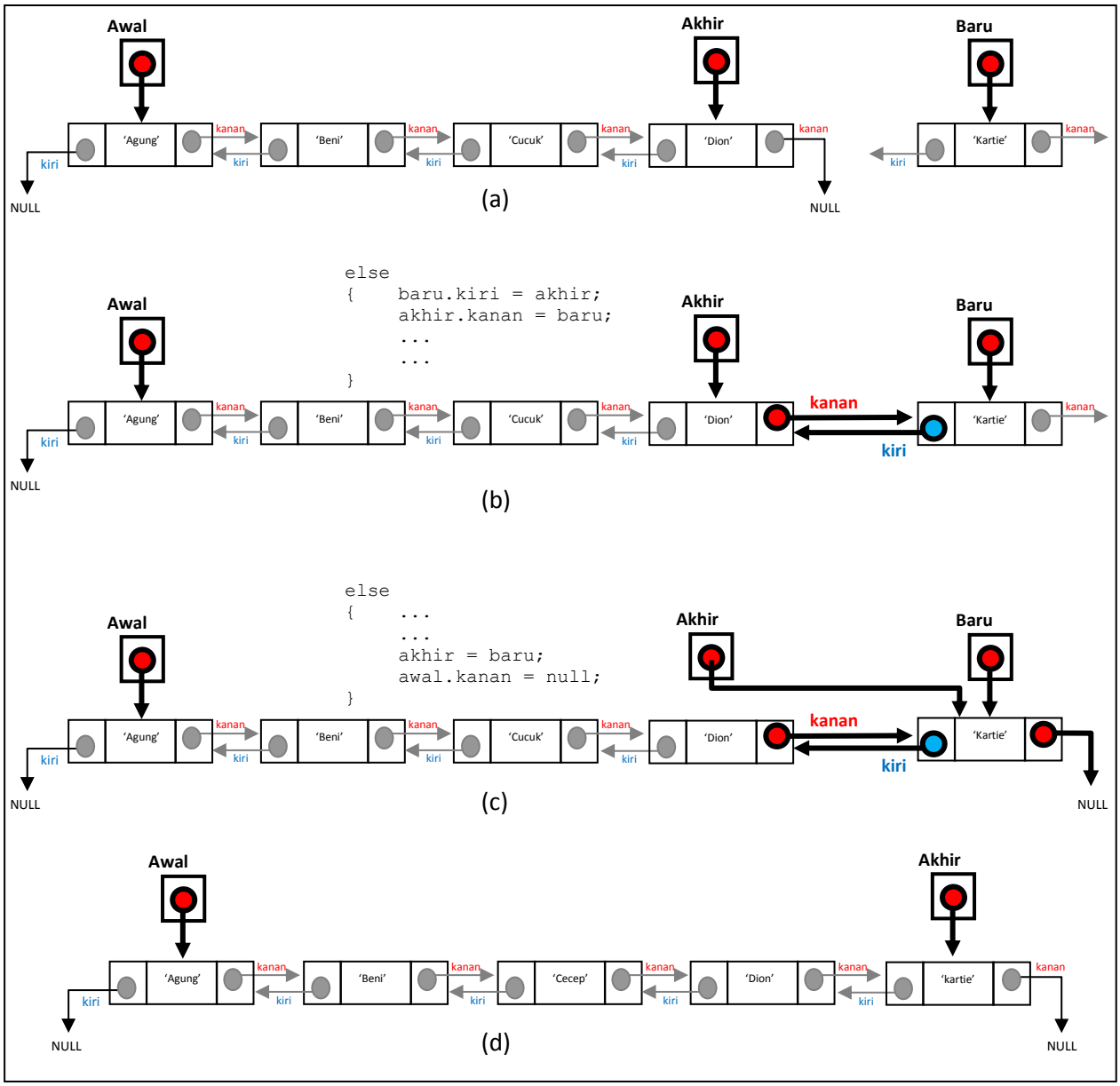
Langkah 4 :

Pada kondisi : Jika senarai masih kosong (ditandai dengan Awal atau Akhir menunjuk ke null), maka heap baru akan menjadi satu-satunya penyimpan data di dalam Double Linkedlist. Karena baru ada satu heap penyimpan data maka pointer Awal dan Akhir harus diarahkan menunjuk ke heap tersebut.



Gambar 10.11

Pada kondisi : Jika senarai tidak kosong (ditandai dengan Awal atau Akhir tidak menunjuk ke null) maka tempatkanlah heap baru sebagai awal dari Double Linkedlist dengan urutan cara sebagai berikut.


Awal
Akhir
Baru
'Agung'
'Beni'
'Cucuk'
'Dion'
'Kartie'
kanan
kiri
NULL
(a)
else
{ baru.kiri = akhir;
akhir.kanan = baru;
...
...
}
(b)
else
{ ...
...
akhir = baru;
awal.kanan = null;
}
(c)
'Cecep'
'kartie'
(d)

Gambar 10.12

10.5. Operasi Tambah Data di Tengah

Operasi untuk menambah data baru pada Double Linkedlist di bagian tengah dilakukan dengan langkah-langkah seperti di bawah ini.

Langkah 1 : Tentukan lokasi penambahan data baru. Lokasi penambahan data baru dientri melalui keyboard dan ditampung dalam sebuah variabel bernama Lokasi. Variabel Lokasi ini nanti akan membantu dalam memberi tanda di manakah data baru harus disisipkan.

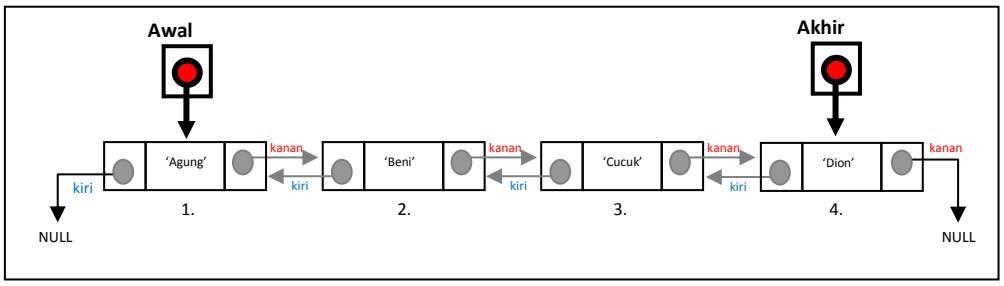
```
Scanner masukan = new Scanner(System.in);
System.out.println("Tentukan Lokasi Penambahan Data");
int LOKASI = masukan.nextInt();

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

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

Sebagai langkah tambahan, nilai Lokasi yang dientri melalui keyboard juga perlu diuji berapa nilainya. Apabila nilai tersebut sama dengan 1 itu artinya sama saja dengan melakukan penambahan data di depan. Apabila nilai Lokasi > Banyaknya heap dalam Double Linkedlist maka itu artinya sama dengan melakukan penambahan data di belakang. Namun apabila nilai Lokasi > 1 dan < banyaknya heap dalam Double Linkedlist maka itu bisa dikategorikan dengan penambahan data di tengah, sehingga proses dapat dilanjutkan ke langkah 2.

Untuk memudahkan pemahaman kita, kita akan menggunakan contoh kasus untuk menambahkan data baru ”Kartie” pada posisi ke 3 dari Double Linkedlist ada. Kondisi awal Double Linkedlist dapat dilihat pada gambar 10.13. Karena Lokasi yang dipilih adalah 3 maka data ”Karti” nantinya akan dimasukkan (baca: disisipkan) di antara ”Beni” dan ”Cucuk”.



Gambar 10.13

Langkah 2 : Membuat variabel-variabel sementara dan menerima masukan data baru dari keyboard. Di sini kita akan mengentri data baru tentang ”Kartie”.

```
//------------bagian entri data dari keyboard--------------
String NAMA;
String ALAMAT;
```

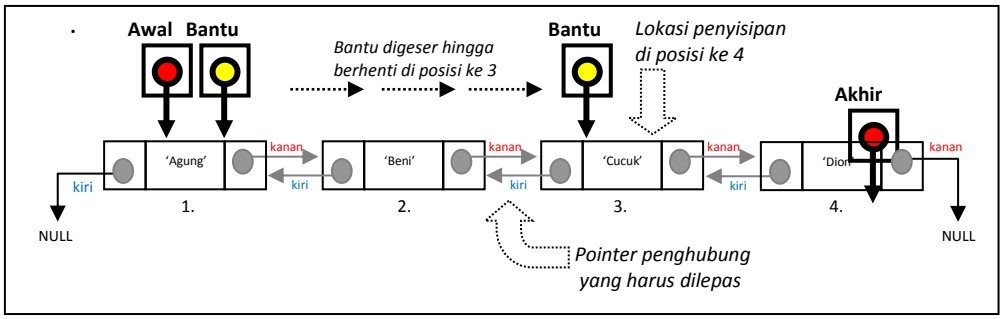
```
int    UMUR;
char   JEKEL;
String HOBI[] = new String[3];
float  IPK;
Scanner masukan = new Scanner(System.in);
int bacaTombol=0;

System.out.println("TAMBAH DEPAN : ");
System.out.print("masukkan nama: ");     NAMA = masukan.nextLine();
System.out.print("masukkan alamat: ");   ALAMAT = masukan.nextLine();
System.out.print("masukkan umur: ");     UMUR = masukan.nextInt();
System.out.print("masukkan Jenis Kelamin: ");
   try   {bacaTombol = System.in.read();}catch(java.io.IOException e){}
   JEKEL = (char)bacaTombol;
System.out.println("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("masukkan IPK: ");      IPK = masukan.nextFloat();
......
......
```

Langkah 3 : Menemukan posisi yang dikehendaki pada Double Linkedlist. Untuk dapat menemukan posisi yang dikehendaki maka diperlukan bantuan sebuah pointer bernama **Bantu**. Pointer Bantu ini nanti akan bertugas untuk ”memegang” (baca:menunjuk) heap yang letaknya tepat pada lokasi penyisipan yang dikehendaki.

Dalam contoh di atas, karena lokasi penyisipan adalah 3 maka **Bantu** harus menunjuk di heap ke-3 (”Cucuk”). Hal ini sedikit berbeda dari cara menyisipkan pada Single Linkedlist di mana harus menempatkan Bantu di samping kiri dari lokasi penyisipan.

Agar pointer **Bantu** dapat berada di lokasi ke 3, penelusuran harus dilakukan mulai dari heap yang paling depan (Bantu = Awal). Selanjutnya pointer Bantu akan digeser selangkah demi selangkah hingga mencapai heap ke-3 (”Cucuk”).



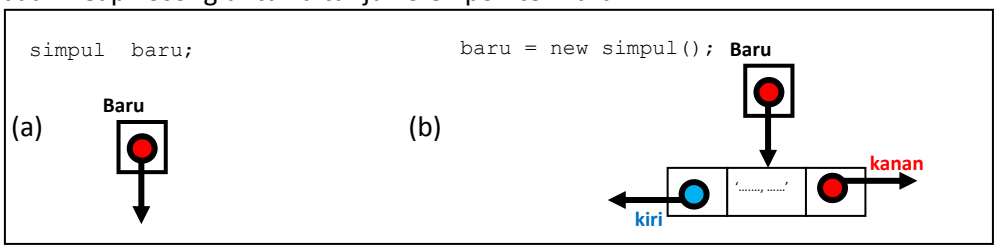
Gambar 10.14

Script program yang digunakan untuk dapat menempatkan Bantu pada posisinya adalah sebagai berikut.

```
//------------bagian menemukan posisi yang dikehendaki--------------
simpul bantu;
bantu = awal;
int i = 1;
while ((i<LOKASI) && (bantu!=akhir))
```

```
{  bantu = bantu.kanan;
   i++;
}
```

Langkah 4 : Membuat sebuah pointer bernama **Baru**, dilanjutkan dengan membuat sebuah heap kosong untuk ditunjuk oleh pointer **Baru**

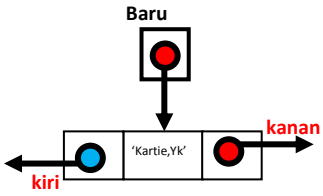


Gambar 10.15

Langkah 5 : Memindahkan isi variabel-variabel sementara (”Kartie”) ke dalam heap baru

```
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;
```

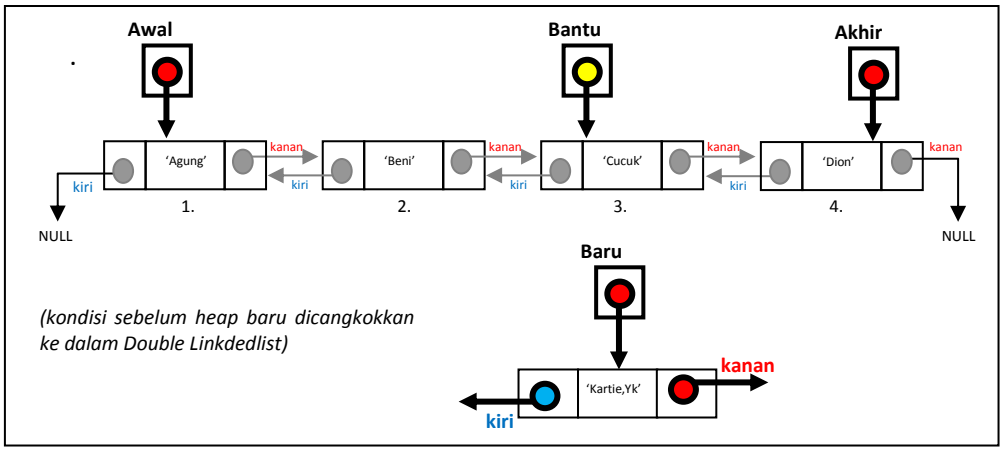
(pada gambar, tidak semua nilai diilustrasikan)



Gambar 10.16

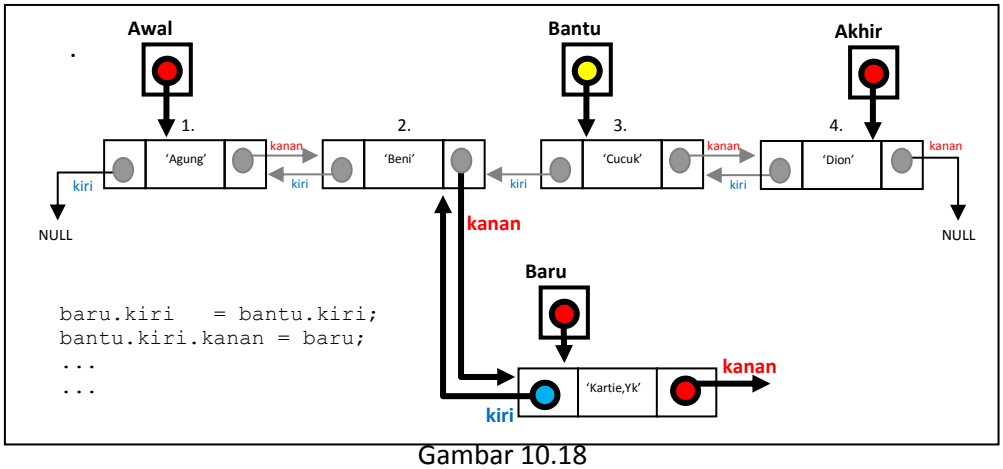
Langkah 6 :

Mencangkokkan heap baru pada Double Linkedlist dengan urutan langkah sebagai berikut



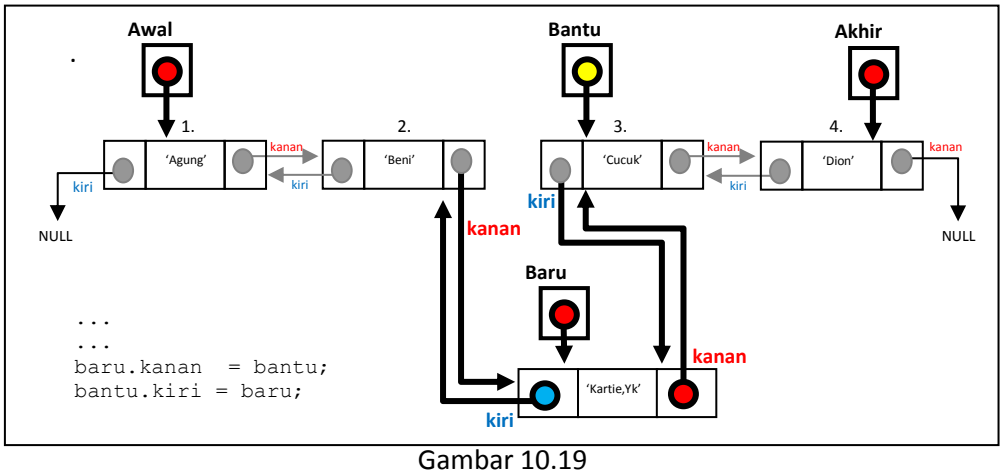
Gambar 10.17

(a) Arahkan pointer **Baru.kiri** menunjuk ke heap yang ditunjuk oleh **Bantu.kiri** dan arahkan pointer **Bantu.kiri.kanan** menunjuk ke heap yang ditunjuk oleh **Baru**.



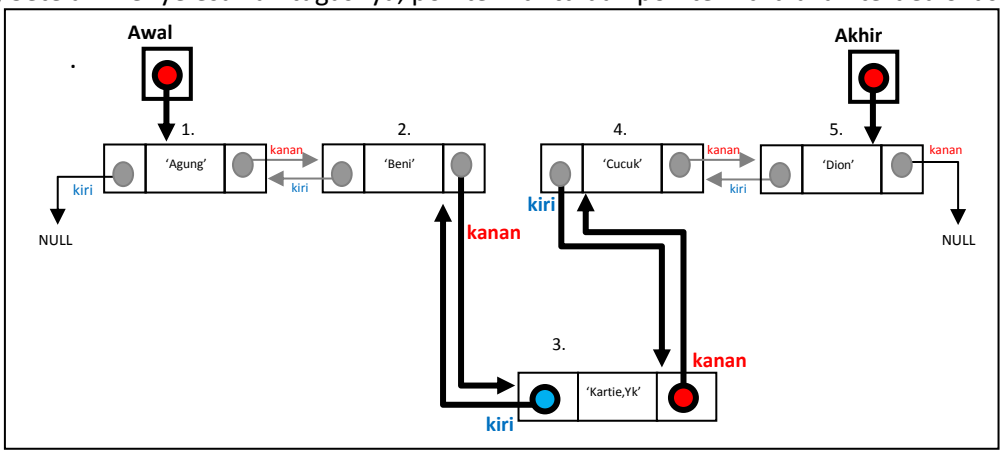
Gambar 10.18

(b) Arahkan pointer **Baru.kanan** menunjuk ke heap yang ditunjuk oleh **Bantu** dan arahkan pointer **Bantu.kiri** menunjuk ke heap yang ditunjuk oleh **Baru**.



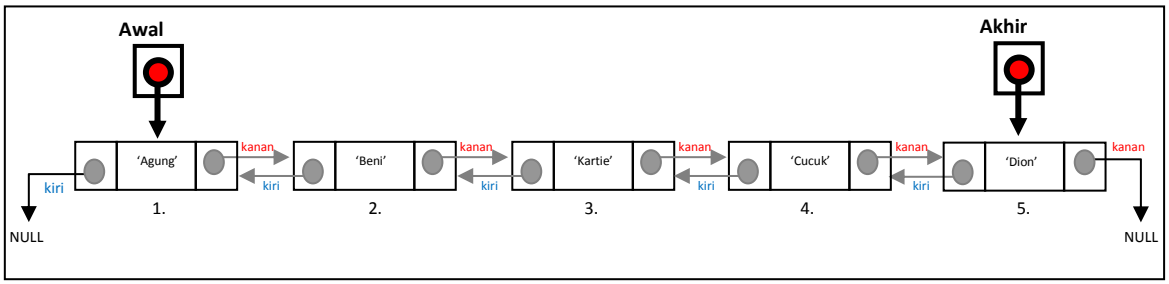
Gambar 10.19

(c) Setelah menyelesaikan tugasnya, pointer **Bantu** dan pointer **Baru** akan terdealokasi.



Gambar 10.20

Hasil akhir yang diperoleh setelah penambahan data baru di tengah Double Linkedlist dapat dilihat Gambar 10.21. Tampak sekarang ”Kartie” telah berada di antara ”Beni” dan ”Cucuk”.



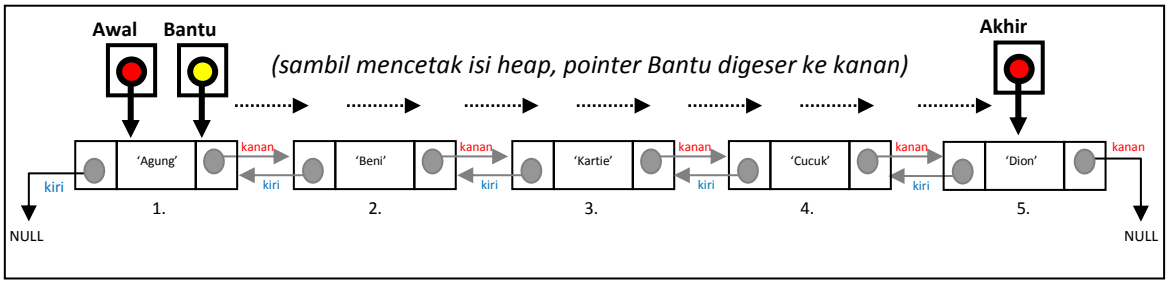
Gambar 10.21

10.6. Operasi Menampilkan Semua Data pada Double Linkedlist

Dalam menampilkan (mencetak) data-data yang ada dalam Double Linkedlist, ada 2 cara yaitu cetak maju dan cetak mundur. Cetak maju adalah mencetak data-data dimulai dari data paling depan hingga belakang, sementara cetak mundur adalah mencetak data-data mulai dari data paling belakang hingga depan.

10.7. Operasi Operasi Cetak Maju

Untuk dapat mencetak maju diperlukan sebuah pointer bernama Bantu yang akan bertugas untuk menunjuk satu persatu heap demi heap dimulai dari heap paling depan hingga paling belakang. Sambil menampilkan data dari heap yang sedang ditunjuk pointer Bantu ini akan digeser terus menerus ke kanan, selangkah demi selangkah, sampai pointer ini menyentuh null yang berada di akhir Double Linkedlist.



Gambar 10.22

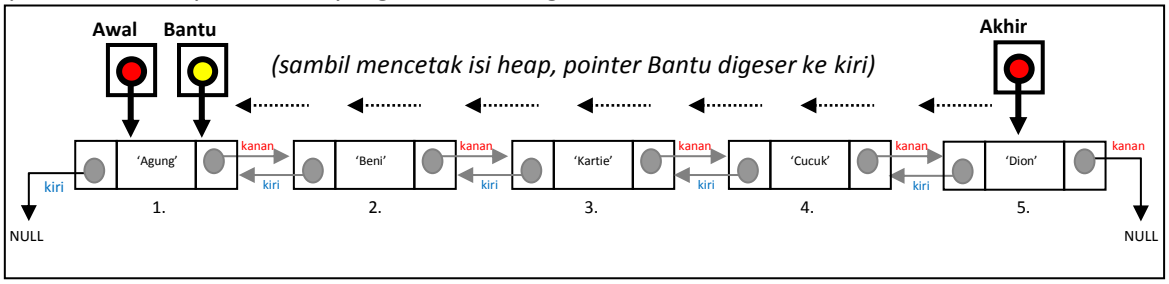
Adapun proses menampilkan isi Double Linkedlist dapat dilihat pada program di bawah ini.

```
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("-------------------------------------------------------");
     }
  }
```

10.8. Operasi Operasi Cetak Mundur

Untuk dapat mencetak mundur diperlukan sebuah pointer bernama Bantu yang akan bertugas untuk menunjuk satu persatu heap demi heap dimulai dari heap paling belakang hingga paling depan. Sambil menampilkan data dari heap yang sedang ditunjuk pointer Bantu ini akan digeser terus menerus ke kiri, selangkah demi selangkah, sampai pointer ini menyentuh null yang berada di bagian awal dari Double Linkedlist.



Gambar 10.23

Adapun proses menampilkan isi Double Linkedlist dapat dilihat pada program di bawah ini.

```
  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("-------------------------------------------------");
    }
}
```

10.9. Operasi Menghapus Data di Depan, Tengah, dan Belakang

Operasi menghapus data pada Double Linkedlist dapat terjadi pada bagian depan, tengah maupun belakang. Namun tidak seperti pada operasi menambah data, dalam menghapus kita tidak dapat menentukan di manakah letak data yang akan dihapus. Hal ini sepenuhnya tergantung pada dimanakah data yang akan dihapus tersebut ditemukan dan hal ini dapat terjadi di mana saja tanpa dapat kita prediksi.

Pada pembahasan ini kita akan kembali menggunakan contoh kasus ”Kartie”, namun contoh tersebut kita sesuaikan dengan kondisi dimana data bisa berada di depan, tengah, maupun belakang dari Double Linkedlist atau bahkan dalam kondisi dimana data tidak ditemukan sama sekali dalam Double Linkedlist.

Berikut ini langkah-langkah menghapus data pada Double Linkedlist.

Langkah 1 : Ujilah, apakah Double Linkedlist kosong. Apabila kosong maka penghapusan data tidak dapat dilakukan. Salah satu kondisi jika Double Linkedlist kosong adalah Awal ataupun Akhir menunjuk ke null. Namun apabila Double Linkedlist tidak kosong proses dapat dilanjutkan ke langkah 2.

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

Langkah 2 : Masukkan data yang akan dihapus sebagai kunci pencarian.

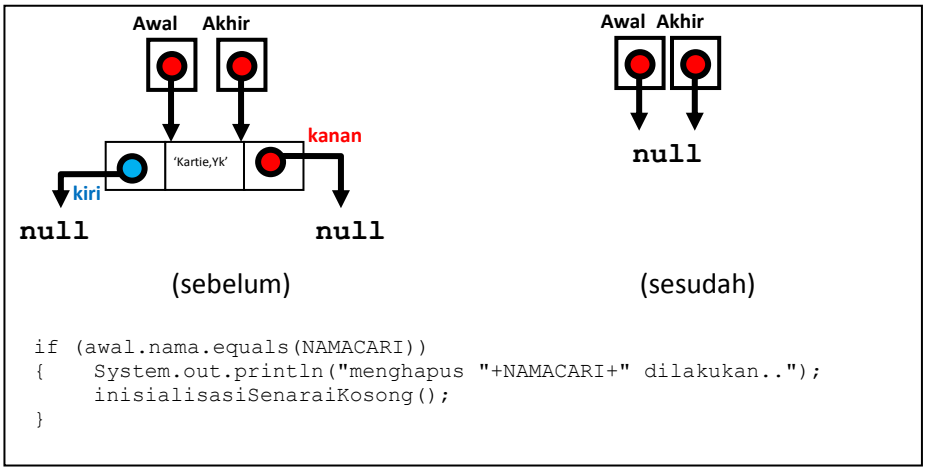
```
        Scanner masukan = new Scanner(System.in);
        System.out.print("Silakan masukkan nama yang ingin dihapus : ");
        String NAMACARI = masukan.nextLine();
```

Langkah 3 : Pada langkah ini terdapat 3 (tiga) situasi yang mungkin terjadi.

Situasi pertama, jika Double Linkedlist hanya berisi 1 (satu) heap (ditandai dengan kondisi dimana Awal=Akhir), maka akan ada 2 kemungkinan yang bisa terjadi yaitu:

Kemungkinan pertama, jika data pada heap tersebut identik dengan kunci pencarian, maka heap dapat dihapus dengan cara Awal dan Akhir kembali dibuat menunjuk null (proses ini sama seperti proses inisialisasi Double Linkedlist yang pernah dibahas pada sub bab 8.4) .

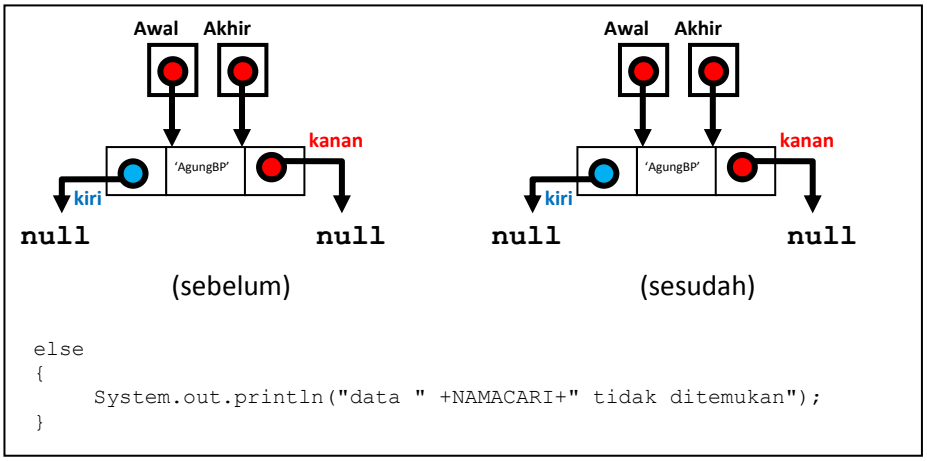
Gambar 10.24 berikut ini adalah contoh kasus di mana data yang dicari ”Kartie” dan data paling depan = ”Kartie”.




```
if (awal.nama.equals(NAMACARI))
{    System.out.println("menghapus "+NAMACARI+" dilakukan..");
     inisialisasiSenaraiKosong();
}
```

Gambar 10.24

Kemungkinan kedua, jika data pada heap tidak sama dengan kata kunci pencarian, maka dapat disimpulkan bahwa data yang dicari tidak ditemukan pada Double Linkedlist. Gambar 10.25 berikut ini adalah contoh kasus di mana Data yang dicari ”Kartie” dan data paling depan = ”AgungBP”.



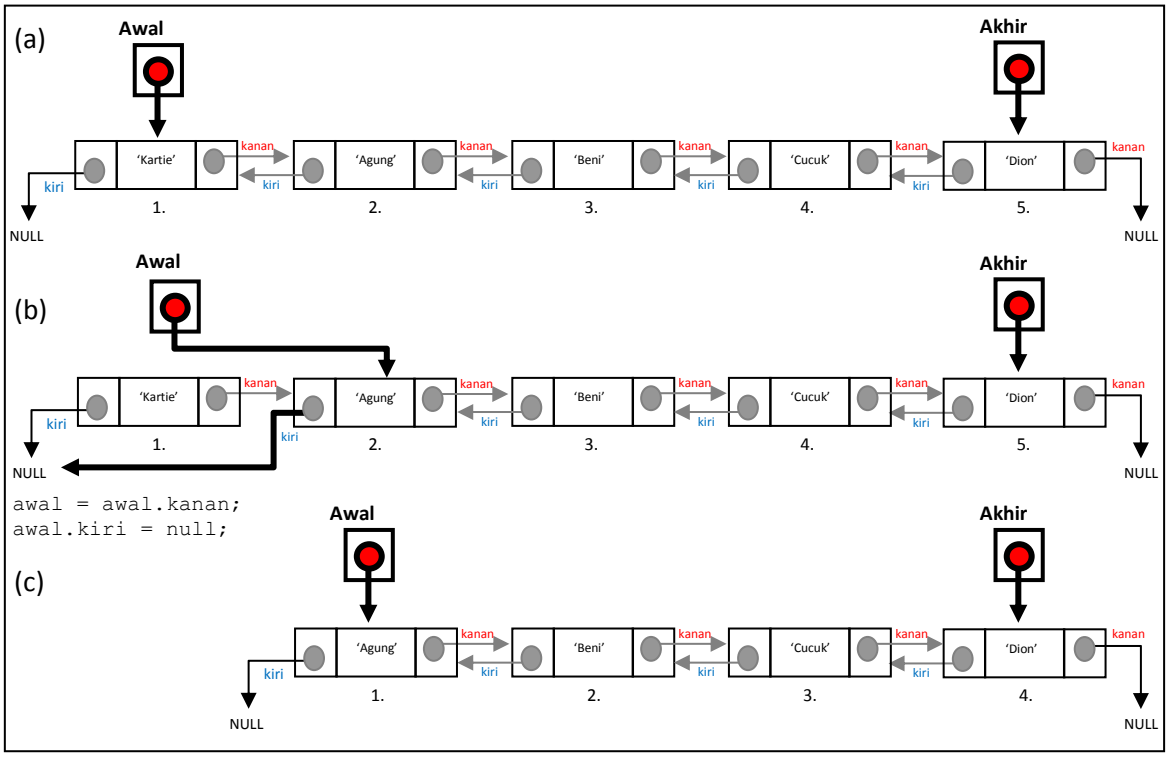

```
else
{
     System.out.println("data " +NAMACARI+" tidak ditemukan");
}
```

Gambar 10.25

Penanganan dari kedua kemungkinan di atas dapat dilihat pada program berikut ini.

```
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");
}
```

Situasi kedua, jika Double Linkedlist berisi lebih dari 1 heap serta data yang dicari **ditemukan pada heap paling depan**, maka proses penghapusan dapat dilakukan dengan cara menggeser pointer Awal dari yang semula menunjuk pada heap terdepan dibuat menunjuk pada heap kedua. Gambar 10.26 berikut ini adalah contoh kasus di mana Data yang dicari ”Kartie” dan data paling depan = ”Kartie”.



Gambar 10.26

Penanganan dari situasi di atas dapat dilihat pada program berikut ini.

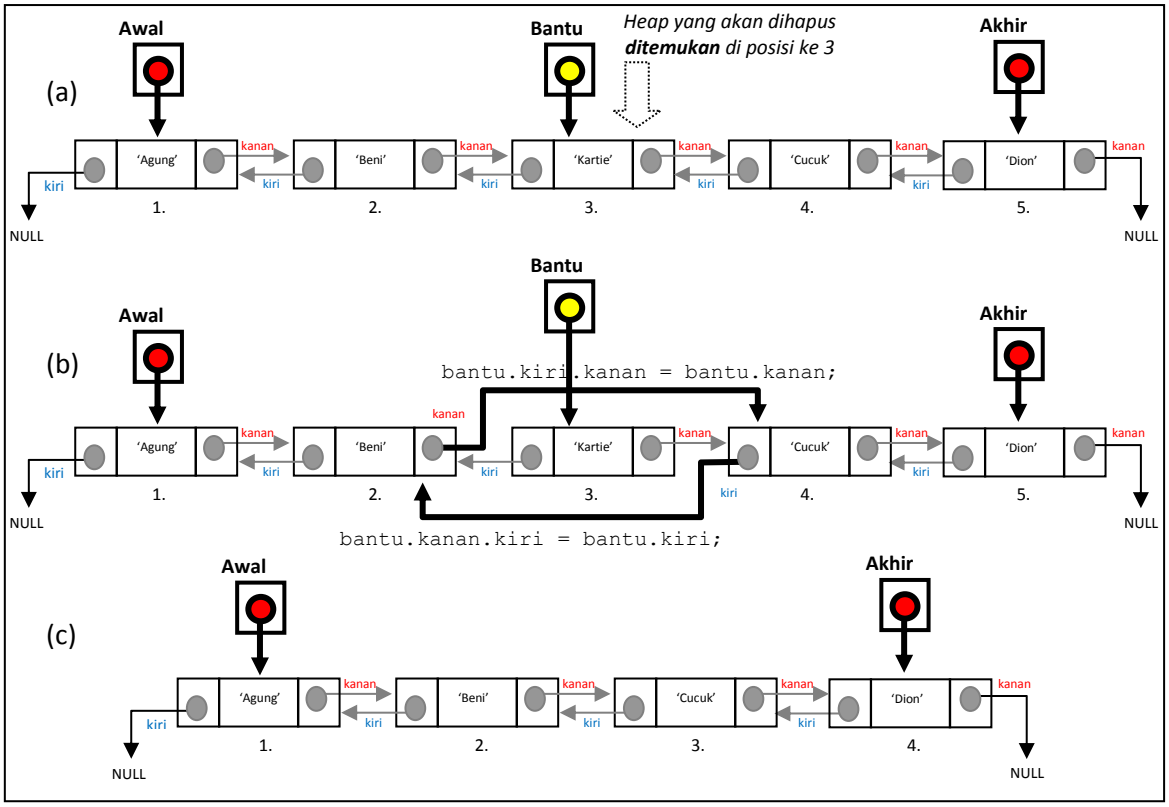
```
else if (awal.nama.equals(NAMACARI))//jika nama ditemukan di awal
{  System.out.println("menghapus "+NAMACARI+" dilakukan..");
   awal = awal.kanan;
   awal.kiri = null;
}
```

Situasi ketiga, jika Double Linkedlist berisi lebih dari 1 heap tetapi **data yang dicari bukan berada pada heap paling depan, bukan berada pada heap paling belakang,** maka perlu dilacak dimanakah data tersebut berada. Pelacakan dilakukan dengan cara menempatkan pointer Bantu mulai dari posisi nomor 2 dari depan untuk menyusuri heap demi heap hingga heap paling belakang. Selama Bantu belum menemukan data yang dimaksud maka Bantu akan bergeser ke heap berikutnya untuk melacak data lagi, demikian seterusnya hingga ditemukan data yang dimaksud ataupun hingga Bantu = Akhir.

```
simpul bantu;
bantu = awal.kanan;
while (bantu.nama.equals(NAMACARI)==false)
{  bantu = bantu.kanan;
   if (bantu.kanan ==  null) break;
}
```

Seiring dengan perjalanan pointer Bantu menyusuri heap-heap di atas, maka akan ditemukan beberapa kondisi sebagai berikut :

Pertama : Jika data yang dicari **ditemukan pada heap kedua, atau ketiga, atau keempat, dan seterusnya hingga heap kedua dari belakang** , maka proses penghapusan dapat dilakukan karena pointer Bantu telah memegang heap yang akan dihapus sebagaimana Gambar 10.27.

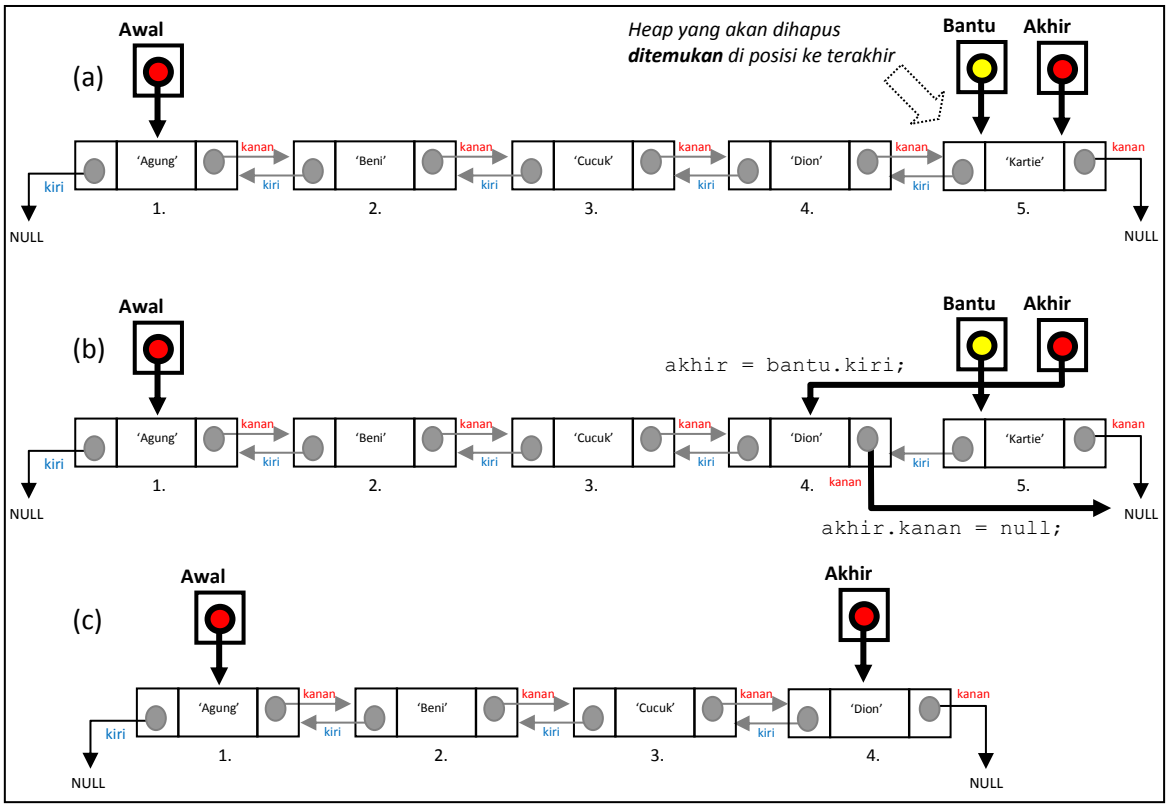


Gambar 10.27

Penanganan dari situasi di atas dapat dilihat pada program berikut ini.

```
            else
            {  System.out.println("menghapus "+NAMACARI+" dilakukan..");
               bantu.kanan = bantu.kanan.kanan;
```

Kedua: jika data **ditemukan pada heap terakhir**, maka dilakukan langkah sebagaimana Gambar 10.28 berikut ini.



Gambar 10.28

Penanganan dari situasi di atas dapat dilihat pada program berikut ini.

```
else if (akhir.nama.equals(NAMACARI))//jika nama ditemukan di akhir
{  akhir = bantu.kiri;
   akhir.kanan = null;
}
```

Ketiga : Jika pointer Bantu telah sampai ke heap paling belakang (ditandai dengan Bantu==Akhir) **dan data yang dicari tidak juga ditemukan** maka dapat

disimpulkan bahwa data yang menjadi kunci pencarian memang tidak ada pada seluruh ini Double Linkedlist.

```
if ((bantu== akhir) && (akhir.nama.equals(NAMACARI)==false))
{  System.out.println("data " +NAMACARI+" tidak ditemukan");
}
```

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;
        bantu.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;
```

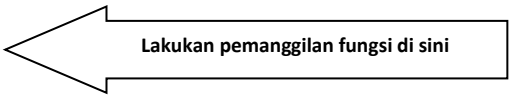
```
            if (bantu.kanan ==  null) break;
         }
         if ((bantu == akhir) && (akhir.nama.equals(NAMACARI)==false))
         {  System.out.println("data " +NAMACARI+" tidak ditemukan");
         }
         else if (akhir.nama.equals(NAMACARI))//jika nama ditemukan di akhir
         {
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



Program 10.1

10.10. Latihan

Latihan 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.

Latihan 2 :

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

Tugas

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).