

BAB 9

PENGELOLAAN DATA PADA SINGLE LINKEDLIST : PENAMBAHAN DATA DI TENGAH & PENGAPUSAN DATA

Tujuan :

Mahasiswa dapat mengimplementasikan penggunaan Single Linked List

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

9.1. Operasi Tambah Data di Tengah

Operasi untuk menambah data baru pada Single 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.

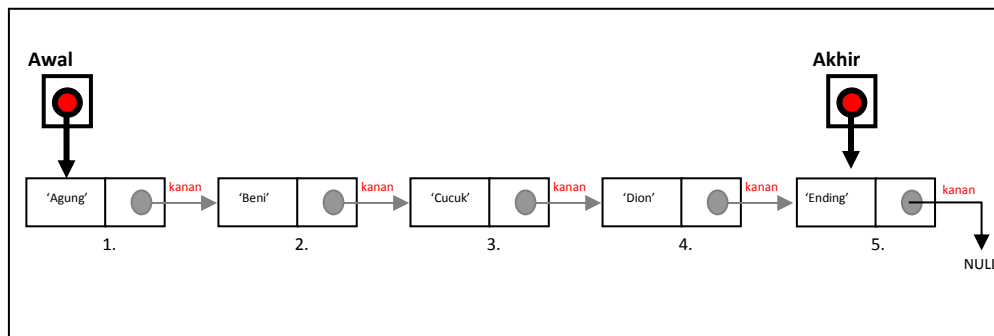
```
Scanner masukan = new Scanner(System.in);
System.out.println("Tentukan Lokasi Penambahan Data");
int LOKASI = masukan.nextInt();

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 Single Linkedlist maka itu artinya sama dengan melakukan penambahan data di belakang. Namun apabila nilai Lokasi > 1 dan < banyaknya heap dalam Single 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 4 dari Single Linkedlist ada. Kondisi data pada Single Linkedlist dapat dilihat pada gambar 9.1. Karena Lokasi yang dipilih adalah 4 maka data "Karti" nantinya akan dimasukkan (baca: disisipkan) di antara "Cucuk" dan "Dion".



Gambar 9.1

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;

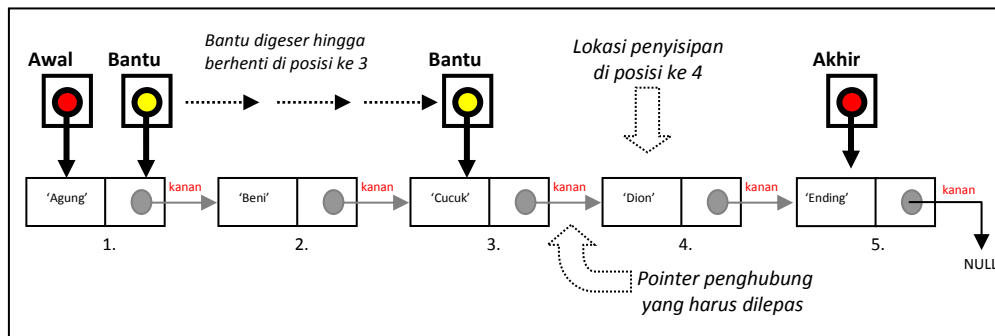
System.out.println("TAMBAH DEPAN : ");
System.out.print("masukkan nama: "); NAMA = masukan.nextLine();
System.out.print("masukkan alamat: "); ALAMAT = masukan.nextLine();
System.out.print("masukkan umur: "); UMUR = masukan.nextInt();
System.out.print("masukkan Jenis Kelamin: ");
try {bacaTombol = System.in.read();}catch(java.io.IOException 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 Single 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 di samping kiri dari Lokasi penyisipan yang dikehendaki.

Dalam contoh di atas, karena lokasi penyisipan adalah 4 maka Bantu harus menunjuk di heap ke-3 ("Cucuk"). Mengapa demikian? Karena nantinya pointer

penghubung antara heap ke-3 ("Cucuk") dan ke-4 ("Dion") harus dilepaskan untuk disisipi dengan heap baru ("Kartie").

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").

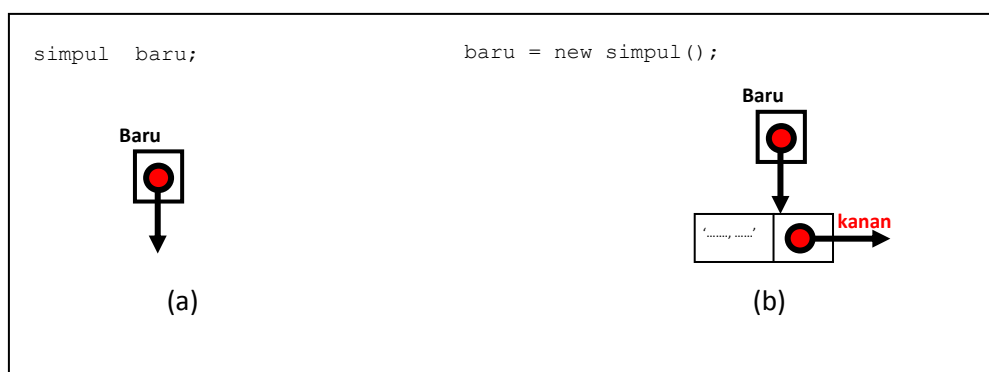


Gambar 9.2

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

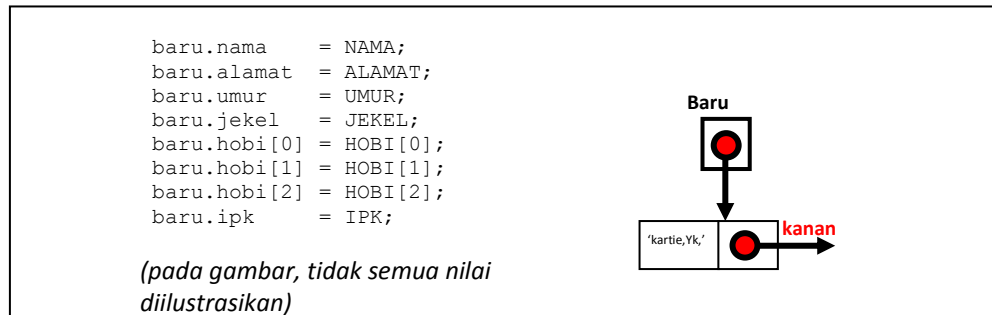
```
//-----bagian menemukan posisi yang dikehendaki-----
simpul bantu;
bantu = awal;
int N = 1;
while ((N<LOKASI-1) && (bantu!=akhir))
{ bantu = bantu.kanan;
  N++;
}
```

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



Gambar 9.3

Langkah 5 : Memindahkan isi variabel sementara ("Kartie") ke dalam heap baru.

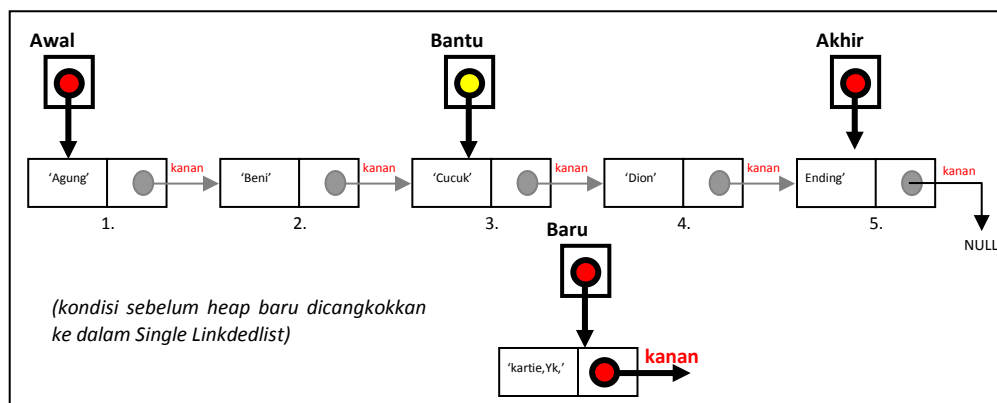


Gambar 9.4

Langkah 6 :

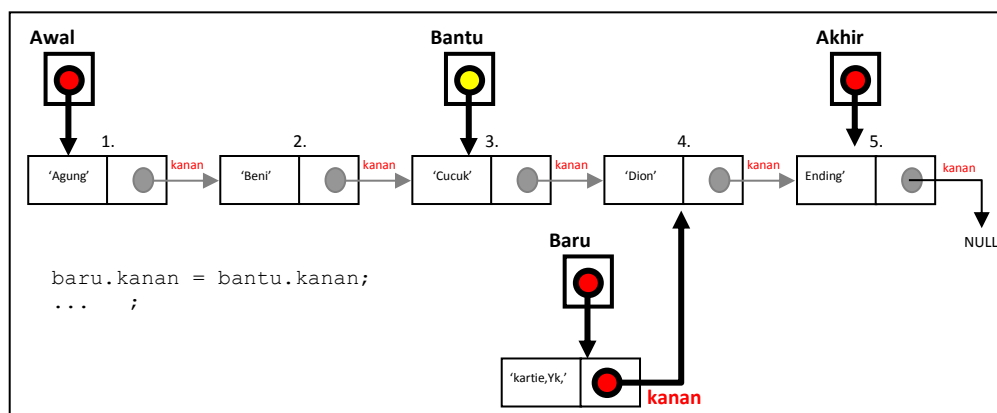
Mencangkokkan heap baru pada Single Linkedlist dengan urutan langkah sebagai berikut.

(a) Kondisi awal di mana pointer **Baru** dan **heapnya** belum tercangkok pada linklist



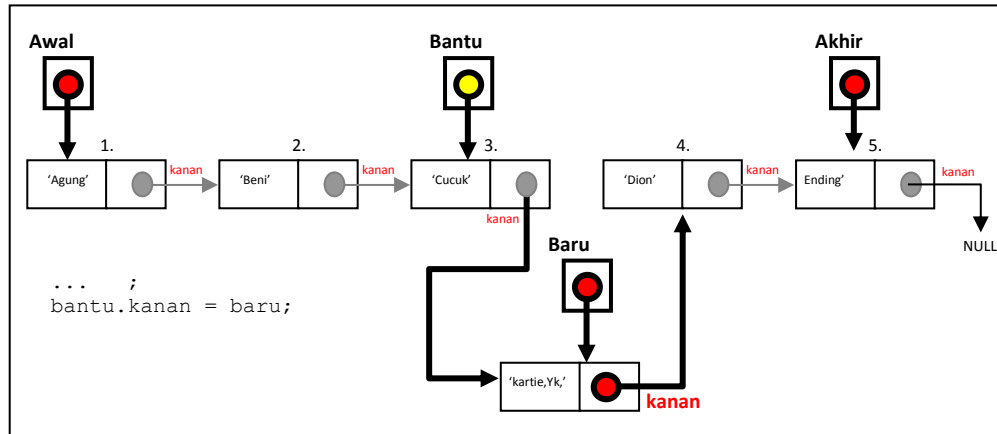
Gambar 9.5

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



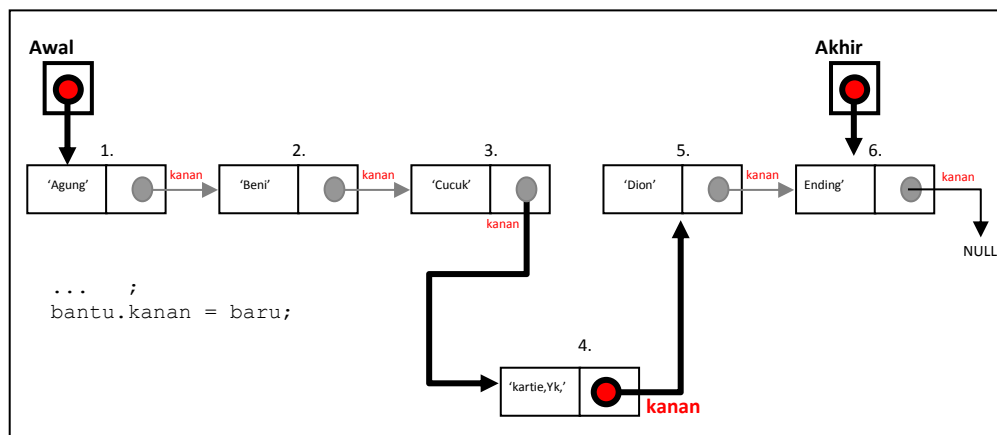
Gambar 9.6

(c) Arahkan pointer **Bantu.kanan** menunjuk ke heap yang ditunjuk oleh **Baru**.



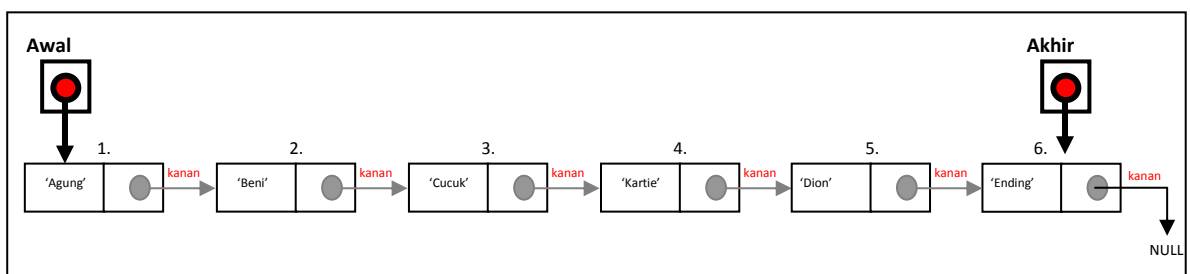
Gambar 9.7

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



Gambar 9.8

Hasil akhir yang diperoleh setelah penambahan data baru di tengah Single Linkedlist dapat dilihat Gambar 9.9. Tampak sekarang "Kartie" telah berada di antara "Cucuk" dan "Dion".



Gambar 9.9

9.2. Operasi Hapus Data di Depan, Tengah, dan Belakang

Operasi menghapus data pada Single 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 Single Linkedlist atau bahkan dalam kondisi dimana data tidak ditemukan sama sekali dalam Single Linkedlist.

Berikut ini langkah-langkah menghapus data pada Single Linkedlist.

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

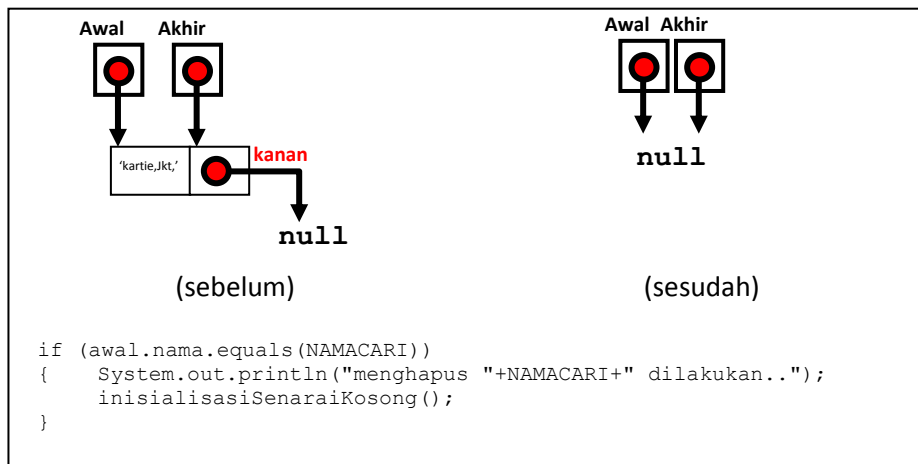
```
Scanner masukan = new Scanner(System.in);
System.out.print("Silakan masukkan nama yang ingin dihapus : ");
String NAMACARI = masukan.nextLine();
```

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

Situasi pertama, jika Single Linkedlist hanya berisi 1 (satu) heap (ditandai dengan Awal = Akhir) , maka hanya ada 2 kemungkinan yang ada. Yang pertama heap tersebut adalah data yang dicari, dan yang kedua adalah heap tersebut bukanlah data yang dicari. Tindakan untuk kedua kemungkinan tersebut adalah sebagai berikut.

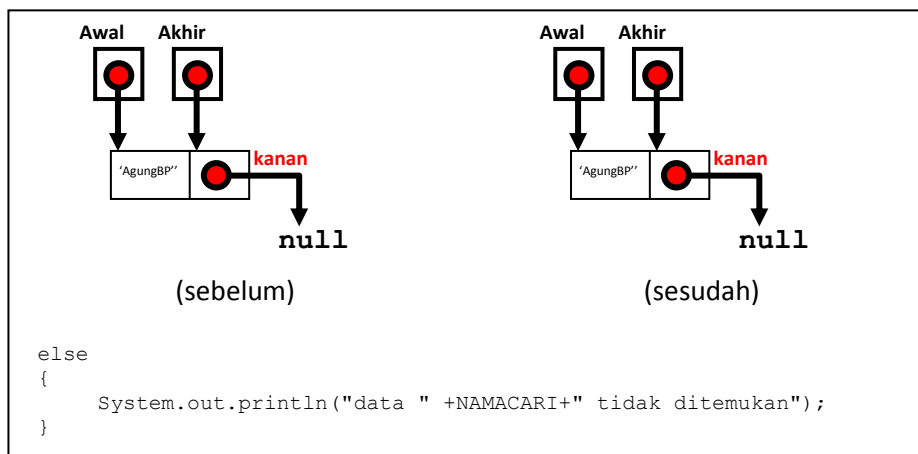
Kemungkinan pertama, jika data pada heap tersebut identik dengan kunci pencarian, maka heap dapat dihapus dengan cara Awal dan Akhir kembali dibuat menunjuk null (proses ini sama seperti proses inialisasi Single Linkedlist yang pernah dibahas pada sub bab 8.4) .

Gambar 9.10 berikut ini adalah contoh kasus di mana Data yang dicari "Kartie" dan data paling depan = "Kartie".



Gambar 9.10

Kemungkinan kedua, jika data pada heap tidak sama dengan kata kunci pencarian, maka kita dapat menyimpulkan bahwa data yang dicari tidak ada pada Single Linkedlist. Gambar 9.11 berikut ini adalah contoh kasus di mana Data yang dicari "Kartie" dan data paling depan = "AgungBP".

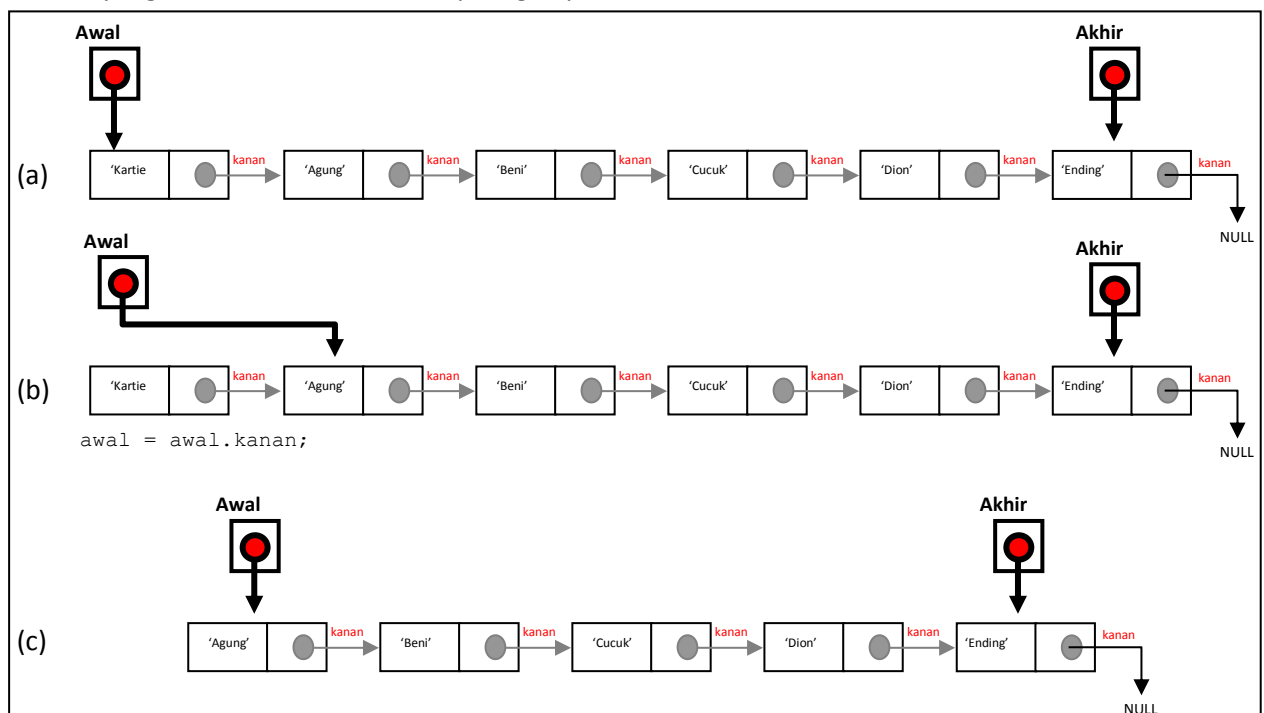


Gambar 9.11

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 Single 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 9.12 berikut ini adalah contoh kasus di mana Data yang dicari "Kartie" dan data paling depan = "Kartie".



Gambar 9.12

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;
}
```

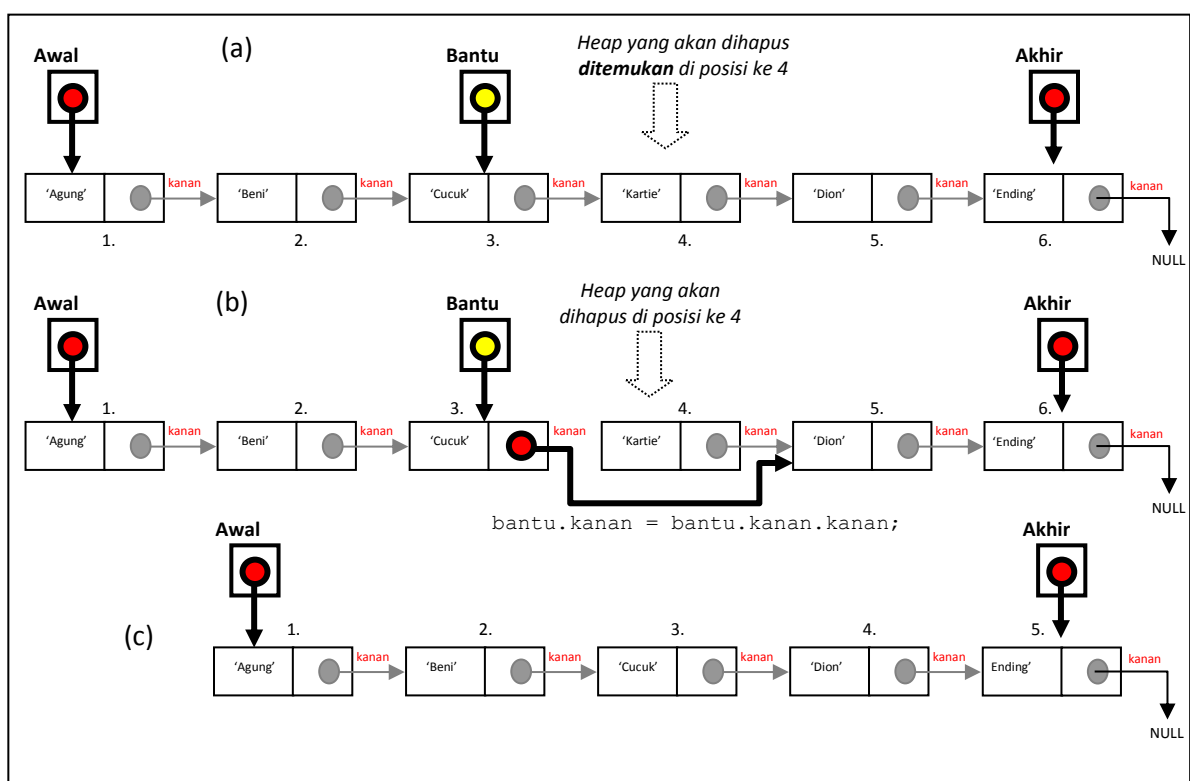
Situasi ketiga, jika Single Linkedlist berisi lebih dari 1 heap tetapi **data yang dicari bukan berada pada heap paling depan, bukan berada pada heap paling belakang**, maka perlu

dilacak dimanakah data tersebut berada. Pelacakan dilakukan dengan cara menempatkan pointer Bantu mulai dari posisi paling 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;
  while (bantu.kanan.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 berada di sebelah kiri dari heap yang akan dihapus sebagaimana Gambar 9.13.



Gambar 9.13

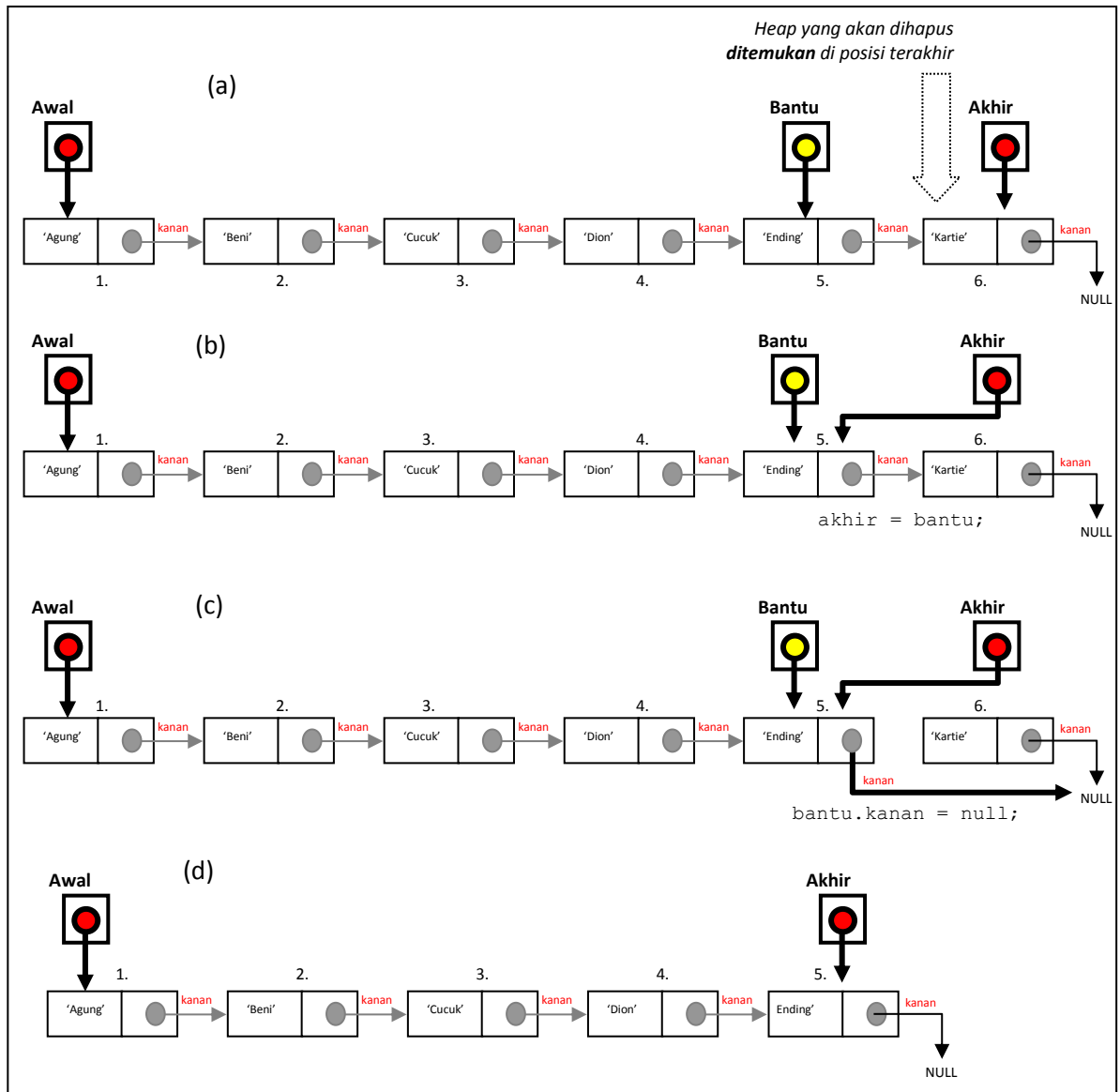
Penanganan dari situasi di atas dapat dilihat pada program berikut ini.

```

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

```

Kedua: jika data **ditemukan pada heap terakhir**, maka dilakukan langkah sebagaimana Gambar 9.14 berikut ini.



Gambar 9.14

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

```

else if (akhir.nama.equals(NAMACARI))//jika nama ditemukan di akhir
{
    akhir = bantu
    bantu.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 Single Linkedlist.

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

9.3. Latihan

Latihan 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;
    }
}

```

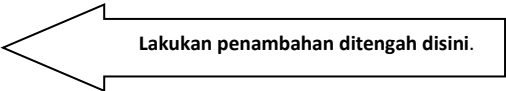
```

        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-----
        baru.kanan = bantu.kanan;
        bantu.kanan = baru;
    }
}

//-----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();

```

Jalankan 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.

Latihan 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

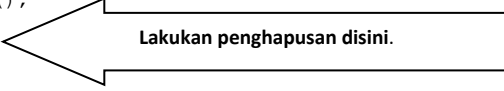
```

```

        { 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();
  }
}

```



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();

```

Jalankan 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

-

Tugas

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). Lakukan hal tersebut pada kertas tersendiri dan ditulis tangan (bukan diketik/print).

