BAB 8
PENGELOLAAN DATA PADA SINGLE LINKEDLIST : PENAMBAHAN DATA DI DEPAN, BELAKANG, & CETAK DATA

Tujuan :

Mahasiswa dapat mengimplementasikan penggunaan Single Linked List

Pada Bab 7 kita telah mempelajari bagaimana sebuah pointer, heap dan linkedlist diciptakan, termasuk juga kita telah mempelajari sekilas apakah Single Linkedlist dan Double Linkedlist itu. Pada bab 8 ini kita akan mempelajari lebih jauh tentang Single Linkedlist.

Single Linkedlist adalah kumpulan heap/ obyek/ simpul/ node yang saling terhubung satu sama lain (*linked*) dengan menggunakan **1 (satu) buah pointer pengait di bagian kanan** yang digunakan sebagai penghubung dengan simpul sejenis yang ada di sebelah kanannya. Pada kedua ujung dari single 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 pada heap paling belakang terkait dengan null.

Kebanyakan orang menyingkat Single Linkedlist (Senarai Berantai Tunggal) hanya dengan sebutan Linkedlist saja (Senarai Berantai).

Single Linkedlist memiliki 3 unsur pendukung yang penting, yaitu :

1. **Penunjuk** (disebut juga dengan ***pointer***),
2. **Simpul** (disebut juga dengan ***heap/ node/ list/ obyek/ senarai/ record***), dan
3. **Senarai Berantai Tunggal** atau Single Linked List itu sendiri.

8.1. Penunjuk (pointer)

Pada bab 7 subbab 7.1 kita telah mempelajari apa dan bagaimana definisi dari sebuah pointer. Pada bab ini kita akan membahas pointer terutama dalam hal penggunaannya di dalam Single Linkedlist.

Dalam Single Linkedlist, pointer dipergunakan untuk menunjuk sebuah simpul atau heap. Berikut ini adalah pointer-pointer yang akan dipergunakan dalam bab ini yaitu Awal, Akhir, Kanan, Baru, dan Bantu.

Pointer **Awal** adalah pointer yang bertugas untuk selalu menunjuk pada heap yang berada pada posisi yang paling depan dari linkedlist.

Pointer **Akhir** adalah pointer yang bertugas untuk selalu menunjuk pada heap yang berada pada posisi yang paling depan dari linkedlist.

Pointer **Kanan** adalah pointer yang bertugas untuk menunjuk pada heap yang berada di sebelah kanan dari heap yang dipegang saat ini. Khusus untuk heap yang berada pada posisi paling belakang, pointer Kanan harus selalu menunjuk ke **null**.

Pointer **Baru** adalah pointer yang bertugas untuk menunjuk kepada heap yang baru saja diciptakan dengan perintah `new`. Pointer ini nantinya akan berperan besar dalam merajut heap baru tersebut saat digabungkan dengan linkedlist yang telah ada.

Pointer **Bantu** adalah pointer yang bertugas untuk melakukan pelacakan data dari heap yang satu ke heap yang lainnya. Sesuai fungsinya, pointer Bantu ini tidak pernah menunjuk heap secara permanen, melainkan akan selalu perpindah-pindah dari satu heap ke heap yang lain.

8.2. Simpul (Heap/ Node)

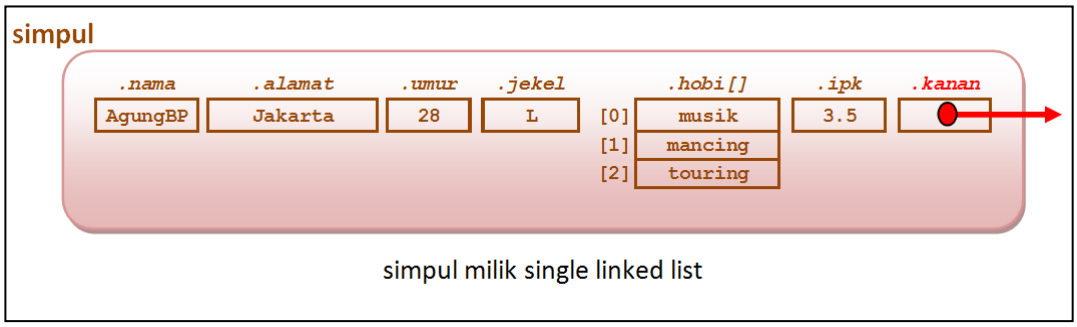
Deklarasi yang digunakan untuk menciptakan heap atau simpul pada Single Linkedlist dapat dilihat di bawah ini.

```
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;
}
```

Dari struktur heap di atas (dalam class simpul) terlihat adanya 2 kelompok anggota heap. Yang pertama adalah kelompok variabel-variabel penyimpan data (bagian inilah yang dimaksud dengan List atau Senarai). Kelompok ini nantinya akan digunakan untuk menyimpan data seperti biodata (`nama`, `alamat`, `umur` dan `jenis kelamin`). Sedangkan bagian yang kedua adalah 1 buah pointer penghubung yang bernama `kanan` yang nantinya akan bertugas untuk merangkaikan setiap heap dengan heap yang lain.

Adapun bentuk simpul yang tercipta dari struktur class seperti di atas dapat dilihat pada Gambar 8.1. di bawah ini.

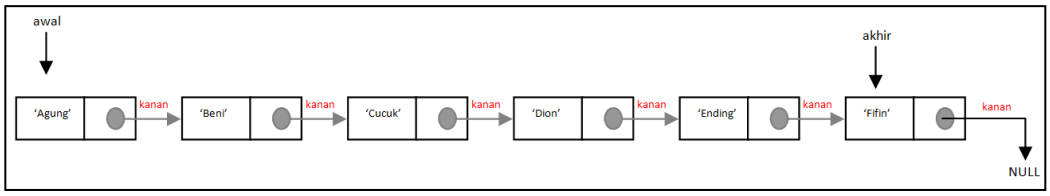


Gambar 8.1

Simpul sebagaimana tergambar pada Gambar 8.1 yang demikian disebut Simpul Tunggal atau Single List. Disebut Simpul Tunggal atau Single List karena simpul ini hanya mempunyai 1 (satu) buah penunjuk, yaitu 'kanan'.

8.3. Single Linked List (Senarai Berantai Tunggal)

Single Linkedlist atau Senarai Berantai Tunggal adalah kumpulan simpul-simpul tunggal yang terhubung satu dengan yang lain seperti yang dicontohkan pada Gambar 8.2 di bawah ini.



Gambar 8.2

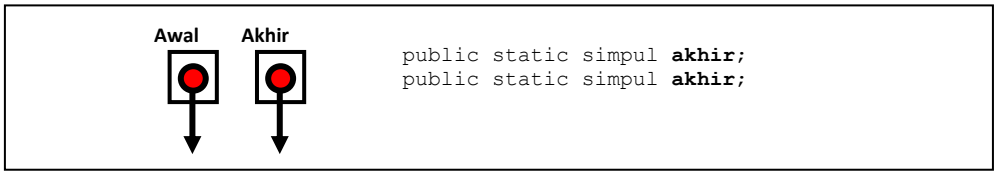
Di dalam mengelola media penyimpan data menggunakan Single LinkedList terdapat beberapa operasi yang dapat dilakukan antara lain :

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

8.4. Deklarasi Single Linked List

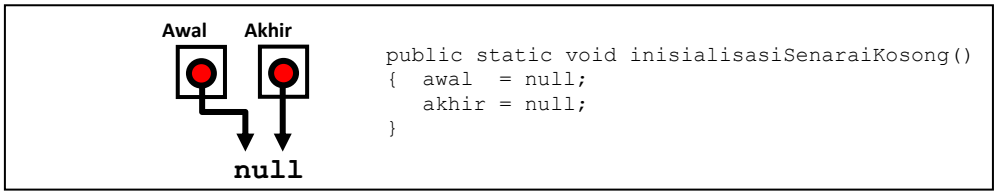
Dalam pembentukannya, Single Linkedlist di **deklarasikan** dan **diinisialisasi** dengan cara sebagai berikut :

Langkah 1: Menciptakan 2 buah pointer yaitu Awal dan Akhiir



Gambar 8.3

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



Gambar 8.4

Dalam script, deklarasi dan inisialisasi Single Linkedlist dituliskan 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()
        {   ..............          }
}
```

8.5. Operasi Tambah Data pada Single Linked List di bagian Depan

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

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

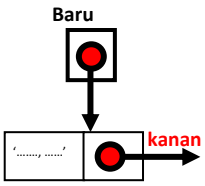
```
//------------bagian entri data dari keyboard--------------
String NAMA;
String ALAMAT;
int    UMUR;
char   JEKEL;
String HOBI[] = new String[3];
float  IPK;
Scanner masukan = new Scanner(System.in);
int bacaTombol=0;

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

```
simpul  baru;
```


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

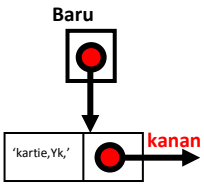


Gambar 8.5

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;
```

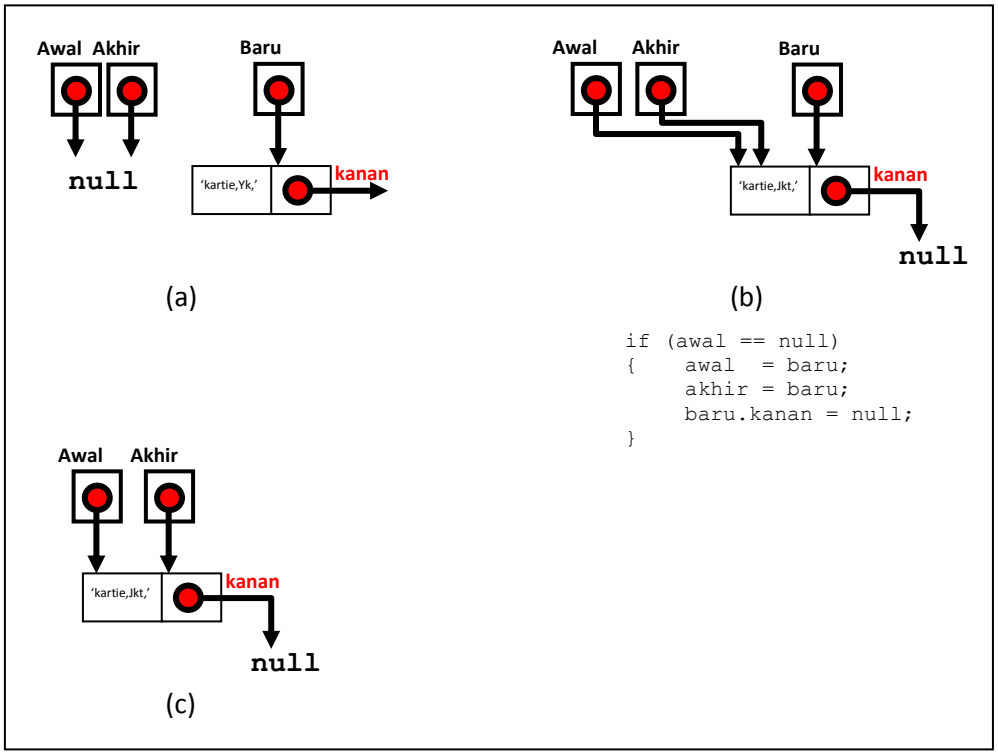
(pada gambar, tidak semua nilai diilustrasikan)



Gambar 8.6

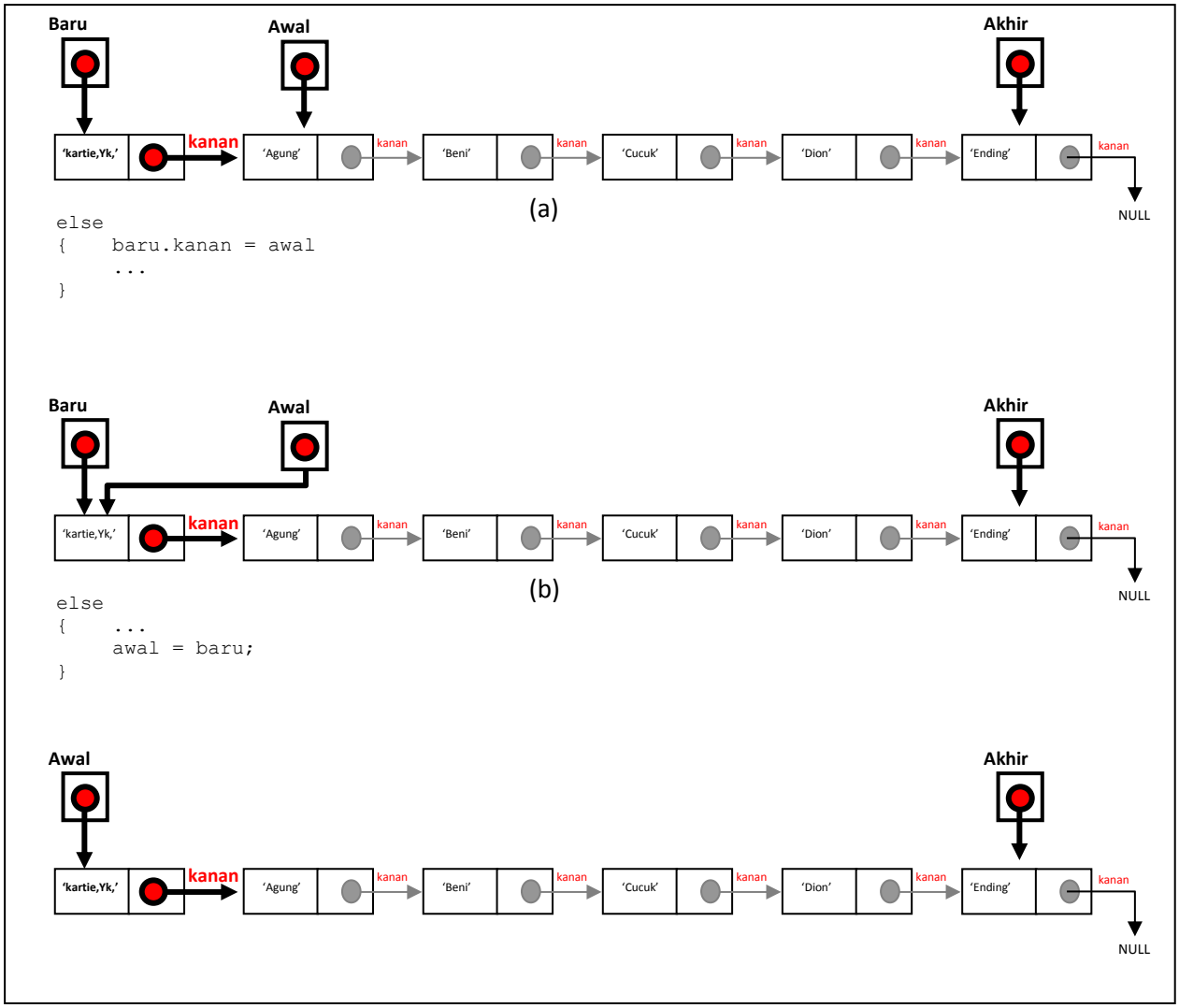
Langkah 4 :

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 Single Linkedlist. Karena baru ada satu heap penyimpan data maka pointer Awal dan Akhir harus diarahkan menunjuk ke heap tersebut.



Gambar 8.7

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



Gambar 8.8

8.6. Operasi Tambah Data pada Single Linked List bagian Belakang

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

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

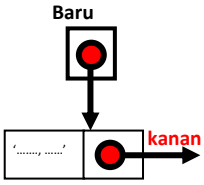
```
//------------bagian entri data dari keyboard--------------
String NAMA;
String ALAMAT;
int    UMUR;
char   JEKEL;
String HOBI[] = new String[3];
float  IPK;
Scanner masukan = new Scanner(System.in);
int bacaTombol=0;
```

```
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;
```


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

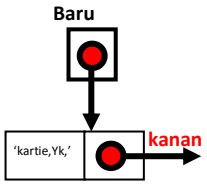


Gambar 8.9

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;
```

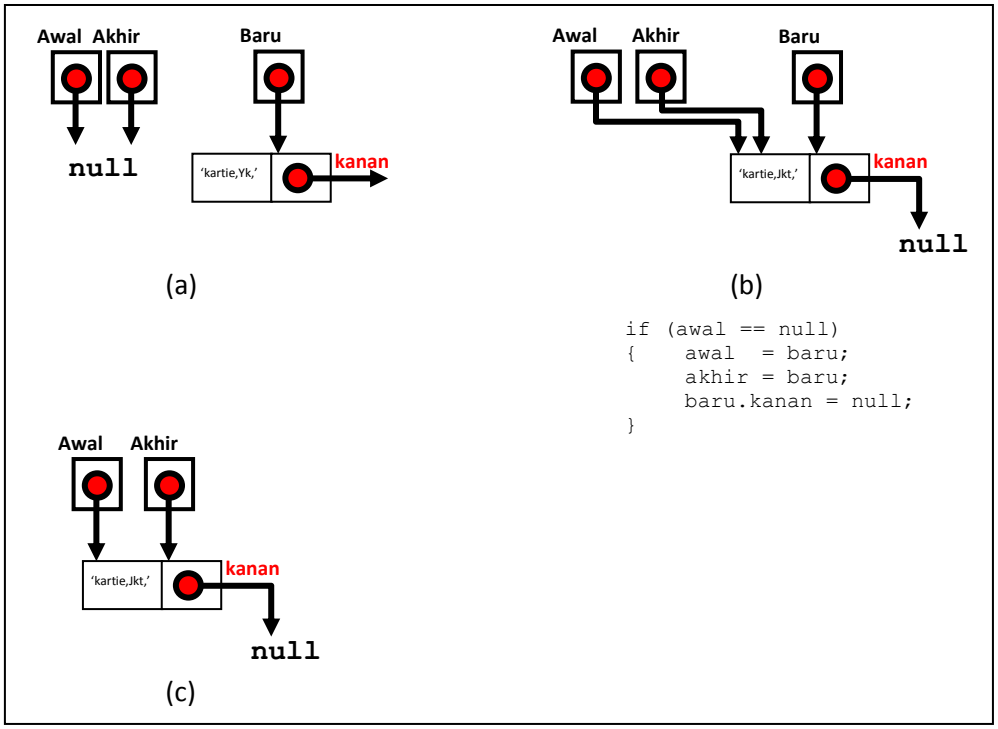
(pada gambar, tidak semua nilai diilustrasikan)



Gambar 8.10

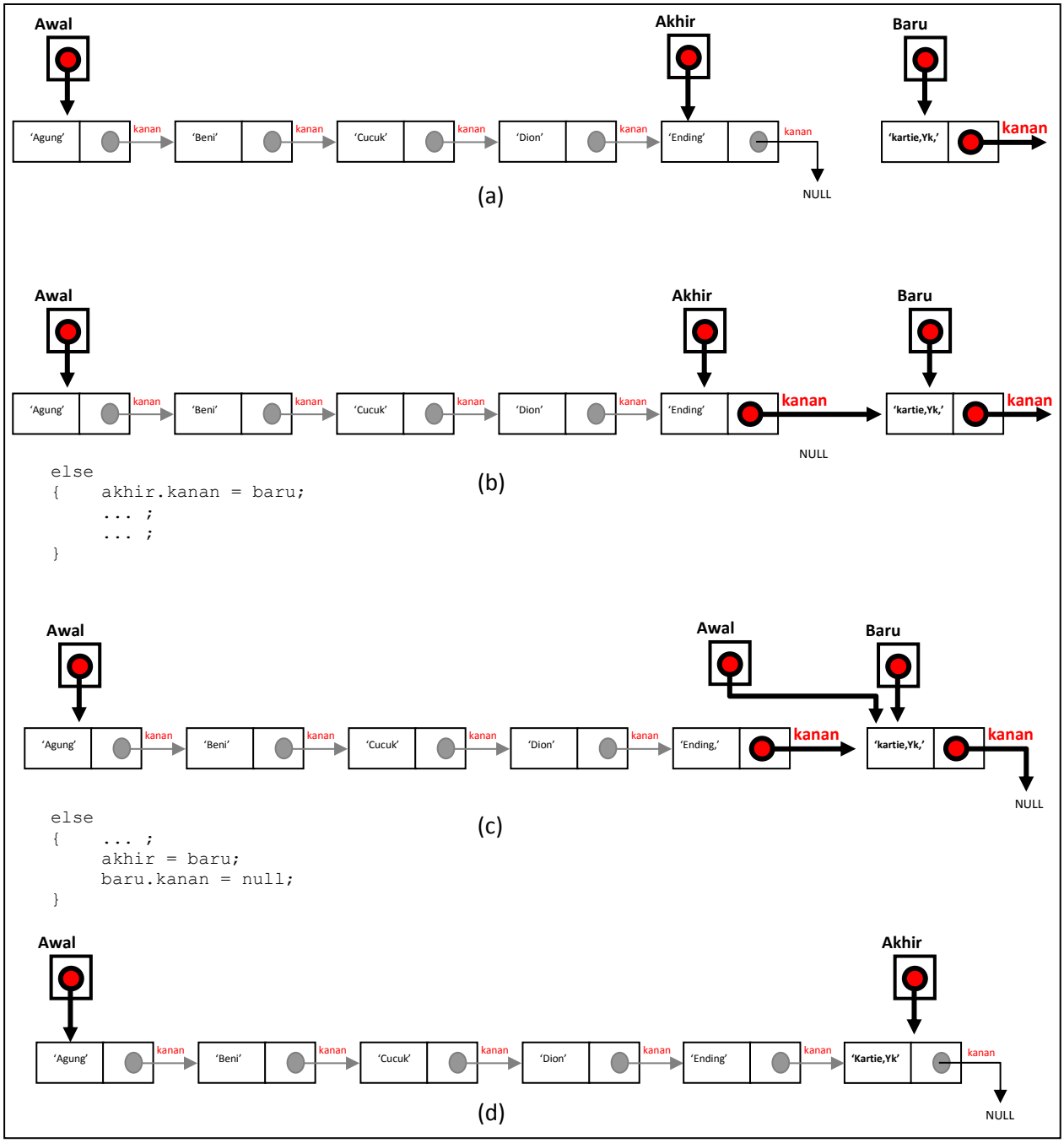
Langkah 4 :

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 Single Linkedlist. Karena baru ada satu heap penyimpan data maka pointer Awal dan Akhir harus diarahkan menunjuk ke heap tersebut.


Awal
Akhir
Baru
null
'kartie,Yk,'
kanan
(a)
Awal
Akhir
Baru
'kartie,Jkt,'
kanan
null
(b)
if (awal == null)
{ awal = baru;
akhir = baru;
baru.kanan = null;
}
Awal
Akhir
'kartie,Jkt,'
kanan
null
(c)

Gambar 8.11

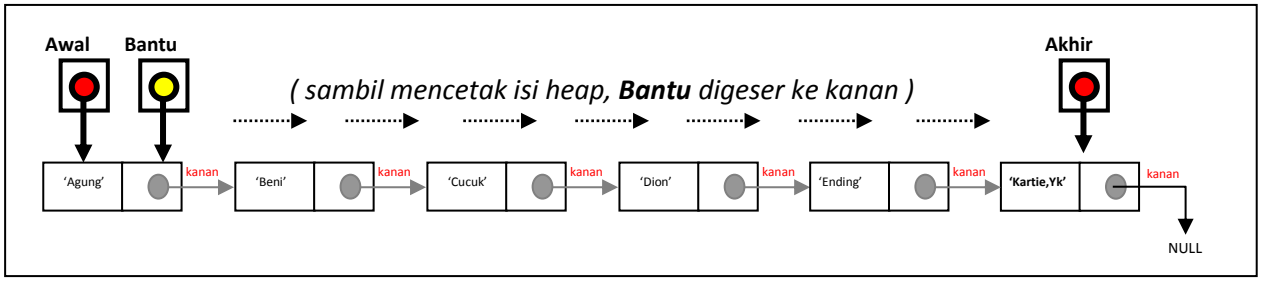
Pada kondisi : Pada kondisi : Jika senarai tidak kosong (ditandai dengan Awal atau Akhir tidak menunjuk ke null) maka tempatkanlah heap baru sebagai akhir dari Single Linkedlist dengan urutan cara sebagai berikut.



Gambar 8.12

8.7. Operasi Menampilkan Semua Data pada Single Linkedlist

Untuk dapat menampilkan data-data yang ada di dalam Single Linkedlist diperlukan sebuah pointer bernama Bantu yang akan bertugas untuk menunjuk satu persatu heap demi heap dimulai dari heap paling depan. Sambil menampilkan data dari heap yang sedang ditunjuk pointer Bantu ini akan digeser terus menerus, selangkah demi selangkah, sampai pointer ini menyentuh null yang berada di akhir Single Linkedlist.



Gambar 8.13

Adapun proses menampilkan isi Single 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("-----------------------------------------------------");
   }
}
```

8.8. Latihan

Latihan 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, ...)

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.

Latihan 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.

Tugas :

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