Implementasi Struktur Data

LABORATORIUM PRIDE

KELOMPOK KEAHLIAN PROGRAMMING

FAKULTAS ILMU TERAPAN

UNIVERSITAS TELKOM

DAFTAR PENYUSUN

Rizza Indah Mega Mandasari, S.Kom, M.T

Cahyana, S.T., M.Kom

LEMBAR REVISI

No	Keterangan Revisi	Tanggal Revisi Terakhir
1	Revisi Bagian Pertama	
2	Revisi Bagian Kedua	

LEMBAR PERNYATAAN

Saya yang bertanggung jawab di bawah ini:

Nama : Rizza Indah Mega Mandasari, S.Kom., M.T

NIP : 15880031

Dosen PJMP : Implementasi Struktur Data

Kelompok Keahlian : Interactive System

Menerangkan dengan sesungguhnya bahwa modul ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Ajaran 2016/2017 di Laboratorium Pride Fakultas Ilmu Terapan Universitas Telkom

Bandung, 29 Desember 2016

Mengetahui,

Ketua Kelompok Keahlian	Dosen PJMP
Hariandi Maulid, S.T., M.Sc	Rizza I. M. Mandasari, S.Kom., M.T
NIP	NIP 15880031

DAFTAR ISI

DAFTAR PENYUSUN 1
LEMBAR REVISI 2
LEMBAR PERNYATAAN 3
DAFTAR ISI 4
DAFTAR GAMBAR 6
DAFTAR PROGRAM 7
DAFTAR TABEL 8
Modul 0 : Running Modul 9
0.1 Tujuan 9
0.2 Peraturan Praktikum 9
0.3 Penilaian Praktikum 10
Modul 1 : Pengantar Struktur Data 11
1.1 Tujuan 11
1.2 Alat & Bahan 11
1.3 Dasar Teori 11
1.3.1 Algoritma dan Struktur Data 11
1.4 Array 12
1.4.1 Apa itu Array? 12
1.4.2 Deklarasi 12
1.4.3 Inisialisasi Array 13
1.5 ArrayList 15
1.6 Generic Types 16
Modul 2 : Singly Linked List 18
2.1 Tujuan 18
2.2 Alat & Bahan 18
2.3 Dasar Teori 18
2.3.1 Abstrack Data Type 18
2.3.2 Linked List 20
2.4 Operasi Dasar 22
Terdapat beberapa macam operasi dasar terhadap ADT pada Linked list. Secara umum operasi-operasi tersebut adalah: 22
2.4.1 Penyisipan Node (Insert) 22
2.4.2 Pencarian (Searching) 24

2.4.3 Penghapusan (Delete) 25
2.4.4 Traversing dan Display 28
2.5 Latihan 28
Modul 3 : Doubly Linked List 29
3.1 Tujuan 29
3.2 Alat & Bahan 29
3.3 Doubly Linked List 29

DAFTAR GAMBAR

Gambar 1.1 Ilustrasi Array 12
Gambar 2.1 Ilustrasi Linked List (Source: Carrano, 2001) 20
Gambar 2.2 Linked List 20
Gambar 2.3 Singly Linked List dengan 3 Node 20
Gambar 2.4 Insert Depan Singly Linked List 23
Gambar 2.5 Insert Belakang Singly Linked List 24
Gambar 2.6 Searching pada Link List 25
Gambar 2.7 Singly Link List Hapus Depan 26
Gambar 2.8 Singly Link List Hapus Belakang 27
Gambar 3.1 Doubly Linked List with 3 Nodes 29

DAFTAR PROGRAM

No table of figures entries found.

DAFTAR TABEL

No table of figures entries found.

Modul 0 : Running Modul

0.1 Tujuan

Setelah mengikuti Running Modul mahasiswa diharapkan dapat:

1. Memahami peraturan kegiatan praktikum.
2. Memahami Hak dan Kewajiban praktikan dalam kegiatan praktikum.
3. Memhami komponen penilaian kegiatan praktikum.

0.2 Peraturan Praktikum

1. Praktikum diampu oleh **Dosen Kelas** dan dibantu oleh **Asisten Laboratorium** dan **Asisten Praktikum**.
2. Praktikum dilaksanakan di Laboraturium Komputer Gedung FIT lantai 2 sesuai jadwal yang ditentukan.
3. Praktikan wajib membawa **modul praktikum, kartu praktikum, dan alat tulis**.
4. Praktikan wajib mengisi **daftar hadir** dan **BAP praktikum** dengan bolpoin **bertinta hitam**.
5. Durasi kegiatan praktikum **D3 = 4 jam (200 menit)**.
    a. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
    b. 60 menit untuk penyampaian materi
    c. 125 menit untuk pengerjaan jurnal dan tes akhir
6. Jumlah **pertemuan praktikum**:
    - 11 kali di lab (praktikum rutin)
    - 1 kali responsi (terkait Tugas Besar dan/atau UAS/COTS)
    - 1 kali berupa presentasi Tugas Besar dan/atau pelaksanaan UAS/COTS
7. Praktikan **wajib hadir minimal 75%** dari seluruh pertemuan praktikum di lab. Jika total kehadiran kurang dari 75% maka nilai COTS/UAS/ Tugas Besar = 0.
8. Praktikan yang datang terlambat :
    - <= 30 menit : diperbolehkan mengikuti praktikum tanpa tambahan waktu Tes Awal
    - > 30 menit : tidak diperbolehkan mengikuti praktikum
9. Saat praktikum berlangsung, asisten praktikum dan praktikan:
    - Wajib menggunakan **seragam** sesuai aturan Institusi.
    - Wajib mematikan/ men-silent semua **alat komunikasi**(smartphone, tab, iPad, dsb).
    - Dilarang membuka **aplikasi yang tidak berhubungan** dengan praktikum yang berlangsung.
    - Dilarang mengubah **setting software maupun hardware** komputer tanpa ijin.
    - Dilarang **membawa makanan maupun minuman** di ruang praktikum.
    - Dilarang **memberikan jawaban ke praktikan lain** (pre-test, TP, jurnal, dan post-test).
    - Dilarang **menyebarkan soal pre-test, jurnal, dan post-test**.
    - Dilarang**membuang sampah/sesuatu apapun** di ruangan praktikum.
10. Setiap praktikan dapat mengikuti praktikum susulan maksimal 2 modul untuk satu praktikum.
    - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan Institusi, yaitu rawat inap di Rumah Sakit

(menunjukkan bukti rawat inap dan resep obat dari RS), tugas dari Institusi (menunjukkan surat dinas dari Institusi), atau mendapat musibah (menunjukkan surat keterangan dari orangtua/ wali mahasiswa).

- Persyaratan untuk praktikum susulan diserahkan sesegera mungkin ke Asisten Praktikum untuk keperluan administrasi.

11. Pelanggaran terhadap peraturan praktikum ini akan ditindak secara tegas secara berjenjang di lingkup Kelas, Laboratorium, Program Studi, Fakultas, hingga Institusi.

0.3 Penilaian Praktikum

1. Komponen penilaian praktikum:
    - 20**%** nilai Tugas Pendahuluan
    - 20% nilai Tugas Awal
    - 50% Jurnal
    - 10% Skill
2. Seluruh komponen penilaian beserta pembobotannya ditentukan oleh dosen **PJMP**
3. Penilaian permodul dilakukan oleh **asisten praktikum**, sedangkan nilai Tugas Besar/ UAS diserahkan kepada **dosen kelas,** dilaporkan ke **PJMP.**
4. Baik praktikan maupun asisten tidak diperkenankan meminta atau memberikan **tugas tambahan** untuk perbaikan nilai.
5. Standar **indeks dan range nilai** ditentukan oleh dosen PJMP atas sepengetahuan Ketua Kelompok Keahlian

Modul 1 : Pengantar Struktur Data

1.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengenal struktur data
2. Menginstal IDE IntelliJ IDEA
3. Mengenal kosep Array dan Arraylist

1.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC.

1.3 Dasar Teori

1.3.1 Algoritma dan Struktur Data

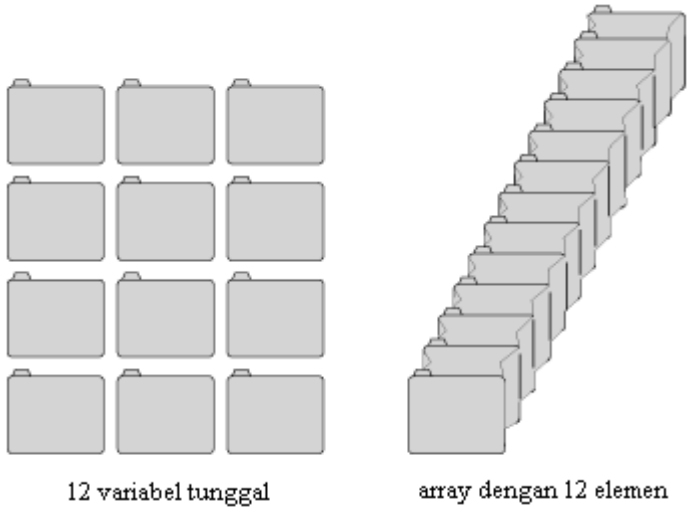
Istilah struktur data tidak dapat terpisah dari algoritma. Jika algoritma adalah suatu langkah atau prosedur yang ditujukan untuk memproses data, maka struktur data adalah bagaimana pengaturan data tersebut dimemori atau disk secara efisien saat akan dilakukan proses pengolahan data tersebut. Saat membuat sebuah algoritma, yang paling ditekankan adalah bagaimana membuat algoritma yang cepat, dan optimum. Dengan pemilhan struktur data yang tepat maka running time terhadap algoritma tersebut akan lebih baik.

*Djikstra Algorithm adalah algoritma yang digunakan untuk menemukan rute terpendek, atau masalah optimasi lintasan terpendek. Tanpa penggunaan struktur data yang baik running time dari Djikstra (impelentasi dengan Adjacency matrix) akan memakan waktu berkali-kali lipat (**$O(N)^2$**) namun dengan pemilihan struktur data yang tepat (implementasikan dengan priority queue) maka, running time yang dapat dimaksimalkan (**$O(N*log^n)$**).*

1.4 Array

1.4.1 Apa itu Array?

Array merupakan sekelompok data sejenis yang disimpan ke dalam suatu variabel yang mana variabel tersebut memiliki indeks untuk pengaksesan datanya. Berikut contoh ilustrasi antara variabel dengan array.



Gambar 1.1 Ilustrasi Array

Jika kita lihat pada gambar yang pertama, kita mendeklarasikan 12 buah variabel dengan nama yang berbeda tentunya untuk tujuan yang sama, misalnya untuk menyimpan total pengeluaran setiap bulan atau menyimpan identitas dari mahasiswa. Sekarang kita lihat gambar yang kedua, kita mendeklarasikan sebuah array dengan 12 element dan fungsinya sama untuk menyimpan total pengeluaran setiap bulan. Jika kita bandingkan tentunya akan lebih mudah jika kita mempergunakan array karena kita hanya mempergunakan satu buah variabel dalam prosesnya nanti, dan juga dalam implementasi ke dalam program kita tidak mungkin akan mendeklarasikan variabel sebanyak seribu buah untuk tujuan yang sama namun dengan menggunakan array untuk mendeklarasikan variabel sebanyak itu akan dapat dilakukan dengan mudah.

1.4.2 Deklarasi

Secara umum bentuk pendeklarasian tipe data array pada bahasa Java dapat dilakukan dengan format sebagai berikut.

```
Tipe_Data_Array  Nama_Array  [ukuran]
```

Layaknya variable, sebuah array juga memiliki tipe data dan perlu diperhatikan suatu array hanya bisa memiliki data yang bertipe sama saja. Tipe data dari suatu array bisa berupa integer, float, char, bahkan tipe objek. Tanda kurung [] pada pendeklarasian array di atas digunakan untuk menunjukkan jumlah elemen larik. Jika dideklarasikan berupa int x[10], maka x merupakan array yang berisi 10 elemen dengan tipe data integer. Selain itu, yang perlu diperhatikan lagi bahwa penghitungan elemen dari suatu array dimulai dari 0, bukan 1.

Namun, perlu diperhatikan bahwa pada Java, deklarasi array tidak otomatis menjadikan variabel array tersebut dapat digunakan, karena pada saat deklarasi hanya nama array saja yang terbentuk, sementara objek array belum terbentuk. Saat deklarasi, nilai array masih null, yang artinya tidak ada objek array tersebut. Agar terbentuk suatu objek array, perlu mengalokasikan array tersebut dengan keyword new. Perhatikan kode berikut.

```
Tipe_Data_Array  Nama_Array  [ukuran];
array-var = new type[size]
```

Pada kode program diatas, setelah melakukan deklarasi array, dilakukan alokasi pada variabel array tersebut dengan menggunakan new. Type pada bagian new harus memiliki tipe data yang sama dengan tipe data pada saat deklarasi array. Jika tidak dilakukan inisialiasi, otomatis akan diisi dengan nilai default masing-masing tipe data (misal, untuk tipe integer akan diisi nol).
Jadi, pembuatan array pada Java membutuhkan dua langkah; pertama deklarasi variabel array, kemudian alokasikan memori untuk variabel tersebut dengan menggunakan keyword new. Karena itu, proses alokasi ini bisa saja dilakukan pada baris program yang terpisah jauh dengan baris program deklarasi array.

1.4.3 Inisialisasi Array

Inisialisasi array dapat dilakukan langsung saat deklarasi array, dengan memasukkan serangkaian nilai yang dipisahkan oleh koma pada dalam kurung kurawal. Koma berfungsi memisahkan nilai antar elemen array. Jika melakukan insialisasi pada saat deklarasi, maka keyword new tidak diperlukan. Perhatikan contoh berikut.

```
package praktikum1;


class InitArray {
   public static void main(String args[]) {
     int jum_hari[] =
                    {31,28,31,30,31,30,31,31,30,31,30,31};
     System.out.println("Februari memiliki " + jum_hari[1] +
" hari.");

     //deklarasi array terpisah
     String nama_hari;
     nama_hari = new String[7];
     nama_hari[0] = "Senin";
     nama_hari[1] = "Selasa";
     }
 }
```

Apakah Outputnya?

……

……

……

Pada program diatas, terdapat dua cara deklarasi dan inisialisasi array. Yang pertama adalah dengan mendeklarasikan dan sekaligus melakukan inisialisasi array (pada array jum_hari), sementara pada array nama_hari, deklarasi dilakukan terpisah dengan inisialisasi. Inisialisasi pada array nama_hari hanya dilakukan pada elemen pertama dan kedua aarray. Selain itu, inisialisasi array juga dapat dilakukan dengan menggunakan perulangan. Perhatikan contoh berikut.

```
package praktikum2;

import.java.util.Scanner;

class DemoArray {
   public static void main(String args[]){
      int[] a = new int[100];
      Scanner in = new Scanner (System.in);

      System.out.println("Masukkan banyaknya nilai : ");
      int x = in.nextInt(); //menerima input dari console

      for (int i = 0; i < x; i++) {
         System.out.println("Input angka ke - "+(i+1)+ " :" );
         a[i] = in.nextInt();
       }

      System.out.println("Angka yang di-input-kan: " );

      for (i = 0; i < x; i++) {
         System.out.println(a[i]);
      }
   }
}
```

Apakah Outputnya?

……

……

……

Perhatikan bahwa deklarasi array dapat dilakukan dengan meletakkan kurung siku pada tipe data (bukan hanya pada nama variabel). Untuk menerima input dari console, digunakan Scanner. Karena Scanner merupakan objek dari kelas util dari java (bukan pada kelas yang sedang dijalankan) maka Scanner tersebut harus di-import dengan perintah import setelah nama package. Berikut contoh lain program yang menggunakan array.

```
package praktikum3;

import.java.util.Scanner;

class AnotherArray{
   publlic static void main (String args[]) {
      int a[] = new int[100];
      int x, sum, ganjil, genap;
      Scanner in = new Scanner (System.in);
      System.out.println("Masukkan banyaknya nilai: ");
      x = in.nextInt();
      sum = ganjil = genap = 0;

      for (int i = 0; i < x; i++) {
        System.out.println("Input angka ke- "+(i+1)+":");
          a[i] = in.nextInt();
          sum += a[i];
          if (a[i] % 2 == 0)
            genap += 1;
          else
            ganjil+=1;
      }
      System.out.println("Angka yang di-input-kan: " );
      for (i = 0; i < x; i++) {
        System.out.println(a[i]);
      }
```

Apakah Outputnya?

……

……

……

……

……

……

```
        System.out.println("Total bilangan : " + sum );
        System.out.println("Jumlah bilangan ganjil : " + ganjil );
        System.out.println("Jumlah bilangan genap : "+ genap);

    }
}
```

1.5 ArrayList

ArrayList merupakan bagian dari data collection yang disediakan oleh Java API untuk menyimpan grup dari objek yang saling berhubungan. Pengaturan data di dalam ArrayList sama seperti pada array, dimana objek dengan tipe data yang sama disimpan secara berurutan. Namun, kita tidak perlu membuat kode program untuk pengaturan data tersebut, karena sudah disediakan oleh Java. Selain itu, dengan menggunakan ArrayList, alokasi data dapat dilakukan secara dinamis (dapat ditambah dan dikurangi, jika menggunakan array tidak bisa). Adapun cara deklarasi ArrayList adalah sebagai berikut.

```
ArrayLIst<T> nama_variabel = new ArrayList<T>
```

Perintah <T> merupakan tipe data yang digunakan untuk ArrayList. Misal, jika ingin membuat suatu ArrayList dari String, dituliskan ArrayList<String>. Seperti pada array, keyword new digunakan untuk mengalokasikan ArrayList pada memori. Method-method yang akan digunakan dalam ArrayList dapat dilihat pada Tabel 1-1.

Table 1-1 Method pada ArrayList

Method	Keterangan
add	Menambahkan elemen di ujung AraryList
get	Mengambil elemen pada indeks tertentu dalam ArrayList
remove	Menghapus elemen yang pertama kali ditemukan pada ArayList
size	Mengembalikan besar ArrayList

Method yang dipelajari pada praktikum ini terbatas untuk add, get dan remove. Penjelasan lebih mendalam mengenai ArrayList akan dipelajari pada mata kuliah Implementasi Struktur Data. Karena ArrayList berada pada kelas lain, maka harus di-import terlebih dahulu. Untuk lebih jelasnya, lihat contoh kode berikut ini.

```
Package praktikum4;

import java.util.ArrayList; // import ArrayList

class DemoArrayList {
   public static void main( String[] args ) {
       ArrayList<String> baju = new ArrayList<String>();
       baju.add( seragam");
```

Apakah Outputnya?

……

……

```
        baju.add("gaun");
        baju.add(0,”kaos kaki");

        /*tampilkan isi array*/
        for ( int i=0; i < items.size(); i++ )
           System.out.print (baju.get( i ) + ” ” );

        baju.add(“gaun”); //menambahkan gaun diujung array
        baju.remove( "gaun" ); // menghapus elemen gaun
pertama yang ditemui
        /*tampilkan isi array*/
        for ( int i=0; i < items.size(); i++ )
           System.out.print (baju.get( i ) + ” ” );
        }
}
```

……

Perintah add akan menambahkan elemen pada ujung array. Namun, jika ditentukan indeks spesifik, seperti pada perintah add (0, “kaos kaki”) maka elemen akan diletakkan pada tempat yang ditunjukkan oleh indeks (dalam hal ini, indeks pertama array). Perintah remove hanya menghapus elemen “gaun” yang ditemukan pertama kali.

1.6 Generic Types

Generic type adalah sebuah Class atau interface umum yang parameternya dapat diganti type data apapun selain type data primitive. Contohnya dapat dilihat pada code dibawah ini.

```
public class Pair<KeyType, ValueType> {

    private final KeyType key;
    private final ValueType value;

    public Pair(KeyType key, ValueType value) {
        this.key = key;
        this.value = value;
    }

    public KeyType getKey() {
        return key;
    }

    public ValueType getValue() {
        return value;
    }

    public String toString() {
        return "(" + key + ", " + value + ")";
    }
}
```

Dalam penggunaannya Generic Class dapat digunakan seperti contoh dibawah ini:

```
Pair<String, Integer> p1 = new OrderedPair<String, Integer>("Even", 8);
Pair<String, String>  p2 = new OrderedPair<String, String>("hello", "world");
```

Case!

Kita dapat menambahkan value ke dalam array selama array tersebut tidak penuh. Selain itu kita dapat menambahkan value ke index tertentu didalam array. Bagaimana jika kita ingin menambahkan value ke dalam index yang telah berisi?

Contoh: add(23,1)

10	0
8	1
-11	2
9	3
3	4
	5
	6

Modul 2 : Singly Linked List

2.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengenal konsep linked list
2. Mengenal konsep singly link list

2.2 Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

2.3 Dasar Teori

2.3.1 Abstrack Data Type

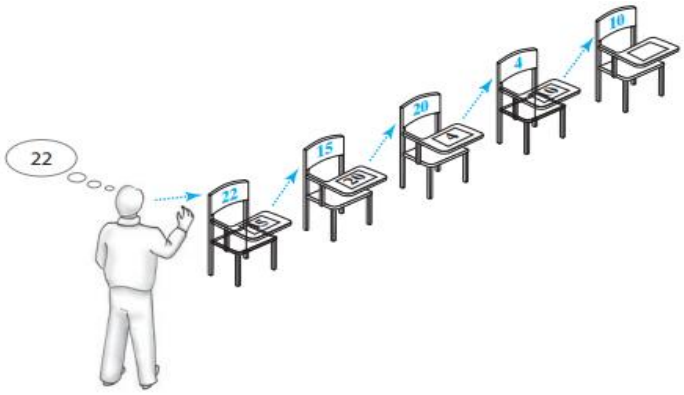
Abstract data type (ADT) adalah spesifikasi suatu set data dan operasi yang dilakukan pada data tersebut. Spesifikasi ini tidak menyebutkan mengenai bagaimana cara menyimpan data, ataupun implementasinya. Contohnya adalah data mahasiswa dibawah ini:

Abstrak Data Type: Mahasiswa		
Data: Nama, NIM, Kelas, IPK, dll.		
Operation		
Pseudocode	UML	Description
isEmpty()	+ isEmpty()	Task: Cek apakah mahasiswa kosong. Input: - Output: True or false berdasarkan data kosong atau tidaknya.
AddFirst(newEntry)	+ AddFirst(newEntry)	Task: Menambahkan data mahasiswa didepan list. Input: newEntry object Output: True or false berdasarkan berhasil atau tidaknya penambahan data.
Remove(newEntry)	+ Remove(newEntry)	Task : Menghapus data mahasiswa Input : newEntry object Output: True or false berdasarkan berhasil atau tidaknya penghapusan data.
Clear()	+ Clear()	Task : Menghapus seluruh data mahasiswa Input : - Output: True or false berdasarkan berhasil atau tidaknya penghapusan seluruh data.
TraverseList	+ TraverseList()	Task : Menyusuri seluruh data mahasiswa.

Semakin detail spesifikasi ADT yang diberikan maka, semakin kompleks dan semakin detail method yang akan digunakan dibahasa pemrograman. Spesifikasi operasi terhadap ADT dapat dikumpulkan menjadi sebuah interface yang mengimplementasikan ADT dalam Bahasa Java. Contoh interface Java yang berisi method untuk ADT diatas dapat dilihat pada code dibawah ini:

```
public interface ListMahasiswa<T> {

    public boolean isEmpty();
    /** Cek apakah list kosong.
     @return true jika kosong, or false tidak */

    public boolean addFirst(T newEntry);
    /** Menyisipkan node didepan list.
     @param newEntry adalah object yang akan ditambahkan
     @return true jika berhasil ditambahkan, or false jika tidak */

    public boolean addLast(T newEntry);
    /** Menyisipkan node didepan list.
     @param newEntry adalah object yang akan ditambahkan
     @return true jika berhasil ditambahkan, or false jika tidak */

    public T findData(String nama);
    /** Menelusuri setiap node didalam list.
     @param nama adalah salah satu data di dalam list
     @return seluruh data dalam node tsb jika ditemukan, null jika tidak
ditemukan */

    public int getPositionOf(String nama);
    /** Menelusuri setiap node didalam list.
     @param nama adalah salah satu data di dalam list
     @return posisi node tsb dalam list, -1 jika tidak ditemukan */

    public T remove(int givenPosition);
    /** Remove salah satu node di posisi tertentu, jika memungkinkan.
     // @param posisi dari node tsb
     @return data dalam node yg dihapus, null jika tidak berhasil di-remove
*/

    public void clear();
    /** Remove semua node dilist. */

    public boolean contains(T anEntry);
    /** Tes apakah node tersebut memiliki data sesuai dengan anEntry
     // @param anEntry data yang ingin dicari
     @return true jika ditemukan anEntry, or false jika tidak ditemukan */

    public void traverseList();
    /** Menyelusuri setiap node didalam list dan menampilkan data yang
menjadi isi node. */
}
```

2.3.2 Linked List



Gambar 2.1 Ilustrasi Linked List (Source: Carrano, 2001)

Linked List (biasa disebut list saja) adalah salah satu bentuk struktur data (representasi penyimpanan) berupa serangkaian elemen data yang saling berkait (berhubungan) tersusun secara linear dengan masing-masing disimpan dalam sebuah Node(simpul). Data yang disimpan dalam sebuah node terdiri atas dua bagian. Pertama adalah bagian yang menyimpan data, sedangkan bagian yang kedua menyimpan referensi atau penunjuk pada node berikutnya. Data pada Linked List dapat berupa data tunggal atau data majemuk.

Gambar 2.2 Linked List

Pada pembahasan kali ini yang akan dibahas adalah singly linked list. Singly linked list merupakan linked list dimana setiap node **hanya** menyimpan referensi terhadap node selanjutya. Sebuah linked list minimal harus memiliki referensi pada Node pertama (head) di list tersebut. Tanpa sebuah referensi yang merujuk ke head, tidak akan ada cara utuk menemukan node dan rangkaiannya.

Gambar 2.3 Singly Linked List dengan 3 Node

Node terakhir di list dikenal dengan istilah tail. Tail dapat ditemukan dengan cara *traversing* linked list tersebut. Traversing (menyusuri) adalah melintasi setiap Node dimulai dari Head menuju node selanjutnya mengikuti referensi dari setiap node hingga ditemukan tail. Tail dapat diidentifikasi sebagai sebuah node yang memiliki Null value untuk referensi selanjutnya. Definisi singkat dari Node dapat dilihat dibawah ini.

```
private class Node {
        private T data; // entry in bag
        private Node next; // link to next node

        private Node(T dataPortion) {
            this(dataPortion, null );
        }
        private Node(T dataPortion, Node nextNode)
        {
            data = dataPortion;
            next = nextNode;
        } // end constructor

        private T getData()
        {
            return data;
        } // end getData

    private void setData(T newData)
        {
            data = newData;
        } // end setData

        private Node getNextNode()
        {
            return next;
        } // end getNextNode

        private void setNextNode(Node nextNode)
        {
            next = nextNode;
        } // end setNextNode

        @Override
        public String toString() {
            return this.data.toString();
        }
}
```

Karena Node dibuat menjadi Inner Class, maka generic type T akan sama dengan jenis generik dinyatakan oleh outer class dari Node tesebut. Dengan demikian, kita tidak menulis <T> setelah Node, namun, jika Node bukan inner class tetapi memiliki akses **package** atau **public**, maka harus dituliskan Node <T>. T dapat diganti sesuai dengan ADT yang telah dibuat. (Lihat Modul 1.6)

Untuk implementasinya, kita akan menggunakan singly linked list berisi Node diatas untuk menghubungkan antara Node satu dan Node lainnya. Pada implementasi ini harus “mengingat” Node pertama(firstnode) dari linked list ini.

```
public class LinkedMahasiswa<T> implements MahasiswaInterface<T>
{
    private Node firstNode; // reference to firstnode
    private Node lastNode; // reference to lastnode
    private int numberOfEntries;

    public LinkedMahasiswa()
    {
        firstNode = null ;
        numberOfEntries = 0;
    }

<< Implementasi dari public methods yang dideklarasikan di Interface. >>

    private class Node // private inner class
    {
        < Lihat implementasi Node diatas >
    } // end Node
} // end
```

2.4 Operasi Dasar

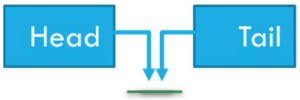
Terdapat beberapa macam operasi dasar terhadap ADT pada Linked list. Secara umum operasi-operasi tersebut adalah:

1. Penyisipan Node (Insert)
2. Penghapusan Node (Delete)
3. Penelusuran Node dan menampilkan isi Node (Traversing)
4. Pencarian Node (Searching)
5. Pengubahan isi Node (Update)

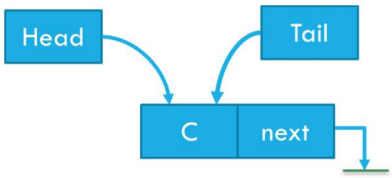
2.4.1 Penyisipan Node (Insert)

Penyisipan Node pada list dapat dilakukan didepan, dibelakang ataupun diantara Node-node tertentu. Pada penyisipan didepan, langkah-langkah dapat dilihat dibawah ini:

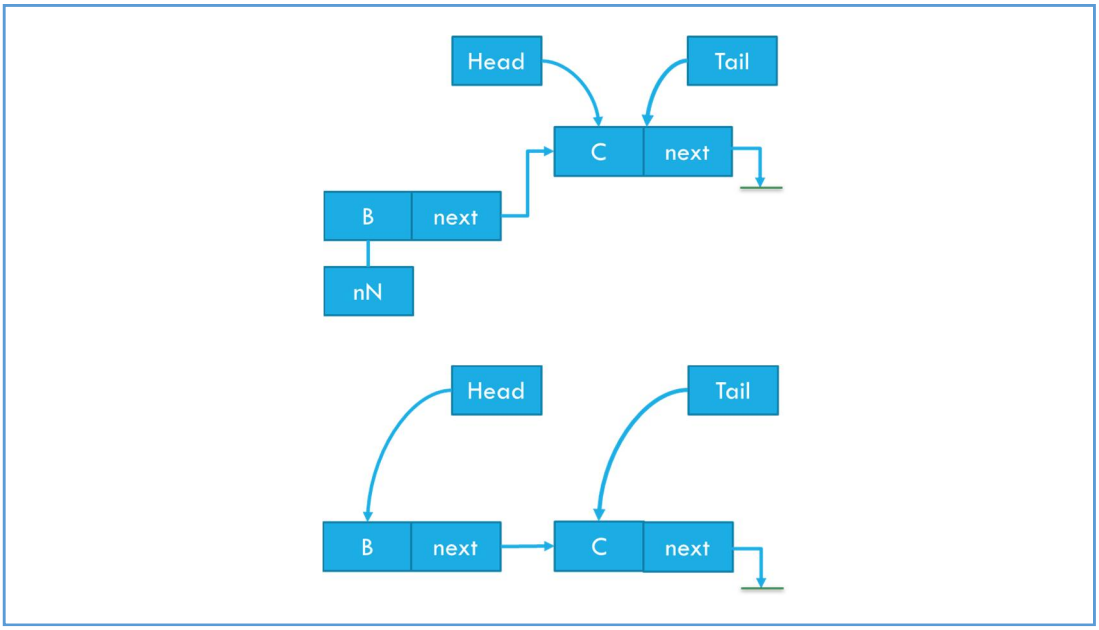
1. Ketika List kosong maka head/ firstnode dan tail/lastnode menunjuk null



2. Ketika disisipkan node baru, maka head dan tail akan menunjuk node tersebut.



3. Ketika kemudian akan disisipkan node baru di depan linked list yang ada, maka newNode next akan menunjuk head. Dan head menunjuk pada newnode.

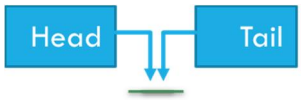


Gambar 2.4 Insert Depan Singly Linked List

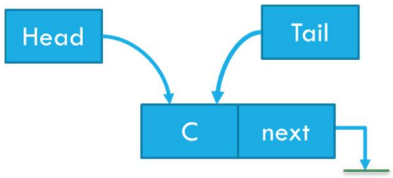
```
public boolean addFirst(T newEntry) {
        // add to beginning of chain:
        Node newNode = new Node(newEntry);
        newNode.setNextNode(firstNode);

        // (firstNode is null if chain is empty)
        if (isEmpty()) {
            firstNode = newNode; // new node is at beginning of chain
            lastNode = newNode;
        }else{
            newNode.setNextNode(firstNode);
            firstNode = newNode;
        }
        numberOfEntries++;
        return true;
    }
```

Sedangkan untuk lngkah-langkah pada penyisipan belakang dapat dilihat dibawah ini:

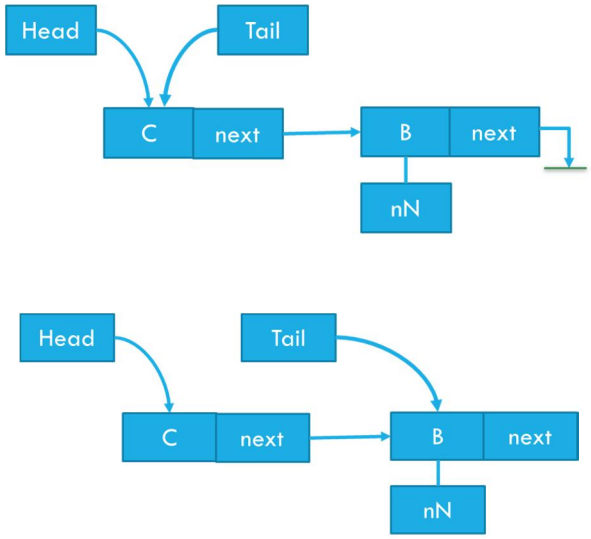
1. Ketika List kosong maka head/ firstnode dan tail/lastnode menunjuk null



2. Ketika disisipkan node baru, maka head dan tail akan menunjuk node tersebut.



3. Ketika kemudian akan disisipkan node baru di depan linked list yang ada, maka tail next akan menunjuk newNode. Dan tail akan menunjuk newNode.



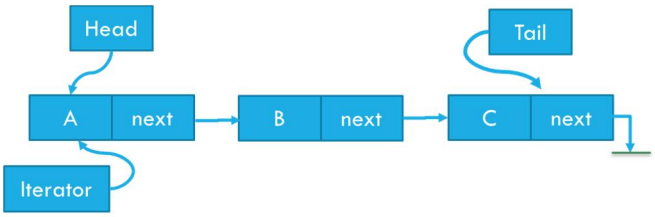
Gambar 2.5 Insert Belakang Singly Linked List

```
public boolean addLast(T newEntry) {
    Node newNode = new Node(newEntry);

    lastNode.setNextNode(newNode);
    lastNode = newNode;
    numberOfEntries++;
    return true;
}
```

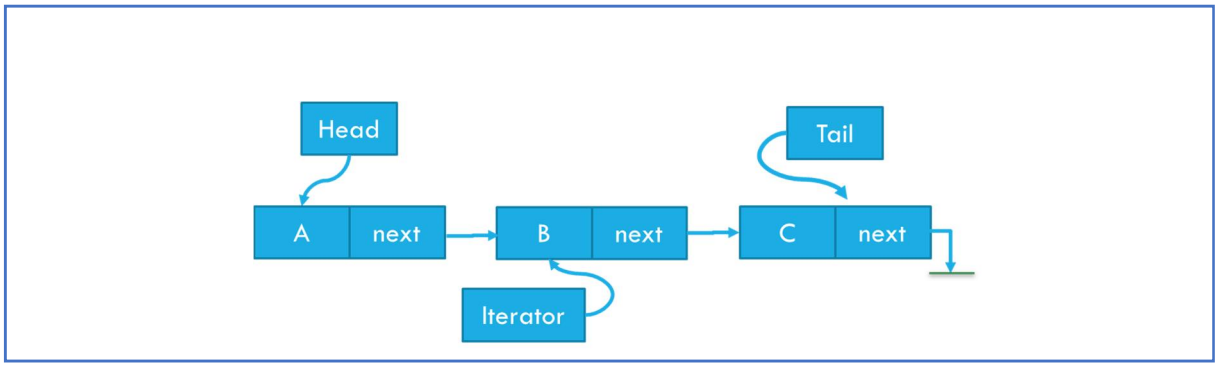
2.4.2 Pencarian (Searching)

Searching merupakan operasi dasar list dengan melakukan aktivitas pencarian terhadap node tertentu. Proses ini berjalan dengan mengunjungi setiap node dan berhenti setelah node yang dicari ketemu. Dengan melakukan operasi searching, operasi-operasi seperti insert after, delete after, dan update akan lebih mudah.

1. Set sebuah iterator pada head.



2. Lakukan perbandingan data yang dikunjungi iterator, bila ditemukan maka kembalikan nilai pada node tersebut, jika tidak ditemukan lakukan step 3.
3. Pindahkan iterator pada node selanjutnya. Lakukan step 2 hingga ditemukan atau iterator telah mencapai akhir list. (iterator mencapai tail)



Gambar 2.6 Searching pada Link List

```
public T findData(String nama) {
    boolean found = false;
    Node iterator = firstNode;
    while (!found && (iterator != null ))
    {
        Mahasiswa mahasiswa = (Mahasiswa)iterator.data;
        if (nama.equals(mahasiswa.getNama())) {
            found = true;
            return iterator.data;
        } else {
            iterator = iterator.next;
        }
    }
    return null;
} // end
```

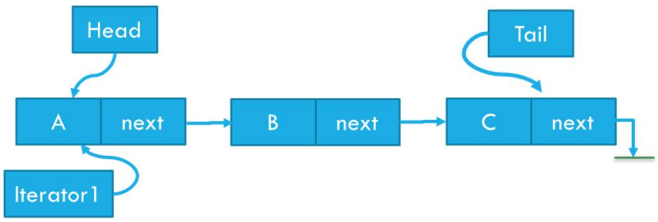
Selain menggunakan cara diatas, pencarian pun dapat dilakukan dengan mengunjungi setiap node, berhenti apabila node tersebut ditemukan dan mengembalikan posisi dari node tersebut. Posisi node tersebut dapat digunakan untuk parameter masukan dalam melakukan penghapusan maupun penyisipan.

```
public int getPositionOf(String nama){
    assert (firstNode !=null); //
    Node current = firstNode;

    for (int i = 1; i < numberOfEntries; i++) { //numberOfEntries = size
                                  //dari link list tsb
        Mahasiswa mahasiswa = (Mahasiswa) current.data;
        if (nama.equals(mahasiswa.getNama()))
            return i;
        else
            current = current.next;
    }
    return -1;
}
```

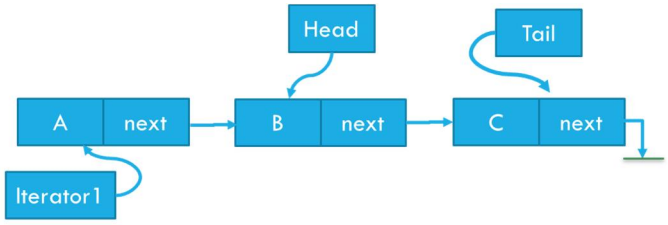
2.4.3 Penghapusan (Delete)

Sama seperti penyisipan Node, penghapusan Node pada list dapat dilakukan didepan, dibelakang ataupun diantara Node-node tertentu. Pada penghapusan sebuah Node disebelah depan, langkah-langkah dapat dilihat dibawah ini:

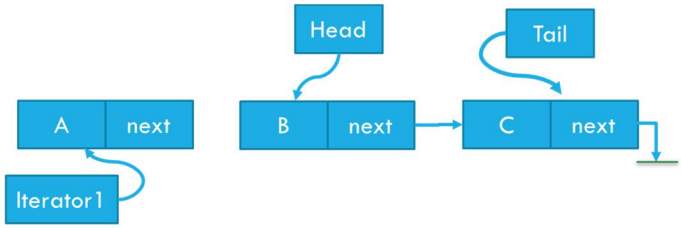
1. Cek apakah list tersebut kosong. Jika tidak kosong lanjutkan menuju step-2.
2. Jika tidak kosong, set Iterator pada Head atau firstNode.



3. Pindahkan Head pada node setelah Head.



4. Set Iterator menjadi null.

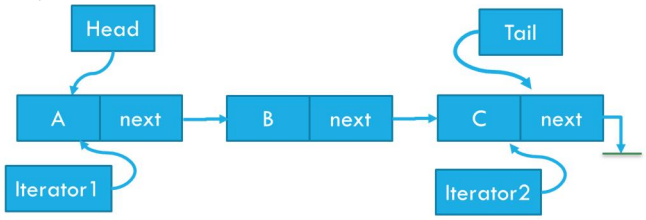


Gambar 2.7 Singly Link List Hapus Depan

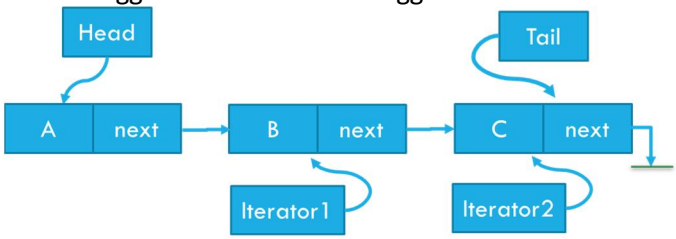
```
public T remove() {
  assert !isEmpty();
  T iterator = null; // return value
          iterator = firstNode.getData(); // save entry to be removed
          firstNode = firstNode.getNextNode();
          if (numberOfEntries == 1)
              lastNode = null ; // solitary entry was removed
      }
      numberOfEntries--;
  return iterator; // return removed entry
}
```

Sedangkan untuk melakukan hapus belakang, dibutuhkan dua buah iterator. Berikut langkah-langkah menghapus node dibelakang.

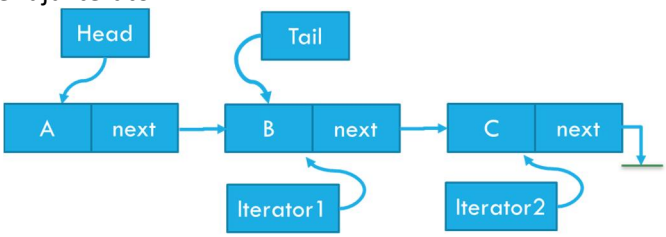
1. Cek apakah list kosong.
2. Set iterator-1 di head, Set iterator-2 di tail.



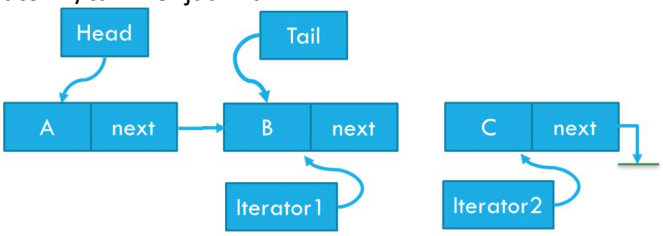
3. Lakukan penelusuran menggunakan iterator-1 hingga satu node sebelum tail.



4. Pindahkan tail menuju iterator-1.



5. Set reference iterator-1/tail menjadi null.



6. Kembalikan nilai iterator-2.

Gambar 2.8 Singly Link List Hapus Belakang

```
public T remove() {
    T result = null; // iterator 2 = result, return value
    assert !isEmpty();
    Node iterator1 = firstNode;
    Node iterator2 = lastNode;
    While(iterator1.getNextNode() == lastNode){
       iterator1 = iterator1.getNextNode;
    }
    lastNode = iterator1;
    iterator1.setNextNode(null);
    result = lastNode.getData(); // save entry to be removed
    numberOfEntries--;
    }
    return result; // return removed entry
}
```

2.4.4 Traversing dan Display

Traversing merupakan operasi dasar pada list yang menelusuri setiap Node dpada list dimulai dari Head menuju node selanjutnya mengikuti referensi hingga ditemukan tail/node terakhir. Traversing dapat digunakan untuk melakukan pencarian, menampilkan semua node dalam list, dan menghapus semua node/mengosongkan list tersebut.

```
public void traverseList() {
    if (this.firstNode == null) {
        return;
    }

    Node node = this.firstNode;

    while (node != null) {
        System.out.println(node + " ");
        node = node.getNextNode();
    }
}
```

2.5 Latihan

Buatlah ADT pegawai dan Singly Linked List yang mengimplementasikan ADT tersebut. Fungsi utama yang terdapat pada Linked List dengan ketentuan:

1. Input data pegawai
2. Melihat data pegawai
3. Menghapus data pegawai
4. Mengupdate data pegawai.
5. Menghapus seluruh data pegawai.