VERSION 1.1

SEPTEMBER 9, 2021

[STRUKTUR DATA]

MODUL 2, ARRAY LIST & LINKED LIST

DISUSUN OLEH: - LIDYA FANKKY OKTAVIA.P
- BELLA DWI MARDIANA

DIAUDIT OLEH: - IR. GITA INDAH M ST M.KOM
- DIDIH RIZKI CHANDRANEGARA, S.KOM., M.KOM

PRESENTED BY: TIM LAB-IT

UNIVERSITAS MUHAMMADIYAH MALANG

[STRUKTUR DATA]

PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi praktikum dengan baik, sesuai dengan materi yang diberikan oleh dosen pengajar dikelas. Terutama dalam penerapan materi OOP JAVA:

1. Class dan Object
2. Enskapsulasi
3. Array

TUJUAN

Mahasiswa mampu menguasai & menjelaskan konsep dari Struktur Data *Array List* & *Linked List.*

TARGET MODUL

Mahasiswa mampu memahami :

1. Collection tanpa menggunakan generics
2. Collection menggunakan generics
3. ArrayList
4. LinkedList

PERSIAPAN SOFTWARE/APLIKASI

1. Java Development Kit
2. Java Runtime Environment
3. IDE (Intellij IDE, Eclipse, Netbeans, dll)

MATERI POKOK

A. Collection

Collection adalah suatu obyek yang bisa digunakan untuk menyimpan sekumpulan obyek. Kelas Collection merupakan kelas generic, sehingga dapat digunakan untuk menampung objek yang memiliki tipe data berbeda.

a. Collection tanpa menggunakan generics

```
public class CellCollection {
  Cell[] cells;
  int index;
  public void add(Cell c) {
    Cells[index]=c;
    index++;
  }
  public Cell get(int i) {
    return cells[i];
  }
//more methods….
}
```

Kita juga memerlukan konstruktor untuk menginisialisasi ukuran array.

```
Public CellCollection(int size) {
          cells = new Cell[size];
}
```

b. Collection menggunakan generics

```
public class CellGenericCollection<T> {
  T[] cells;
  int index;
  public void add(T c) {
    cells[index]=c;
    index++;
  }
  public T get(int i) {
    return cells[i];
  }
}
```

B. ArrayList

ArrayList adalah sebuah class untuk array yang resizeable, class ini mengimplementasikan Interface List. ArrayList hanya mendukung tipe data object dan tidak mendukung tipe data primitif. Index dalam ArrayList dimulai dari 0 dan di akhiri dengan n-1 .

- **Interface List**
  List digunakan untuk menyimpan sekumpulan objek berdasarkan urutan masuk (ordered) dan menerima duplikat. Cara penyimpanannya seperti array, oleh sebab itu memiliki posisi awal dan posisi akhir, menyisipkan objek pada posisi tertentu, mengakses dan menghapus isi list, dimana semua proses ini selalu didasarkan pada urutannya. Class-class yang mengimplementasikan interface List adalah Vector, Stack, Linked List dan Array List.
- **Method untuk ArrayList**
  Beberapa method yang ada untuk penggunaan ArrayList

No.	Class Metthods	Description
1.	`boolean` **`add(E e)`**	Appends the specified element to the end of this list.
2.	`void` **`add(int index, E element)`**	Inserts the specified element at the specified position in this list.
3.	`boolean` **`addAll(Collection<? extends E> c)`**	Appends all of the elements in the specified collection to the end of this list, in the order that they.
4.	`boolean` **`addAll(int index, Collection<? extends E> c)`**	Inserts all of the elements in the specified collection into this list, starting at the specified position.
5.	`void` **`clear()`**	Removes all of the elements from this list.
6.	`object` **`clone()`**	Returns a shallow copy of this ArrayList instance.
7.	`boolean` **`contains(Object o)`**	Returns true if this list contains the specified element.
8.	`void` **`ensureCapacity(int minCapacity)`**	Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.
9.	`E` **`get(int index)`**	Returns the element at the specified position in this list.
10.	`int` **`indexOf(Object o)`**	Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
11.	`boolean` **`isEmpty()`**	Returns true if this list contains no elements.
12.	`int` **`lastIndexOf(Object o)`**	Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
13.	`E` **`remove(int index)`**	Removes the element at the specified position in this list.
14.	`boolean` **`remove(Object o)`**	Removes the first occurrence of the specified element from this list, if it is present.
15.	`protected void` **`removeRange(int fromIndex, int toIndex)`**	Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive.
16.	`E` **`set(int index, E element)`**	Replaces the element at the specified position in this list with the specified element.
17.	`int` **`size()`**	Returns the number of elements in this list.
18.	`Object[]` **`toArray()`**	Returns an array containing all of the

		elements in this list in proper sequence (from first to last element).
19.	`<T> T[] toArray(T[] a)`	Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array.
20.	`void trimToSize()`	Trims the capacity of this ArrayList instance to be the list's current size.

C. LinkedList

LinkedList dapat diilustrasikan seperti kereta api, dimana kereta api terdiri dari gerbong-gerbong yang saling terhubung yang dapat mengangkut penumpang (Data). Gerbong (Node/Simpul) disini berfungsi untuk menyimpan data. LinkedList merupakan struktur data yang setiap datanya memiliki pointer ke data berikutnya dan data sebelumnya.
Terdapat 3 jenis LinkedList : Single LinkedList , Double LinkedList , Circular LinkedList

- **Single LinkedList**
  Setiap node pada LinkedList mempunyai field yang berisi pointer ke node berikutnya dan juga memiliki field yang berisi data. Akhir LinkedList ditandai dengan node terakhir akan menunjuk ke null yang akan digunakan sebagai kondisi berhenti saat pembacaan LinkedList.
- **Double LinkedList**
  LinkedList dimana setiap node memiliki 3 field yaitu 1 field pointer yang menunjuk ke pointer berikutnya (next) , 1 field pointer yang menunjuk ke pointer sebelumnya (prev) dan field yang berisi data dari Node tersebut. Pointer next dan prev menunjuk ke null.
- **Circular LinkedList**
  Dalam Circular LinkedList ini masih dibagi lagi menjadi beberapa bagian diantaranya yaitu :
  - Single Circular LinkedList
    LinkedList yang pointer next-nya menunjuk ke diri nya sendiri, jika terdiri dari beberapa node maka pointer terakhir nya akan menunjuk ke pointer terdepan nya.
  - Double Circular LinkedList
    Double LinkedList yang pointer next nya dan prev nya menunjuk ke diri nya sendiri secara circular.

D. Implementasi penggunaan beberapa method dalam ArrayList

```
import java.util.ArrayList;

public class CthArrayList {

    public static void main(String args[]) {
        ArrayList<Integer> data = new ArrayList<>();
```

```
        data.add(1);
        data.add(2);
        data.add(3);
        data.add(4);
        data.add(5);


        System.out.println(data.get(0));
        System.out.println(data.get(1));
        System.out.println(data.get(2));
        System.out.println(data.get(3));
        System.out.println(data.get(4));

    }
}
```

E. Membuat LinkedList Secara Manual

1. Membuat kelas Node

```
public class Node {

    int data;
    Node next;
    Node prev;
    public Node (int data){
        this.data = data;
    }
}
```

2. Membuat Kelas Link

```
public class Link {
    Node head;

    public void add(int data){
        if (head == null){
            head = new Node(data);
        }

        Node current = head;
        while (current.next != null){
            current = current.next;
        }
        current.next = new Node(data);
    }
```

```
    public void showData(){
        if (head == null){
            System.out.println("Linklist is Empty");
            return;
        }
        Node current = head;
        while (current.next != null){
            current = current.next;
            int data = current.data;
            System.out.println(data);
        }
    }
}
```

3. Membuat kelas Main

```
public class LinkedList {
    public static void main(String[] args) {
        Link myLink = new Link();
        myLink.add(6);
        myLink.add(8);
        myLink.add(4);
        myLink.add(5);
        myLink.showData();
    }
}
```

LATIHAN PRAKTIKUM

LATIHAN 1

Penggunaan interface List pada kelas LinkedList :

Buatlah dua objek List (ArrayList) yaitu objek Hewan dan DeleteHewan. Objek ini berisi hewan-hewan, buatlah sebagian ada yang sama. Lakukan penghapusan data yang terdapat pada objek Hewan yang sama dengan data warna yang terdapat pada objek DeleteHewan, selanjutnya tampilkan.

```
Hewan :
[Sapi, Kelinci, Kambing, Unta, Domba]
Hewan yang dihapus :
[Kelinci, Kambing, Unta]

Output
Hewan :
[Sapi, Domba]
```

TUGAS

KEGIATAN 1

Buatlah sebuah ArrayList of String dengan nama-nama Hewan, Lalu isilah ArrayList tersebut dengan ketentuan sebagai berikut ini :

Index 0 = Angsa

Index 1 = Bebek

Index 2 = Cicak

Index 3 = Domba

Index 4 = Elang

Index 5 = Gajah

1. Buatlah object kosong untuk menambahkan ketentuan diatas yang bertipe data String. Boleh menggunakan Library.
2. Tambahkan elemen lagi “Badak” dan ”Bebek”, hitung jumlah elemen “Bebek” dan tampilkan posisi index dari elemen “Bebek” pada object kosong yang telah dibuat.

```
Output :

[Angsa, Bebek, Cicak, Domba, Elang, Gajah, Badak, Bebek]
Bebek = 2
Posisi Index : 1, 4
```

3. Hapus posisi “Bebek” yang pertama.

```
Output :

[Angsa, Cicak, Domba, Elang, Gajah, Badak, Bebek]
```

4. Tampilkan elemen pada index ke-0 dan ke-2, selanjutnya hapus index ke-0.

```
Output :

Index ke-0 : Angsa
Index ke-2 : Domba

[Cicak, Domba, Elang, Gajah, Badak, Bebek]
```

5. Ubahlah index ke-0 dari “Cicak” Menjadi “Ular”, selanjutnya tambahkan elemen baru pada index ke-2 dengan “Itik”.

```
Output :

[Ular, Domba, Elang, Gajah, Badak, Bebek]
[Ular, Domba, Itik, Elang, Gajah, Badak, Bebek]
```

6. Hapus elemen diantara index ke-1 dan ke-5.

```
Output :

[Ular, Domba, Badak, Bebek]
```

7. Tampilkan elemen pertama dan terakhir.

```
Output :

Elemen Pertama : Ular
Elemen Terakhir : Bebek
```

8. Tampilkan jumlah elemen pada ArrayList.
9. Carilah posisi index dari “Badak”.
10. Sampaikan pemahamanmu mengenai ArrayList kepada asisten masing-masing

KEGIATAN 2

Buatlah sebuah program LinkedList secara manual **tidak diperkenankan menggunakan library** dengan ketentuan dibawah ini :

Ketika user menambahkan Node baru bernilai *n* pada LinkedList, maka program akan memeriksa apakah nilai dari *n* tersebut mempunyai nilai lebih besar daripada data yang lain. Sehingga program LinkedList yang dibuat dapat mengurutkan node dari yang terkecil hingga terbesar.

Contoh :

```
LinkedList list = new LinkedList()
list.add(8);
list.add(7);
list.add(1);
list.add(4);
list.add(6);
list.add(2);
list.add(3);
list.printList();
```

Output :

CATATAN

Kegiatan 1 : Gunakan Library untuk mengerjakan

Kegiatan 2 : Buat Linkedlist secara manual tidak diperkenankan menggunakan library

Aturan umum penulisan bahasa JAVA agar mudah di koreksi oleh asisten:

1. Untuk nama kelas,interface,enum, dan yang lainnya biasakanmenggunakangaya CamelCase (diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: Kursi , JalanRaya, ParkiranGedung, dan lain seterusnya.
2. Untuk penulisan nama method, dan attribute diawali dengan huruf kecil di awal kata dan menggunakan huruf besar untuk kata setelahnya, seperti: getNamaJalan, namaJalan, harga, setNamaJalan, dan lain seterusnya.
3. Jika menggunakan IDE Intellij jangan lupa untuk memformat penulisan kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok. Silahkan dikerjakan tanpa **copy – paste**.

DETAIL PENILAIAN TUGAS

Soal	Nilai
Kegiatan 1	50
Kegiatan 2	40
Pemahaman Materi Keseluruhan	10