

MODUL PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK



Disusun oleh :

Ultach Enri, S.Kom., M.Kom.

Yuyun Umaidah, S.Kom., M.Kom.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2018

DAFTAR ISI

Pertemuan 1 : Class, Object, Variable dan Method	3
Pertemuan 2 : Tipe Data, Parameter, Konstruktor dan Interface	6
Pertemuan 3 : Packages, Encapsulation dan Inheritance	11
Pertemuan 4 : Polymorphism.....	15
Pertemuan 5 : Input dan Branching.....	18
Pertemuan 6 : Looping.....	23
Pertemuan 7 : Array	26
Pertemuan 8 : GUI	30
Pertemuan 9 : ArrayList.....	36

Class, Object, Variable dan Method

1.1. Pendahuluan

- a) Class adalah cetak biru(*blue print*) dari sesuatu.

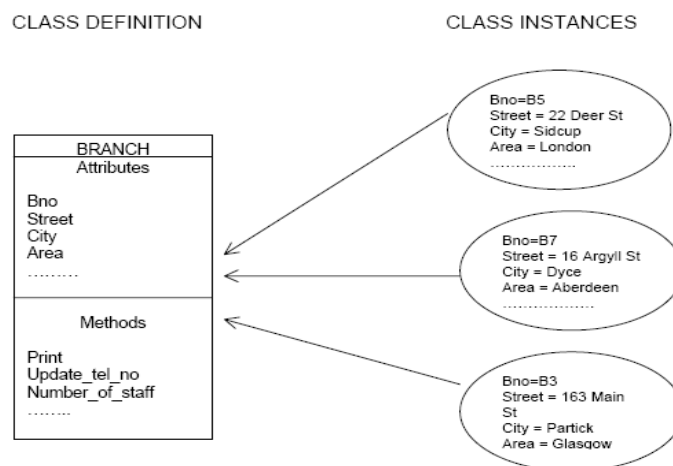
- Class adalah rancangan atau konsep atau ide.

Contoh : Class mobil

- b) Object adalah instance dari class, bentuk (contoh) nyata dari class, object memiliki sifat independen dan dapat digunakan untuk mengakses attribute dan juga method.

- Wujud, kejadian, kenyataan dari rancangan atau konsep ataupun ide.

Contoh : terdapat 3 mobil masing memiliki warna, tahun produksi atau pun nomor polisi yang berbeda beda.



1.2. Variable

- a) Variable digunakan untuk menyimpan nilai yang nantinya akan digunakan pada program

- b) Nilai variabel dapat diubah di pernyataan manapun di dalam program

- c) Cara mendeklarasikan Variable :

tipe_data nama_variabel

contoh :

String kota;

int nilaiAbsen, nilaiTugas;

Variabel dipanggil berdasarkan lingkungannya, dimulai dari blok yang paling kecil, kemudian blok di luar itu :

1. Local Variable: digunakan di dalam method atau blok pernyataan yang lebih kecil dari itu
2. Parameter: variabel yg ada di dalam pernyataan (argument) method
3. Instance Variable: variabel yang memiliki nilai yang berbeda di setiap objek
4. Class Variable: variabel yang berlaku di suatu class dan seluruh instan dari class tersebut (objek). Ciri class variable adalah menggunakan keyword static

1.3. Method

- a) Method adalah urutan instruksi yang mengakses data dari object
- b) Method melakukan:
 - Manipulasi data
 - Perhitungan matematika
 - Memonitor kejadian dari suatu event

1.4. Contoh Program

Contoh 1 :

```
public class Halo{
    public static void main(String[] args){
        System.out.println("Halo Karawang");
    }
}
```

Contoh 2 :

```
public class Sepeda {
    String warna;
    int tahunProduksi;
}
```

Contoh 3 :

```
public class SepedaAksi{
    public static void main(String[] args){
        // Membuat object
        Sepeda sepedaku = new Sepeda();
        /* memanggil atribut dan memberi nilai */

        sepedaku.warna = "Hitam";
        sepedaku.tahunProduksi = 2006;
        System.out.println("Kring-kring Sepedaku...");
        System.out.println("Sepedaku berwarna : " + sepedaku.warna);
        System.out.println("Diproduksi tahun    : " + sepedaku.tahunProduksi);
    }
}
```

Contoh 4 :

```
public class Matematika {
    int a, b;
    int tambah(){
        int hasil = a+b;
        return hasil;
    }
    void info(){
        System.out.println("Penambahan "+a+" + "+b+" = "+tambah());
    }
}
```

Contoh 5 :

```
public class DemoMatematika {  
    public static void main(String[] args) {  
        Matematika mtk = new Matematika();  
  
        mtk.a = 10;  
        mtk.b = 2;  
        mtk.info();  
    }  
}
```

Latihan :

1. Buat class Mahasiswa yang berisi tiga method:

```
membaca()  
nyontek()  
modifikasi()
```

Isi masing-masing method dengan tampilan status menggunakan System.out.println()

Buat class MahasiswaBeraksi, dan panggil method-method diatas dalam class tersebut

2. Buat class Nilai yang berisi method :

```
Nilai()  
CetakNilai()
```

Buat class DemoNilai dan isi variabel serta panggil method-method dalam class Nilai tersebut dengan hasil tampilan sbb :

NIM	: xxxxxxxx
Nama	: xxxxxxxx
Nilai Absen [10%]	: 0000000
Nilai Tugas [20%]	: 0000000
Nilai UTS [30%]	: 0000000
Nilai UAS [40%]	: 0000000
Nilai Akhir	: 0000000

Tipe Data, Parameter, Konstruktor dan Interface

2.1 Tipe Data

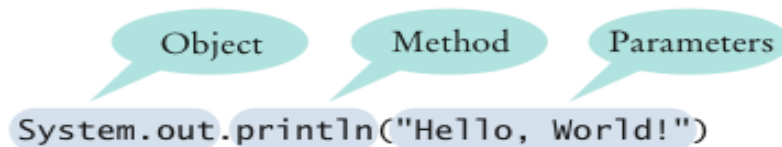
1. Tipe Data Primitif

- Tipe data yang merupakan kata kunci di Java (tertanam di compiler Java), sehingga pemrosesan jauh lebih cepat
- Menggunakan huruf kecil (lowercase)
- Contoh: int, double

2. Tipe Data Reference (Class)

- Tipe data berupa class yang ada di library Java (java.lang)
- Menggunakan huruf capital
- Contoh: String, Integer, Double

2.2 Parameter



- Sepeda akan berguna apabila ada object lain yang berinteraksi dengan sepeda tersebut
- Object software berinteraksi dan berkomunikasi dengan object lain dengan cara mengirimkan message atau pesan
- Pesan adalah suatu method, dan informasi dalam pesan dikenal dengan nama parameter

Syntax *accessSpecifier returnType methodName(parameterType parameterName, . . .)*

```
{
    method body
}
```

Example

These methods are part of the public interface.

```

public void deposit(double amount)
{
    balance = balance + amount;
}

public double getBalance()
{
    return balance;
}
                
```

This method does not return a value.

A mutator method modifies an instance variable.

This method has no parameters.

An accessor method returns a value.

Contoh 1 :

```

public class Sepeda{
    int gir;

    // method (mutator) dengan parameter
    void setGir(int pertambahanGir) {
        gir= gir+ pertambahanGir;
    }
    // method (accessor)
    int getGir() {
        return gir;
    }
}

```

Contoh 2 :

```

public class SepedaBeraksi{
    public static void main(String[] args) {

        Sepeda sepedaku = new Sepeda();
        sepedaku.setGir(1); // menset nilai gir = 1 (sebelumnya 0)
        System.out.println("Gir saat ini: " + sepedaku.getGir());
    }
}

```

2.3 Konstruktor

- Method yang digunakan untuk memberi nilai awal pada saat object diciptakan
- Dipanggil secara otomatis ketika new digunakan untuk membuat instan class
- Sifat konstruktor:
 - Nama konstruktor sama dengan nama class
 - Tidak memiliki nilai balik dan tidak boleh ada kata kunci void
- Kata kunci this digunakan pada pembuatan class dan digunakan untuk menyatakan object sekarang

Contoh 3 :

```

class PersegiPanjang {
    int panjang, lebar;

    public PersegiPanjang(int panjang, int lebar){
        this.panjang = panjang;
        this.lebar = lebar;
    }
    public void info(){
        System.out.println("Panjang Persegi Panjang : " + panjang);
        System.out.println("Lebar Persegi Panjang : " + lebar);
    }
}

```

Contoh 4 :

```
public class PersegiPanjangKonstruktor{
    public static void main(String[] args){
        Persegi Panjang pp = new PersegiPanjang(2, 10);
        pp.info();
    }
}
```

2.4 Interface

- Interface digunakan apabila kita ingin menentukan apa yang harus dilakukan oleh suatu class tapi tidak menentukan bagaimana cara untuk melakukannya
- Interface sebenarnya sama dengan class, tapi hanya memiliki deklarasi method tanpa implementasi

Syntax

```
public interface InterfaceName
{
    method signatures
}
```

Example

```
public interface Measurable
{
    double getMeasure();
}
```

The methods of an interface are automatically public.

No implementation is provided.

Syntax

```
public class ClassName implements InterfaceName, InterfaceName, . . .
{
    instance variables
    methods
}
```

Example

```
public class BankAccount implements Measurable
{
    . . .
    public double getMeasure()
    {
        return balance;
    }
    . . .
}
```

BankAccount instance variables

Other BankAccount methods

List all interface types that this class implements.

This method provides the implementation for the method declared in the interface.

Contoh 5 :

```
interface InterfaceLampu{
    public static final int KEADAAN_HIDUP=1;
    public static final int KEADAAN_MATI=0;
    public abstract void hidupkan();
    public abstract void matikan();
}
```

Contoh 6 :

```
public class Lampu implements InterfaceLampu{
    int statusLampu;
    public void hidupkan(){
        if (statusLampu == KEADAAN_MATI){
```



```

        statusLampu = KEADAAN_HIDUP;
        System.out.println("Hidupkan Lampu! --> Lampu Hidup");
    }else{
        System.out.println("Hidupkan Lampu! --> Lampu Sudah Hidup Kok");
    }
}

public void matikan(){
    if (statusLampu == KEADAAN_HIDUP){
        statusLampu = KEADAAN_MATI;
        System.out.println("Matikan Lampu! --> Lampu Mati");
    }else{
        System.out.println("Matikan Lampu! --> Lampu Sudah Mati Kok");
    }
}
}

```

Contoh 7 :

```

public class LampuBeraksi{
    public static void main(String[] args){
        Lampu lampuKamar = new Lampu();

        System.out.println("Status Lampu Saat Ini: Mati");

        lampuKamar.hidupkan(); //Hidupkan Lampu
        lampuKamar.matikan(); //Matikan Lampu
        lampuKamar.matikan(); //Matikan Lampu
        lampuKamar.hidupkan(); //Hidupkan Lampu
        lampuKamar.hidupkan(); //Hidupkan Lampu
    }
}

```

Latihan

1. Buat Class bernama Matematika, yang berisi method dengan dua parameter:

- pertambahan(int a, int b)
- pengurangan(int a, int b)
- perkalian(int a, int b)
- pembagian(int a, int b)

Buat Class bernama MatematikaBeraksi, yang mengeksekusi method dan menampilkan:

- Pertambahan : $20 + 10 = 30$
- Pengurangan : $10 - 5 = 5$
- Perkalian : $10 * 3 = 30$
- Pembagian : $21 / 2 = 10$

2. Terapkan interface untuk soal no 1 diatas.
3. Terapkan konstruktor untuk tampilan dibawah ini :

There was a farmer who had a dog,
And Bingo was his name-o.
B-I-N-G-O
B-I-N-G-O
B-I-N-G-O
And Bingo was his name-o.

There was a farmer who had a dog,
And Bingo was his name-o.
(clap)-I-N-G-O
(clap)-I-N-G-O
(clap)-I-N-G-O
And Bingo was his name-o.

There was a farmer who had a dog,
And Bingo was his name-o.
(clap)-(clap)-N-G-O
(clap)-(clap)-N-G-O
(clap)-(clap)-N-G-O
And Bingo was his name-o.

There was a farmer who had a dog,
And Bingo was his name-o.
(clap)-(clap)-(clap)-G-O
(clap)-(clap)-(clap)-G-O
(clap)-(clap)-(clap)-G-O
And Bingo was his name-o.

There was a farmer who had a dog,
And Bingo was his name-o.
(clap)-(clap)-(clap)-(clap)-O
(clap)-(clap)-(clap)-(clap)-O
(clap)-(clap)-(clap)-(clap)-O
And Bingo was his name-o.

There was a farmer who had a dog,
And Bingo was his name-o.
(clap)-(clap)-(clap)-(clap)-(clap)
(clap)-(clap)-(clap)-(clap)-(clap)
(clap)-(clap)-(clap)-(clap)-(clap)
And Bingo was his name-o.

PACKAGES, ENCAPSULATION DAN INHERITANCE

3.1 Packages

- Package adalah koleksi dari beberapa class dan interface yang berhubungan, dan menyediakan proteksi akses dan pengelolaan namespace
- 1 package adalah 1 folder di file system
- Package berguna untuk mengorganisir file dalam suatu project atau library
- Nama package menggunakan lowercase
- Nama package mengikuti nama domain (perusahaan) dengan susunan terbalik
 - Contoh: id.ac.unsika
- Keyword: *package name*;

Syntax **package** *packageName*;

Example

The classes in this file
belong to this package.

package com.horstmann.bigjava;

A good choice for a package name
is a domain name in reverse.

- **import** : digunakan untuk memanggil pustaka fungsi yang tersedia
 - Contoh import java.io.*;

Contoh 1 :

```
package kelasku;
    public class Dilan{
        public void info(){
            System.out.println("Hi namaku Dilan");
        }
    }
```

Contoh 2 :

```
package kelasku;
    public class Milea{
        public void info(){
            System.out.println("Hi saya Milea");
        }
    }
```

Contoh 3 :

```
package dilan1990;

import kelasku.Dilan;
```

```

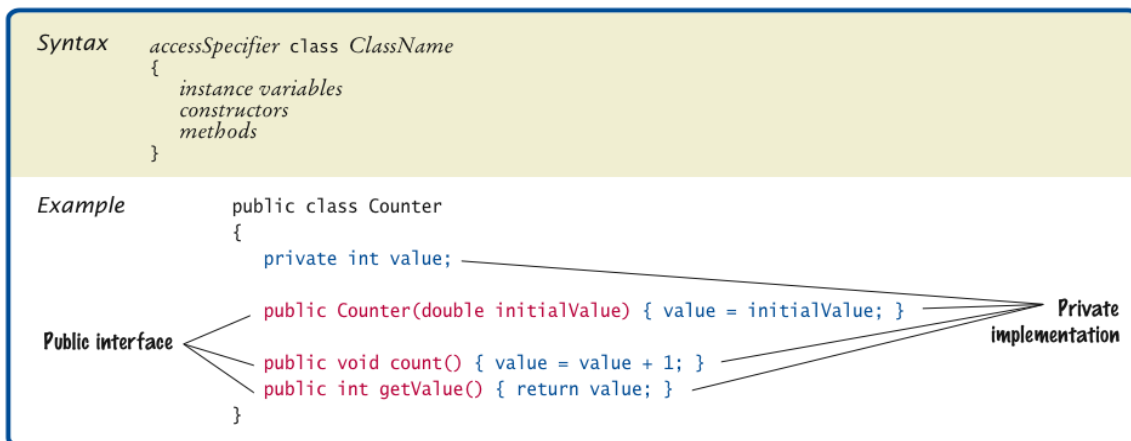
public class PaketBeraksi{
    public static void main(String[] args){

        Dilan dln = new Dilan();
        dln.info();
    }
}

```

3.2 Encapsulation

- Mekanisme menyembunyikan suatu proses dan data dalam sistem untuk menghindari interferensi, dan menyederhanakan penggunaan proses itu sendiri
- Class access level (public, protected, privat) adalah implementasi dari konsep encapsulation
- Enkapsulasi data dapat dilakukan dengan cara:
 - mendeklarasikan instance variable sebagai private
 - mendeklarasikan method yang sifatnya public untuk mengakses variable tersebut



- **Access Modifier**
 - **Default**
Menyatakan bahwa class / interface/ method / constructor/attribute/inner class tersebut dapat diakses/diimplementasikan oleh class lain yang berada dalam satu package(class turunannya / implentasinya harus berada dalam satu package)
 - **Private**
Menyatakan bahwa method / constructor / attribute / inner class tersebut tidak dapat di akses sama sekali oleh class lain bahkan juga tidak dapat di turunkan.
Suatu attribute dideklarasikan private :
 1. Bila class lain tidak memerlukan attribute tersebut
 2. Melindungi suatu attribute dari kemungkinan nilai nya di ubah oleh method lain dari class lain
 - **Public**
Menyatakan bahwa class / interface/ method / constructor/attribute/inner class tersebut dapat diakses/diimplementasikan oleh class lain dimanapun

- *Protected*
Menyatakan bahwa method / constructor / attribute/ inner class tersebut dapat diakses/diimplementasikan oleh class lain yang berada dalam satu package atau turunan nya

Contoh 4 :

```
public class Sepeda{
    // access modifier private pada instance variable
    private int gir;

    void setGir(int penambahanGir) {
        gir= gir+ penambahanGir;
    }

    int getGir() {
        return gir;
    }
}
```

Contoh 5 :

```
public class DemoSepeda{
    public static void main(String[] args) {
        Sepeda sepedaku = new Sepeda();

        sepedaku.setGir(1);

        System.out.println("Gir saat ini: " + sepedaku.getGir());
    }
}
```

3.3 Inheritance

- Suatu class dapat mewariskan atribut dan method kepada class lain (subclass), serta membentuk class hierarchy
- Contoh di dunia nyata : Anak-anak memiliki perilaku dan atribut yang mirip dengan orangtuanya
- Penting untuk Reusability (penggunaan ulang kode program, tidak perlu menuliskan kode yang sama berulang-ulang)
- Java Keyword: *extends*

Contoh 6 :

```
public class Sepeda{
    private int gir;

    void setGir(int penambahanGir) {
        gir= gir+ penambahanGir;
    }
}
```

```

        int getGir() {
            return gir;
        }
    }

```

Contoh 7 :

```

public class SepedaGunung extends Sepeda{

    private int sadel;
    void setSadel (int jumlah) {
        sadel = getGir() - jumlah;
    }

    int getSadel(){
        return sadel;
    }
}

```

Contoh 8 :

```

public class SepedaGunungBeraksi {
    public static void main(String[] args) {
        SepedaGunung sg=new SepedaGunung();
        sg.setGir(3);
        System.out.println(sg.getGir());
        sg.setSadel(1);
        System.out.println(sg.getSadel());
    }
}

```

Latihan

1. Buat class Matematika2 yang merupakan inherit dari class Matematika (soal latihan pertemuan 2), tambahkan method modulus(int a, int b) yang menghitung modulus dari a dan b
Buat class MatematikaInheritance yang memanggil semua method dari class Matematika dan Matematika2
2. Buat class KonversiSuhu() yang mempunyai method celciusToFahrenheit() dan celciusToReamur()
Buat class KonversiSuhu2() yang inherit dari class KonversiSuhu() dan mempunyai method fahrenheitToReamur()
Buat class DemoKonversiSuhu() yang memberikan nilai dan memanggil semua method dari class KonversiSuhu() dan class KonversiSuhu1()

POLYMORPHISM

- Kemampuan untuk memperlakukan object yang memiliki perilaku (bentuk) yang berbeda
- Implementasi konsep polymorphism:
 1. Overloading: Kemampuan untuk menggunakan nama yang sama untuk beberapa method yang berbeda parameter (tipe dan atau jumlah)
 2. Overriding: Kemampuan subclass untuk menimpa method dari superclass, yaitu dengan cara menggunakan nama dan parameter yang sama pada method

4.1 Overloading

Contoh 1 :

```
public class Circle {
    double x, y, r;

    public Circle(double x, double y, double r) {
        this.x = x;
        this.y = y;
        this.r = r;
    }
    public Circle(double x, double y) {
        this.x = x;
        this.y = y;
        r = 1;
    }
    public double keliling(){
        return 2*3.14 *r;
    }
    public double luas(){
        return 3.14 * r * r;
    }
}
```

Contoh 2 :

```
public class DemoCircle {
    public static void main(String[] args) {
        Circle cr = new Circle(2, 3, 5);
        Circle cr1 = new Circle (2, 3);

        System.out.println("Lingkaran Konstruktor 3 Parameter");
        System.out.println("Keliling Lingkaran : "+cr.keliling());
    }
}
```

```

        System.out.println("Luas Lingkaran    : "+cr.luas());

        System.out.println("Lingkaran Konstruktor 2 Parameter");
        System.out.println("Keliling Lingkaran : "+cr1.keliling());
        System.out.println("Luas Lingkaran    : "+cr1.luas());
    }
}

```

4.1 Overriding

Contoh 3 :

```

public class Employee{
    private String name;
    private double salary;

    protected Employee(String name, double salary){
        this.name = name;
        this.salary = salary;
    }
    protected String getDetails(){
        return "Name : "+name+ "\nSalary : "+salary;
    }
    public void cetak(){
        System.out.println("Percobaan di Class Employee");
    }
}

```

Contoh 4 :

```

public class Manager extends Employee{
    private String dept;

    public Manager (String name, double salary, String dept){
        super (nama, salary);
        this.dept = dept;
    }
    public String getDept(){
        return dept;
    }
    public String getDetails(){
        return super.getDetails()+"\nDepartment : "+getDept();
    }
    public void cetak(){
        System.out.println(" Percobaan di Class Manager");
    }
}

```


Contoh 5 :

```
public class view{  
    public static void main(String[] args){  
        Employee em = new Employee("Dilan", 4000000);  
        Manager mn = new Manager("Milea", 5000000, "Marketing");  
  
        System.out.println("Data Employee : \n"+em.getDetails());  
        em.cetak();  
        System.out.println("\nData Manager : \n"+mn.getDetails());  
        mn.cetak();  
    }  
}
```

Latihan

Buat Program untuk menghitung volume dan luas permukaan dari balok dan kubus dengan menggunakan konsep polimorfisme!

INPUT DAN BRANCHING

5.1 Input

- Merupakan proses menerima masukan dari keyboard dan menyimpannya kedalam suatu variable dan kemudian menampilkannya ke layar.
- Stream standar yang biasa digunakan antara lain :
 - `System.in` : menangani pembacaan dari keyboard (standard input)
 - `System.out` : mengirimkan keluaran ke layar (standard output)
 - `System.err` : mengirimkan kesalahan (standard error)
- Untuk membaca input dari keyboard, menggunakan class `Scanner` (`java.util.Scanner`) dengan method sebagai berikut:
 - `nextInt()` : untuk menerima tipe data integer
 - `nextShort()` : untuk menerima tipe data short
 - `nextLong()` : untuk menerima tipe data long
 - `nextDouble()` : untuk menerima tipe data double
 - `nextFloat()` : untuk menerima tipe data float
 - `nextLine()` : untuk menerima tipe data string
 - `nextBoolean()` : untuk menerima tipe data boolean

Contoh1 :

```
import java.util.Scanner;
public class SalamKenal {
    public static void main( String[] args ){
        Scanner masukan = new Scanner(System.in);
        System.out.print("Masukkan Nama Anda: ");
        String nama = masukan.nextLine();
        System.out.println("Halo, Salam Kenal sdr " + nama + "!");
    }
}
```

Contoh2 :

```
import java.util.Scanner;
public class Perkalian{
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan bilangan pertama: ");
        int bilangan1 = input.nextInt();

        System.out.print("Masukkan bilangan kedua: ");
        int bilangan2 = input.nextInt();
        System.out.print("Hasil perkalian: " +
            (bilangan1 * bilangan2));
    }
}
```

5.2 Brancing

- Statemen kondisional dalam Java memiliki 4 macam syntax :
 - **If (kondisi) statemen ;**
 - **If (kondisi) {blok statemen} ;**
 - **If (kondisi) statemen1 else statemen2;**
 - **If (kondisi) {blok statemen -1} else {blok statemen-2} ;**
- Blok Statemen : kumpulan statemen yang berada dalam blok { }
- Jenis Struktur Seleksi
 - **Struktur seleksi sederhana (if)**
Bentuk ini merupakan bentuk yang paling sederhana dari keseluruhan struktur seleksi yang ada. Pada bentuk ini, jika memiliki nilai true saja yang akan diproses

Perintah :

```
If (Variabel=NilaiVariabel)
{
    Blok statement kondisi yang benar
}
```

Contoh3 :

```
public class PernyataanIF{
    public static void main(String[] args){
        int diskon =0, totalBelanja = 500000;
        if(totalBelanja>= 100000){
            diskon = totalBelanja/10;
        }
        System.out.println("Diskon = " + diskon);
    }
}
```

- **Struktur seleksi tersarang (nested if)**
Bentuk ini, baik kondisi bernilai true ataupun false diikuti oleh proses khusus. Tetapi yang harus diperhatikan adalah bahwa proses khusus pada keadaan true tidak mungkin akan diproses pada keadaan false dan sebaliknya.

Perintah dua Kondisi :

```
If (Variabel=NilaiVariabel)
{
    Blok statement kondisi yang benar
}
Else
{
    Blok statement kondisi yang salah
}
```

Contoh4 :

```
public class PernyataanIFELSE{
    public static void main(String[] args){
        int diskon =0, totalBelanja = 500000;
```

```

        if(totalBelanja >= 100000){
            diskon = totalBelanja/10;
        } else{
            diskon = 0;
        }
        System.out.println("Diskon = " + diskon);
    }
}

```

Perintah tiga kondisi atau lebih:

```

If (Variabel-1=Nilai Variabel-1)
{
    Blok statement-1 yang benar
}
Else
    If (Variabel-2=Nilai Variabel-2)
    {
        Blok statement-2 yang salah
    }
    Else
    {
        Blok statement-3 kondisi salah
    }
}

```

Contoh 5 :

```

public class PernyataanIFELSEIF{
    public static void main(String[] args) {
        int skorUjian= 86; char nilai;
        if (skorUjian >= 90) {
            nilai = 'A';
        } else if (skorUjian >= 80) {
            nilai = 'B';
        } else if (skorUjian >= 70) {
            nilai = 'C';
        } else {
            nilai = 'D';
        }
        System.out.println("Nilai = " + nilai);
    }
}

```

▪ **Struktur Switch - case**

Statement Switch merupakan statement alternatif untuk melakukan pemilihan statement. Statement ini biasanya digunakan untuk menyederhanakan kompleksitas statement IF yang banyak mengandung kondisi.

Tetapi, switch disini masih memiliki batasan yaitu: Variabel penentu harus bertipe Integer, atau yang cocok dengan Integer seperti character, boolean, byte, short dan long. Batasan variabel yang bisa diperiksa antara 0 sampai 255.

Perintah :

Switch (Variabel)

```
{
case nilai1 :
Blok statement 1 yang benar ;
break ;
case nilai2 :
Blok statement 2 yang benar ;
break ;
case nilai3 :
Blok statement 3 yang benar ;
break ;
default :
blok statement 4 yang salah ;
}
```

Contoh6 :

```
public class PernyataanSWITCH1{
    public static void main(String[] args){
        int pilihan = 3;
        switch(pilihan){
            case 1:
                System.out.println("Soto Ayam");
                break;
            case 2:
                System.out.println("Gule Kambing");
                break;
            case 3:
                System.out.println("Nasi Goreng");
                break;
            default:
                System.out.println("Silakan Pilih 1, 2 atau 3");
        }
    }
}
```

Latihan

1. Buatlah program yang menghasilkan tampilan Input dan Output sebagai berikut :

Tampilan Input

NPM : <input>
 Nama Mahasiswa : <input>
 Nilai Kehadiran : <input>
 Nilai Tugas : <input>
 Nilai UTS : <input>
 Nilai UAS : <input>

Tampilan Output

NPM Mahasiswa : <tampil otomatis>
 Nama Mahasiswa : <tampil otomatis>
 Nilai Rata-rata : <tampil otomatis>
 Grade : <tampil otomatis>
 Keterangan : <tampil otomatis>

Ketentuan Soal

Nilai akhir : (10% x Nilai Absen) + (20% x Nilai Tugas) + (30% x Nilai Tugas) + (40% x Nilai UAS)

Nilai akhir	grade	keterangan
0 – 45	E	KURANG SEKALI
46 – 55	D	KURANG
56 – 65	C	CUKUP
66 – 75	B	BAIK
76 – 100	A	ISTIMEWA

2. Buat program untuk menentukan besarnya potongan dari pembelian barang yang diberikan seorang pembeli, dengan kriteria :
 - a. jika total pembelian kurang dari Rp. 50.000,- potongan yang diterima sebesar 5% dari total pembelian.
 - b. Jika total pembelian lebih dari atau sama dengan Rp. 50.000,- potongan yang diterima sebesar 20% dari total pembelian.

output

Total pembelian Rp. = 50000 (input)

Besarnya potongan Rp. 10000 (otomatis)

Jumlah yang harus dibayarkan Rp. 40000 (otomatis)

3. Buat program untuk menentukan kriteria kegemukan dengan menggunakan Indeks Massa Tubuh (IMT), yang dihitung berdasarkan rumus :

$$IMT = \frac{\text{Berat Badan (kg)}}{\text{Tinggi Badan (m)} \times \text{Tinggi Badan (m)}}$$

Nilai IMT	Kriteria
18,4 kebawah	Barat Badan Kurang
18,5 – 24,9	Berat Badan Ideal
25 – 29,9	Berat Badan Lebih
30 – 39,9	Gemuk
40 Keatas	Sangat Gemuk

LOOPING

Perulangan(looping) adalah suatu proses didalam program yang dapat mengeksekusi beberapa statement yang sama dengan berulang-ulang sampai ada kondisi untuk berhenti.

Terdapat beberapa jenis perulangan pada Java, yaitu :

- for
- while
- do-while

6.1 For

- Struktur for pada umumnya digunakan untuk pengulangan yang banyaknya sudah diketahui sebelumnya. Proses pengulangan akan terus berjalan selama kondisi bernilai true, dan jika bernilai false maka proses pengulangan akan dihentikan secara otomatis.

Perintah :

```
for (inisialisasi; kondisi; kenaikan_penurunan){  
    pernyataan  
}
```

Contoh 1 :

```
public class PernyataanFOR{  
    public static void main(String[] args){  
        int i;  
        for(i=1; i<=10; i++){  
            System.out.println("Hasil Fungsi FOR ke - "+i);  
        }  
    }  
}
```

Contoh 2 :

```
public class PernyataanFORArray{  
    public static void main(String[] args){  
        int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
        for (inti : numbers) {  
            System.out.println(i);  
        }  
    }  
}
```

Contoh 3 :

```

public class Segitiga {
    private int lebar;
    public Segitiga(int lebar){
        this.lebar = lebar;
    }
    public String gambarSegitiga(){
        String r = "";
        for (int i = 1; i <= lebar; i++){
            for (int j = 1; j <= i; j++){
                r = r + "[";
                r = r + "\n";
            }
            return r;
        }
    }
}

```

Contoh 4 :

```

public class SegitigaBeraksi {
    public static void main(String[] args) {
        Segitiga kecil = new Segitiga(5);
        System.out.println(kecil.gambarSegitiga());
        Segitiga besar = new Segitiga(15);
        System.out.println(besar.gambarSegitiga());
    }
}

```

6.2 While

Pada while pengulangan dikondisikan di awal blok, jadi apabila kondisi tidak terpenuhi (bernilai false) maka proses pengulangan tidak akan pernah dilakukan atau tidak berjalan.

Perintah :

```

while (kondisi) {
    pernyataan
}

```

Contoh 5 :

```

class pernyataanWhile{
    public static void main(String[] args){
        int i=1;
        while (i <= 5){
            System.out.println(i);
            i++;
        }
    }
}

```


6.3Do - While

Struktur do-while sebenarnya tidak beda jauh dengan while. Perbedaannya hanyalah terletak pada penempatan kondisinya saja. Pada while kondisi diletakan di awal blok pengulangan sedangkan pada do-while kondisinya berada di akhir blok. Jadi pada proses pengulangan do-while akan dilakukan minimal sekali meskipun ternyata kondisinya tidak terpenuhi atau bernilai false.

Perintah :

```
do {  
    pernyataan  
} while (kondisi);
```

Contoh 6 :

```
class pernyataanDoWhile{  
    public static void main(String[] args){  
        int jumlah = 1;  
        do{  
            System.out.print(jumlah);  
            jumlah++;  
        }  
        while(jumlah >=10);  
    }  
}
```

Latihan

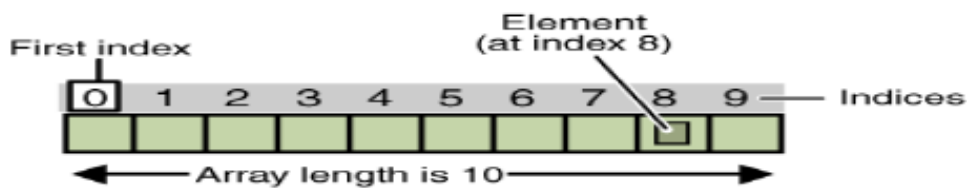
1. Buatlah program dengan menggunakan semua bentuk perulangan untuk :
 - a. Menghitung Deret bilangan prima dan bukan dari 0 – 20 dengan Hasilnya
 - b. Menghitung Deret bilangan ganjil dan genap dari 0 – 20 dengan Hasilnya
 - c. Huruf Z – A
 - d. Lagu “Anak Ayam Turun N”
2. Buatlah suatu program yang menghasilkan suatu tabel perkalian $n \times n$ (dimana nilai n lebih kecil atau sama dengan 10). Sebagai contoh, jika n bernilai 4 (empat), maka akan ditampilkan tabel perkalian berikut :

	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

ARRAY

7.1 Array

- Array adalah objek yang dapat digunakan untuk menyimpan sejumlah data dalam tipe sama dengan jumlah tetap



- Elemen yang disimpan pada array dapat berupa tipe primitif atau objek

7.2 Mendeklarasikan Array

- Langkah menciptakan array :
 - Mendeklarasikan array
 - Contoh : `String[] namaMahasiswa;`
`int[] nilaiUTS;`
`int[][] twoD;`
`char[][][] threeD;`
 - Menciptakan objek array
 - Contoh : `namaMahasiswa = new String[40];`
`nilaiUTS = new int[40];`
`twoD = new int[512][128];`
`threeD = new char[8][16][24];`
- Bentuk singkat deklarasi variable dan objek array :
 - `String[] namaMahasiswa = new String[40];`
 - `int[] nilaiUTS = new int[40];`
 - `int[][] twoD = new int[512][128];`
 - `char[][][] threeD = new char[8][16][24];`

7.3 Array 1 Dimensi

Contoh 1 :

```
public class public class ArrayKota{
    public static void main(String[] args){
        //deklarasi variabel array
        String[] kota;

        // membuat objek array
        kota = new String[3];

        // mengisi elemen array
        kota[0] = "Jakarta";
        kota[1] = "Surabaya";
        kota[2] = "Semarang";

        // menampilkan elemen array
        System.out.println(kota[0]);
        System.out.println(kota[1]);
        System.out.println(kota[2]);
    }
}
```

Contoh 2 :

```
public class ArrayKota2{
    public static void main(String[] args){
        String[] kota = {"Jakarta", "Surabaya", "Semarang"};

        // menampilkan elemen array
        System.out.println(kota[0]);
        System.out.println(kota[1]);
        System.out.println(kota[2]);
    }
}
```

Contoh 3 :

```
public class Kota {
    public static void main(String[] args) {
        String[] kota = new String[20];
        Scanner input = new Scanner(System.in);
        int i;

        System.out.print("Masukkan Jumlah Data : ");
        int jml = input.nextInt();
        for (i=0;i<jml;i++){
            System.out.println("Data ke " + (i+1));
        }
    }
}
```

```

        System.out.print("Masukkan nama Kota : ");
        kota[i]=input.next();
    }

    System.out.println("\nNama - nama Kota ");
    for (i=0;i<jml;i++){
        System.out.print((i+" . "));
        System.out.println("Nama KOta : "+kota[i]);
    }
}
}

```

7.4 Array Multi Dimensi

Contoh 4 :

```

class ArrayMultidimensi {
    public static void main(String[] args) {
        String[][] nama = {
            {"Pak ", "Bu ", "Mbak"},
            {"Joko", "Susi"}
        };
        System.out.println(nama[0][0] + nama[1][0]);
        System.out.println(nama[0][1] + nama[1][1]);
        System.out.println(nama[0][2] + nama[1][0]);
    }
}

```

Tugas

Buatlah program seperti berikut :

```
1.001.
TOKO SERBA ADA
*****
Masukkan Item Barang : 3
Data ke 1
Masukkan Kode      : a001
Masukkan jumlah Beli : 3
Data ke 2
Masukkan Kode      : a002
Masukkan jumlah Beli : 2
Data ke 3
Masukkan Kode      : a003
Masukkan jumlah Beli : 5

TOKO SERBA ADA
*****
No  Kode Barang  Nama Barang  Harga  Jumlah Beli  Jumlah Bayar
=====
1   a001         Buku         3000    3           9000
2   a002         Pensil        4000    2           8000
3   a003         Pulpen        5000    5          25000
=====
Total Bayar                                42000
=====
```

Dengan Ketentuan :

- Inputan :
Item Barang, Kode, dan Jumlah Beli
- Output :
Seperti tampilan diatas
- Penentuan Nama Barang serta Harga Berdasarkan Kode Barang

GUI (GRAPHICAL USER INTERFACE)

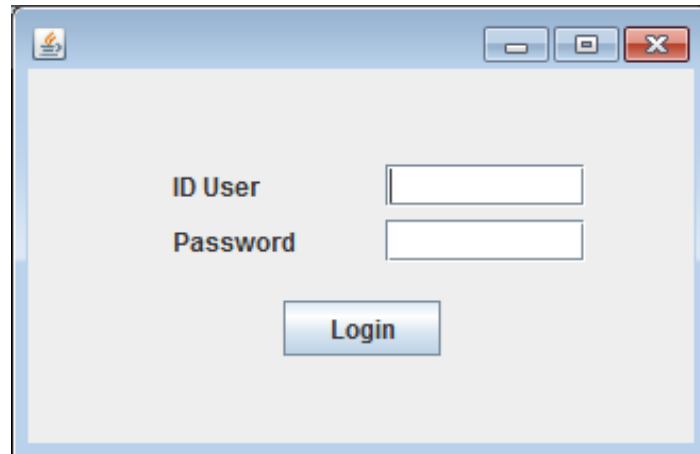
Contoh 1 : *OperasiPertambahan2Angka.java*



Source Code :

```
private void tambahBTActionPerformed(java.awt.event.ActionEvent evt) {  
    int a = Integer.parseInt(angka1TF.getText());  
    int b = Integer.parseInt(angka2TF.getText());  
    int c = a+b;  
    hasilTF.setText(c+"");  
}  
private void hapusBTActionPerformed(java.awt.event.ActionEvent evt) {  
    angka1TF.setText("");  
    angka2TF.setText("");  
    hasilTF.setText("");  
    angka1TF.requestFocus();  
}  
private void keluarBTActionPerformed(java.awt.event.ActionEvent evt) {  
    System.exit(0);  
}
```

Contoh 2 : Login.java



Source Code :

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    if((userTF.getText().equals("ultach"))&&  
        (String.valueOf(passwordTF.getPassword()).equals("ultach"))){  
        new OperasiDuaAngka().setVisible(true);  
        dispose();  
    }else{  
        JOptionPane.showMessageDialog(null,userTF.getText()+"", password anda salah  
            ", "pesan Kesalahan",JOptionPane.ERROR_MESSAGE);  
        userTF.setText("");  
        passwordTF.setText("");  
        userTF.requestFocus();  
    }  
}
```

Contoh 3 :

a. AplikasiPenentuNilaiMahasiswa.java



Source Code :

```
import gui.*;
import java.awt.event.KeyEvent;
import java.io.BufferedWriter;
import java.io.FileWriter;
import javax.swing.JOptionPane;

public class AplikasiPenentuNilaiMahasiswa extends javax.swing.JFrame {
    NilaiMhs2 nm = new NilaiMhs2();

    private void keluarBTActionPerformed(java.awt.event.ActionEvent evt) {
        int reply = JOptionPane.showConfirmDialog(
            null, "Yakin ingin keluar?", "Konfirmasi Keluar Aplikasi",
            JOptionPane.YES_NO_OPTION);
        if(reply == JOptionPane.YES_OPTION){
            System.exit(0);
        }else{
            namaTF.requestFocus();
        }
    }

    private void bersihBTActionPerformed(java.awt.event.ActionEvent evt) {
        namaTF.setText(" ");
        utsTF.setText(" ");
        tmTF.setText(" ");
        uasTF.setText(" ");
        namaLB.setText("");
        rataLB.setText("");
        gradeLB.setText("");
        hasilLB.setText("");
        namaTF.requestFocus();
    }

    private void utsTFKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (!((Character.isDigit(c) || (c == KeyEvent.VK_BACK_SPACE) || (c ==
            KeyEvent.VK_DELETE)))){
            getToolkit().beep();
            JOptionPane.showMessageDialog(null, "Masukkan hanya 0-100");
            evt.consume();
        }
    }
}
```



```

private void hitungBTActionPerformed(java.awt.event.ActionEvent evt) {
    namaLB.setText(namaTF.getText());
    nm.uts = Double.parseDouble(utsTF.getText());
    nm.tm = Double.parseDouble(tmTF.getText());
    nm.uas = Double.parseDouble(uasTF.getText());
    rataLB.setText(""+nm.nilaiRata());
    gradeLB.setText(""+nm.gradeMhs());
    hasilLB.setText(""+nm.hasil());
}

private void simpanBTActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        BufferedWriter out = new BufferedWriter(new
            FileWriter("hitungnilai.txt"));
        String hasil = namaLB.getText()+" ' ' +'\n'+
            rataLB.getText()+" ' ' +'\n'+
            gradeLB.getText()+" ' ' +
            hasilLB.getText();
        out.write(hasil);
        JOptionPane.showMessageDialog(null, "Berhasil disimpan dalam file");
        out.close();
    }catch (Exception e){
        System.err.println("Error : "+e.getMessage());
    }
}
}

```

b. NilaiMhs.java

```

public class NilaiMhs {
    double tm, uas, uts, rata; char grade;
    double nilaiRata(){
        rata = (tm+uts+uas)/3;
        return rata;
    }
}

```

C. NilaiMhs2.java

```

public class NilaiMhs2 extends NilaiMhs{
    String ket;

    char gradeMhs(){
        if (rata >=80)
            grade = 'A';
        else if (rata >=65)
            grade = 'B';
        else if (rata >= 56)
            grade = 'C';
    }
}

```

```

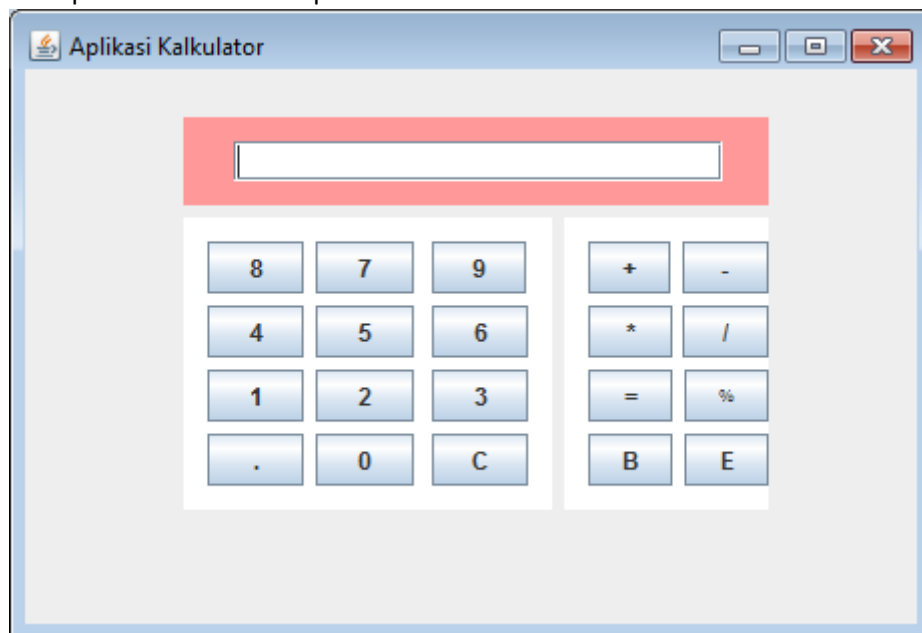
else if(rata >=40)
    grade = 'D';
else
    grade = 'E';
return grade;
}

String hasil(){
if (rata >=56)
    ket = "Lulus";
else
    ket = "Tidak Lulus";
return ket;
}
}

```

Latihan

- 1) Buatlah Aplikasi Kalkulator seperti berikut :



- 2) Buatlah tampilan seperti berikut (*PenentuJumlahHari.java*) :



Dengan ketentuan sebagai berikut :

- Buatlah sebuah class *HitungHari* yang mempunyai method *hitung()* yang berfungsi untuk melakukan proses untuk mendapatkan jumlah hari dari inputan yang di berikan oleh user (Tahun dan Bulan)
- Panggil method *Hitung()* tersebut pada class *PenentuJumlahHari.java*
- Button hapus berfungsi untuk mengosongkan tahun, bulan dan juga hasil perhitungan
- Tombol Simpan berfungsi untuk menyimpan hasil perhitungan ke dalam bentuk .txt
- Hasil ketika di running



ARRAY LIST

Class `ArrayList` mengelola urutan object, yang dapat bertambah dan berkurang sesuai dengan keperluan dan juga menyediakan banyak method untuk berbagai keperluan, misalnya menambah atau menghapus elemen.

Contoh penerapan `ArrayList`.

a. Class `Mahasiswa`

```
public class Mahasiswa {  
    private String NIM, Nama, Alamat;  
  
    public Mahasiswa(String NIM, String Nama, String Alamat) {  
        this.NIM = NIM;  
        this.Nama = Nama;  
        this.Alatam = Alamat;  
    }  
  
    public String getAlamat() {  
        return Alamat;  
    }  
  
    public String getNIM() {  
        return NIM;  
    }  
  
    public String getNama() {  
        return Nama;  
    }  
}
```

b. Class `InputDataMahasiswa`

```
import java.util.ArrayList;  
  
public class InputDataMahasiswa {  
    ArrayList<Mahasiswa> listmahasiswa;  
  
    public InputDataMahasiswa() {  
        listmahasiswa = new ArrayList ();  
    }  
  
    public void insertData(String NIM, String Nama, String Alamat) {  
        Mahasiswa mhs = new Mahasiswa(NIM, Nama, Alamat);  
        listmahasiswa.add(mhs);  
    }  
}
```

```

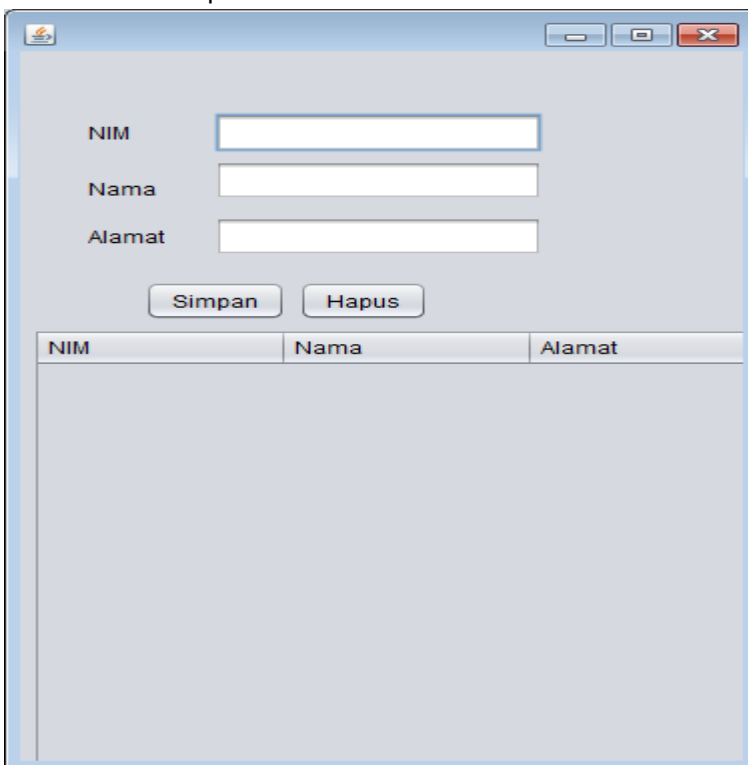
        public ArrayList<Mahasiswa> getAll() {
            return listmahasiswa;
        }

        public void deleteData(int index) {
            listmahasiswa.remove(index);
        }
    }

```

c. JFrame ArrayList

Buatlah Tampilan sbb :



```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class ArrayListTable extends javax.swing.JFrame {
    DefaultTableModel modelmahasiswa;
    InputDataMahasiswa datamahasiswa;

    public ArrayListTable() {
        initComponents();
        datamahasiswa = new InputDataMahasiswa();
    }

```

```

        viewDataTabel();
    }
    public final void viewDataTabel(){
        String [] namakolom = {"NIM", "Nama", "Alamat"};
        Object [][] objekmahasiswa = new Object[datamahasiswa.getAll().size()] [3];
        int i = 0;
        for (Mahasiswa mhs: datamahasiswa.getAll()) {
            String arrayMahasiswa[] = {
                mhs.getNIM(),
                mhs.getNama(),
                mhs.getAlamat()
            };
            objekmahasiswa[i] = arrayMahasiswa;
            i++;
        }
        modelmahasiswa = new DefaultTableModel(objekmahasiswa, namakolom);
        mahasiswaTB.setModel(modelmahasiswa);
    }

    public void ClearTextField(){
        nimTF.setText("");
        namaTF.setText("");
        alamatTF.setText("");
        nimTF.requestFocus();
    }

    private void hapusBTActionPerformed(java.awt.event.ActionEvent evt) {
        datamahasiswa.deleteData(mahasiswaTB.getSelectedRow()+1);
        viewDataTabel();
    }

    private void simpanBTActionPerformed(java.awt.event.ActionEvent evt) {
        datamahasiswa.insertData(
            nimTF.getText(),
            namaTF.getText(),
            alamatTF.getText()
        );
        viewDataTabel();
        ClearTextField();
    }
}

```

Latihan

Buatlah Program seperti berikut ini :

The image shows a Java Swing window with a light blue title bar and standard Windows-style window controls (minimize, maximize, close). The window contains a form for entering student data. On the left, there are four labels: "NIM", "Nama", "Alamat", and "Mata Kuliah", each followed by a text input field. On the right, there are five labels: "Nilai 1 [10%]", "Nilai 2 [15%]", "Nilai 3 - UTS[25%]", "Nilai 4 [15%]", and "Nilai 5 [35%]", each followed by a text input field. Below the "Mata Kuliah" field, there are two buttons: "Simpan" and "Hapus". At the bottom of the window, there is a table with five columns: "NIM", "Nama", "Alamat", "Mata Kuliah", and "Nilai Akhir". The table is currently empty.

NIM	Nama	Alamat	Mata Kuliah	Nilai Akhir
-----	------	--------	-------------	-------------