UIN Maulana
Malik Ibrahim
Malang

MODUL PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

ALLIN JUNIKHAH, M.T.

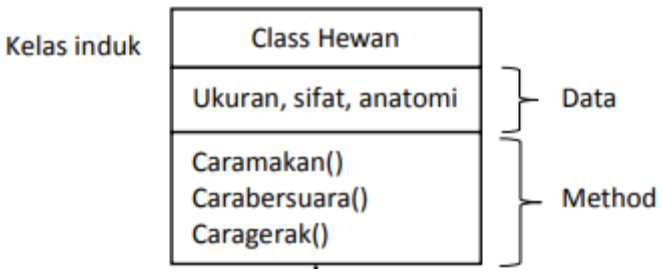
DAFTAR ISI

MODUL 1

Pengenalan Dasar PBO
Class, Object, Method

MATERI

Pemrograman Berorientasi Objek (PBO) atau *Object Oriented Programming* (OOP) adalah suatu metode pemrograman yang mengelola perintah program menjadi objek-objek tertentu. Objek yang dimaksud dalam konsepnya disamakan dengan objek di dunia nyata, contohnya sekelompok kode untuk Mobil, Orang dll. Yang suatu saat dipanggil dengan menyertakan atribut-atribut yang dimilki (Mobil:merk,bahanBakar - Orang:nama, jenisKelamin). Dengan konsep ini kode akan lebih fleksibel dan akan lebih mudah dipahami sesama developer.



Programming Paradigms Comparison

Object Oriented Programming	Procedural Programming
Program is divided into small parts called objects	Program is divided into small parts called functions
Supports inheritance: more code reusability	No inheritance: limited code reusability
Function overloading is supported	Function overloading is not supported
Access specifiers are supported	Access specifiers are not supported
More secure as it provides data hiding	Less secure as it does not provide a proper mechanism to hide data
Data is more important than function	Function is more important than data
Examples: C++, Java, Python, C# etc.	Examples: C++, C, FORTRAN, Pascal, Basic etc.

- Terdapat 3 ciri mendasar dari Pemrograman Berorientasi Objek, yaitu : *Encapsulation* (Enkapsulasi), *Inheritance* (Pewarisan), *Polymorphisme*

Pemrograman Berorientasi Objek yang akan pelajari dalam praktikum ini menggunakan Bahasa Pemrograman Java. Bahasa Pemrograman Java memiliki beberapa komponen, antara lain :

a. Class : wadah yang didalamnya terdapat method-method, tempat mendeklarasikan variabel
b. Object : entiti yang memiliki keadaan/ tingkah laku
c. Attribute : elemen dari sebuah objek yang berisi informasi tentang objek
d. Method : tingkah laku dari objek/ sub program
e. Constructor : method yang digunakan untuk membuat objek baru

Contoh program dengan Konsep Prosedural.

```
class PenampungMobil{
    public static void main(String args[]){
        int tahun_produksi_nissan_1 = 2018;
        int harga_produksi_nissan_1 = 2000000000000000;
        String pemilik_mobil_nissan_1 =  "Tono"
        int tahun_stnk_mobil_nissan_1 = 2022;

        int tahun_produksi_toyota_1 = 2015;
        int harga_produksi_toyota_1 = 3000000000000000;
        String pemilik_mobil_toyota_1 =  "Budi"
        int tahun_stnk_mobil_toyota_1 = 2025;

        int tahun_produksi_suzuki_1 = 2014;
        int harga_produksi_suzuki_1 = 4000000000000000;
        String pemilik_mobil_suzuki_1 =  "Andi"
        int tahun_stnk_mobil_suzuki_1 = 2020;

    }
}
```

Contoh program dengan Konsep PBO.

```
class PenampungMobil{
    public static void main(String args[]){
      Mobil nissan = new Mobil(2018,2000000000000000 , "Tono" , 2022 );
      Mobil toyota = new Mobil(2015,3000000000000000 , "Budi" , 2025 );
      Mobil suzuku = new Mobil(2014,4000000000000000 , "Andi" , 2020 );
    }
}
```

Bahasa pemrograman Java dikenal dengan case sensitive, artinya ada banyak aturan dalam Pemrograman Java yang harus benar-benar diperhatikan. Berikut ini beberapa aturan dalam pemberian nama class :

a. Nama file halaman kerja Java harus sama dengan nama class
b. Nama class boleh terdiri dari huruf kecil dan huruf besar
c. Nama class yang terdiri dari dua suku kata, dapat dituliskan dengan tanda underscore (_) sebagai pemisah. Contoh : Belajar_Java
d. Nama class tidak boleh terdiri dari angka
e. Nama class boleh ditulis dengan huruf abjad dan diikuti dengan angka. Contoh : Belajar_Java1

Pada salah satu CLASS dimana terdapat kode baris berikut :

```
public static void main(String[] args){
}
```

Mengindikasikan nama suatu method dalam class yang bertindak sebagai method utama. Method utama adalah titik awal dari suatu program Java. Semua program kecuali applet yang ditulis dalam bahasa Java dimulai dengan method utama.

Class

Class adalah prototype, atau blueprint, atau rancangan yang mendefinisikan variable dan method-method pada seluruh objek tertentu. Class berfungsi untuk menampung isi dari program yang akan di jalankan, di dalamnya berisi atribut / type data dan method untuk menjalankan suatu program.

Object

Objek adalah sebuah perangkat lunak yang berisi sekumpulan atribut dan method yang berhubungan. Objek merupakan instance (keturunan) dari kelas. Setiap objek dibangun dari sekumpulan data atribut untuk menjabarkan karakteristik khusus dari objek dan sekumpulan method untuk menjabarkan tingkah laku dari objek. Satu kelas bisa memiliki beberapa objek, setiap objek memiliki sifat yang sama persis seperti yang didefinisikan dalam class tersebut.

Method/Prosedur/Fungsi

Method adalah adalah Prosedur atau fungsi yang dimiliki oleh suatu objek. Karena pada dasarnya merupakan prosedur/fungsi, maka method ini akan mengolah atau mengubah data/variabel yang ada didalamnya sesuai dengan operasi yang telah ditentukan.

PRAKTIKUM

Praktikum 1. Membuat Class

```
public class Kelasku {
    public static void main(String[] args) {
    String nama,alamat;
    nama="Miley Cyrus";
    alamat ="Jl. Teluk Aru 11 Arjosari";
    System.out.println(" Nama : " +nama);
    System.out.println(" Alamat : " +alamat);
    }
}
```

Kemudian **RUN** hasil kode tersebut. Pahami struktur kode yang ada.

Praktikum 2. Membuat Object

Buatlah object dari Class **Kelasku** berikut ini.

```
public class Kelasku { //nama kelas
    String nama;    //atribut atau varibel
    String jurusan;
    public static void main(String[] args){
        Kelasku mahasiswa1= new Kelasku(); //object
        mahasiswa1.nama="Cinta Laura";
        mahasiswa1.jurusan="Teknik Informatika";
        Kelasku mahasiswa2= new Kelasku(); //object
        mahasiswa2.nama="Miley Cyrus";
        mahasiswa2.jurusan="Teknik Arsitektur";
```

```
        System.out.println(" Nama Mahasiswa : " +mahasiswa1.nama);
        System.out.println(" Jurusan : " +mahasiswa1.jurusan);
        System.out.println(" Nama Mahasiswa : " +mahasiswa2.nama);
        System.out.println(" Jurusan : " +mahasiswa2.jurusan);
    }
}
```

Kemudian **RUN** hasil kode tersebut. Pahami struktur kode yang ada. Bandingkan dengan kode berikut ini. Praktikkan kode dibawah ini.

```
class Mahasiswa{ //nama kelas
    String nama; //atribut atau varibel
    String jurusan;
}

public class Kelasku {
    public static void main(String[] args){
        Mahasiswa mahasiswa1= new Mahasiswa(); //object
        mahasiswa1.nama="Cinta Laura";
        mahasiswa1.jurusan="Teknik Informatika";
        Mahasiswa mahasiswa2= new Mahasiswa(); //object
        mahasiswa2.nama="Miley Cyrus";
        mahasiswa2.jurusan="Teknik Arsitektur";

        System.out.println(" Nama Mahasiswa : " +mahasiswa1.nama);
        System.out.println(" Jurusan : " +mahasiswa1.jurusan);
        System.out.println(" Nama Mahasiswa : " +mahasiswa2.nama);
        System.out.println(" Jurusan : " +mahasiswa2.jurusan);
    }

}
```

Kemudian **RUN** hasil kode tersebut. Pahami struktur kode yang ada.

ASK : Bagaimanakah hasil dari kedua kode sebelumnya?. Analisa perbedaan kedua kode tersebut.

Praktikum 3. Memodifikasi Atribut

```
public class Kelasku {
    String nama ="Cinta Laura";
    String jurusan ="Teknik Informatika";
    public static void main(String[] args){
       Kelasku mahasiswa= new Kelasku(); //object
        mahasiswa.nama="Jerome Polin";

        System.out.println(" Nama Mahasiswa : " +mahasiswa.nama);
```

```
        System.out.println(" Jurusan : " +mahasiswa.jurusan);
    }
}
```

RUN kode tersebut. Pahami struktur kode yang ada.

ASK : Apa yang terjadi dengan isi dari atribut ?

Praktikum 4. Membuat Method

```
class Kali{
    double a;
    double b;

    public void HasilKali(){
        double c;
        c=a*b;
        System.out.println(" Hasil Perkalian : "+a+" x " +b+" = " +c);
    }
}
public class Kelasku {
    public static void main(String[] args){
        Kali objKali = new Kali();
        objKali.a=34;
        objKali.b=21;
        objKali.HasilKali();
    }

}
```

RUN kode tersebut. Pahami struktur kode yang ada. **HasilKali()** merupakan METHOD dari CLASS **Kali.** Untuk memanggil method tersebut yaitu dengan memanggil *object[.]namamethod*.

TUGAS

```
1. public class Main {
2.   final int x = 10;
3.
4.   public static void main(String[] args) {
5.     Main myObj = new Main();
6.     myObj.x = 25;
7.     System.out.println(myObj.x);
8.   }
9. }
```

1. Analisa, berapakah output dari Kode di atas ? Jika error mengapa demikian ?
2. Dengan menerapkan class, object, dan method, buatlah program Java menghitung:
   - Luas isi tabung
   - Luas persegi panjang
   - Luas trapesium

MODUL 2

Input dari Keyboard, Method Lanjutan

MATERI

A. Input dari Keyboard

Seperti yang kita ketahui, program komputer terdiri dari tiga komponen utama, yaitu: input, proses, dan output.

- Input: nilai yang kita masukan ke program
- Proses: langkah demi langkah yang dilakukan untuk mengelola input menjadi sesuatu yang berguna
- Output: hasil pengolahan

Semua bahasa pemrograman telah menyediakan fungs-fungsi untuk melakukan input dan output.

Java sendiri sudah menyediakan beberapa class untuk mengambil input:

- Class Scanner;
- Class BufferReader;
- Class Console.
- JOptionPane

Tiga class pertama untuk mengambil input pada program berbasis teks (console). Sedangkan untuk GUI menggunakan class JOptionPane dan inputbox pada form.

Sementara untuk outputnya, Java menyediakan fungsi print(), println(), dan format().

B. Static Vs Public Method

```
public class panggilmethod {
    // Static method
    static void myStaticMethod() {
      System.out.println("Static method dapat dipanggil tanpa membuat
object");
    }
    // Public method
    public void myPublicMethod() {
      System.out.println("Public method harus dipanggil dengan membuat
object");
    }
    // Main method
    public static void main(String[] args) {
      myStaticMethod();

      panggilmethod myObj = new panggilmethod();
      myObj.myPublicMethod();
    }
  }
```

C. Konsep Method/Function

- Fungsi adalah bagian dari kode program yang mempunyai tugas spesifik.
- Fungsi akan dipanggil ke program utama atau fungsi yang lain bila dibutuhkan.
- Sebuah fungsi biasanya digunakan untuk menangani suatu proses tertentu yang ada di dalam sebuah program.

➢ **Bentuk penulisan fungsi/method**

Kita bisa menuliskan fungsi atau method dengan beberapa cara, dan method juga mempunyai beberapa macam tipe yaitu:

- void
- string
- double
- integer

Dari beberapa tipe method diatas semua harus mempunyai nilai kembalian kecuali method yang bertipe void.

1. Method yang tidak mempunyai parameter

```
Void hitung()
{
 // Statement atau perintah yang dikerjakan
}
```

2. method yang mempunyai parameter

```
Void hitung(int a,int b)  // Method yang mempunyai parameter
{
// Statement atau perintah yang dikerjakan
}
```

PRAKTIKUM

Praktikum 1. Input dengan Scanner

```
import java.util.Scanner;

public class input1 {
    public static void main(String[] args){
        String nama, prodi, fakultas;
        Scanner input=new Scanner(System.in);

        System.out.print("Nama : ");
        nama = input.nextLine();
        System.out.print("Prodi : ");
        prodi = input.nextLine();
        System.out.print("Fakultas : ");
        fakultas = input.nextLine();
    }
}
```

Praktikum 2. Input dengan BufferReader

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
public class input2 {
    public static void main(String[] args) throws IOException{
       String nama,prodi,fakultas;
       InputStreamReader isr= new InputStreamReader(System.in);
       BufferedReader br= new BufferedReader(isr);

        System.out.print("Nama : ");
        nama = br.readLine();
        System.out.print("Prodi : ");
        prodi = br.readLine();
        System.out.print("Fakultas : ");
        fakultas = br.readLine();
    }
}
```

Praktikum 3. Input dengan GUI JOptionPane

```
import javax.swing.JOptionPane;

public class input3 {
    public static void main(String[] args){

        String nama,prodi,fakultas;
        nama=JOptionPane.showInputDialog("Nama : ");
        prodi=JOptionPane.showInputDialog("Prodi : ");
        fakultas=JOptionPane.showInputDialog("Fakultas : ");
        String msg = "Nama : " + nama + "\nProdi :" + prodi+"\nFakultas :" +
fakultas;
        JOptionPane.showMessageDialog(null, msg);;

    }
}
```

Jalankan 3 source code praktikum di atas. Analisa dan pelajari perbedaan penggunaan ketiga program tersebut.

Praktikum 4. Contoh Method Void dengan Parameter

```
public class ContohMethod {
    static void cetak2(int a){
        System.out.print(" Nilai x: "+a);
    }
    public static void main(String[] args) throws Exception {
        int x;
        for(x=0; x<3; x++){
            cetak2(x);
        }
        System.out.println(" Nilai x terakhir: " +x);
```

```
    }
}
```

TUGAS

1. Gunakan salah satu cara inputan di atas, buatlah Program dengan output seperti di bawah ini.

```
==========INPUT DATA==============
Nama : Allin Junikhah
Prodi : Teknik Informatika
Fakultas : Sains dan teknologi
Nilai Praktikum/Tugas : 85
Nilai UTS : 77
Nilai UAS : 95
============CETAK DATA=============
 Nama : Allin Junikhah
 Prodi : Teknik Informatika
 Jurusan : Sains dan teknologi
 Nilai Prak/Tugas : 85.0
 Nilai UTS : 77.0
 Nilai UAS : 95.0
============NILAI AKHIR=============
Nilai Akhir Mahasiswa : 86.6
```

2. Gunakan Method untuk perhitungan **nilaiAkhirMahasiswa()**
   Dengan persentase nilai :
   Praktikum/Tugas 30%
   UTS 30%
   UAS 30%

MODUL 3
Return Method, Constructor

MATERI
A. Method
Dalam java terdapat dua buah method yaitu :

- **Fungsi**, merupakan method yang memiliki nilai balik (**return**) jika method tersebut dipanggil, cara pembuatan sebuah fungsi adalah dengan cara menentukan nilai baliknya.
- **Prosedur**, merupakan method yang tidak memiliki nilai balik, cara pembuatan prosedur sama dengan fungsi namun bedanya, nilai baliknya menggunakan kata kunci void.

```
public class metodx {
// Fungsi
    public double luas_lingkaran(int diameter) {
        int jari2=diameter/2;
        double luas=Math.PI * Math.pow(jari2,2);
        return luas;
    }
//Prosedur
    public void hitungLuasLingkaran(int jari2){
        double luas=Math.PI * Math.pow(jari2,2);
        System.out.println(luas);
    }
    public static void main(String[]args) {
        metodx mt=new metodx();
        mt. hitungLuasLingkaran(10);
        System.out.print(mt.luas_lingkaran(20));
        System.exit(0);
    }
}
```

B. Constructor
Constructor adalah method khusus yang akan dieksekusi pada saat pembuatan objek (instance). Biasanya method ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek.

```
Source  History
1  package konstruktor;
2
3  public class User {
4      private String username;
5      private String password;
6
7      public User() {
8          System.out.println("eksekusi method constructor...");
9      }
10
11 }
12
```

ini adalah constructor

Cara membuat constructor adalah dengan menuliskan nama method constructor sama seperti nama class.

Contoh.

```
public class lingkaran {
    double radius;
    public lingkaran(){
    radius = 1.0;
    }
}
```

PRAKTIKUM

Contoh 1. Contoh method tidak mengembalikan nilai dan tanpa parameter

```
public class App {
    static void cetak1(){
        System.out.print("Saya Mahasiswa Informatika UIN Malang...");
    }
    public static void main(String[] args) {
        cetak1();
        System.out.println(" Hello, World!");
    }
}
```

Contoh 2. Contoh method mengembalikan nilai (**return**).

```
public class App {
    static int kuadrat(int bil){
        return bil*bil;
    }
    public static void main(String[] args) {
        int x=5;
        System.out.println( x +" kuadrat adalah "+ kuadrat(x));
    }
}
```

Contoh 3. Contoh method mengembalikan nilai menggunakan 2 parameter

```
public class App {
    static int penjumlahan(int bil1, int bil2){
        return bil1+bil2;
    }
    public static void main(String[] args) throws Exception {
        int x1=2;
        int x2=3;
        System.out.println( x1 + " + " + x2 + " = " + penjumlahan(x1, x2) );
    }
}
```

Contoh 4. Membuat Constructor

```
public class lingkaran {
    double radius;

    public lingkaran(){
    radius=10.0;
    System.out.println("Testing....");
    }
    public static void main(String[] args){
        lingkaran o = new lingkaran();
        System.out.println("radius = "+ o.radius);
    }
}
```

TUGAS

- Buatlah Program kalkulator sederhana dengan mengimplementasikan method **Penjumlahan**(), **Pengurangan**(), **Perkalian**() dan **Pembagian()** menggunakan 2 parameter!
- Tampilkan Hasil operasi dengan 2 parameter tersebut berdasarkan method yang ada !
- Implementasikan inputan dari keyboard
- Implementasikan return method

```
=========== INPUT X dan Y ===========
X : 24500
Y : 17450
=========== Hasil Operasi ===========
Penjumlahan : 41950.0
Pengurangan : 7050.0
Perkalian : 4.27525E8
Pembagian : 1.4040114613180517
```

MODUL 4
Encapsulation

MATERI

Istilah *encapsulation* (pembungkusan) adalah menggunakan **private** variable agar data tidak bisa diakses secara langsung dari luar class, tetapi kita masih dapat mengakses data tersebut dengan method **GET** dan **SET** dan mengupdate nilai dari **private** variable.

Method **GET** akan mengembalikan (**return**) nilai variable, dan method **SET** akan mengisi nilai variable.

Fungsi dari *encapsulation*:

1. Untuk meningkatkan keamanan data;
2. Agar lebih mudah dalam mengontrol atribut dan dan method;
3. Class bisa kita buat menjadi read-only (jika hanya menggunakan method GET) dan write-only (jika hanya menggunakan method SET);
4. dan fleksibel: programmer dapat mengganti sebagian dari kode tanpa harus takut berdampak pada kode yang lain.

Cara Membuat Method Setter dan Getter

Cara membuat method setter dan getter sama saja seperti membuat method biasa.

```
public class Person {
  private String name; // private = restricted access

  // Getter
  public String getName() {
    return name;
  }

  // Setter
  public void setName(String newName) {
    this.name = newName;
  }
}
```

Method setter dan getter harus diberikan modifier **public**, karena method ini akan diakses dari luar class.

Ciri Setter Getter:

- Method **setter** tidak memiliki nilai kembalian void (kosong). Karena tugasnya hanya untuk mengisi data ke dalam atribut.
- Sedangkan method **getter** memiliki nilai kembalian sesuai dengan tipe data yang akan diambil.

Setelah kita membuat method setter dan getter, kita bisa mengakses atau menggunakannya seperti method biasa.

PRAKTIKUM

Laksanakan percobaan beberapa program di bawah ini dan analisa hasil dari masing-masing output program.

Program 1.

```
class data {
    public String username;
    public String password;
}

public class user {
    public static void main(String[] args){
        data obj = new data();
        // menggunakan method setter
        obj.username = "blackpink";
        obj.password = "TeknikInformatika";
        // menggunakan method getter
        System.out.println("Username: " + obj.username);
        System.out.println("Password: " + obj.password);
    }
}
```

ASK. Apakah program-1 berjalan dengan baik ? Bandingkan dengan program ke-2

Program 2.

```
class data {
    private String username;
    private String password;
}

public class user {
    public static void main(String[] args){
        data obj = new data();
        // menggunakan method setter
        obj.username = "blackpink";
        obj.password = "TeknikInformatika";
        // menggunakan method getter
        System.out.println("Username: " + obj.username);
        System.out.println("Password: " + obj.password);
    }
}
```

ASK. Apakah program-2 berjalan dengan baik ? Analisa dan jelaskan mengapa demikian!

Program 3.

```
public class user2 {
    private String username;
    private String password;

    // ini method setter
    public void setUsername(String username){
        this.username = username;
    }

    public void setPassword(String password){
        this.password = password;
    }

    // ini method getter
    public String getUsername(){
        return this.username;
    }

    public String getPassword(){
        return this.password;
    }

    public static void main(String[] args){
        user2 obj = new user2();
        // menggunakan method setter
        obj.setUsername("blackpink");
        obj.setPassword("TeknikInformatika");
        // menggunakan method getter
        System.out.println("Username: " + obj.getUsername());
        System.out.println("Password: " + obj.getPassword());
    }
}
```

ASK. Apakah program-3 berjalan dengan baik ? Analisa dan jelaskan mengapa demikian!

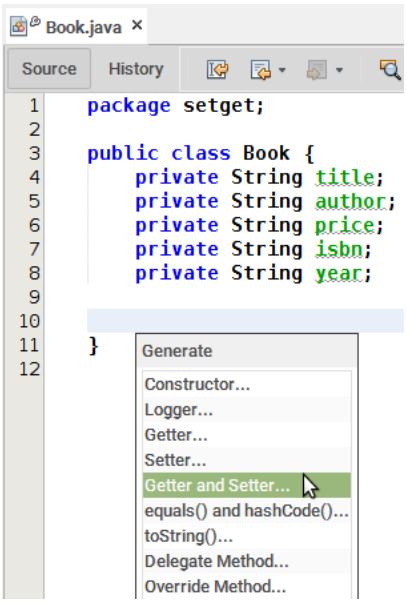
Praktik 4. (Sesuaikan dengan IDE yang Anda Gunakan)

A. Membuat method getter setter di netbean secara otomatis.

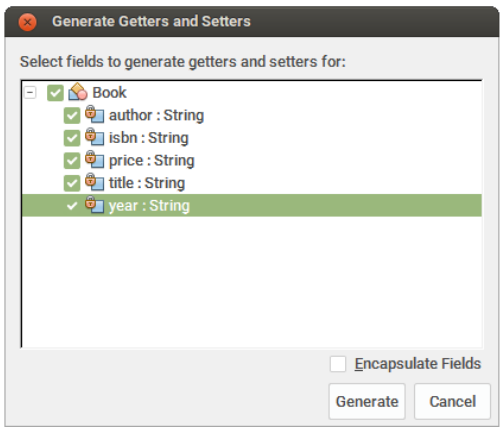
Pertama buat dulu atribut data yang akan dibuatkan method setter dan getter seperti ini:

```
class Book {
    private String title;
    private String author;
    private String price;
    private String isbn;
    private String year;


}
```

Setelah itu klik kanan, lalu pilih **Insert Code…** atau tekan tombol **Alt+Insert** sehingga akan muncul menu seperti ini:



Pilih Setter and Getter untuk masuk ke menu generator Setter dan Getter. Lalu tentukan atribut yang akan dibuatkan method setter dan getter-nya.



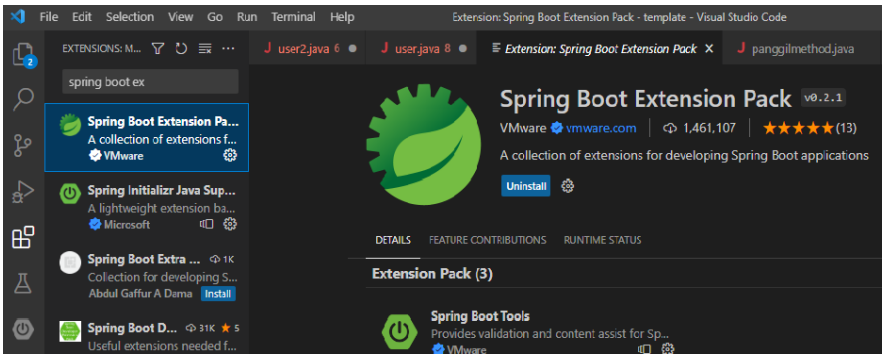
Kemudian KLIK **Generate.**

Book.java

```java
package setget;

public class Book {
    private String title;
    private String author;
    private String price;
    private String isbn;
    private String year;

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }
```

B. Membuat method getter setter di Visual Studio Code secara otomatis.

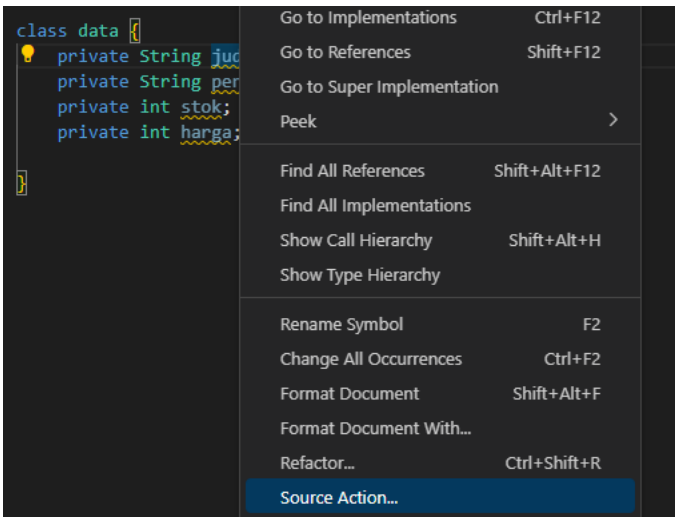
- Lakukan installasi extention **Spring Boot Extension Pack**



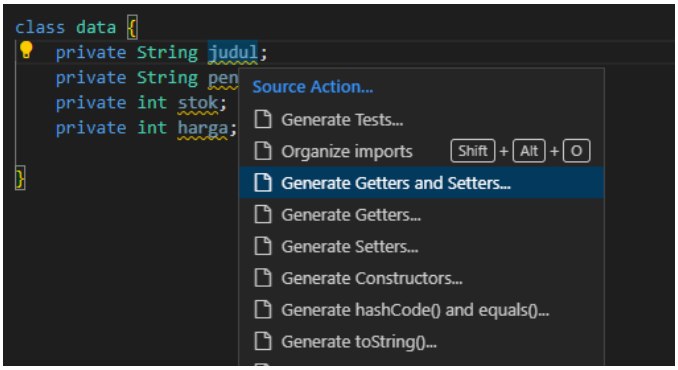
- Buat dulu atribut data yang akan dibuatkan method setter dan getter seperti ini:

```java
class data {
    private String judul;
    private String pengarang;
    private int stok;
    private int harga;

}
```

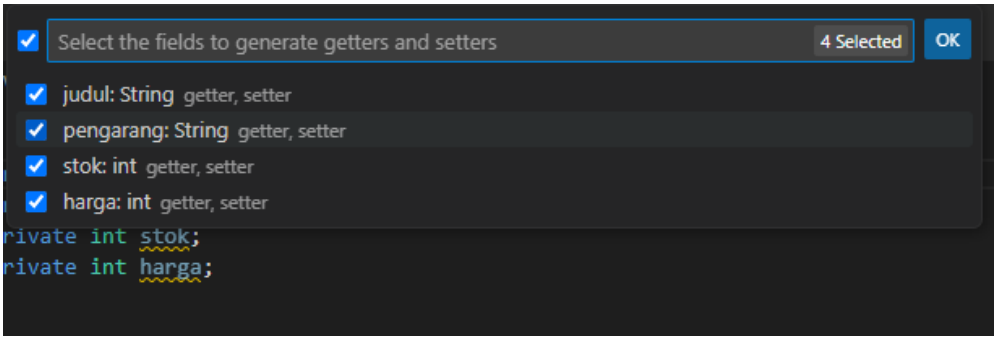
- Klik kanan pada variable, Pilih **Source Action**



- Kemudian pilih **Generate Getters and Setters**



- Kemudian pilih dengan **mencentang variable** yang akan digenerate getter dan setter nya.



- Hasil generate getter dan setter.

```java
class data {
    private String judul;
    public String getJudul() {
        return judul;
    }
    public void setJudul(String judul) {
        this.judul = judul;
    }
    public String getPengarang() {
        return pengarang;
    }
    public void setPengarang(String pengarang) {
        this.pengarang = pengarang;
    }
    public int getStok() {
        return stok;
    }
    public void setStok(int stok) {
        this.stok = stok;
    }
    public int getHarga() {
```

TUGAS

- Buatlah program yang mengimplementasikan **Encapsulation** / **Getter Setter** seperti output berikut ini.
- Gunakanlah modifier **PRIVATE,** dan Inputan data dari keyboard.

```
========== INPUT PEMBELIAN BUKU ==========
Judul Buku : Mudah Belajar Java
Pengarang : Budi Rahardjo
Harga Buku (Rp) : 65000
Jumlah Pembelian : 3
========== DATA PEMBELIAN ==========
Judul Buku: Mudah Belajar Java
Pengarang: Budi Rahardjo
Harga Buku (Rp): 65000
Jumlah Pembelian: 3
========== TOTAL PEMBAYARAN ==========
Total (Rp): 195000
```

- Buatlah Laporan dari Hasil Praktikum dan Tugas Anda.

MODUL 5

Inheritance (Subclass and Superclass)

MATERI

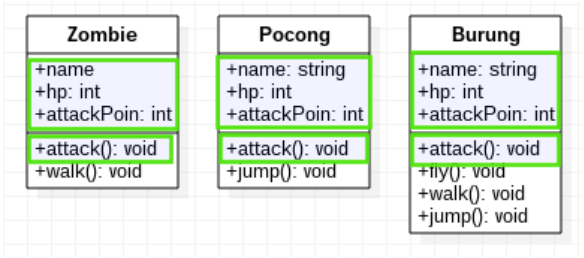
Dalam bahasa pemrograman Java, suatu atribut dan *method* dapat **diwariskan** dari satu *Class* ke *Class* lainnya. Berikut ini beberapa istilah dalam konsep pewarisan/*inheritance*:

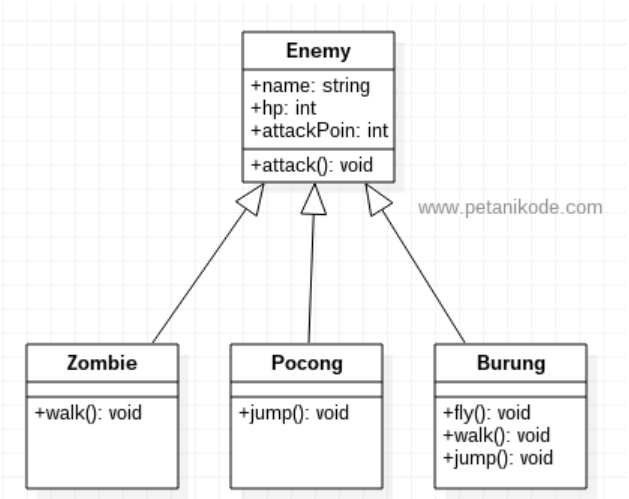
- **subclass (child)** - the class that inherits from another class
- **superclass (parent)** - the class being inherited from

Dengan konsep inheritance, sebuah class dapat mempunyai class turunan. Suatu class yang mempunyai class turunan dinamakan **superclass/parent** class. Sedangkan class turunan itu sendiri seringkali disebut **subclass/child**.

Suatu subclass dapat mewarisi apa saja yang dimiliki oleh parent class-nya. Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (**extend**) parent class-nya.

- Gambaran Implementasi Inheritance dalam Class Diagram.





- Implementasi inheritance dalam source code.

Contoh:

```
public class B extends A {
...
}
```

Contoh diatas memberitahukan kompiler Java bahwa kita ingin meng- extend class A ke class B. Dengan kata lain, class B adalah subclass (class turunan) dari class A, sedangkan class A adalah parent class dari class B

Suatu parent class dapat tidak mewariskan sebagian member-nya kepada subclass-nya. Sejauh mana suatu member dapat diwariskan ke class lain, ataupun suatu member dapat diakses dari class lain, sangat berhubungan dengan access control (kontrol pengaksesan). Di dalam java, kontrol pengaksesan dapat digambarkan dalam tabel berikut ini:

Modifier	class yang sama	package yang sama	subclass	class manapun
private	√			
default	√	√		
protected	√	√	√	
public	√	√	√	√

Referensi :

[1] "MODUL PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK TEKNIK INFORMATIKA UIN MAULANA MALIK IBRAHIM MALANG." Accessed: Mar. 27, 2023. [Online]. Available: http://informatika.uin-malang.ac.id/wp-content/uploads/2018/12/Modul-Praktikum-PBO-Lengkap.pdf

[2] "Java Tutorial." https://www.w3schools.com/java/default.asp (accessed Mar. 27, 2023).

[3] Ahmad Muhardian, "Belajar Java OOP: Memahami Inheritance dan Method Overriding," Dec. 25, 2017. https://www.petanikode.com/java-oop-inheritance/ (accessed Mar. 27, 2023).

PRAKTIKUM

Laksanakan percobaan beberapa program di bawah ini dan analisa hasil dari masing-masing output program.

Program-1.

```
class Vehicle {
    protected String brand = "Ford";
    public void honk() {
      System.out.println("Tuut, tuut!");
    }
  }

  class Car extends Vehicle {
    private String modelName = "Mustang";
    public static void main(String[] args) {

      Car myCar = new Car();
      myCar.honk();
      System.out.println(myCar.brand + " " + myCar.modelName);
    }
  }
```

Program-2.

```
class Vehicle {
    private String brand = "Ford";
    public void honk() {
      System.out.println("Tuut, tuut!");
    }
  }

  class Car extends Vehicle {
    private String modelName = "Mustang";
    public static void main(String[] args) {

      Car myCar = new Car();
      myCar.honk();
      System.out.println(myCar.brand + " " + myCar.modelName);
    }
  }
```

ASK. Apakah program - 2 diatas berjalan dengan baik ? Bandingkan dan Analisa kedua program diatas !

Program-3.

```
class Komputer {
    String processor = "Intel Core i9";

    String cekKomputer() {
        return "Ini berasal dari class Komputer";
    }
}

class Laptop extends Komputer {
    String merk = "Asus";

    String cekLaptop() {
        return "Ini berasal dari class Laptop";
    }
}

class belajarInheritance {
  public static void main(String args[]){
    Laptop laptopAndi = new Laptop();

    System.out.println(laptopAndi.processor);
    System.out.println(laptopAndi.merk);
    System.out.println(laptopAndi.cekKomputer());
    System.out.println(laptopAndi.cekLaptop());
  }
}
```

TUGAS

Latihan Trial and Error Modifier dan Inheritance dari source code yang ada. Pelajari dan analisa.

MODUL 6

Polymorphism

MATERI

Poly artinya banyak, **morfisme** artinya bentuk. **Polimorfisme** (bahasa inggris *polymorphism*) adalah sebuah prinsip dalam biologi di mana oraganisme atau spesias dapat memiliki banyak bentuk atau tahapan (stages).

Prinsip ini juga diadopsi pada pemrograman berorientasikan objek. Sehingga kita dapat definisikan sebagai berikut:

- Polimorfisme dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak "bentuk" method yang berbeda-beda meskipun namanya sama.
- "Bentuk" di sini dapat kita artikan: isinya berbeda, parameternya berbeda, dan tipe datanya berbeda.
- Objek dapat mendefenisikan beberapa hal yang berbeda dengan cara yang sama. Dengan adanya polimorfisme, kita dapat melihat beberapa kesamaan antara sebuah kelas dengan kelas yang lain.

Polimorfisme pada Java memiliki 2 macam yaitu diantaranya:

1) **Static Polymorphism** (Polimorfisme statis) → menggunakan method overloading
2) **Dynamic Polymorphism** (Polimorfisme dinamis) → menggunakan method overriding

Method overloading terjadi pada sebuah class yang memiliki nama method yang sama tapi memiliki parameter dan tipe data yang berbeda.
Tujuan dari method overloading yaitu memudahkan penggunaan atau pemanggilan method dengan fungsionalitas yang mirip.

Method overriding identik dengan menggunakan pewarisan (inheritance), implementasi interface bahkan abstrak class.

Interface adalah class kosong yang berisi nama-nama method dan nantinya harus diimplementasikan pada class lain. Dalam pengimplementasiannya, tiap-tiap class akan mengimplementasikan secara berbeda dengan nama method yang sama.

Abstrak adalah class yang masih dalam bentuk bayangan. Ia tidak bisa dibuat langsung menjadi objek karena bentuknya masih bayangan atau abstrak. Tentunya abstrak ini induk dan jika ini konkrit, kamu mesti mengimplementasikan method-method tersebut. Ini bisa kamu lakukan dengan melakukan teknik pewarisan (inheritance).

Referensi :

[1] "Pengertian Polimorfisme Dalam Pemrograman Java." https://www.dicoding.com/blog/pengertian-polimorfisme-dalam-pemrograman-java/ (accessed April. 03, 2023).
[2] "Java Polymorphism." https://www.programiz.com/java-programming/polymorphism (accessed April. 03, 2023).
[3] Ahmad Muhardian, "Belajar Java OOP: Memahami Prinsip Polimorfisme dalam OOP," Dec. 27, 2019. https://www.petanikode.com/java-oop-polimorfisme/ (accessed April. 03, 2023).

PRAKTIKUM

Laksanakan percobaan beberapa program di bawah ini dan analisa hasil dari masing-masing output program.

Program-1.

```
package modul6;

class Komputer {
    public void tampil() {
        System.out.println("Ini Komputer...");
    }
}

class Laptop extends Komputer {
    public void tampil() {
        System.out.println("Ini Laptop...");
    }
}

class PC extends Komputer {
    public void tampil() {
        System.out.println("Ini PC...");
    }
}

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

        Laptop lp = new Laptop();
        lp.tampil();

        PC pc = new PC();
        pc.tampil();
    }
}
```

ASK. Perhatikan, analisa dan pahami program di atas, Morf/bentuk apa yang memiliki kesamaan? Apakah output menampilkan kesamaan?

Program-2. Method Overloading

```
package modul6;

class Hitung {
    static int Perkalian(int a, int b) {
        return a * b;
```

```
    }

    static double Perkalian(double a, double b) {
        return a * b;
    }
}

class polymorfOverload {
    public static void main(String[] args) {
        System.out.println(Hitung.Perkalian(2, 4));
        System.out.println(Hitung.Perkalian(5.5, 6.3));
    }
}
```

ASK. Perhatikan, analisa dan pahami program di atas, Morf/bentuk apa yang memiliki kesamaan? Adakah perbedaan dari method yang ada, analisa output yang ada!

Program-3. Method Overriding

```
package modul6;

abstract class Hewan {
    // Mendeklrasikan class dan method tipe abstract
    protected abstract void munculSuara();
}

class Kucing extends Hewan {
    // Menggunakan method dari kelas induk abstrak
    @Override
    protected void munculSuara() {
        System.out.println("Suara Kucing: Meow...meow..meow.");
    }

}

class Burung extends Hewan {
    // Menggunakan method dari kelas induk abstrak
    @Override
    protected void munculSuara() {
        System.out.println("Suara Burung: Cit...cit..cit.");
    }

}
```

```
public class polymorfOveride {
    public static void main(String[] args) {
        Hewan kucing = new Kucing();
        kucing.munculSuara();

        Hewan burung = new Burung();
        burung.munculSuara();
    }
}
```

ASK. Morf/bentuk apa yang memiliki kesamaan? Apakah yang sama dan apakah yang berbeda???

TUGAS

- Membuat program *Polymorphism* dan *Inheritance* sederhana yang dapat menampilkan jenis-jenis alat musik seperti : piano, biola, gitar, drum, saxophone, dan trumpet.
- Gunakanlah variabel dan method yang tepat, agar konsep polimorfisme dapat diterapkan dengan baik.

MODUL 7

Abstract, Interface

MATERI

Java Abstraction

Abstract Classes and Methods

Data **abstraction** is the process of hiding certain details and showing only essential information to the user.
Abstraction can be achieved with either **abstract classes** or **interfaces**

The abstract keyword is a non-access modifier, used for classes and methods:

- **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

An abstract class can have both abstract and regular methods:

```
abstract class Animal {
  public abstract void animalSound();
  public void sleep() {
    System.out.println("Zzz");
  }
}
```

From the example above, it is not possible to create an object of the Animal class:

```
Animal myObj = new Animal(); // will generate an error
```

To access the abstract class, it must be inherited from another class.

Java Interface

Interfaces

Another way to achieve abstraction in Java, is with interfaces.

An interface is a completely "**abstract class**" that is used to group related methods with empty bodies:

Example.

```
// interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void run(); // interface method (does not have a body)
}
```

To access the interface methods, the interface must be "implemented" (kinda like inherited) by another class with the implements keyword (instead of extends). The body of the interface method is provided by the "implement" class:

Referensi :
[1] https://www.w3schools.com/java/java_abstract.asp
[2] https://www.w3schools.com/java/java_interface.asp

PRAKTIKUM

Laksanakan percobaan beberapa program di bawah ini dan analisa hasil dari masing-masing output program.

Program-1. ABSTRACT

```
// Abstract class
abstract class Animal {
  // Abstract method (does not have a body)
  public abstract void animalSound();
```

```
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}

// Subclass (inherit from Animal)
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig(); // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

Program-2. INTERFACE

```
// Interface
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void sleep(); // interface method (does not have a body)
```

```
}

// Pig "implements" the Animal interface
class Pig implements Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
  public void sleep() {
    // The body of sleep() is provided here
    System.out.println("Zzz");
  }
}

class Main {
  public static void main(String[] args) {
    Pig myPig = new Pig();  // Create a Pig object
    myPig.animalSound();
    myPig.sleep();
  }
}
```

MODUL 8

GUI (*Graphical User Interface*)

MATERI

Swing adalah API (*Application Programming Interface*) untuk membuat GUI (*Graphical User Interface*) untuk aplikasi yang dibuat dengan Java.

Package yang bisa digunakan : javax.swing.*

Beberapa komponen Swing antara lain :

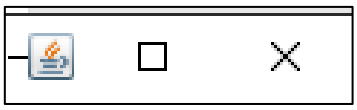
1. JComponent : class induk untuk semua komponen Swing
2. JFrame : Class yang dapat membuat frame.
3. JPanel : Class yang dapat digunakan untuk menampung komponen lain.
4. JLabel : Class yang digunakan untuk menampilkan label.
5. JButton : class untuk membuat sebuah tombol
6. JCheckBox : Class untuk membuat pilihan ya/tidak
7. JTextField : Class untuk mengisi data text

Langkah-langkah membuat aplikasi GUI dengan Swing :

- Membuat objek dengan class JFrame. Isi judul frame sebagai parameternya
- Atur setVisible dengan nilai true.

```
import javax.swing.*;

public class Demo {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo Swing");
        f.setVisible(true);
    }
}
```

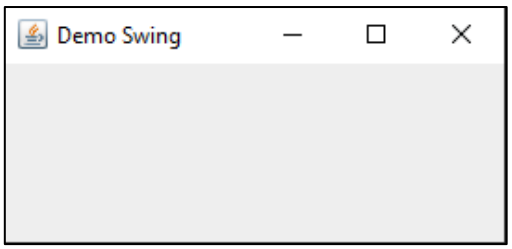
Output:



- Method setSize memiliki parameter lebar (width) dan tinggi (height). Lebar dan tinggi dalam satuan pixel.

```
import javax.swing.*;

public class Demo2 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo Swing");
        f.setSize(200, 100);
        f.setVisible(true);
    }
}
```

Output:



- Jika kita mengklik tombol **Close windows** pada aplikasi berbasis Swing, maka windows akan tertutup tetapi aplikasinya belum benar-benar tertutup.
- Untuk benar-benar menutup maka kita harus mengatur **setDefaultCloseOperation** dengan konstanta **JFrame.EXIT_ON_CLOSE**.

```
import javax.swing.*;

public class Kalkulator {
    public static void main(String[] args) {
        JFrame f = new JFrame("Demo Swing");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(200, 100);
        f.setVisible(true);
    }
}
```

Langkah untuk menambah sebuah komponen ke sebuah Frame adalah :

- Buat objeknya (sesuai dengan komponen yang diinginkan).
- Tambahkan ke Frame dengan memanggil method **add()** milik Frame.

```
import javax.swing.*;

public class GUI1 {
    public static void main(String[] args) {
        ......................
```

```
        JButton tombol = new JButton("Tombol");
        f.add(tombol);
        ......................
    }
}
```

Output:

PRAKTIKUM

Laksanakan percobaan beberapa program di bawah ini dan analisa hasil dari masing-masing output program. Mulai dengan membuatlah project baru.

Program-1. Membuat jendela program

```
import javax.swing.JFrame;

public class App {
    public static void main(String[] args) throws Exception {

        // membuat objek jendela
        JFrame myWindow = new JFrame();

        // berikan judul pada jendela
        myWindow.setTitle("Aplikasi Kasir");

        // tentukan ukuran jendela
        myWindow.setSize(400, 200);

        // tampilkan jendela ke layar
        myWindow.setVisible(true);
    }

}
```

Output:

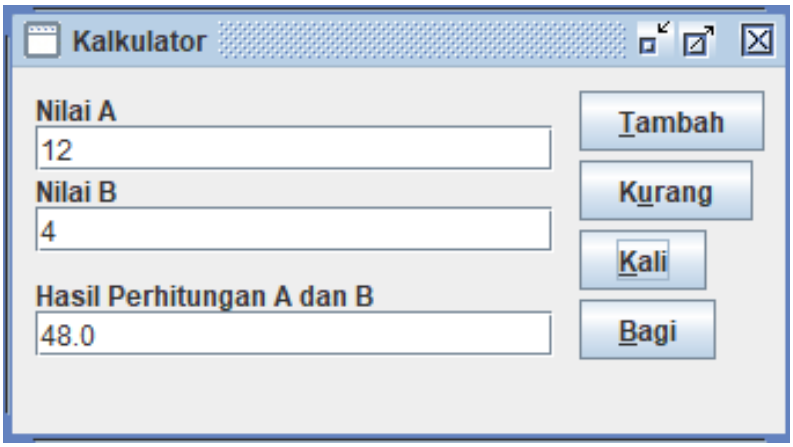
Program-1. Membuat Program Kalkulator Sederhana

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Kalkulator extends JFrame implements ActionListener {
    public JLabel label1, label2, label3;
    public JTextField text1, text2, text3;
    public JButton button1, button2, button3, button4;

    public Kalkulator() {
        label1 = new JLabel("Nilai A ");
        label1.setLocation(10, 10);
        label1.setSize(label1.getPreferredSize());

        text1 = new JTextField(20);
        text1.setLocation(10, 25);
        text1.setSize(text1.getPreferredSize());

        label2 = new JLabel("Nilai B ");
        label2.setLocation(10, 45);
        label2.setSize(label2.getPreferredSize());

        text2 = new JTextField(20);
        text2.setLocation(10, 60);
        text2.setSize(text2.getPreferredSize());

        label3 = new JLabel("Hasil Perhitungan A dan B");
        label3.setLocation(10, 90);
        label3.setSize(label3.getPreferredSize());

        text3 = new JTextField(20);
        text3.setLocation(10, 105);
        text3.setSize(text3.getPreferredSize());

        button1 = new JButton("Tambah");
        button1.setLocation(245, 10);
        button1.setSize(button1.getPreferredSize());
        button1.addActionListener(this);
        button1.setMnemonic('T');

        button2 = new JButton("Kurang");
        button2.setLocation(245, 40);
        button2.setSize(button2.getPreferredSize());
        button2.addActionListener(this);
        button2.setMnemonic('u');
```

```
        button3 = new JButton("Kali");
        button3.setLocation(245, 70);
        button3.setSize(button3.getPreferredSize());
        button3.addActionListener(this);
        button3.setMnemonic('K');

        button4 = new JButton("Bagi");
        button4.setLocation(245, 100);
        button4.setSize(button4.getPreferredSize());
        button4.addActionListener(this);
        button4.setMnemonic('B');
    }

    public void actionPerformed(ActionEvent ae) {
        double a = 0.0, b = 0.0, c = 0.0;

        // baca data dari textfield
        try {
            a = Double.parseDouble(text1.getText());
            b = Double.parseDouble(text2.getText());

        } catch (NumberFormatException nfe) {
            nfe.printStackTrace();
        }

        if (ae.getSource() == button1) {
            c = a + b;
        } else if (ae.getSource() == button2) {
            c = a - b;
        } else if (ae.getSource() == button3) {
            c = a * b;
        } else {
            c = a / b;
        }

        // menampilkan hasil pada textfield
        text3.setText(new String().valueOf(c));

    }

    public void tampilan() {
        JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame frame = new JFrame("Kalkulator");
        frame.setLayout(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.getContentPane().add(label1);
        frame.getContentPane().add(text1);
        frame.getContentPane().add(label2);
```

```
        frame.getContentPane().add(text2);
        frame.getContentPane().add(label3);
        frame.getContentPane().add(text3);
        frame.getContentPane().add(button1);
        frame.getContentPane().add(button2);
        frame.getContentPane().add(button3);
        frame.getContentPane().add(button4);

        frame.setBounds(0, 0, 350, 200);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }

    public static void main(String[] args) throws Exception {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                Kalkulator app = new Kalkulator();
                app.tampilan();
            }
        });

    }

}
```

Output:



TUGAS

1. Tentukan judul program dari Tugas Akhir Praktikum PBO Anda
2. Tugas Individu, kumpulkan pada asisten praktikum Anda. Pastikan judul antara praktikan satu dengan yang lain tidak sama.