**LAPORAN PRAKTIKUM**

**DASAR-DASAR PEMROGRAMAN PERTEMUAN KE-12**

Object Oriented Program (OOP)

****

Disusun Oleh:

Muhammad Fitroh Amrilla

(11190910000022)

Program Studi Teknik Informatika

Fakultas Sains dan Teknologi

2019/2020

1. **Pembahasan**
2. Pengertian OOP
3. Konsep OOP
4. Konsep Abstrak, encapsulation, inheritance, dan polymorphism
5. **Dasar Teori**

**Pengertian OOP**

OOP (Object Oriented Programming) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, nah objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Saya ambil contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dll. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

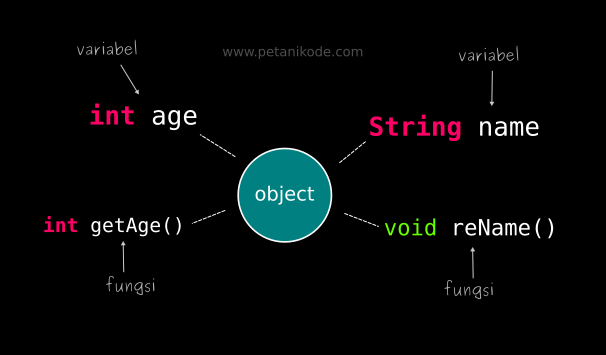
**Konsep OOP**

* Abstrak Class
  1. Kelas merupakan deskripsi abstrak informasi dan tingkah laku dari sekumpulan data.
  2. Kelas dapat diilustrasikan sebagai suatu cetak biru(blueprint) atau prototipe yang digunakan untuk menciptakan objek.
  3. Kelas merupakan tipe data bagi objek yang mengenkapsulasi data dan operasi pada data dalam suatu unit tunggal.
  4. Kelas mendefinisikan suatu struktur yang terdiri atas data kelas (data field), prosedur atau fungsi (method), dan sifat kelas (property).
* Encapsulation
  1. Istilah enkapsulasi sebenarnya adalah kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail informasi.
  2. Proses enkapsulasi memudahkan kita untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci.
  3. Enkapsulasi menekankan pada antarmuka suatu kelas, atau dengan kata lain bagaimana menggunakan objek kelas tertentu.
  4. Contoh: kelas mobil menyediakan antarmuka fungsi untuk menjalankan mobil tersebut, tanpa kita perlu tahu komposisi bahan bakar, udara dan kalor yang diperlukan untuk proses tersebut.
* Inheritance
  1. Kita dapat mendefinisikan suatu kelas baru dengan mewarisi sifat dari kelas lain yang sudah ada.
  2. Penurunan sifat ini bisa dilakukan secara bertingkattingkat, sehingga semakin ke bawah kelas tersebut menjadi semakin spesifik.
  3. Sub kelas memungkinkan kita untuk melakukan spesifikasi detail dan perilaku khusus dari kelas supernya.
  4. Dengan konsep pewarisan, seorang programmer dapat menggunakan kode yang telah ditulisnya pada kelas super berulang kali pada kelas-kelas turunannya tanpa harus menulis ulang semua kodekode itu.
* Polymorphism
  1. Polimorfisme merupakan kemampuan objekobjek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.
  2. Polimorfisme juga dapat dikatakan kemampuan sebuah objek untuk memutuskan method mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.
  3. Method overriding.
  4. Method name overloading.

## Apa itu OOP?

OOP (Object Oriented Programming) atau dalam bahasa indonesia dikenal dengan pemrograman berorientasikan objek (PBO) merupakan sebuah paradigma atau teknik pemrograman yang berorientesikan Objek.

Pada OOP, Fungsi dan variabel **dibungkus** dalam sebuah **objek** atau class yang dapat saling brinteraksi, sehingga membentuk sebuah program.



Variabel dalam objek akan menyimpan data dari objek. Sedangkan fungsi akan menentukan operasinya.

Contoh objek dalam dunia nyata: Mobil, Burung, Drone, Meja, Pohon, dll.

------------------OBJEK

Drone

------------------Variabel/Atribut

energi = 100;

ketinggian = 200;

kecepatan = 29;

------------------Fungsi

terbang();

matikanMesin();

turun();

maju();

mundur();

belok();

------------------

Semua objek di dunia nyata yang memiliki sifat dan tingkah laku, bisa kita representasikan dalam kode.

Kata kunci yang perlu diingat:

**“Objek isinya data dan fungsi”**

## Class dan Object

Class adalah rancangan atau blue print dari sebuah objek.

Sedangkan objek dalam pemrograman adalah sebuah variabel yang merupakan instance dari Class.

Instance bisa diartikan sebagai wujud dari class.

Class berisi definisi variabel dan fungsi yang menggambarkan sebuah objek.

Dalam OOP:

* **Variabel** disebut **atribut** atau **properti**;
* **Fungsi** disebut **method**.

Contoh Class:

class NamaClass {

String atribut1;

String atribut2;

void namaMethod(){ ... }

void namaMethodLain(){ ... }

}

Lalu, Kita biasanya membuat objek (instance) seperti ini:

NamaClass namaObj = new NamaClass();

Kata kunci new berfungsi untuk membuat objek baru dari class tertentu.

Setelah membuat objek, kita bisa mengakses atribut dan method dari objek tersebut.

Contoh:

namaObj.namaMethod();

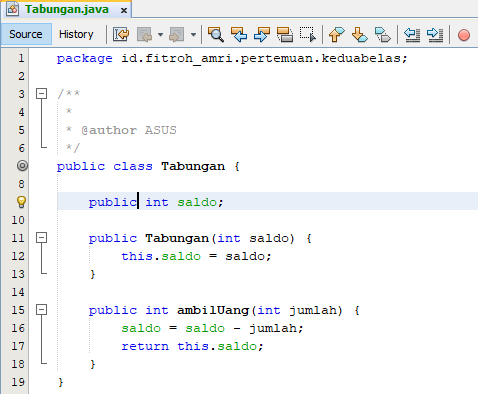
namaObj.atribut1;

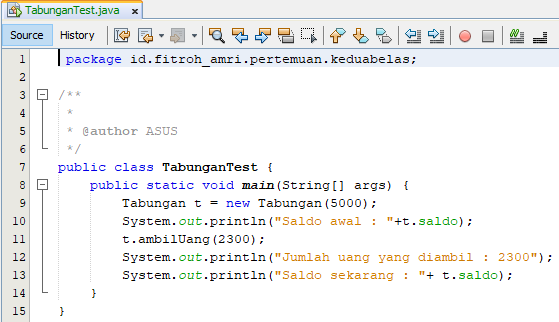
Tanda titik (.) berfungsi untuk mengakses atribut dan method

1. **Latihan**
2. Mengimplementasikan UML class diagram dalam program untuk class Tabungan

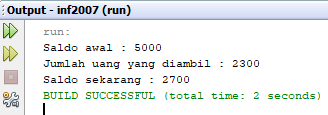
|  |
| --- |
| Tabungan |
| +saldo : int |
| +Tabungan (saldo : int) |
| +ambilUang(jumlah : int) |
|  |

Source Code :





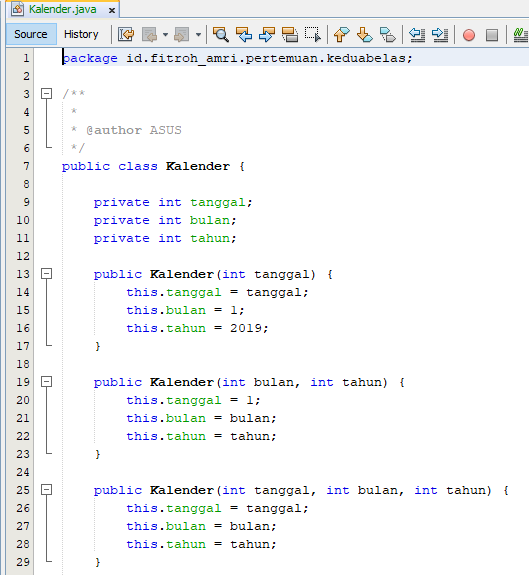
Output :

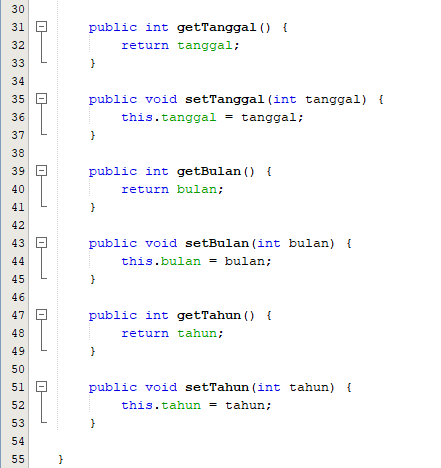


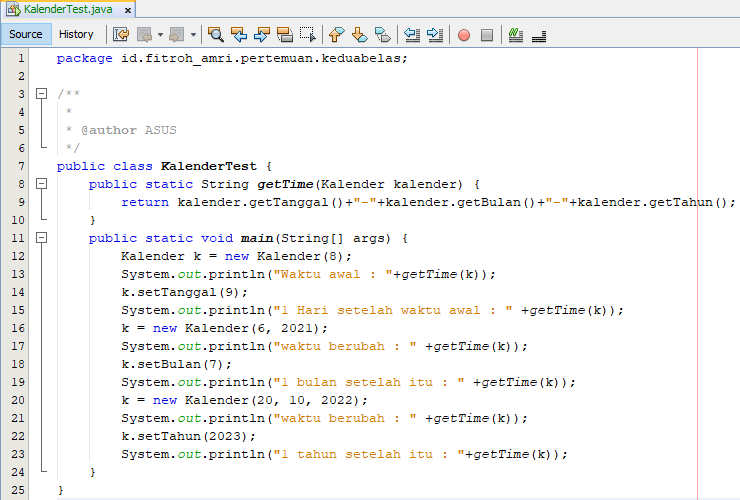
1. Mengimplementasikan UML class diagram dalam program untuk class Kalender

|  |
| --- |
| Kalender |
| -tanggal : int |
| -bulan : int |
| -tahun : int |
| +Kalender(tanggal : int) |
| +Kalender (bulan : int , tahun : int) |
| +Kalender (tanggal :int , bulan : int , tahun : int) |

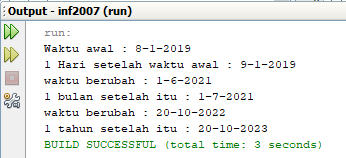
Source Code :







Output :

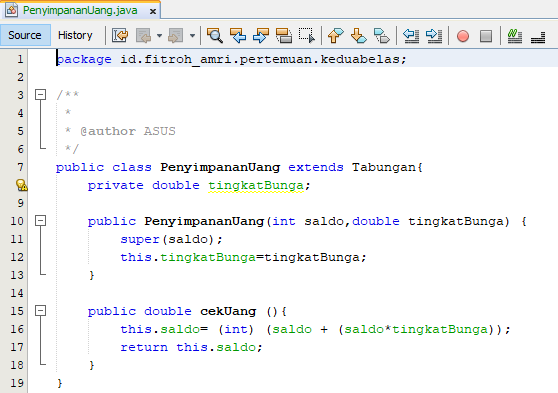


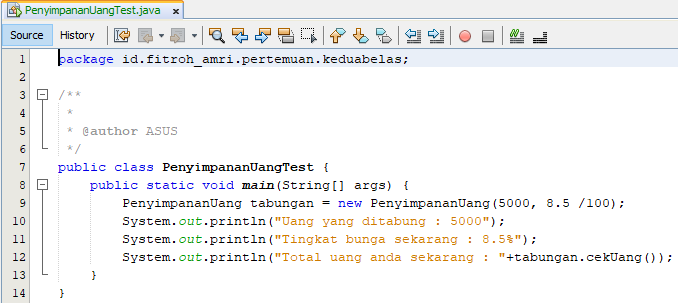
1. Mengimplementasikan UML class diagram dalam program untuk class Penyimpanan Uang

|  |
| --- |
| Tabungan |
| #saldo : int |
| +Tabungan (saldo : int) |
| +ambilUang(jumlah : int) |

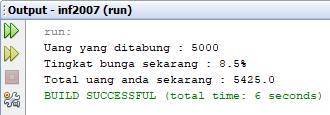
|  |
| --- |
| PenyimpananUang |
| -tingkatBunga : double |
| +PenyimpananUang (saldo : int, tingkatBunga : double) |
| +cekUang() : double |

Source Code :





Output :



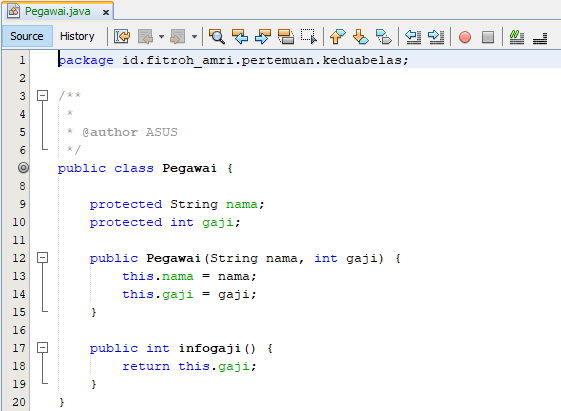
1. Mengimplementasikan UML class diagram dalam program untuk class Pegawai

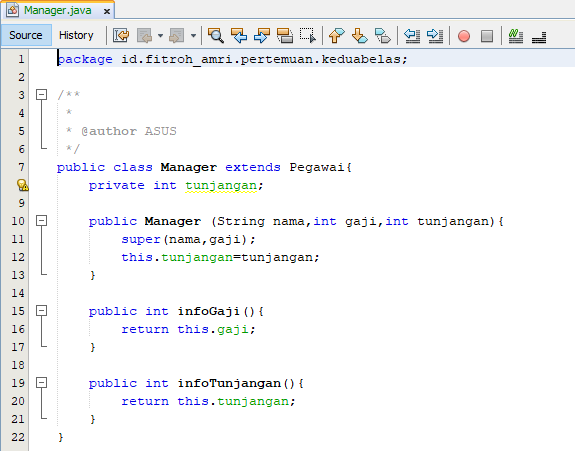
|  |
| --- |
| Pegawai |
| #nama : String |
| #Gaji : int |
| +Pegawai(nama : String , gaji : int) |
| +InfoGaji (): int |

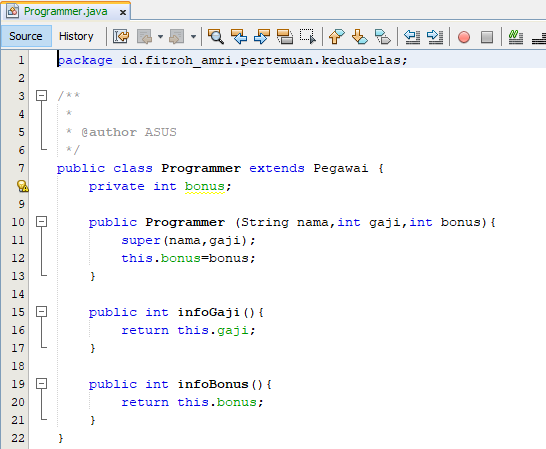
|  |
| --- |
| Manager |
| -tunjangan : int |
| +Manager (nama : String, gaji: int, tunjangan : int) |
| +infoGaji() : int |
| +infoTunjangan() : int |

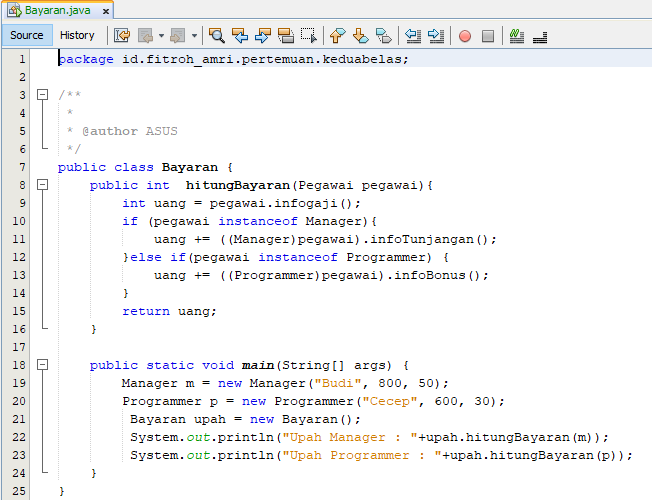
|  |
| --- |
| Programmer |
| bonus : int |
| +Programmer(nama : String, gaji : int, bonus : int) |
| +infoGaji() : int |
| +infoBonus() : int |

Source Code :









Output :

