**LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL VI**

**POLIMORFISME (*POLYMORPHISM)***

****

**DOSEN PENGAMPU :**

Agus Priyanto, S. Kom., M.Kom.

**DISUSUN OLEH :**

Filfimo Yulfiz Ahsanul Hulqi

19102143

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**FAKULTAS INFORMATIKA**

**PRODI S1 INFORMATIKA**

**2020**

**BAB I**

**Tujuan Praktikum**

Mahasiswa diharapkan mampu untuk :

1. Mengerti prinsip polimorfisme dalam Java.
2. Mengerti tentang prinsip polimorfisme dan cara pemakaianya dalam membentuk suatu kelas.
3. Dapat mengimplementasikannya pada permasalahan yang nyata.

**BAB II**

**Dasar Teori**

1. Polimorfisme

Polimorfisme dapat berarti ”mempunyai banyak bentuk” dapat disimpulkan, Polimorfisme adalah kemampuan untuk meminta objek yang berbeda untuk melaksanakan tugas yang sama dan meembuat objek tahu bagaimana untuk mencapainya dengan caranya sendiri. Polimorfisme menunjukkan kemampuan untuk menangani dua atau lebih bentuk objek yang berlainan saat eksekusi berlangsung.

1. Macam-macam Polimorfisme

Polimorfisme dalam OOP merupakan sebuah konsep OOP di mana class memiliki banyak ”bentuk” method yang berbeda-beda, meskipun namanya sama. Maksud dari ”bentuk” adalah isi, tipe data dan parameternya berbeda.

Polimorfisme pada Java memiliki 2 macam yaitu :

1. Polimorfisme Statis
2. Polimorfisme Dinamis

Perbedaan keduanya terletak pada cara membuat Polimorfisme. Polimorfisme statis mengginakan *method overloading*, sedangkan Polimorfisme dinamis menggunakan *method overriding*.

1. Abstract Class

Di dalam Java, class yang memiliki method tanpa definisi atau method kosong dapat kita jadikan kelas abstrak. Pendefinisan kelas abstrak biasanya digunakan untuk Polimorfisme di mana *subclass* dari kelas abstrak yang mendefinisikan method kosong dari kelas abstrak. Kelas abstrak tidak bisa di-*instantiate* (di-create menjadi objek), tetapi bisa me-refer objek konkrit yang kelasnnya diturunkan dari dirinya.

1. Abstract Method

Method yang didefinisikan dengan *keyword* **abstract**. Di dalam sebuah kelas abstark, kita dapat membuat *abstract method*. *Abstract method* adalah method yang tidak ada implementasinya. Implementasi dari semua *abstract method* yang dimiliki kelas abstrak dilakukan di kelas konkrit yang diturunkan dari kelas abstrak tersebut. Method abstrak ini hanya dapat dimiliki oleh kelas abstrak. Sebuah kelas konkrit tidak dapat memiliki method abstrak.

**BAB III**

**PENJELASAN**

**GUIDED**

1. Buat tiga kelas yang akan dihubungkan dengan *interface* Notifikasi, yakni EmailNotification, SMSNotification, dan PushNotification, dan buat satu kelas sebagai *main* untuk *running* program.

package notif;

public interface InterfaceNotifikasi {

void sendMessage(String receiver, String content);

}

*Source code interface notifikasi*

package notif;

public class EmailNotification implements InterfaceNotifikasi {

@Override

public void sendMessage(String receiver, String content){

System.out.println("Mengirim eemail ke : " + receiver);

System.out.println("Dengan isi : " + content);

System.out.println();

}

}

*Source code class Email Notification*

package notif;

public class SMSNotification implements InterfaceNotifikasi {

@Override

public void sendMessage(String receiver, String content){

System.out.println("Mengirim SMS ke : " + receiver);

System.out.println("Dengan isi : " + content);

System.out.println();

}

}

*Source code class* *SMS Notification*

*Source code class Push Notification*

package notif;

public class Notif {

public static void main(String[] args) {

String emailPenerima = "yulfisahsan@gmail.com";

String nomerHp = "082137639278";

String mobileId = "IOS 14.2";

EmailNotification emailNotif = new EmailNotification();

SMSNotification smsNotif = new SMSNotification();

PushNotification pushNotif = new PushNotification();

String message = "Ini adalah pesan Ibu";

emailNotif.sendMessage(emailPenerima, message);

smsNotif.sendMessage(nomerHp, message);

pushNotif.sendMessage(mobileId, message);

}

}

package notif;

public class PushNotification implements InterfaceNotifikasi {

@Override

public void sendMessage(String receiver, String content){

System.out.println("Mengirim Pushh Notif ke : " + receiver);

System.out.println("Dengan isi : " + content);

System.out.println();

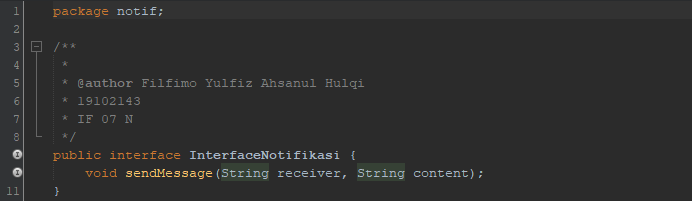
}

}

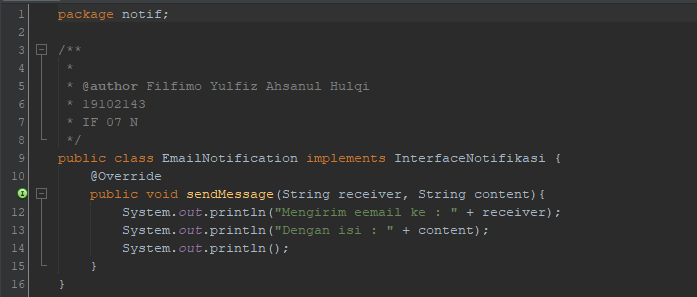
*Source code class* notif untuk *running program*

Screenshot Program :

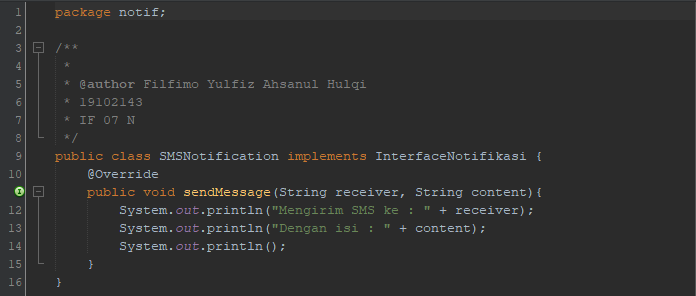
* Interface Notifikasi



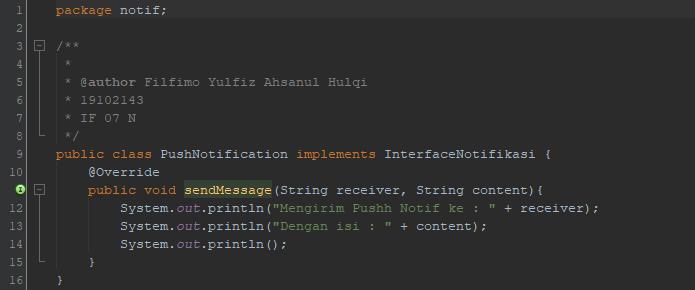
* *Class Email* Notifikasi



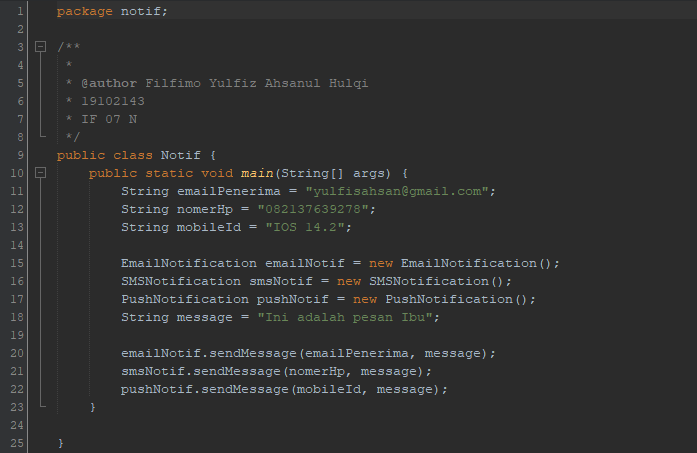
* *Class* *SMS* Notifikasi



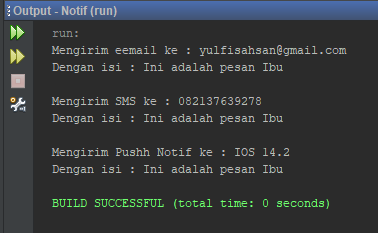
* *Class Push* Notifikasi



* *Main*



* *Output*



Penjelasan Program :

Pada program diatas adalah contoh implementasi dari polimorfisme pada Java, membuat 1 Interface, Interface tersebut nantinya akan diimplementasikan pada kelas lainnya.

*Interface* Notifikasi, hanya berisi satu *method void*.

*Class Email* *Notification*, didalamnya melakukan *override method void* dari *Interface* Notifikasi, dengan isi yang berbeda.

*Class SMS Notification*, didalamnya juga melakukan *override method void* dari *Interface* Notifikasi, dengan isi yang berbeda.

*Class Push Notification*, sama seperti dua kelas sebelumnya didalamnya juga melakukan *override method void* dari *Interface* Notifikasi, dengan isi yang berbeda juga.

Dan yang terakhir adalah *main class* untuk melakukan eksekusi program, didalamnya terdapat pendeklarasian fields yaitu (emailPenerima, nomerHp, mobileId dan message), kemudian terdapat pembuatan objek untuk ketiga *class* tadi. Selanjutnya lakukan pemanggilan *method void sendMessage* dari *interface* Notifikasi, maka akan bisa memunculkan ouput walaupun *method* yang digunakan sama.

1. Buat empat kelas yang akan dihubungkan dengan *Abstract Class* Binatang, yakni Burung, Kucing, Anjing, dan Kambing buat satu kelas sebagai *main* untuk *running* program.

package hewan;

public abstract class Binatang {

private String jenis;

Binatang(String jenis){

this.jenis = jenis;

}

protected abstract void suara();

public String toString(){

return "Seekor " + jenis;

}

}

*Source code Abstract Class* Binatang

package hewan;

public class Burung extends Binatang {

private String nama;

Burung(String nama){

super("Burung");

this.nama = nama;

}

@Override

public void suara(){

System.out.println("Berkicau");

}

@Override

public String toString(){

return super.toString()+" "+nama;

}

}

*Source code class* Burung

package hewan;

public class Kucing extends Binatang {

private String nama;

Kucing(String nama){

super("Kucing");

this.nama = nama;

}

*Source code class* Kucing bagian 1

*Source code class* Kucing bagian 2

package hewan;

/\*\*

\*

\* @author Filfimo Yulfiz Ahsanul Hulqi

\* 19102143

\* IF 07 N

\*/

public class Anjing extends Binatang {

private String nama;

Anjing(String nama){

super("Anjing");

this.nama = nama;

}

@Override

public void suara(){

System.out.println("Menggonggong");

}

@Override

public String toString(){

return super.toString()+" "+nama;

}

}

@Override

public void suara(){

System.out.println("Mengeong");

}

@Override

public String toString(){

return super.toString()+" "+nama;

}

}

*Source code class* Anjing

package hewan;

public class Kambing extends Binatang {

private String nama;

Kambing(String nama){

super("Kambing");

this.nama = nama;}

*Source code class* Kambing bagian 1

*Source code class* Kambing bagian 2

package hewan;

import java.util.Random;

public class CobaPolimorphic {

public static void main(String[] args) {

Binatang[] peliharaanku = {new Burung("Kakak Tua"),

new Kambing("Etawa"), new Anjing("Bulldog"), new Kucing("Anggora")};

Binatang kesayangan;

Random pilihan = new Random();

kesayangan = peliharaanku[pilihan.nextInt(peliharaanku.length)];

System.out.println("Binatang Kesayangan anda : " + kesayangan);

System.out.println("Suaranya : " );

kesayangan.suara();

}

}

@Override

public void suara(){

System.out.println("Mengembik");

}

@Override

public String toString(){

return super.toString()+" "+nama;

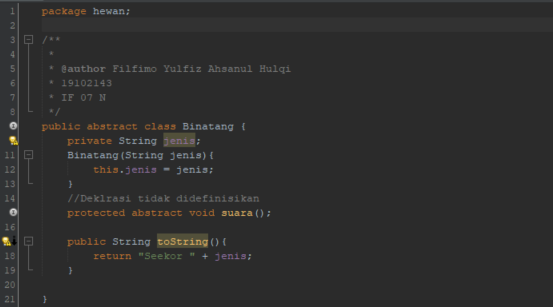
}

}

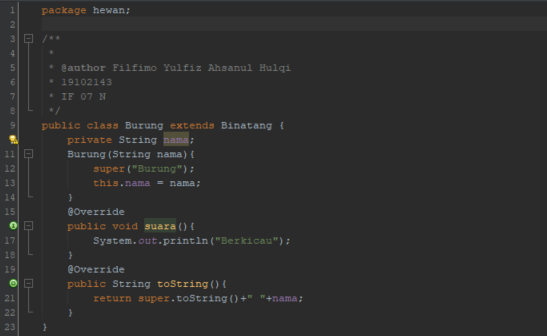
*Source code class* CobaPolimorphic untuk *running program*

Screenshot Program :

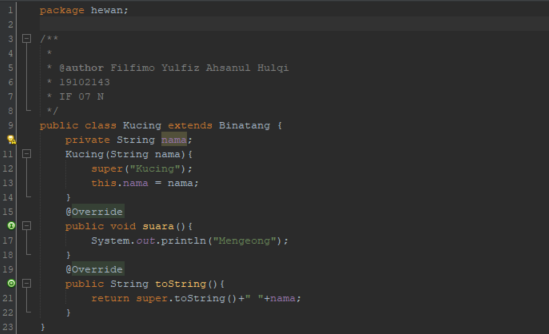
* *Abstract Class* Binatang



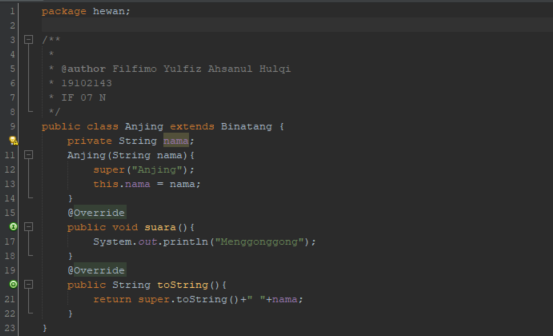
* *Class* Burung



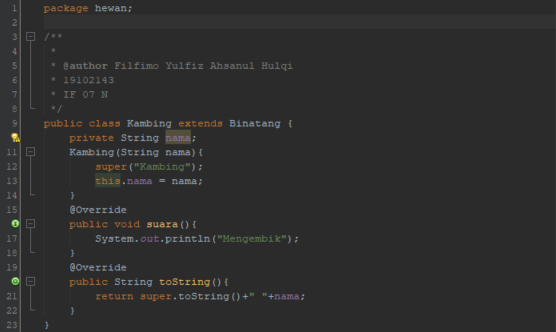
* *Class* Kucing



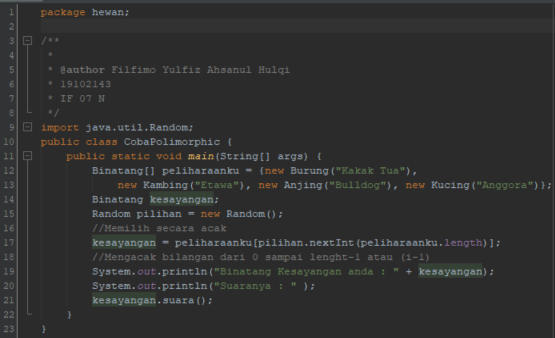
* *Class* Anjing



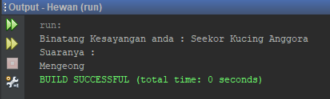
* *Class* Kambing



* *Class* CobaPolimorphic



* *Output*



Penjelasan Program :

Pada program diatas adalah contoh implementasi dari polimorfisme pada Java juga, membuat satu *Abstract Class*, yang didalamnya terdapat *method void* nantinya akan dapat implementasikan pada kelas lainnya.

*Abstract Class* Binatang, hanya berisi konstruktor dan *method void*.

*Class* Burung, didalamnya melakukan *override method void* dari *Abstract Class* Binatang, dengan isi yang berbeda.

*Class Kucing*, didalamnya juga melakukan *override method void* dari *Abstract Class* Binatang, dengan isi yang berbeda.

*Class Anjing*, didalamnya juga melakukan *override method void* dari *Abstract Class* Binatang, dengan isi yang berbeda.

*Class Kambing*, didalamnya juga melakukan *override method void* dari *Abstract Class* Binatang, dengan isi yang berbeda.

Dan yang terakhir adalah *main class* untuk melakukan eksekusi program, didalamnya terdapat *import library* *random,* membuat array dan objek dengan nama kesayangan dan peliharaanku yang berisi *class-class* yang sudah dibuat tadi (Burung, Kucing, Anjing, dan Kambing), kemudian terdapat pengunaan *library random* untuk variabel kesayangan. Selanjutnya lakukan pemanggilan *method void suara* dan *toString* dari *Abstract Class* Binatang, maka akan memunculkan ouput walaupun *method* yang digunakan sama dan hasil output tadi akan acak/random.

**UNGUIDED**

1. Mencoba program Polimorfisme mebuat data Pegawai, yaitu Direktur untuk menampilkan informasi seperti nama, jabatan, gaji, dividen, dan total gaji dengan ketentuan seperti pada modul.

package pegawai;

public abstract class Pegawai {

private String namaPeg;

public Pegawai(String nama){

namaPeg = nama;

}

public String namaPegawai() {

return namaPeg;

}

public abstract double income();

}

*Souce* *code* *class* pegawai

package pegawai;

public final class Direktur extends Pegawai {

private double gajiDirektur;

private double dividenSaham;

public Direktur(String nama, double gaji, double dividen) {

super(nama);

setGajiDirektur(gaji);

setDividen(dividen);

}

public void setGajiDirektur(double gaji){

if(gaji>0)

gajiDirektur = gaji;

else

gajiDirektur = 0;

}

public void setDividen(double dividen){

if(dividen>0)

dividenSaham = dividen;

else

dividenSaham = 0;

}

public String nama(){

return super.namaPegawai();

}

*Souce* *code* *class* direktur bagian 1

*Source code class* direktur bagian 2

package pegawai;

import java.text.DecimalFormat;

public class Test {

public static void main(String args[]){

Pegawai pgw;

String ouput = "";

Direktur d = new Direktur("Udin", 12000000.00, 7000000.00);

DecimalFormat digitPresisi = new DecimalFormat("0.00");

pgw = d;

System.out.println("\nDEMO INHERITANS, ENKAPSULASI, POLIMORFI");

System.out.println("\n=======================================");

System.out.println("Nama : " + d.namaPegawai() + "\n" + "Jabatan : "

+ d.jabatan() + "\n" + "Gaji : "

+ digitPresisi.format(d.gajiPerbulan()) + "\n" + "Dividen : "

+ digitPresisi.format(d.labaDividen()) + "\n" + "Total : "

+ digitPresisi.format(d.income()) + "\n");

System.exit(0);

}

}

public String jabatan(){

return "Direktur";

}

public double gajiPerbulan(){

return gajiDirektur;

}

public double labaDividen(){

return dividenSaham;

}

@Override

public double income(){

return(gajiDirektur+dividenSaham);

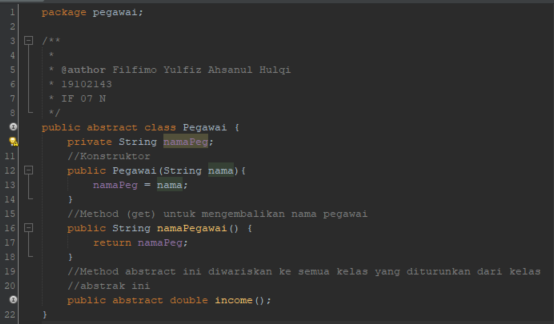
}

}

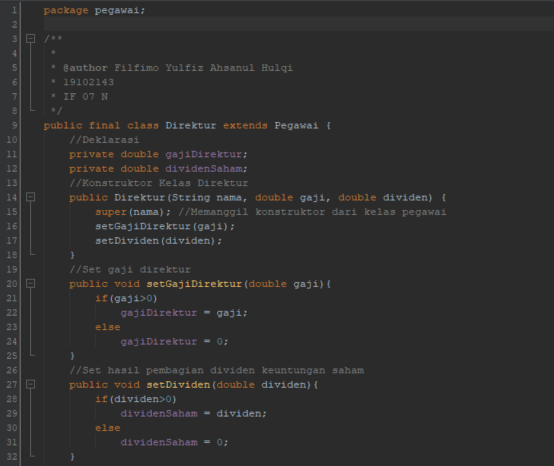
*Source code class test* untuk *running program*

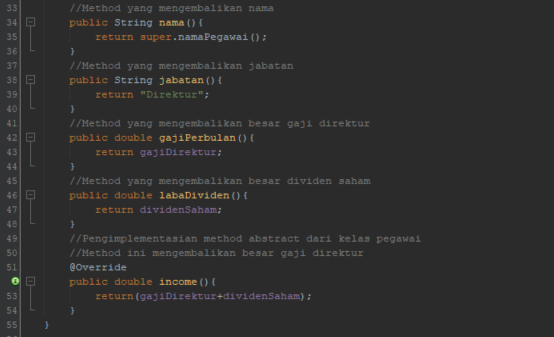
Screenshot Program

* *Class* Pegawai

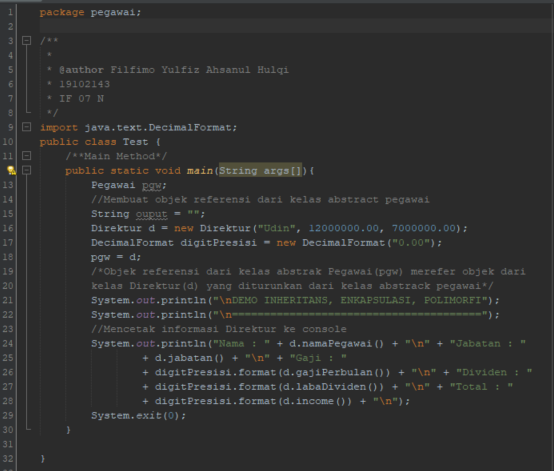


* *Class* Direktur

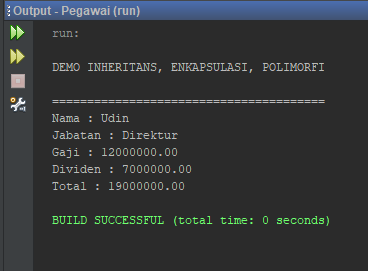




* *Class* Test



* *Ouput*



Penjelasan Program :

Pada program diatas pertama membuat satu *Abstract Class* yaitu Pegawai, yang didalamnya terdapat *method get* dan *method void* nantinya akan dapat implementasikan pada kelas lainnya. Selanjutnya buat satu *class* yaitu Direktur, serta satu class untuk *main* *running* program

*Abstract Class* Pegawai, hanya berisi konstruktor, *method get* untuk nama dan *method abstract* untuk menghitung *income*.

*Class* Direktur, didalamnya terdapat deklarasi *fields* yaitu gajiDirektur, dan dividenSaham, selanjutnya pembuatan konstruktor dengan parameter, tidak lupa untuk meemberikan keyword *super* untuk dapat mengakses *super class*, kemudian membuat *void* untuk *set* gaji, dan *set* dividen, selanjutnya membuat *method* untuk mengembalikan nilai nama dan jabatan, kemudian buat *method* untuk mengembalikan nilai besar gaji dan dividen saham dan yang terakhir adalah implementasi Polimorfisme dengan melakukan *overriding* pada *method abstract* *income* dari *abstract class* Pegawai.

Dan yang terakhir adalah *main class* untuk melakukan eksekusi program, didalamnya ada *import library decimal format* untuk mendapatkan angka yang presisi, disini digunakan untuk *method* gajiPerbulan, labaDividen, dan income(), Kemudian membuat objek referensi dari kelas *abstract* Pegawai, objek referensi dari kelas *abstract* Employee merefer objek dari kelas Direktur yang diturunkan dari kelas *abstract* Pegawai, kemudian akan mencetak informasi dari Direktur, dengan pemanggilan dari masing-masing *method*.

1. Buatlah program untuk sebuah perusahaan yang didalamnya terdapat banyak *employess*, diantaranya *Salaried Employee*, *Comission Employee*, dan *Project Planner*, dengan ketentuan seperti pada modul dan juga menggunakan prinsip-prinsip Polimorfisme.

package employee;

public abstract class Employee {

protected String namaPeg, nipPeg;

public Employee(String nama, String nip) {

namaPeg = nama;

nipPeg = nip;

}

public String namaPegawai(){

return namaPeg;

}

public String nipPegawai(){

return nipPeg;

}

public abstract void cetakInformasi();

public abstract double hitungGaji();

}

*Source code Interface Employee*

package employee;

public class Salaried extends Employee {

public double upahMingguan;

public Salaried(String nama, String nip, double gaji) {

super(nama, nip);

setupahMingguan(gaji);

}

private void setupahMingguan(double gaji){

if(gaji>0)

upahMingguan = gaji;

else

upahMingguan = 0;

}

public String nama(){

return super.namaPegawai();

}

public String nip(){

return super.nipPegawai();

}

*Source code Class Salaried* bagian 1

*Source code Class Salaried* bagian 2

package employee;

private double gajiPokokC, komisiC;

private int totalPenjualanC;

public Comission(String nama, String nip, double gaji, double komisi, int totalPenjualan) {

super(nama, nip);

setGajiPokok(gaji);

setKomisi(komisi);

setTotalPenjualan(totalPenjualan);

}

public void setGajiPokok(double gaji){

if(gaji>0)

gajiPokokC = gaji;

else

gajiPokokC = 0;

}

public void setKomisi(double komisi){

if(komisi>0)

komisiC = komisi;

else

komisiC = 0;

}

//Implementasi method abstract

@Override

public double hitungGaji(){

return upahMingguan;

}

@Override

public void cetakInformasi(){

System.out.println("===== SALARIED EMPLOYEE =====");

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

System.out.println("NIP : " + nip());

System.out.println("Gaji : " + hitungGaji());

}

}

*Source code Class Comission* bagian 1

*Source code Class Comission* bagian 2

package employee;

public class Project extends Employee {

private double gajiPokokP, komisiP, pajak;

private int totalHasilProyekP;

public Project(String nama, String nip, double gaji, double komisi, int totalHasilProyek) {

super(nama, nip);

setGajiPokok(gaji);

setKomisi(komisi);

setTotalHasilProyek(totalHasilProyek);

}

public void setGajiPokok(double gaji){

if(gaji>0)

gajiPokokP = gaji;

else

gajiPokokP = 0;

}

private void setTotalPenjualan(int totalPenjualan){

if(totalPenjualan>0)

totalPenjualanC = totalPenjualan;

else

totalPenjualanC = 0;

}

public String nama(){

return super.namaPegawai();

}

public String nip(){

return super.nipPegawai();

}

@Override

public double hitungGaji(){

return (gajiPokokC + (komisiC\*totalPenjualanC));

}

@Override

public void cetakInformasi(){

System.out.println("===== COMISSION EMPLOYEE =====");

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

System.out.println("NIP : " + nip());

System.out.println("Gaji : " + hitungGaji());

}

}

*Source code Class Project* bagian 1

*Source code Class Project* bagian 2

package employee;

public class Run {

public static void main(String args[]){

System.out.println("DATA PEGAWAI PERUSAHAAN PBO ");

System.out.println();

Employee emp;

Salaried s = new Salaried("Filfimo", "19102143", 253725.32);

emp = s;

s.cetakInformasi();

System.out.println();

public void setKomisi(double komisi){

if(komisi>0)

komisiP = komisi;

else

komisiP = 0;

}

private void setTotalHasilProyek(int totalHasilProyek){

if(totalHasilProyek>0)

totalHasilProyekP = totalHasilProyek;

else

totalHasilProyekP = 0;

}

public String nama(){

return super.namaPegawai();

}

public String nip(){

return super.nipPegawai();

}

@Override

public double hitungGaji(){

pajak = gajiPokokP \* 0.5;

return (gajiPokokP + (komisiP\*totalHasilProyekP) - pajak);

}

@Override

public void cetakInformasi(){

System.out.println("===== PROJECT PLANNER =====");

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

System.out.println("NIP : " + nip());

System.out.println("Gaji : " + hitungGaji());

}

}

*Source code Class Run* bagian 1

*Sourcee code Class Run* bagian 2

Comission c = new Comission("Yulfiz", "34120191", 153856.14, 103618.02, 7);

emp = c;

c.cetakInformasi();

System.out.println();

Project p = new Project("Ahsanul", "IF07N", 185630.76, 195621.32, 5);

emp = p;

p.cetakInformasi();

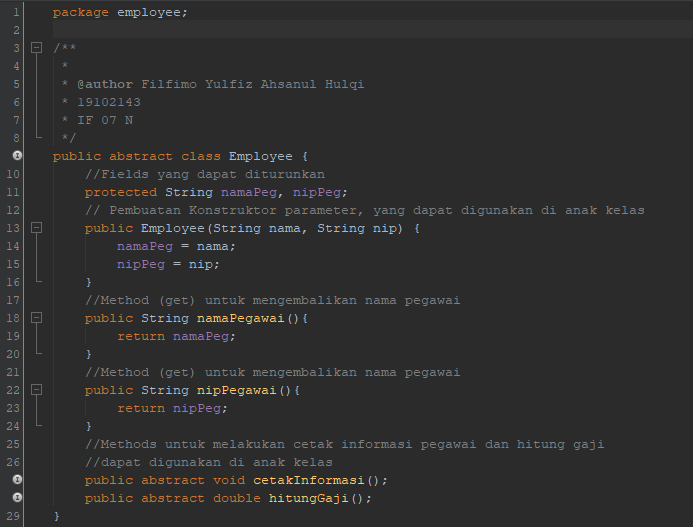
System.exit(0);

}

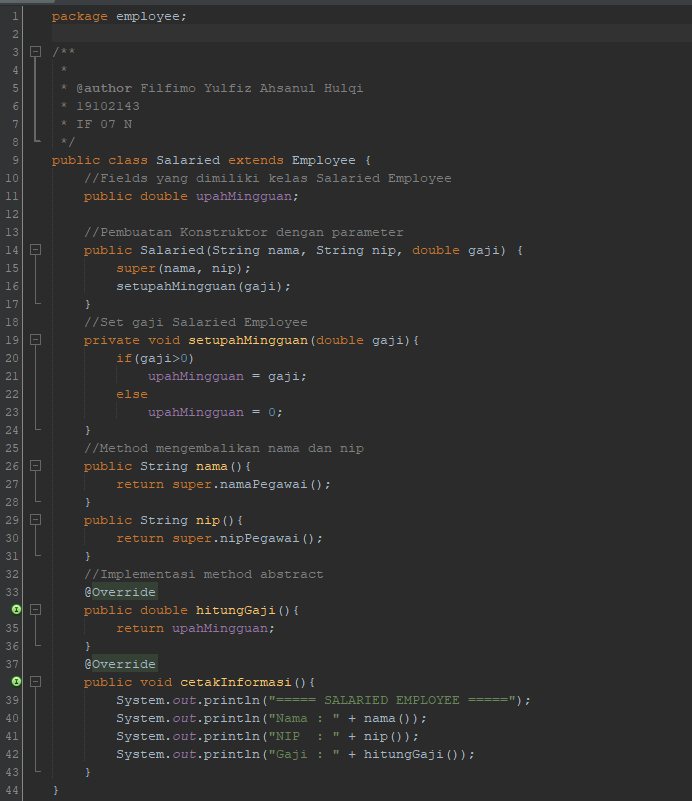
}

Screenshot Program :

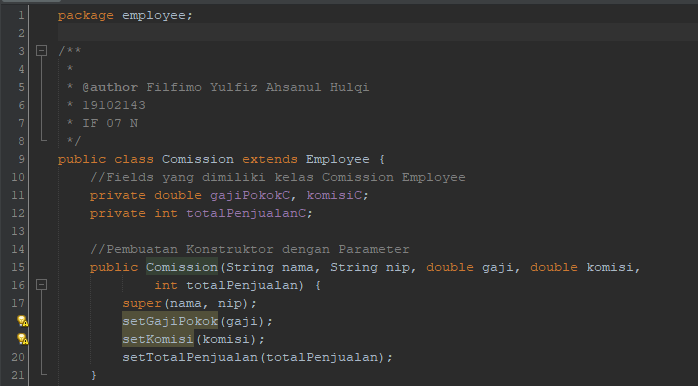
* *Abstract Employee*

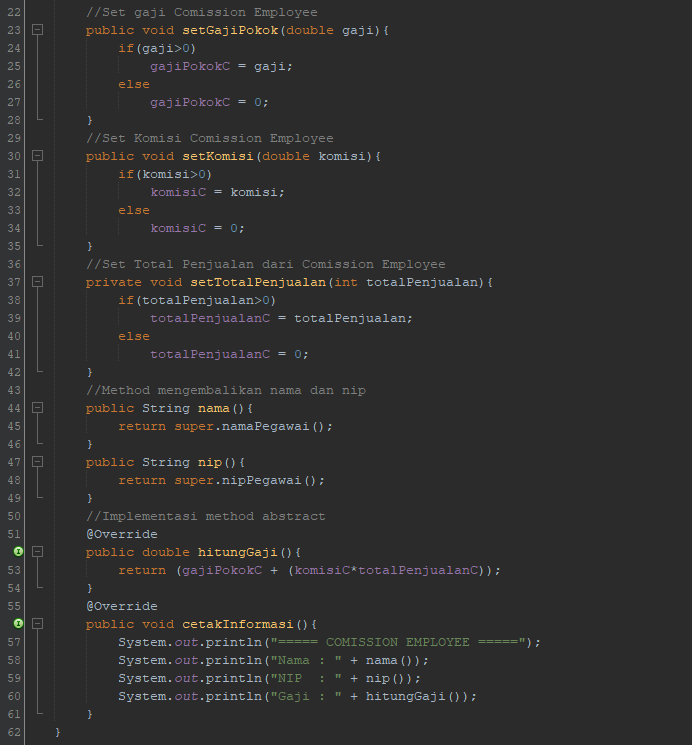


* *Class Salaried*

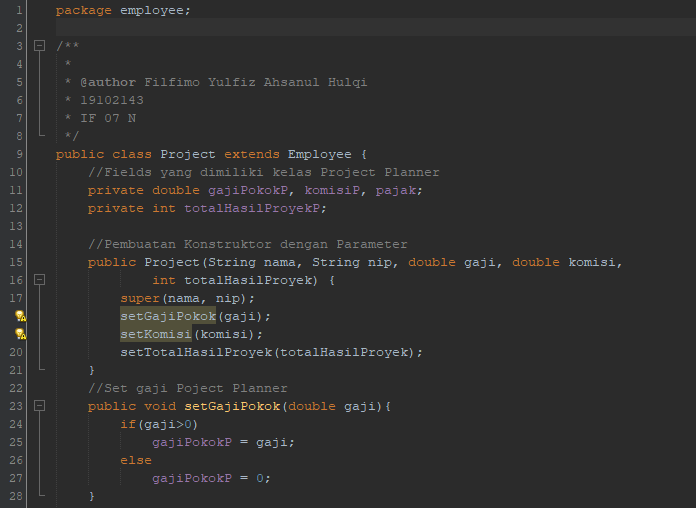


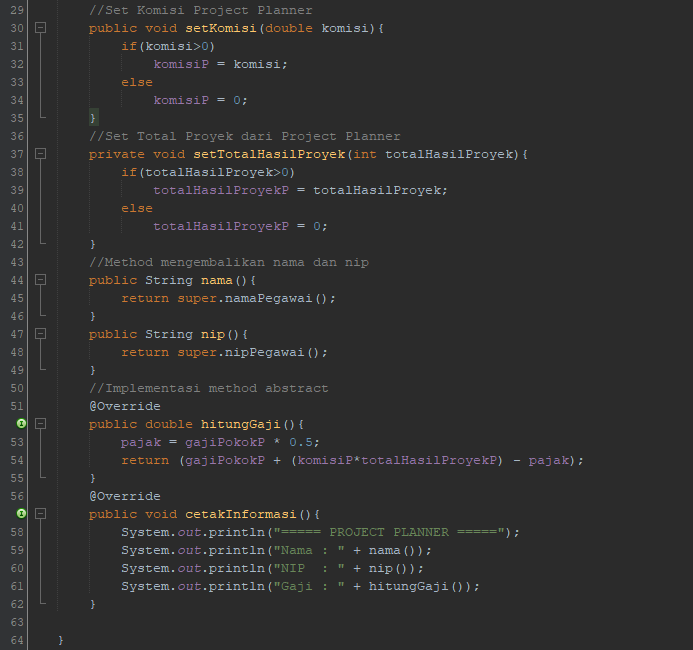
* *Class Comission*



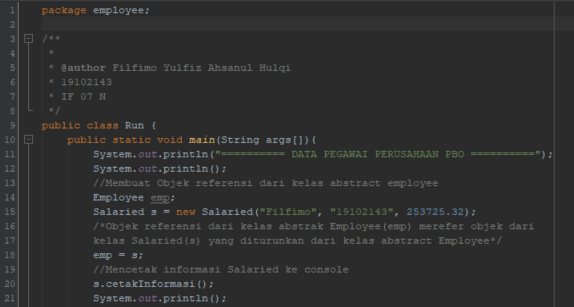


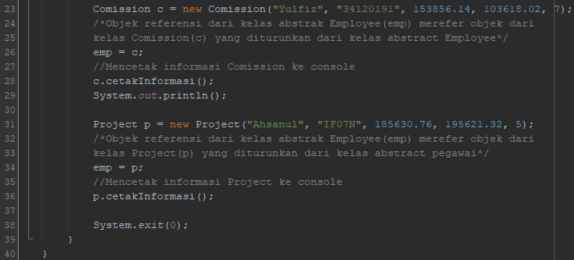
* *Class Project*



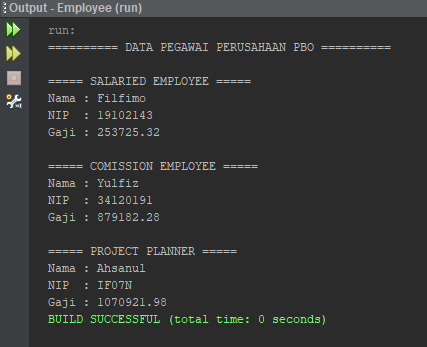


* *Class Run*





* *Output*



Penjelasan Program :

Pada program diatas pertama membuat satu *Abstract Class* yaitu *Employee*, yang didalamnya terdapat *method get* dan *method void* nantinya akan dapat implementasikan pada kelas lainnya. Selanjutnya buat tiga *class* lain yaitu *Salaried*, *Comission*, dan *Project* serta satu class untuk *main* *running* program

*Abstract Class Employee*, hanya berisi konstruktor, *method get* untuk nama dan nip dan *method void* dan *abstract* untuk menghitung gaji dan cetak informasi pegawai.

*Class Salaried*, didalamnya terdapat deklarasi *fields* yaitu upahMingguan, selanjutnya pembuatan konstruktor dengan parameter, tidak lupa untuk meemberikan keyword *super* untuk dapat mengakses *super class*, kemudian membuat *void* untuk *set* upah, membuat *method* untuk mengembalikan nilai nama dan nip, dan yang terakhir adalah implementasi Polimorfisme dengan melakukan *overriding* pada *method abstract* hitungGaji dan cetakInformasi dari *abstract class Employee*.

*Class Comission*, didalamnya terdapat deklarasi *fields* yaitu gajiPokoC, komisiC, dan totalPenjualanC, selanjutnya pembuatan konstruktor dengan parameter, tidak lupa untuk memberikan keyword *super* untuk mengakses *super class*, kemudian membuat *void* untuk set gajiPokok, set komisi, dan set totalPenjualan, selanjutnya membuat *method* untuk mengembalikan nilai nama dan nip, dan yang terakhir adalah implementasi Polimorfisme dengan melakukan *overriding* pada *method abstract* hitungGaji dan cetakInformasi darii *abstract class Employee.*

*Class Project*, sama seperti kedua kelas sebelumnya, didalamnya terdapat deklarasi *fields* yaitu gajiPokokP, komisiP, pajak, dan totalHasilProyekP, selanjutnya pembuatan konstruktor dengan parameter, membuat *void* untuk set gajiPokok, set komis, dan set totalHasilProyek, selanjutnya membuat *method* untuk mengembalikan nilai nama dan nip, dan yang terakhir adalah implementasi Polimorfisme dengan melakukan *overriding* pada *method abstract* hitungGaji dan cetakInformasi darii *abstract class Employee.*

Dan yang terakhir adalah *main class* untuk melakukan eksekusi program, didalamnya membuat objek referensi dari kelas *abstract* Employee, kemudian objek referensi dari kelas *abstract* Employee merefer objek dari kelas Salaried/Comission/Project yang diturunkan dari kelas *abstract* Employee, kemudian akan mencetak informasi dari masing-masing pegawai, dengan pemanggilan dari *method* milik *abstract class*.

**BAB IV**

**Kesimpulan**

Pada Praktikum kali ini dapat diambil beberapa kesimpulan yaitu :

1. Dengan Polimorfisme, dapat menghidari duplikasi *object*.
2. Membuat *code* lebih simpel, karena tidak perlu menulis ulang *method* baru, karena dapat menggunakan *method* yang sudah dibuat sebelumnya dan melakukan *overriding* pada *method* tersebut.
3. Walaupun semua isi *method* akan sama, namun nanti pada implementasinya akan berbeda-beda, dan dapat digunakan dalam *method* yang sama