**LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL IV**

**PEWARISAN (*INHERITANCE)***

****

**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. Memahami tentang konsep pewarisan tunggal (*Inheritance*).
2. Mampu mewujudkan bentuk pewarisan tunggal dalam bahasa Java.

**BAB II**

**Dasar Teori**

1. Pewarisan Tunggal

Konsep yang mendasari *Inheritance* adalah *Generalization*. *Generalization* digambarkan sebagai hubungan dari *subclass* ke *supeclass*, sedangkan *Inheritance*

adalah proses pewarisan data dan *methods* dari *superclass* ke *subclass*.

Pewarisan (*Inheritance*) merupakan proses pembentukan kelas baru dari kelas yanh sudah ada (*Reusability*). Kelas yang mewariskan disebut sebagai *Super Class* (Kelas Induk), sedangkan kelas yang mendapat pewarisan disebut *Sub Class* (Kelas Anak). Pewarisan ini bersifat menyeluruh, sehingga semua data dan *methods* yang dimiliki kelas induk akan seluruhnya diturunkan kepada kelas anak.

1. Konstruktor

Konstruktor kelas dasar tidak diwariskan kepada kelas turunan. Kelas turunan menggunakan konstruktor sendiri baik *deffault constructor* maupun membuat konstruktor pada kelas turunan itu sendiri. Inisialisasi atribut kelas dasar dilakukan dengan menggunakan konstruktor kelas dasar yang dipanggil secara eksplisit dalam konstruktor kelas turunan.

Kelas turunan dapat mengakses setiap *public member* kelas dasar, kelas lain juga dapat mengakses member kelas dasar secara langsung. Kelas turunan tidak dapat mengakses *private member* kelas dasar, kelas lain juga tidak dapat mengakses member kelas dasar secara langung. Kelas turunan dapat mengakses setiap *protected member* kelas dasar, tetapi kelas lain tidak dapat mengakse kelas dasar secara langsung.

1. *Overriding* (Redefinisi)

*Overriding* atau disebut juga redefinisi adalah kemamuan suatu kelas anak untuk melakukan modifikasi data dan *methods* dari kelas induknya. Proses ini akan mengubah data *methods* dari keduanya, kelas induk dan kelas anaknya.

**BAB III**

**PENJELASAN**

**GUIDED**

1. Buatlah program sederhana tentang *inheritance*, seperti pada modul dengan nama kelas yang berbeda dan *methods* yang berbeda.

*Source code* *class* induk

package binatang;

public class Anjing extends Binatang {

public void anjing(){

System.out.println("Anjing Pitbull");

}

public void kesukaanAnjing(){

System.out.println(nama + " suka Tulang");

}

}

package binatang;

public class Binatang {

protected String nama;

public void makan(){

System.out.println(nama + " makan");

}

System.out.println(nama + " minum");

}

}

*Source code class* anjing sebagai anak

*Source code class* kucing sebagai anak

package binatang;

public class Kucing extends Binatang {

public void kucing(){

System.out.println("Kucing Persia");

}

public void kesukaanKucing(){

System.out.println(nama + " suka Whiskas");

}

}

package binatang;

public class Hamster extends Binatang {

public void hasmter(){

System.out.println("Hamster Campbell");

}

public void kesukaanHamster(){

System.out.println(nama + " suka Biji");

}

}

*Source code class* hamster sebagai anak

package binatang;

public class MainBinatang {

public static void main(String[] args) {

Anjing a = new Anjing();

Kucing b = new Kucing();

Hamster c = new Hamster();

a.nama = "Anjing";

b.nama = "Kucing";

c.nama = "Hamster";

*Source code main class* bagian 1

*Source* *code* *main* *class* bagian 2

System.out.println("===== Hewan Favorit Filfimo =====");

System.out.println();

a.anjing();

a.makan();

a.kesukaanAnjing();

a.minum();

System.out.println();

b.kucing();

b.makan();

b.kesukaanKucing();

b.minum();

System.out.println();

c.hasmter();

c.makan();

c.kesukaanHamster();

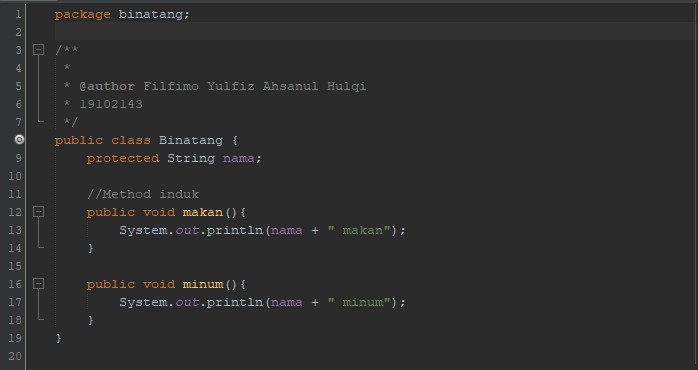
c.minum();

}

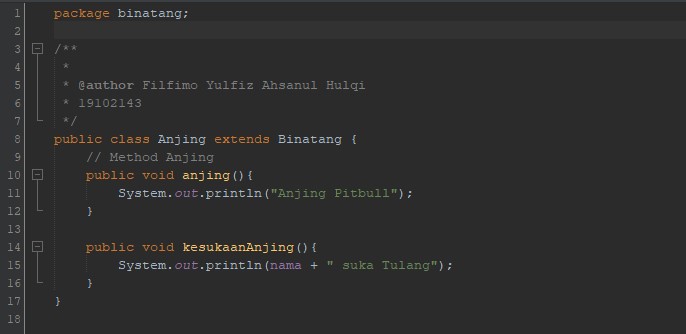
}

Screenshot program :

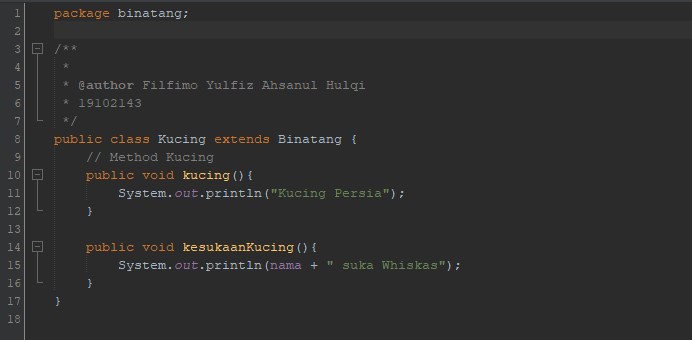
* *Class* Induk



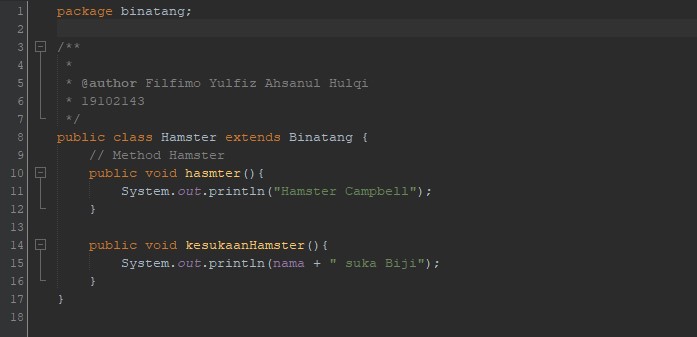
* *Class* Anak (Anjing)



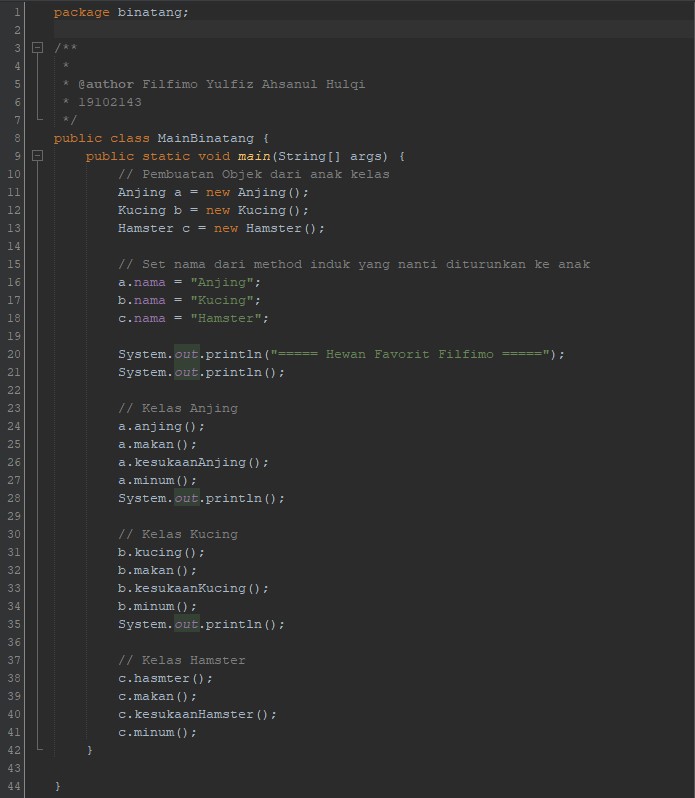
* *Class* Anak (Kucing)



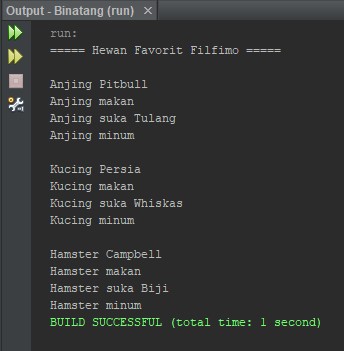
* *Class* Anak (Hamster)



* *Main* *Class*



* Output



Penjelasan Program :

Pada program diatas, adalah program implementasi *inheritance* mirip seperti pada modul, objek pada modul yaitu cong, kun, yu, dan sund, diganti menjadi anjing, kucing, dan hamster.

Objek-objek tersebut dapa mengakses atribut nama dan method makan(), minum() dari kelas binatang, Karena objek-objek tersebut mendapatkan pewarisan dari kelas induknya yaitu kelas binatang, sehingga seluruh *fields*, dan *methods* yang berada pada kelas induk atau kelas binatang, dapat digunakan pada kelas anak yang mendapatkan pewarisan dari kelas induk.

Kelas anak tidak selalu menggunakan *fields* dan *methods* dari kelas induk, kelas anak juga dapat menggunakan *fields* dan *methods* sendiri, pada program diatas, masing-masing kelas mempunyai *methods* sendiri yaitu *methods* kesukaanyang tidak ada pada kelas induk.

Pada program diatas, binatang mempunyai *fields* nama dengan aksesibiltas *protected*, yang artinya akses diberikan kepada kelas itu sendiri dan kelas anak yang diturunkan darinya. Sehingga kelas anak dapat mengakses *fields* nama tersebut. Dan juga *methods* makan() dan minum().

1. Buat program pewarisan tentang pegawai, memiliki tiga kelas anak yaitu pegawai tetap, pegawai tidak tetap, dan pegawai magang, untuk ketentuan sama seperti pada modul.

package karyawan;

public class Karyawan {

protected String NIP, nama, jenisKelamin;

protected int gajiPokok;

public Karyawan(String NIP, String nama, String jenisKelamin, int gajiPokok){

this.NIP = NIP;

this.nama = nama;

this.jenisKelamin = jenisKelamin;

this.gajiPokok = gajiPokok;

}

public void cetakKaryawan(){

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

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

System.out.println("Jenis Kelamin : " +this.jenisKelamin);

System.out.println("Gaji Pokok : " +this.gajiPokok);

}

}

*Source* *code* kelas karyawan sebagai kelas induk

package karyawan;

public class KaryawanTetap extends Karyawan{

public String pengangkatan;

public String jabatan;

public int tunjanganJabatan;

public KaryawanTetap(String pengangkatan, String jabatan, int tunjanganJabatan, String NIP, String nama, String jenisKelamin, int gajiPokok){

super(NIP, nama, jenisKelamin, gajiPokok);

this.pengangkatan = pengangkatan;

this.jabatan = jabatan;

*Source* *code* kelas karyawan tetap sebagai anak bagian 1

*Source* *code* kelas karyawan tetap sebagai anak bagian 2

this.tunjanganJabatan = tunjanganJabatan;

}

public void cetakKaryawanTetap(){

System.out.println("========== KARYAWAN TETAP =========");

cetakKaryawan();

System.out.println("Pengangkatan : " +this.pengangkatan);

System.out.println("Jabatan : " +this.jabatan);

System.out.println("Tunjangan : " +this.tunjanganJabatan);

System.out.println();

}

}

package karyawan;

public class KaryawanTakTetap extends Karyawan{

public String masukKerja;

public int bonus;

public KaryawanTakTetap(String masukKerja, int bonus, String NIP, String nama, String jenisKelamin, int gajiPokok){

super(NIP, nama, jenisKelamin, gajiPokok);

this.masukKerja = masukKerja;

this.bonus = bonus;

}

public void cetakKaryawanTakTetap(){

System.out.println("========== KARYAWAN TIDAK TETAP =========");

cetakKaryawan();

System.out.println("Tahun Masuk : " +this.masukKerja);

System.out.println("Bonus : " +this.bonus);

System.out.println();

}

}

*Source* *code* kelas karyawan tidak tetap sebagai anak

*Source* *code* kelas magang sebagai anak

package karyawan;

public class Magang extends Karyawan{

public String masukMagang;

public String lamaMagang;

public int bonus;

public Magang(String masukMagang, String lamaMagang, int bonus, String NIP, String nama, String jenisKelamin, int gajiPokok){

super(NIP, nama, jenisKelamin, gajiPokok);

this.masukMagang = masukMagang;

this.lamaMagang = lamaMagang;

this.bonus = bonus;

}

public void cetakMagang(){

System.out.println("========= MAGANG ========");

cetakKaryawan();

System.out.println("Mulai Magang : " +this.masukMagang);

System.out.println("Masa Magang : " +this.lamaMagang);

System.out.println("Bonus : " +this.bonus);

System.out.println();

}

}

package karyawan;

public class MainKaryawan{

public static void main(String[] args) {

KaryawanTetap a = new KaryawanTetap(“ISI PARAMATER TETAP”);

KaryawanTakTetap b = new KaryawanTakTetap(“ISI PARAMATER TIDAK TETAP”);

Magang c = new Magang("ISI PARAMETER MAGANG);

System.out.println("\*\*\*\*\*\*\*\*\*\* DISPLAY DATA KARYAWAN \*\*\*\*\*\*\*\*\*\*\*");

a.cetakKaryawanTetap();

b.cetakKaryawanTakTetap();

c.cetakMagang();

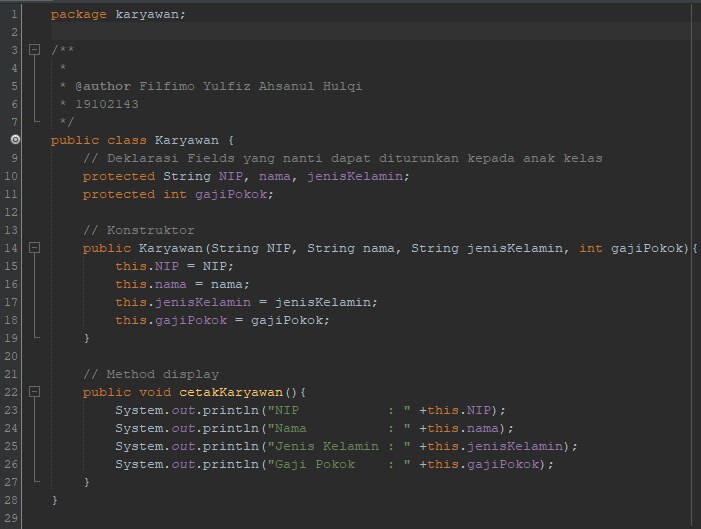
}

}

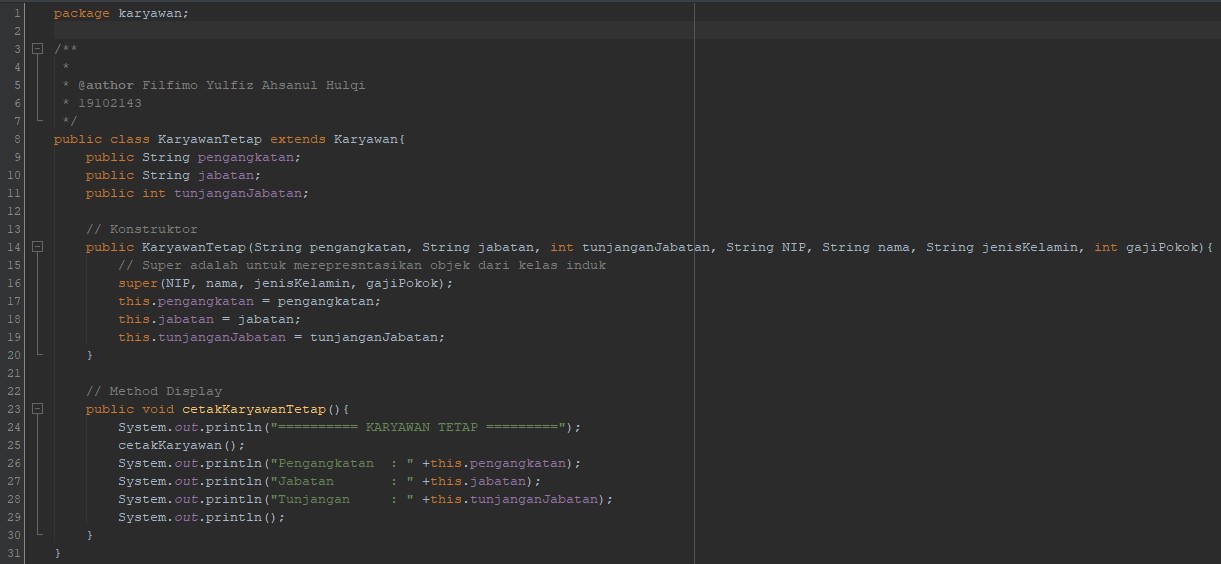
*Source* *code* *main* *class*

Screenshot Program :

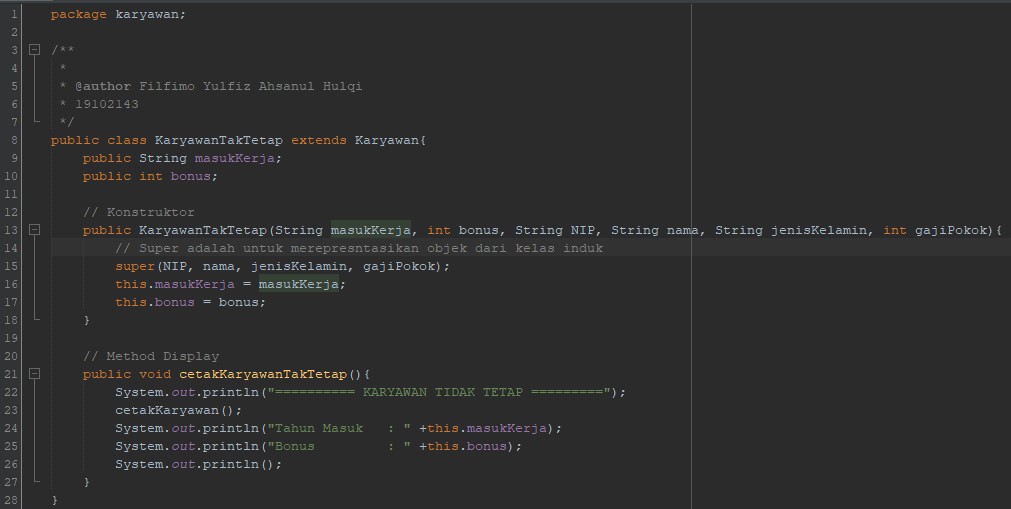
* *Class* Induk



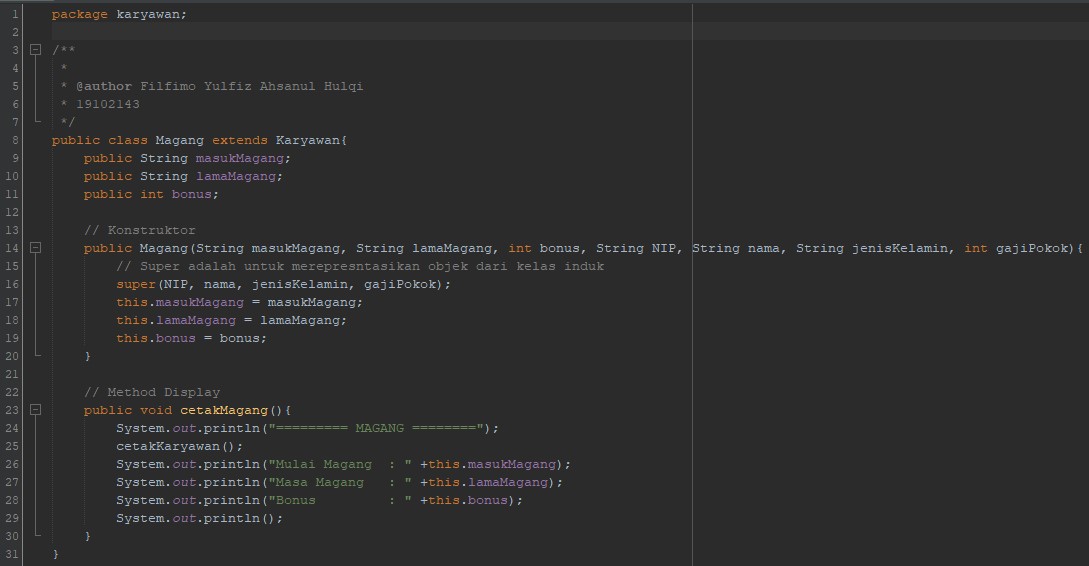
* *Class* Anak (Karyawan tetap)



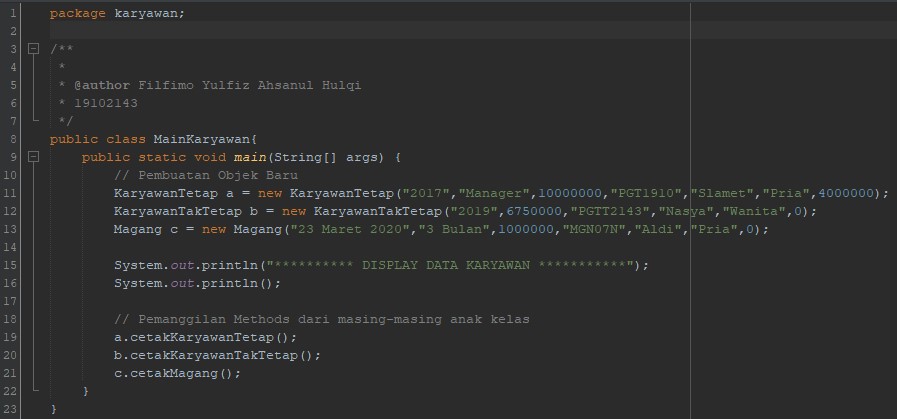
* *Class* Anak (Karyawan tidak tetap)



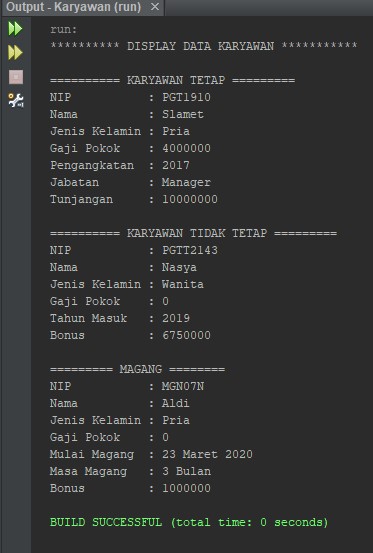
* *Class* Anak (Magang)



* *Main* *Class*



* Ouput



Penjelasan Program :

Pada program diatas merupakan salah satu contoh implementasi pewarisan (*inheritance*), mempunyai tiga anak kelas yaitu karyawan tetap, karyawan tidak tetap,

dan magang, mempunayai kelas induk yaitu karyawan.

Pada kelas induk memiliki *fields* dan *methods* yang dapat diturunkan kepada anak kelas, juga mempunyai konstruktor dengan paramater, konstuktor ini juga dapat digunakan pada anak kelas.

Pada kelas anak karyawan tetap, *fields* yang dimiliki adalah pengangkatan, jabatan, dan tunjangan, selanjutnya *fields* tersebut dibuat konstruktor, ada tambahan atribut *super* di dalam konstruktor, atribut *super* tersebut merujuk pada kelas induk. Kemudian yang terakhir adalah *methods* untuk menampilkan karyawan tetap, *methods* milik kelas induk dapat digunakan juga pada anak kelas.

Pada kelas anak karyawan tidak tetap, *fields* yang dimiliki adalah masuk kerja, dan bonus, selanjutnya *fields* tersebut dibuat konstruktor, memiliki atribut *super* di dalamnya. Kemudian yang terakhir *methods* untuk menampilkan karyawan tidak tetap.

Pada kelas anak magang, sama seperti dua kelas anak sebelumnya, *fields* yang dimiliki adalah masuk magang, lama magang, dan bonus, selanjutnya *fields* tersebut dubuat konstruktor, memiliki atribut *super* di dalamnya. Kemudian yang terakhir *methods* untuk menampilkan karyawan magang.

Setelah semua kelas sudah selesai dibuat, selanjutnya adalah melakukan pembuatan objek dari masing-masing kelas di *main class*, objek yang dibuat harus memiliki paramater mengikuti dengan konstruktor. Lalu lakukan pemanggilan *methods* dari masing-masing anak kelas.

**UNGUIDED**

Buatalah program untuk perusahaan asuransi yang memiliki banyak pegawai, antara lain Sales, Satpan, dan Manajer, dengan spesifikasi sesuai dengan modul. Buatlah sebuah Kelas Pegawai yang menjadi induk Kelas Satpam, Kelas Sales, dan Kelas Manajer, lalu implementasikan konsep pewarisan (*inheritance*) untuk menghitung gaji akhir pegawai melalui method **HitungGajiAkhir()** dan menampilkkan informasi data pegawai perusahaan Asuransi tersebut pada tahun 2021.

package asuransi;

public class Pegawai {

protected String NIP, nama, alamat;

protected int gajiPokok, masukKerja;

public Pegawai(String NIP, String nama, String alamat, int gajiPokok, int masukKerja) {

this.NIP = NIP;

this.nama = nama;

this.alamat = alamat;

this.gajiPokok = gajiPokok;

this.masukKerja = masukKerja;

}

public void cetakPegawai(){

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

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

System.out.println("Jenis Kelamin : " +this.alamat);

System.out.println("Masuk Kerja : " +this.masukKerja);

System.out.println("Gaji Pokok : " +this.gajiPokok);

}

}

*Source code* kelas pegawai sebagai induk

*Source* *code* kelas manajer sebagai anak

package asuransi;

public class Manajer extends Pegawai {

public String divisi;

public int tunjangan, gajiAkhirManajer;

public Manajer(String divisi, String NIP, String nama, String alamat, int gajiPokok, int masukKerja) {

super(NIP, nama, alamat, gajiPokok, masukKerja);

this.divisi = divisi;

}

private void gajiAkhirManajer(){

int masaJabatan;

masaJabatan = 2021 - masukKerja;

if(masaJabatan > 3){

tunjangan = tunjangan + (gajiPokok \* 5/100);

gajiAkhirManajer = gajiPokok + tunjangan;

} else if (masaJabatan > 5){

tunjangan = tunjangan + (gajiPokok \* 10/100);

gajiAkhirManajer = gajiPokok + tunjangan;

}

System.out.println("Gaji Akhir : " +this.gajiAkhirManajer);

}

public void cetakManajer(){

System.out.println("========== DATA MANAJER =========");

cetakPegawai();

System.out.println("Divisi : " +this.divisi);

gajiAkhirManajer();

System.out.println();

}

}

*Source* *code* kelas sales sebagai anak

package asuransi;

public class Sales extends Pegawai {

public int jumlahPelanggan, komisi, gajiAkhirSales;

public Sales(int jumlahPelanggan, String NIP, String nama, String alamat, int gajiPokok, int masukKerja) {

super(NIP, nama, alamat, gajiPokok, masukKerja);

this.jumlahPelanggan = jumlahPelanggan;

}

private void gajiAkhirSales(){

komisi = this.jumlahPelanggan \* 50000;

gajiAkhirSales = komisi + gajiPokok;

System.out.println("Gaji Akhir : " +gajiAkhirSales);

}

public void cetakSales(){

System.out.println("========== DATA SALES =========");

cetakPegawai();

System.out.println("Pelanggan : " +this.jumlahPelanggan);

gajiAkhirSales();

System.out.println();

}

}

package asuransi;

public class Satpam extends Pegawai {

public int jamLembur, bonus, gajiAkhirSatpam;

public Satpam(int jamLembur, String NIP, String nama, String alamat, int gajiPokok, int masukKerja) {

super(NIP, nama, alamat, gajiPokok, masukKerja);

this.jamLembur = jamLembur;

}

public void gajiAkhirSatpam(){

bonus = this.jamLembur \* 10000;

gajiAkhirSatpam = bonus + gajiPokok;

System.out.println("Gaji Akhir : " +gajiAkhirSatpam);

}

*Source* *code* kelas satpam sebagai anak bagian 1

*Source* *code* kelas satpam sebagai anak bagian 2

public void cetakSatpam(){

System.out.println("========== DATA SATPAM =========");

cetakPegawai();

System.out.println("Jam Lembur : " +this.jamLembur +" Jam");

gajiAkhirSatpam();

System.out.println();

}

}

package asuransi;

public class MainAsuransi {

public static void main(String[] args) {

Manajer a = new Manajer (“MASUKAN PARAMETER KONSTRUKTOR”);

Sales b = new Sales (“MASUKAN PARAMETER KONSTRUKTOR”);

Satpam c = new Satpam (“MASUKAN PARAMETER KONSTRUKTOR”);

System.out.println("\n=== DISPLAY DATA KARYAWAN ===");

System.out.println();

a.cetakManajer();

b.cetakSales();

c.cetakSatpam();

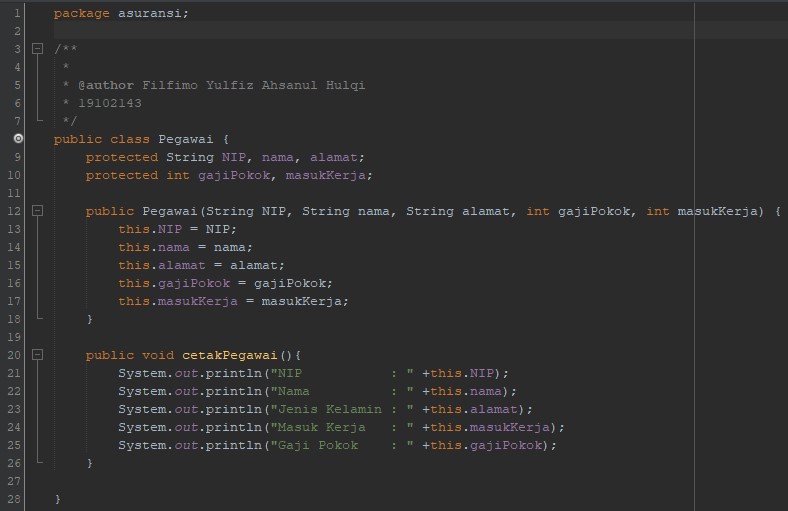
}

}

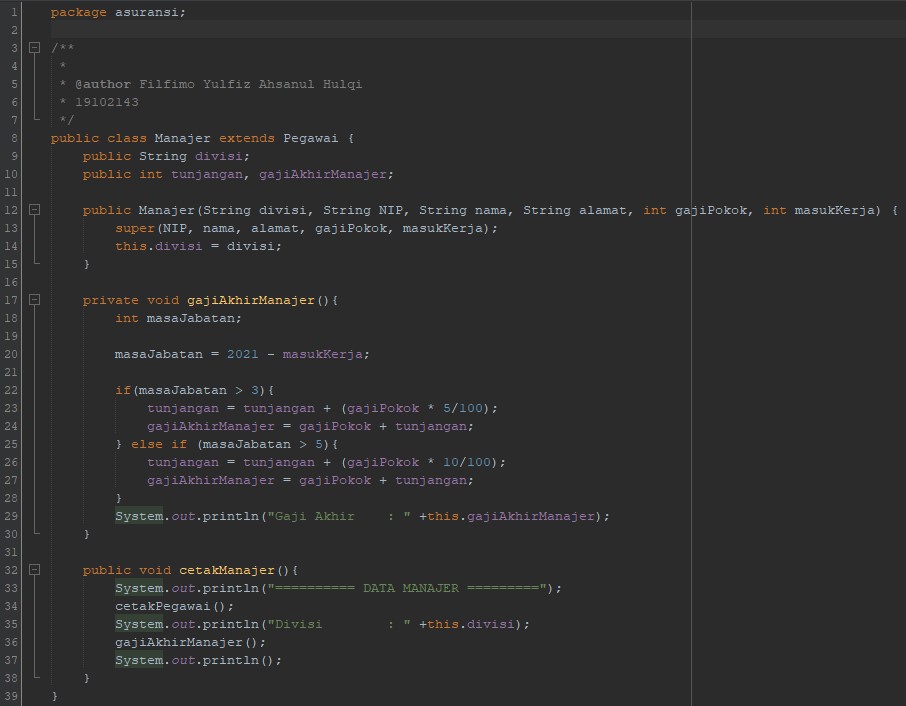
*Source* *code* *main* *class*

Screenshot Program :

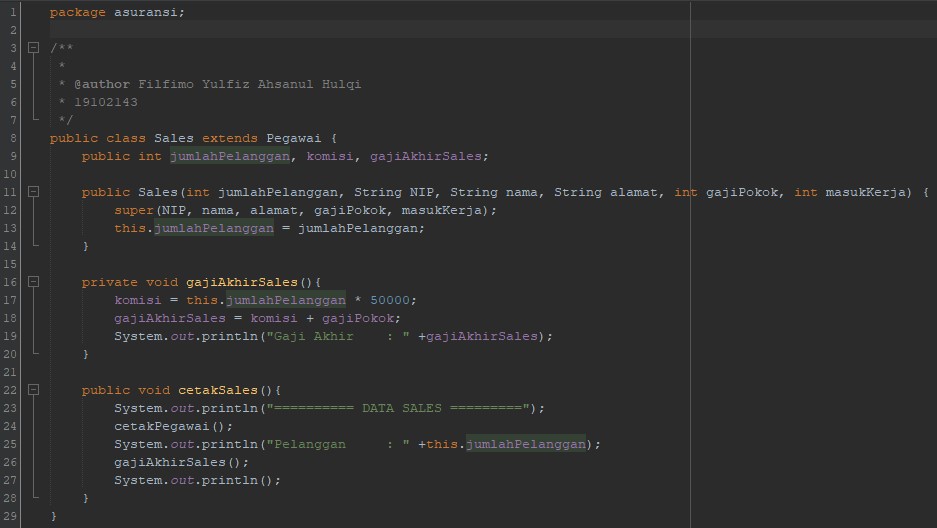
* *Class* Induk



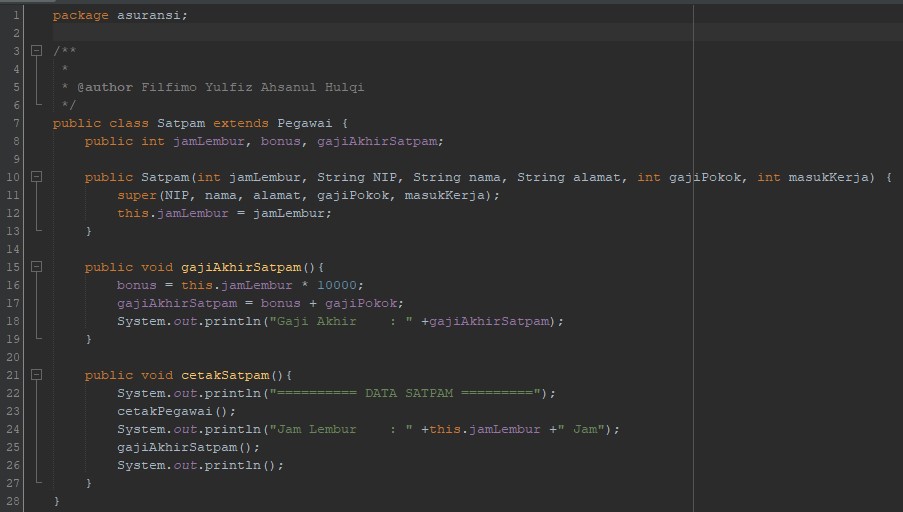
* *Class* Anak (Manajer)



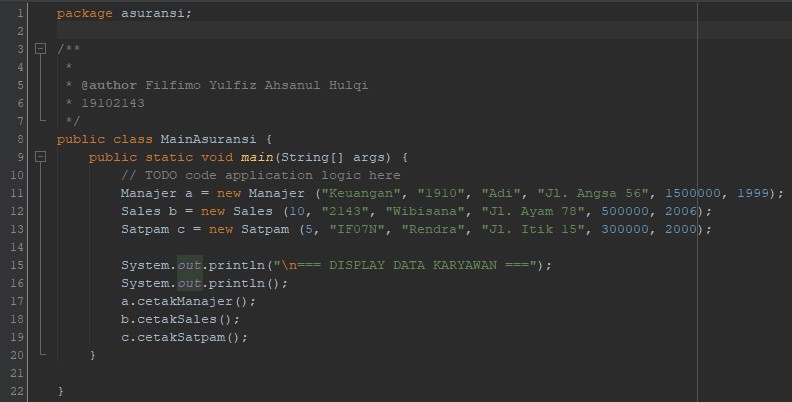
* *Class* Anak (Sales)



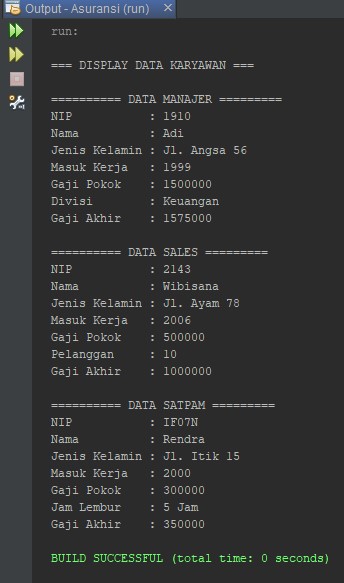
* *Class* Anak (Satpam)



* *Main* *Class*



* Ouput



Penjelasan Program :

Pada program diatas, diminta untuk membuat program dengan implementasi pewarisan (*inheritance*) untuk perusahaan asuransi. Mempunyai satu kelas induk, kelas induk ini bernama ”Pegawai”, mepunyai *fields peotected* yaitu NIP, nama, alamat, gaji pokok, dan masuk kerja, yang nantinya isi dari kelas pegawai ini dapat diturunkan ke kelas lain. Didalam kelas pegawai ini juga ada konstruktor dengan parameter, dan *methods* untuk cetak data pegawai.

Program ini mempunyai tiga anak kelas yaitu ”Manajer”, ”Sales”, dan ”Satpam”, para anak kelas ini merupakan bagian atau anak dari kelas ”Pegawai”.

Pada kelas anak Manajer, *fields* yang dimiliki adalah divisi, tunjangan, dan gaji akhir manajer, selanjutnya *fields* tersebut dibuat konstruktor, ada tambahan atribut *super* didalam konstruktor, atribut *super* tersebut merujuk pada kelas induk. Kemudian didalamnya terdapat dua *methods* yaitu *methods* untuk melakukan perhitungan gaji akhir dari manajer, dimana menggunakan perkondisian *if else* untuk menentukan jumlah tunjangan yang akan diberikan, apakah mendapat 5% atau 10% dari gaji pokok, sesuai dengan lama kerja dari manajer. *Methods* yang kedua adalah *methods* untuk menampilkan data dari manajer, didalam *methods cetak* tersebut ada *methods* dari kelas induk yaitu *methods* cetakpegawai*,* dan juga *methods* dari anak kelas manajer yaitu *methods* gaji akhir manajer*.*

Pada kelas anak Sales, *fields* yang dimiliki adalah jumlah pelanggan, komisi, dan gaji akhir sales, selanjutnya *fields* tersebut dibuat konstruktor, ada tambahan atribut *super* didalam konstruktor. Kemudian didalamnya terdapat dua *methods* yaitu *methods* untuk melakukan perhitungan gaji akhir dari sales, dimana sales bisa mendapatkan tambahan komisi, jika sales mampu mendapatkan pelanggan, satu pelanggan diberi harga 50.000. *Methods* yang kedua adalah *methods* untuk menampilkan data dari sales, didalam *methods cetak* tersebut ada *methods* dari kelas induk yaitu *methods* cetak pegawai, dan juga *methods* dari anak kelas sales yaitu *methods* gaji akhir sales.

Pada kelas anak Satpam, *fields* yang dimiliki adalah jam lembur, bonus, dan gaji akhir satpam, selanjutnya *fields* tersebur dibuat konstruktor, ada tambahan atribut *super* didalam konstruktor. Kemudia didalamnya terdapat dua *methods* yaitu *methods* untuk melakukan perhitungan gaji akhir dari satpam, dimana satpam bisa mendapatkan tambahan bonus, jika satpam melakukan jam lembur, setiap satu jam lembur yang dilakukan mendapatkan harga 10.000. *Methods* yang kedua adalah *methods* untuk menampilkan data dari satpam, didalam *methods cetak* tersebut ada *methods* dari kelas induk yaitu *methods* cetak pegawai, dan juga *methods* dari anak kelas yaiut *methods* gaji akhir satpam.

Dapat dilihat maing-masing anak kelas memiliki format yang mirip, hanya berbeda dibagian *fields,* konstruktor, dan *methods* perhitungan gaji akhir.

Setelah semua kelas sudah selesai dibuat, selanjutnya adalah melakukan pembuatan objek dari masing-masing kelas di *main class*, objek yang dibuat harus memiliki paramater mengikuti urutan penulisan pada konstruktor. Lalu lakukan pemanggilan *methods* dari masing-masing objek yang telah dibuat tadi, maka akan muncul ouput sesuai *screenshot* diatas program pun selesai.

**BAB IV**

**Kesimpulan**

Pada Praktikum kali ini dapat diambi beberapa kesimpulan yaitu :

1. *Subclass* menyediakan *state*/*behaviour* yang spesifik membedakannya dengan *superclass*, hal ini akan memingkinkan *programmer* untuk menggunakan ulang *source code* nya dari *super class*  yang sudah ada sebelumnya.
2. Dapat mendeifinisikan *superclass* khusus yang bersifat generik, yang disebut *abstract class*, untuk mendefinisikan *class* dengan *behaviour* dan *state* secara umum.