**LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL III**

**PRINSIP PERANCANGAN KELAS**

****

**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. Mengenali dan memahami tentang pemrograman berorientasi objek.
2. Mengenali dan memahami Kelas dan Objek.
3. Menerjemahkan Kelas dan Objek dalam bahasa pemrograman.
4. Memahami tentang perancangan Kelas yang baik.
5. Memahami penggunaan *Accessor* dan *Mutator*.

**BAB II**

**Dasar Teori**

1. *Constructor*

*Method* ini digunakan untuk inisialisasi atau mempersiapkan data untuk objek. *Method* *constructor* harus memiliki nama yang sama dengan kelas.

1. Visibilitas bagi *Atributes* dan *Methods*

Dalam suatu kelas diperlukan kontrol akses untuk mengatur siapa saja yang dapat mengakses, mengubah nilai dari *atributes* atau *methods* dari kelas tersebut, sehingga penyalahgunaan *atributes* atau *methods* dalam kelas tersebut dapat dihindari. Untuk menentukan kontrol akses dari *atributes* atau *methods* dalam kelas perlu ditambahkan *Acces Modifier* atau juga disebut sebagai *Acces Specifier* terhadap *atributes* atau *methods* tersebut.

Kontrol akses yang sering digunakan, antara lain :

1. *Public*

Kontrol akses yang memberikan akses penuh kepada *atributes* dan *methods*, dapat diakses oleh objek dalam kelas itu sendiri atau objek dari kelas lain.

1. *Private*

Kontrol akses yang memberikan akses tertutup kepada *atributes* dan *methods*, hanya dapat diakses oleh objek dalam kelas itu sendiri dan objek dari kelas lain tidak dapat mengaksesnya.

1. *Protected*

Kontrol akses yang memberikan akses terbata kepada *atributes* atau *methods*, hanya dapat diakses oleh objek dalam kelas itu sendiri dan objek dari kelas lain yang merupakan turunan dari kelas yang memiliki *atributes* atau *methods* tersebut.

1. *Accessor* dan *Mutator*
2. *Acessor*

Merupakan fungsi untuk mendapatkan *property* dari suatu objek, mengembalikan nilai atau value dari suatu atribut (*get*).

1. *Mutator*

Merupakan fungsi untuk mengubah *property* dari suatu objek, mengubah nilai atau *value* dari sebuah atribut (*set*).

1. *Method* *Overloading* dan Operator *Overloading*

*Method Overloading* adalah membuat dua nilai atau lebih *methods* dengan nama yang sama dalam satu kelas. *Method-Method* tersebut harus dapat dibedakan antara satu dengan yang lain dalam jumlah atau tipe datanya.

1. Melewatkan Argumen/Parameter ke *Method*
2. Melewatkan secara nilai (*pass by value*)

Diterapkan pada argumen bertipe data primitif (*byte, short, int, float,* dsb). Prosesnya adalah *compiler* hanya menyalin isi memori (yang telahh dialokasikan untuk suatu variabel), dan kemudian mengirim salinan tersebut kepada *method* yang bersangkutan (isi memori merupakan data sesungguhnya yang akan dioperasikan), karena yang disampaikan hanya salinan dari isi memori, maka perubahan yang terjadi pada variabel akibat proses didalam method tidak mempengaruhi nilai variabel asalnya di dalam memori.

1. Melewatkan secara referensi (*pass by reference*)

Diterapkan pada argumen bertipe data non primitif (*array, class, object, string, dsb*). Isi memori pada variabel tersebut merupakan penunjuk alamat memori yang mengandung data sesungguhnya yang akan dioperasikan. Dengan kata lain, variabel tersebut menyimpan alamat memori bukan isi memori, maka perubahan yang terjadi pada variabel akibat proses didalam method akan mempengaruhi nilai variabel asalanya

1. *Resposibility Driven Design*

Bahwa dalam perancangan kelas, semua fungsi/*methods* yang ada harus mencerminkan perilaku lengkap yang dimiliki kelas tersebut. Semua fungsi/*methods* tersebut bertanggung jawab terhadap *maintanance* atribut yang dimiliki kelas.

**BAB III**

**Penjelasan**

**UNGUIDED**

1. Buat program dalam bahasa Java dimana diminya untuk membuat biodata diri dengan menggunakan *setter getter* dan *constructor*.

package orang;

public class Orang {

public String nama;

public int umur;

public String gender;

public String alamat;

public String kerja;

public Orang(){

}

public Orang(String nama, int umur, String gender, String alamat, String kerja) {

this.nama = nama;

this.umur = umur;

this.gender = gender;

this.alamat = alamat;

this.kerja = kerja;

}

public String getNama() {

return nama;

}

public void setNama(String nama) {

this.nama = nama;

}

public int getUmur() {

return umur;

}

*Source code* *sub class* bagian 1

*Source code sub class* bagian 2

public void setUmur(int umur) {

this.umur = umur;

}

public String getGender() {

return gender;

}

public void setGender(String gender) {

this.gender = gender;

}

public String getAlamat() {

return alamat;

}

public void setAlamat(String alamat) {

this.alamat = alamat;

}

public String getKerja() {

return kerja;

}

public void setKerja(String kerja) {

this.kerja = kerja;

}

void Display(){

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

System.out.println("Umur : " + getUmur());

System.out.println("Gender : " + getGender());

System.out.println("Alamat : " + getAlamat());

System.out.println("Kerja : " + getKerja());

System.out.println();

}

}

package orang;

public class MainOrang {

public static void main(String[] args) {

Orang orang[] = new Orang[3];

orang[0] = new Orang("Fiflimo Yulfiz Ahsanul Hulqi", 19, "Brebes", "Laki-laki", "Mahasiswa");

orang[1] = new Orang("Prawono Budianto", 20, "Bumiayu", "Laki-laki", "Sejarahwan");

orang[2] = new Orang("Reo Natanegeri", 21, "Dukuhturi", "Laki-laki", "Guru");

System.out.println("==== BIODATA DIRI ====");

for(int i=0; i<orang.length; i++){

orang[i].Display();

}

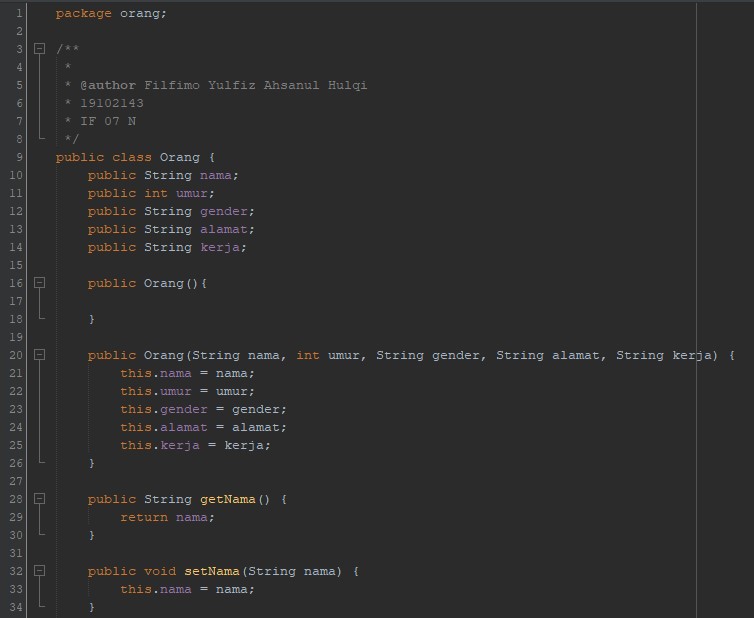
}

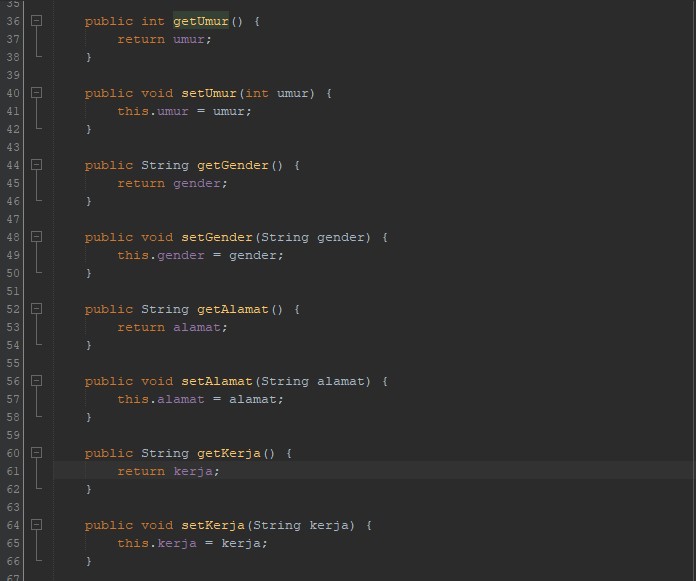
}

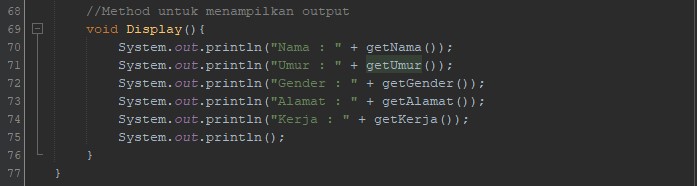
*Source code main class*

**Screenshot Program :**

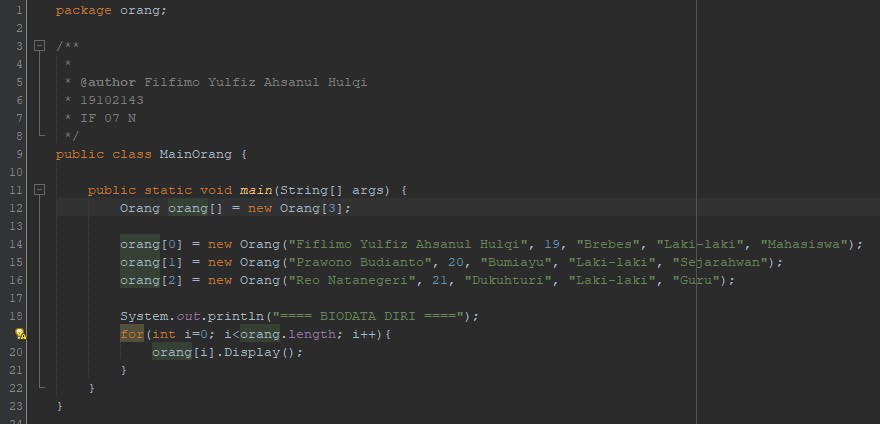
* *Sub class*

**

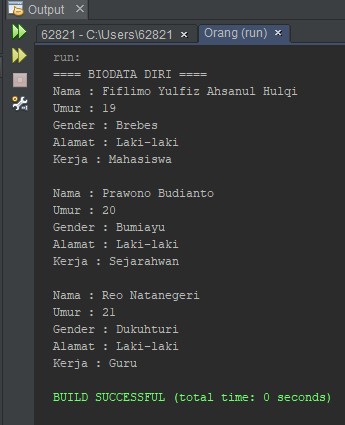




* *Main class*



* Output



Penjelasan Program :

Pada program diatas diminta untuk membuat dua buah *class*, untuk *sub class* disini berisi deklarasi *fields*, *default constructor*, *constructor*, *setter*, *getter*, dan *methods*.

Sementara itu pada *main class* berisi sebuah perulangan objek dengan array, memiliki 3 buah objek baru.

Pada *main class* untuk pembuatan objek menggunakan kurung kotak ([]), karena akan menggunakan, *array*, setelah itu isi objek baru tersebut dengan ketentuan model, tidak lupa untuk memperhatikan paramater, isi nilai paramater sesuai dengan *sub class*.

Setelah selesai membuat objek dan paramater, selanjutnya buat sebuah kode program untuk melakukan perulangan, nilai yang akan diulang adalah objek orang.

Panggil *methods* *displays* dari *sub class* didalam perulangan, *methods display* akan menampilkan data yang telah di-*set* pada program, dan diulang sebanyak tiga kali.

1. Buatlah program dalam bahasa Java dimana diminta untuk melakukan pemesanan pada restoran.

package restoran;

public class Item {

private String nama;

private String jenis;

private int harga;

public Item(String nama, String jenis, int harga){

this.nama = nama;

this.jenis = jenis;

this.harga = harga;

}

public String getNama() {

return nama;

}

public String getJenis() {

return jenis;

}

public int getHarga() {

return harga;

}

public String ShowDetail(){

return getNama() + "( " + getJenis() + " ) - Rp " + getHarga();

}

}

*Source code item class*

*Source code* pembeli *class*

package restoran;

Pembeli {

private String nama;

private Item[] historyPemesanan = new Item[99];

private int pesanan = 0;

public Pembeli(String nama){

this.nama = nama;

}

public String getNama(){

return nama;

}

public void lakukanPemesanan(Item item){

historyPemesanan[pesanan] = item;

pesanan++;

System.out.println(item.getNama() + " berhasil ditambahkan " + this.getNama());

}

public void lihatHistoryPemesanan(){

if(pesanan > 0){

System.out.println("History pesanan sdr " + this.getNama() + ":");

for(int i = 0; i < pesanan; i++){

Item order = historyPemesanan[i];

System.out.println("- " + order.getNama() + " (" + order.getJenis() + ") - Rp " + order.getHarga());

}

System.out.println();

}

else{

System.out.println("Anda belum memesan");

System.out.println();

}

}

public void kosongkanHistory(){

System.out.println("Reset History.....");

this.historyPemesanan = new Item[99];

this.pesanan = 0;

System.out.println("History pemesanan telah dikosongkan");

System.out.println();

}

}

*Source code main* restoran *class* bagian 1

package restoran;

import java.util.Scanner;

public class MainRestoran {

private static void tampilkanMenu(){

int nomorItem = 1;

for(Item item : menu){

System.out.println(nomorItem++ + ". " + item.ShowDetail());

}

System.out.println();

}

private static Item[] menu = {

new Item("Mendoan", "Makanan", 1000),

new Item("Es Teh", "Makanan", 2500),

new Item("Aneka Sayur", "Makanan", 5000),

new Item("Paha Ayam", "Makanan", 7500),

new Item("Sate Ayam", "Makanan", 15000),

};

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

boolean keepLooping = true;

System.out.println("Masukan Nama");

String namaPembeli = input.nextLine();

Pembeli pembeli = new Pembeli(namaPembeli);

System.out.println();

do {

System.out.println("===== MENU UTAMA =====");

System.out.println("1. Pesan Makanan");

System.out.println("2. Lihat history pesanan");

System.out.println("3. Kosongkan history");

System.out.println("4. Keluar");

System.out.println("> ");

int nav = input.nextInt();

System.out.println();

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

switch(nav){

case 1:

tampilkanMenu();

System.out.println("Pilih makanan yang ingin dipesan");

System.out.println("Input angka 1-5");

int indexItem = input.nextInt() -1;

if (indexItem < 0 || indexItem >= menu.length){

System.out.println("Mohon masukan input yang benar");

}

else{

pembeli.lakukanPemesanan(menu[indexItem]);

}

System.out.println();

break;

case 2:

pembeli.lihatHistoryPemesanan();

System.out.println();

break;

case 3:

pembeli.kosongkanHistory();

System.out.println();

break;

case 4:

System.out.println("Program Keluar");

System.out.println();

keepLooping = false;

break;

default:

System.out.println("Tolong masukan input yang sesuai");

System.out.println();

break;

}

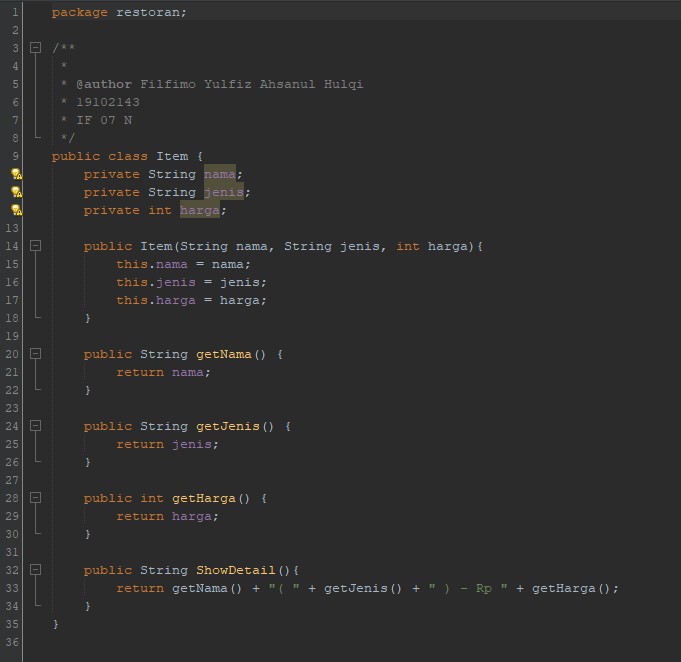
} while(keepLooping);

}

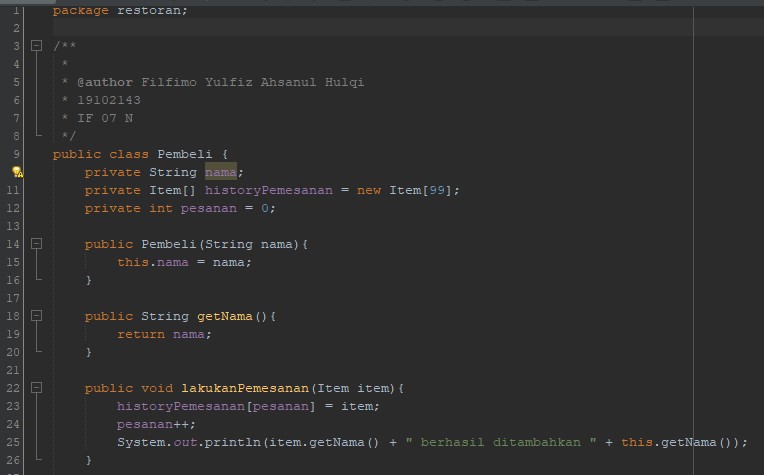
}

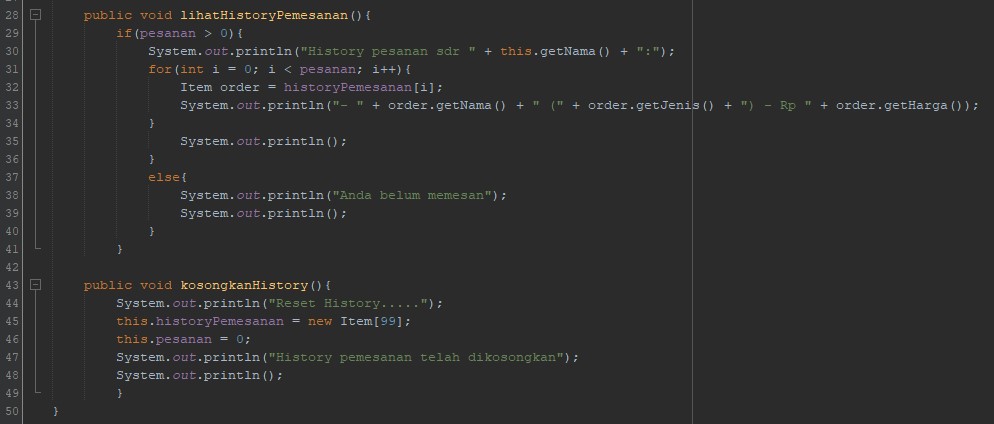
**Screenshot Program :**

* *Sub class (*item*)*

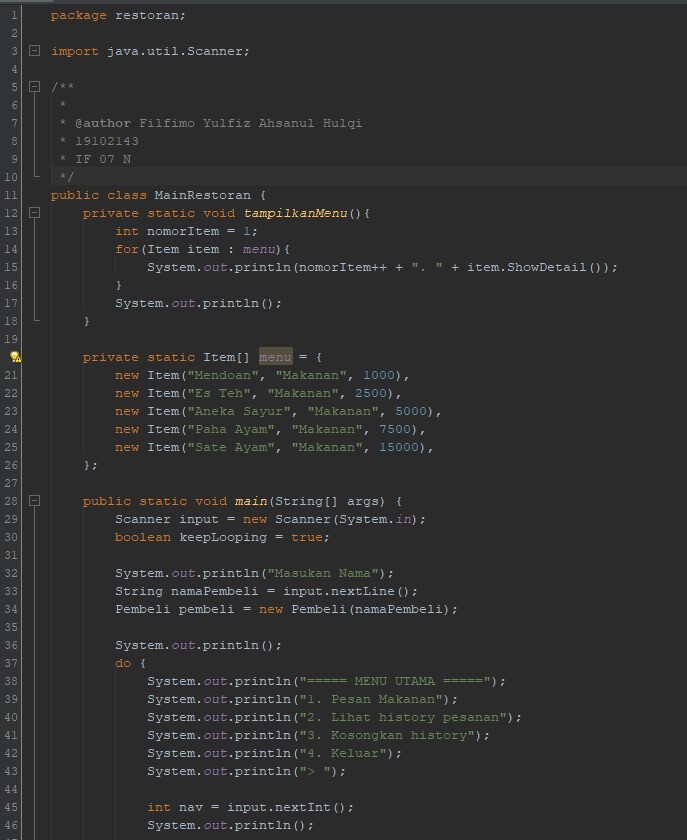


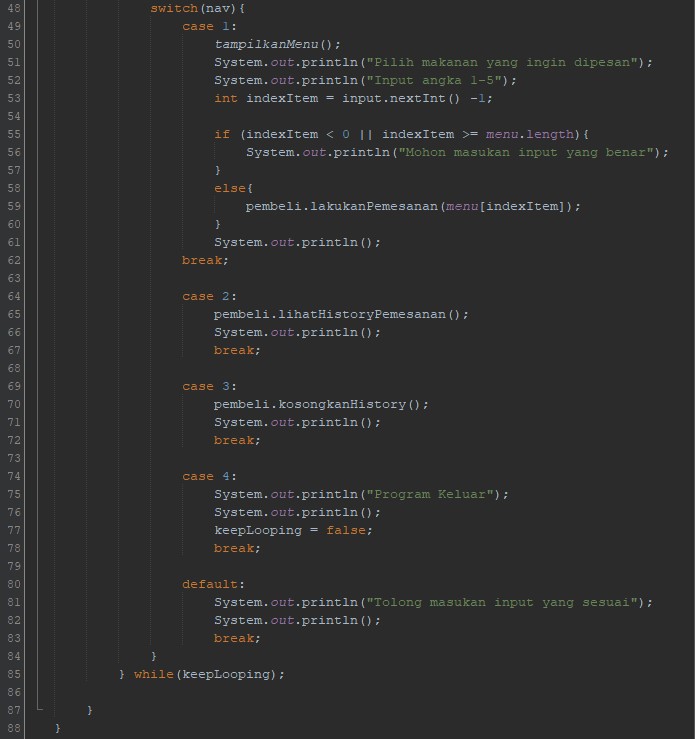
* *Sub class (*pembeli*)*



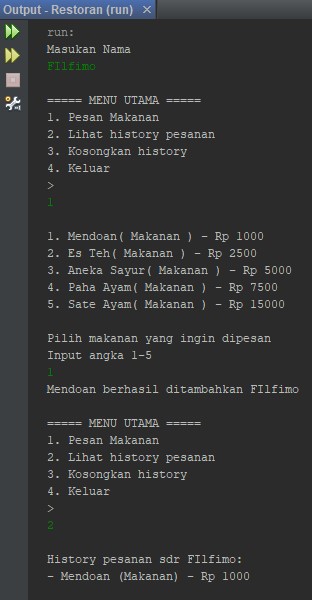


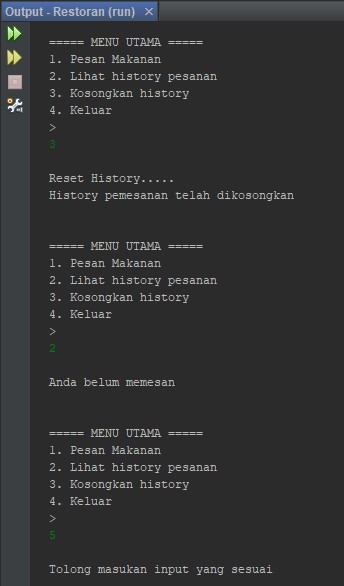
* *Main class*

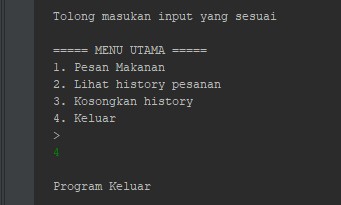




* *Output*







**Penjelasan Program :**

Pada program diatas, diminta membuat tiga *class*, dua sebagai *sub class* dan satu sebagai *main class*, masing-masing *class* memiliki fungsi sendiri-sendiri dan akan saling berhubungan satu sama lain.

*Class* *item,* pada *class* ini berisi *fields*, *setter, getter,* dan *methods*, *class* ini berfungsi untuk menampilkan item.

*Class pembeli*, pada *class* ini berisi *fields, setter, getter, overloading* dan *methods*, *class* ini bukan hanya menampilkan data pembeli secara umum. *Class* ini berhubungan dengan *class item*, *class pembeli* dapat melakukan pemesanan terhadap item yang akan dibeli, melihat *history* pemesanan, dan mengkosongkan *history* pesanan.

*Class main resotran,* ini adalah *class* utama dari program restoran, pada *class* ini berisi *import library scanner, fields, methods, looping,* pembuatan objek, dan pemanggilan *methods*.

Saat program dijalankan, program *running* pada *class main*, lalu *user* akan diminta memasukan nama, nama ini akan diproses ke *class pembeli*, setelah *user* mengisi nama, selanjutnya *user* akan ditampilkan dengan menu, *user* dapat memilih menu, dan melakukan *input*, untuk keluaran, akan disesuaikan pada *looping* kondisi. Jika *input user* sesuai arahan dari *program*, maka *output* yang ditampilkan akan muncul, jika *input* 1, *user* akan diarahkan untuk memilih makanan, jika *input* 2, *user* akan melihat *history* pesanan, jika *user* belum memesan, akan muncul informasi bahwa *user* belum melakukan pemesana, jika *input* 3, *user* dapat mengkosongkan *history* pesanan, dan jika *input 4* maka program akan selesai, misalkan *input* dari *user* tidak sesuai, maka akan muncul pesan bahwa *input* tidak tersedia.

Masing-masing dari *input* tadi, megarah pada *methods* dan *methods* tersebut berasal dari *subclass* jadi informasi yang dimasukan pada program *main* akan diproses, dan akan saling berhubungan dengan *sub class* kemudian informasi nanti dikembalikan kembali pada *main*.

1. Buat program dalam bahasa Java untuk mencatat data pegawai. Atribut data pegawai meliputi NIP, nama, alamat, jumlah hari lembur, gaji pokok dan total gaji. Nilai- nilai dari atribut pegawai di-*set* dalam program (tidak ada operasi *input*). Lakukan perhitungan total gaji lalu tampilkan data pegawai tersebut.

package pegawai;

public class Pegawai {

private int NIP;

private String nama;

private String alamat;

private int jmlHL;

private int gaji;

private double tot;

private double tun;

public Pegawai(){

}

public int getNIP() {

return NIP;

}

public void setNIP(int NIP) {

this.NIP = NIP;

}

public String getNama() {

return nama;

}

public void setNama(String nama) {

this.nama = nama;

}

public String getAlamat() {

return alamat;

}

*Source code sub class* bagian 1

*Source code sub class* bagian 2

public void setAlamat(String alamat) {

this.alamat = alamat;

}

public int getJmlHL() {

return jmlHL;

}

public void setJmlHL(int jmlHL) {

this.jmlHL = jmlHL;

}

public int getGaji() {

return gaji;

}

public void setGaji(int gaji) {

this.gaji = gaji;

}

public void PegawaiN(double totTun,double totGaji){

tun = gaji\*jmlHL\*0.01;

totTun = tun;

}

public void PegawaiN(double totGaji){

totGaji = 0;

tot = gaji + (gaji\*jmlHL\*0.01);

}

public void cetak(){

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

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

System.out.println("Alamat : " + getAlamat());

System.out.println("Jumlah hari lembur : " + getJmlHL());

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

System.out.println("Tunjangan : " + tun);

System.out.println("Total Gaji : " + tot);

}

}

*Source code main class*

package pegawai;

public class MainPegawai {

public static void main(String[] args) {

Pegawai P = new Pegawai();

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

P.setNIP(19102143);

P.setNama("Filfimo Yulfiz Ahsanul Hulqi");

P.setAlamat("Brebes");

P.setJmlHL(7);

P.setGaji(5000000);

P.PegawaiN(1,2);

P.PegawaiN(2);

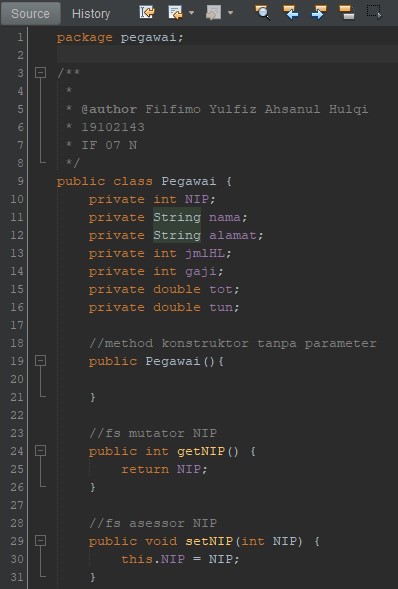
P.cetak();

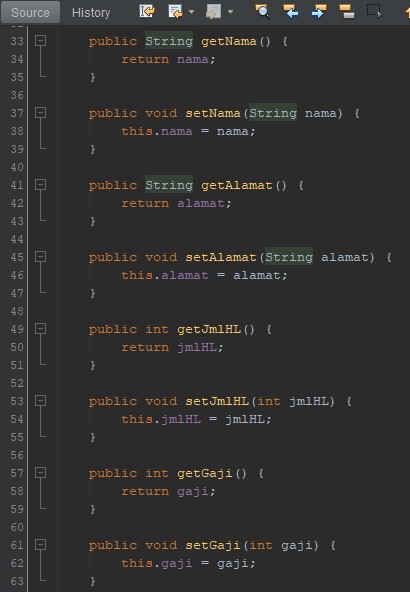
}

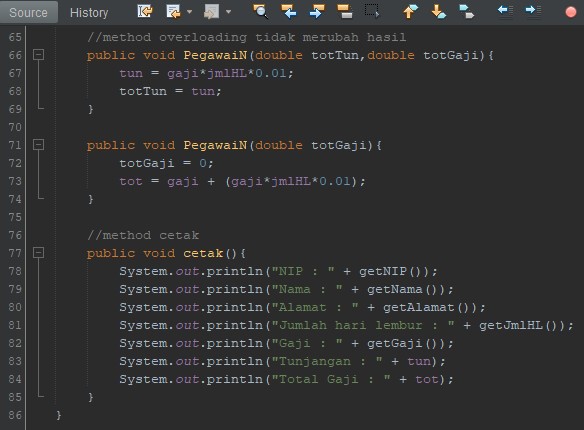
}

Screenshot Program :

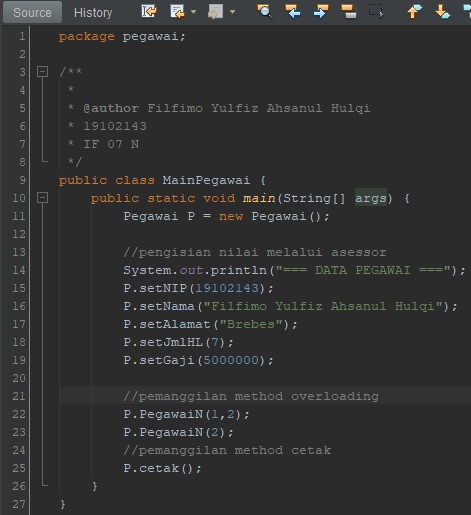
* Sub class



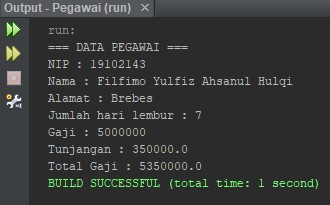




* Main class



* Output



Penjelasan Program :

Pada program diatas, diminta membuat dua *class*.

*Class* *pegawai,* pada *class* ini berisi *fields*, *setter, getter, overloading* dan *methods*, *class* ini berfungsi untuk menyimpan data/paramater dari *main class* serta untuk melakukan *merhods*.

*Class main pegawai*, pada *class* ini berisi *fields, setter,* pembuatan objek, dan pemanggilan *methods* dari *sub class*, pada *class* ini *user* dapat *set* nilai dari data pegawai.

Juga pemanggilan *methods* dari *sub class* untuk melakukan pemanggilan *output* data pegawai, dan *methods* untuk melakukan perhitungan total gaji pegawai.

**BAB IV**

**Kesimpulan**

Pada Praktikum kali dapat diambil beberapa kesimpulan yaitu :

1. Pemrogram berorientasi objek (OOP), membuat kode program lebih mudah dibaca karena karakteristiknya, pada praktik kali ini yang ditunjukkan adalah karakteristik *encapsulation*.
2. Pada Java untuk membuat OOP, diperlukan beberapa *class*, yang nanti *class* tersebut akan saling berhubungan.
3. Kelas dan Objek saling berhubungan, dan harus sangat diperhatikan saat pembuatannya, terutama penulisannya, baik itu *fields,* parameter *constructor*, *constructor*, *methods¸*dan *overloading*.
4. *Accessor* dan *Mutator* dapat digunakan untuk mendapatkan nilai dan mengembalikan nilai.
5. *Overloading*, *methods* yang sangat berguna, dapat menjalankan 2 buah *method* dengan nama yang sama namun memiliki paramter yang berbeda.