**LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL V**

**PEWARISAN JAMAK (*MULTIPLE 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 jamak (*Inheritance*).
2. Mampu mewujudkan bentuk pewarisan jamak dalam bahasa Java.
3. Mengerti tentang *interface* dan implementasnya.

**BAB II**

**Dasar Teori**

1. *Method* *Overriding*

*Method Overriding* juga termasuk dalam bagian pewarisan jamak, *method* ini dilakukan saat akan membuat ulang sebuah *method* dari *class* induk pada *sub*-*class*, atau *class* anak. *Mehod Overriding* dapat dibuat dengan menambahkan anotasi *@Override* di atas nama *method* atau sebelum pembuatan *method*. Jadi *Method Overriding* itu memodifikasi *method* yang sama, namun isi atau *behaviour*nya berbeda-beda.

1. *Multiple Inheritance*

Pada pewarisan tunggal setiap *class* anak mewarisi data maupun *method* dari *class* induk. *Multiple Inheritance* memungkinkan suatu *class* mewarisi data maupun *method* lebih dari satu *class* induk.

1. *Interface*

Pewarisan jamak, dapat digunakan bahasa C++, untuk bahasa Java tidak dapat digunakan pewarisan jamak. Pada pemrograman Java, solusi atas prinsip pewarisan jamak (*Multiple Inheritance*) adalah dengan menggunakan *interface*.

*Interface* adalah salah satu contoh konsep OOP, yang berisi kumpulan *method* secara struktur di bahasa pemrograman Java, yang berfungsi untuk tempat menyimpan *method* kosong (*Method Abstract*) dan atribut bersifat *constant* atau final (bisa disebut juga *template* *method*), serta tidak mempunyai implementasi atau tidak memiliki *body*. Sama halnya seperti *class* *abstract*, kita hanya dapat menggunakan atau mengimplementasikan (*implements*) *interface* dengan *class* lain sebagai *class* turunannya, serta dapat mewariskannya (*extends*) dengan *interface* yang lain.

**BAB III**

**PENJELASAN**

**GUIDED**

1. Projek *Interface*, memiliki 4 *interface* dengan nama internet, sosail, handphone, dan *time*, yang masing-masing *interface* memiliki *method* abstak sendiri. Buatlah kelas *Smartphone* untuk mengimplementasikan *interface*, dan *main class*, untuk menjalankan program.

package smartphone;

public interface Internet {

void Browse();

void Streaming();

}

*Interface* Internet

package smartphone;

public interface Sosial {

void Pesan();

void Vidcall();

}

*Interface* Sosial

package smartphone;

public interface Handphone extends Internet, Sosial{

void Players();

void Telepon();

void Contact();

}

*Interface* *Handphone*

*Interface* *Time*

package smartphone;

public interface Time {

void Kalender();

void Jam();

void Alarm();

}

package smartphone;

public class Smartphone implements Handphone, Time {

@Override

public void Players(){

System.out.println("Layar Nyala");

}

@Override

public void Telepon(){

System.out.println("Berdering....");

}

@Override

public void Contact(){

System.out.println("Mencari Kontak");

}

@Override

public void Browse(){

System.out.println("Pencarian Google");

}

@Override

public void Streaming(){

System.out.println("Memulai Netflix");

}

*Class* *Smartphone* bagian 1

*Class* *Smartphone* bagian 2

@Override

public void Pesan(){

System.out.println("Ketik Pesan");

}

@Override

public void Vidcall(){

System.out.println("Memanggil...");

}

@Override

public void Kalender(){

System.out.println("Program dibuat 5/9/2021");

}

@Override

public void Jam(){

System.out.println("Program dibuat jam 12.06");

}

@Override

public void Alarm(){

System.out.println("Alarm Nyala");

}

}

package smartphone;

import java.util.Scanner;

public class MainSmartphone {

public static void main(String[] args) {

Smartphone a = new Smartphone();

Scanner baca = new Scanner(System.in);

System.out.println("Selamat Datang Di Program Handphone");

String aksi;

*Class* *Main* bagian 1

*Class* *Main* bagian 2

while(true){

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

System.out.println("[1] Browse");

System.out.println("[2] Streaming");

System.out.println("[3] Players");

System.out.println("[4] Telepon");

System.out.println("[5] Contact");

System.out.println("[6] Pesan");

System.out.println("[7] Vidcall");

System.out.println("[8] Kalender");

System.out.println("[9] Jam");

System.out.println("[10] Alarm");

System.out.println("[0] Keluar");

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

System.out.println("Pilih Input >");

aksi = baca.nextLine();

if(aksi.equalsIgnoreCase("1")){

a.Browse();

System.out.println();

} else if (aksi.equalsIgnoreCase("2")){

a.Streaming();

System.out.println();

} else if (aksi.equalsIgnoreCase("3")){

a.Players();

System.out.println();

} else if (aksi.equalsIgnoreCase("4")){

a.Telepon();

System.out.println();

} else if (aksi.equalsIgnoreCase("5")){

a.Contact();

System.out.println();

} else if (aksi.equalsIgnoreCase("6")){

a.Pesan();

System.out.println();

} else if (aksi.equalsIgnoreCase("7")){

a.Vidcall();

System.out.println();

} else if (aksi.equalsIgnoreCase("8")){

a.Kalender();

System.out.println();

*Class* *Main* bagian 3

} else if (aksi.equalsIgnoreCase("9")){

a.Jam();

System.out.println();

} else if (aksi.equalsIgnoreCase("10")){

a.Alarm();

System.out.println();

} else if (aksi.equalsIgnoreCase("0")){

System.exit(0);

} else {

System.out.println("Kamu memilih input yang salah!");

System.out.println();

}

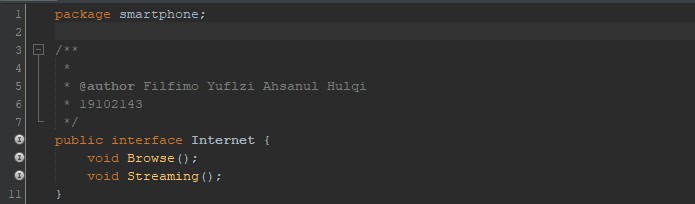
}

}

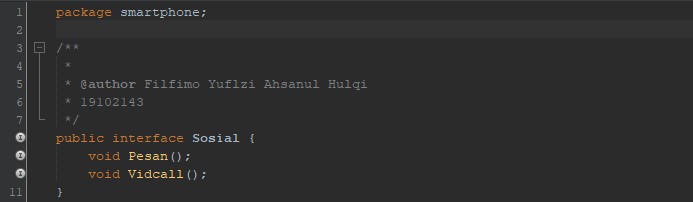
}

Screenshot program :

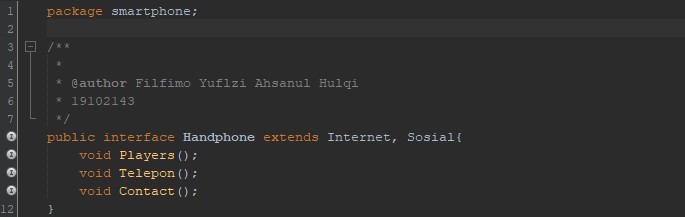
* *Interface* Internet

**

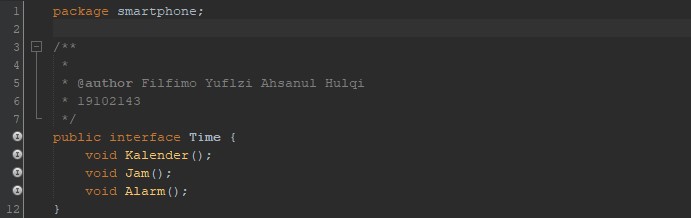
* *Interface* Sosial

**

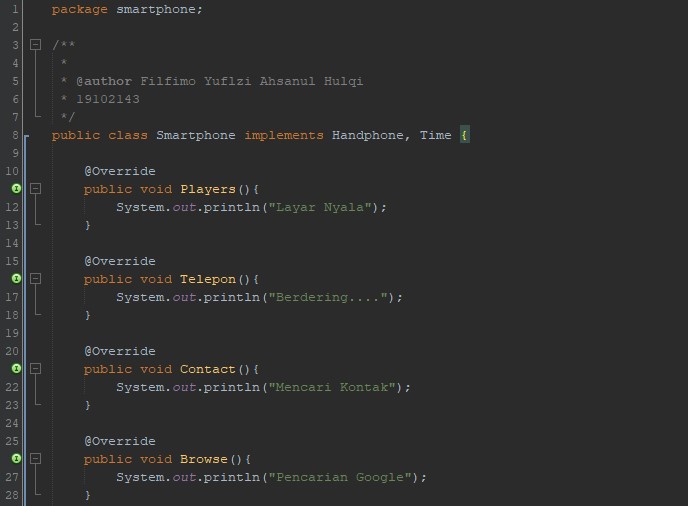
* *Interface Handphone*

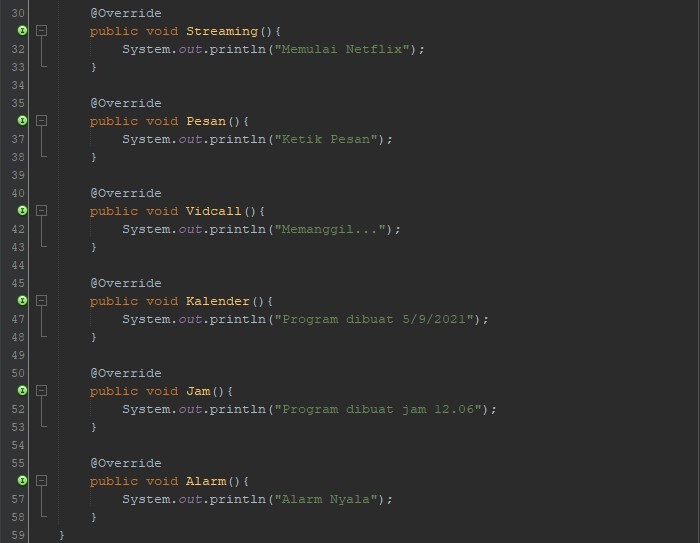
**

* *Interface Time*

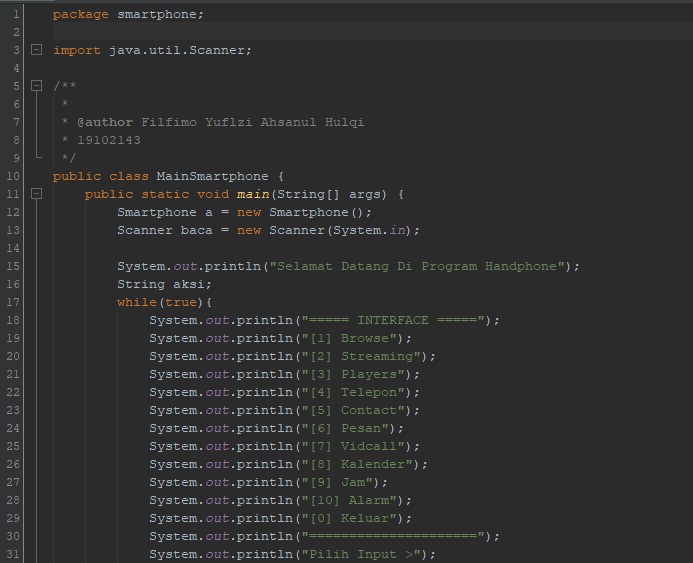
**

* *Class Smartphone*

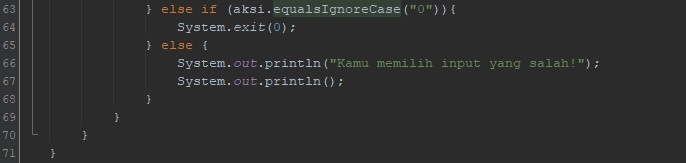
**

**

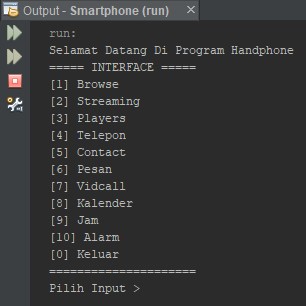
* *Main Class*

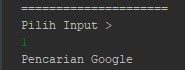
**

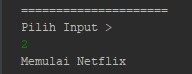
**

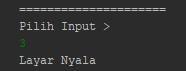
**

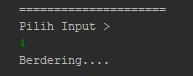
* Output :

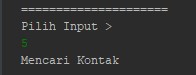


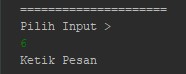


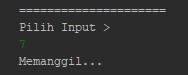


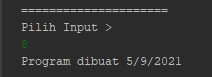


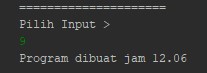


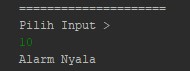


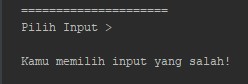


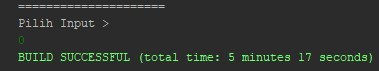












Penjelasan Program :

Pada program diatas adalah contoh implementasi dari pewarisan jamak pada Java. Membuat 4 *interface*, masing-masing dari *interface* tersebut dapat digunakan pada kelas lain.

*Interface* *Internet*, berisi dua *method void* yaitu *browse* dan *streaming.*

*Interface* Sosial, berisi dua *method void* yaitu pesan dan *vidcall*.

*Interface Time*, berisi tiga *method void* yaitu kalender, jam, dan *alarm*.

Pada *Interface Handphone*, barulah memulai implementasi pewarisan jamak, *interface handphone* melakukan *extends* dari *interface internet* dan *interface* sosial, didalamnya juga terdapat tiga *method void* yaitu *players*, telepon, dan *contact*.

Selanjutnya adalah *class smartphone*, didalamnya menggunakan *keywords* *implements* untuk mengimplementasikan *interface* dari *handphone* dan *time*. Karena *interfacw* adalah *class* yang hanya mengandung deklarasi *method* tanpa memiliki implementasi dan semua properti yang dimilikinya bersifat final. Disini juga dapat melakukan *Overriding* terhadap *method* yang berada pada *interface*, jadi didalam *overriding, method* tadi akan memiliki *implementasi*.

Dan yang terkahir adalah *main class* untuk melakukan eksekusi program. Didalamnya terdapat pembuatan objek seperti biasa, dan dibuat sebuah tampilan menu sesuai permintaan dari modul, memiliki *library scanner* karena akan ada operasi *input* dari *user* dan untuk menentukan menu yang akan dipilih menggunakan perkondisian *if else*.

Untuk tampilan menu dan *ouput* dari masing-masing *input* tertera seperti pada gambar diatas.

1. Buat dua objek yang akan dihubungkan dengan *interface*, yakni *Handphone* dan *PhoneUser*, kemudian buat *class* lain untuk masin-masing merek *handphone* yaitu Xiaomi, iPhone, Samsung, dan Oppo.

package phone;

public class PhoneUser {

protected Phone phone;

public PhoneUser(Phone phone){

this.phone = phone;

}

void turnOnThePhone(){

this.phone.powerOn();

}

void turnOffThePhone(){

this.phone.powerOff();

}

void makePhoneLouder(){

this.phone.volumeUp();

}

void makePhoneSilent(){

this.phone.volumeDown();

}

}

*Source code class PhoneUser*

*Interface Phone*

package phone;

public class Xiaomi implements Phone {

private int volume;

private boolean isPowerOn;

public int getVolume() {

return volume;

}

//Set Volume Awal

public Xiaomi(){

this.volume = 50;

}

@Override

public void powerOn(){

isPowerOn = true;

System.out.println("Handphone Menyala");

System.out.println("Device Xiaomi");

}

package phone;

public interface Phone {

int MAX\_VOLUME = 100;

int MIN\_VOLUME = 0;

void powerOn();

void powerOff();

void volumeUp();

void volumeDown();

}

*Class* Xioami bagian 1

*Class* Xiaomi bagian 2

@Override

public void powerOff(){

isPowerOn = false;

System.out.println("Handphone Dimatikan");

}

@Override

public void volumeUp(){

if(isPowerOn){

if(this.volume == MAX\_VOLUME){

System.out.println("Volume Penuh");

System.out.println("Sudah " +this.getVolume() + "%");

} else {

this.volume += 10;

System.out.println("Volume sekarang : " +this.getVolume() + "%");

}

} else{

System.out.println("Nyalakan Handphone terlebih dahulu");

}

}

@Override

public void volumeDown(){

if(isPowerOn){

if(this.volume == MIN\_VOLUME){

System.out.println("Volume = 0%");

} else {

this.volume -= 10;

System.out.println("Volume sekarang : " +this.getVolume() + "%");

}

} else {

System.out.println("Nyalakan Handphone terlebih dahulu");

}

}

}

*Main class*

package phone;

import java.util.Scanner;

public class MainPhone {

public static void main(String[] args) {

Phone xiao = new Xiaomi();

Phone ipho = new iPhone();

Phone sams = new Samsung();

Phone oppo = new Oppo();

PhoneUser nama = new PhoneUser(xiao);

nama.turnOnThePhone();

Scanner input = new Scanner(System.in);

String aksi;

while(true){

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

System.out.println("[1] Nyalakan HP");

System.out.println("[2] Matikan HP");

System.out.println("[3] Perbesar Volume");

System.out.println("[4] Kecilkan Volume");

System.out.println("[0] Keluar");

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

System.out.println("Pilih Input >");

aksi = input.nextLine();

if(aksi.equalsIgnoreCase("1")){

nama.turnOnThePhone();

} else if (aksi.equalsIgnoreCase("2")){

nama.turnOffThePhone();

} else if (aksi.equalsIgnoreCase("3")){

nama.makePhoneLouder();

} else if (aksi.equalsIgnoreCase("4")){

nama.makePhoneSilent();

} else if (aksi.equalsIgnoreCase("0")) {

System.exit(0);

} else {

System.out.println("Kamu memilih input yang salah!");

}

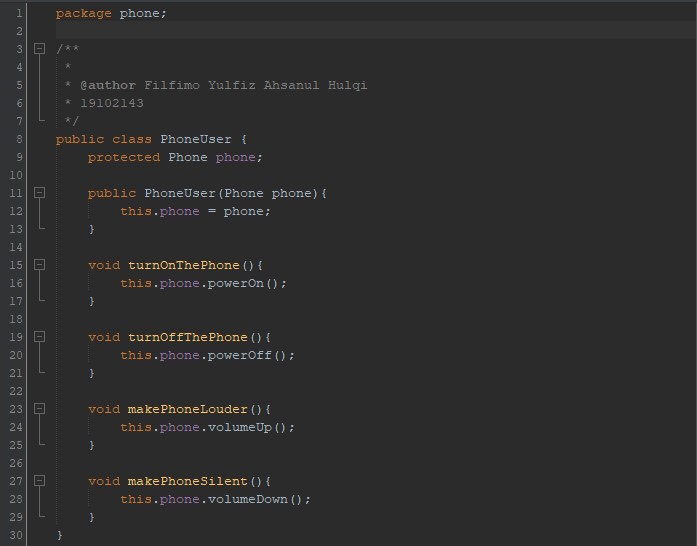
}

}

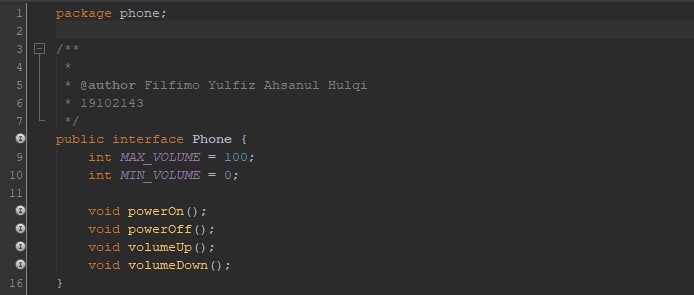
}

Screenshot program :

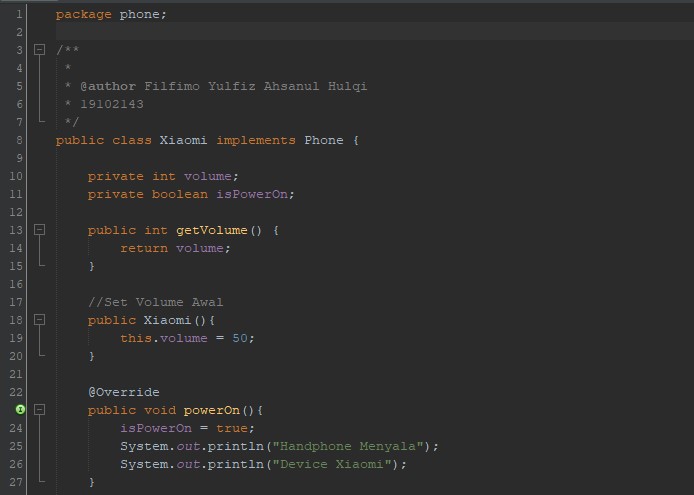
* *Class PhoneUser*

**

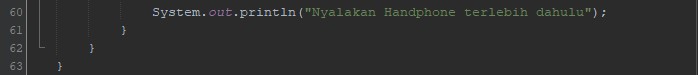
* *Interface Phone*

**

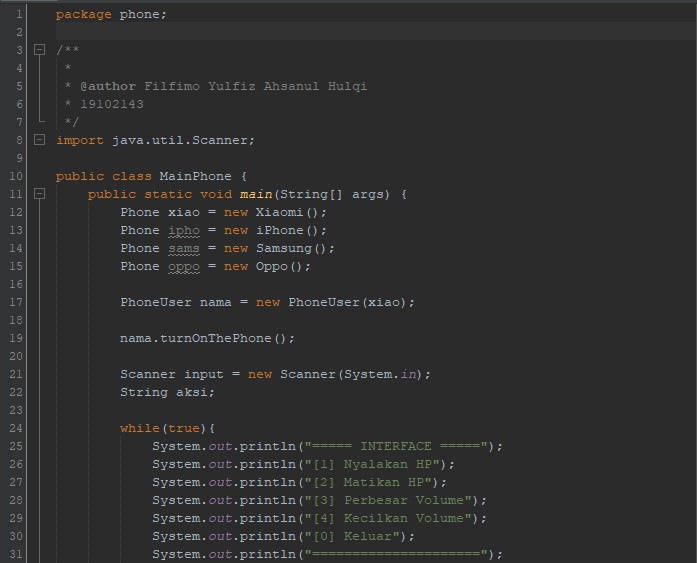
* *Class Xiaomi*

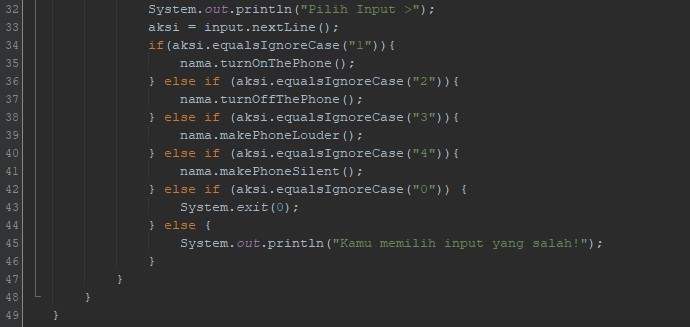
**

**

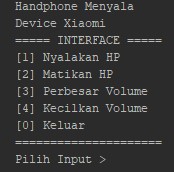
**

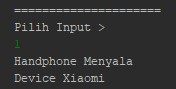
* *Main Class*

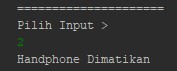
**

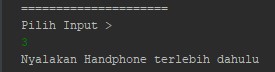
**

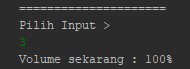
* Output

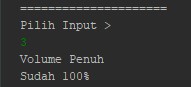
**

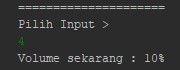
**

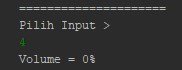
**

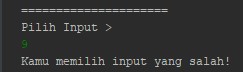
**

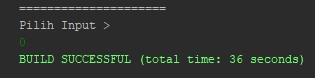
**

**

**

**

**

**

Penjelasan Program :

Pada program diatas memiliki tujuh kelas, satu *interface* dan lima kelas dan satu *main class*.

Pertama buat *interface Phone*, yang didalamnya terdapat *fields* dan *method void* yang akan digunakan pada *class* implementasi.

Buat lagi kelas *PhoneUser*, didalamya terdapat *fields*, *method void* dan konstruktor.

Berikutnya membuat *class* untuk implemnetasi dari *interface Phone,* yaitu Xiaomi, didalamnya terdapat *fields,* konstruktor, dan *overriding* dari *method* *void* milik *interface* *Phone*. Untuk tambahan dari modul kita buat *class* serupa seperti Xiaomi dengan nama yang berbeda yaitu iPhone, Samsung, dan Oppo, untuk isi dari *source code* masih sama, hanya nama dari fungsi dan *class* nya saja yang berubah.

Terakhir buat *class main* untuk melakukan eksekusi program, didalam *class main* terdapat pembuatan objek, menggunakan *library* *scanner* karena akan menggunakan *input* dari *user* dan untuk *ouput* perkondisian *if else* pada menu. Membuat *fields* aksi untuk *input* dari *scanner*. Dan yang terakhir pembuatan menu dan perkondisian *if else*, *output* dari *if else* akan sesuai dengan *input*/aksi, didalamnya akan memanggil *method* dari *PhoneUser.*

**UNGUIDED**

* Carilah sebuah solusi dari program implementasi pewarisan jamak pada modul dengan kasus ”*The Diamond Problem*”, dengan ketentuan dari program adalah sebagai berikut :

1. Buat kelas Mobil dengan ketentuan sebagai berikut :
2. Memiliki atribut : noPlat, merk, dan pajak.
3. Memiliki *method* tampilInfo(), untuk menampilkan informasi dari Mobil.
4. Buat kelas Bus dengan ketentuan seebagai berikut :
5. Kelas Bus adalah kelas turunan dari kelas Mobil.
6. Memiliki atribut : kapasitasPenumpang, dan kapasitasBagasi.
7. Memiliki *method* infoBus(), untuk menampilkan informasi dari Bus.
8. Memiliki *method* tampilInfo(), untuk menampilkan informasi keseluruhan dari Bus (memanggil *method* tampilInfo() dari kelas Mobil dan *method* infoBus()).
9. Buat kelas Sedan dengan ketentuan sebagai berikut :
10. Kelas Seda adalah kelas turunan dari kelas Mobil.
11. Memiliki atribut : fasilitasKeamanan, kapasitasCC, dan fasilitasKenyamanan.
12. Memiliki *method* infoSedan(), untuk menampilkan informasi dari Sedan.
13. Memiliki *method* tampilInfo(), untuk menampilkan informasi keseluruhan dari Sedan (memanggil *method* tampilInfo() dari kelas Mobil dan *method* infoSedan());
14. Memiliki *method* float hitungPajak(), untuk mengembalikan perhitungan besar pajak dengan rumus : pajak + (pajak + (pajak \* kapasitasCC \* 0.00005)).
15. Buat kelas MiniBus dengan ketentuan sebagai berikut :
16. Kelas MiniBus adalah kelas turunan dari kelas Sedan, dan Bus.
17. Memiliki atribut : tipe.
18. Memiliki *method* infoMiniBus(), untuk menampilkan informasi MiniBus (jika tipe MiniBus yang digunakan adalah ”**Pribadi**” maka tampilkan ”Tipe MiniBus yang digunakan adalah Pribadi”, sedangkan jika tipe yang digunakan adalah ”**Wagon**” maka tampilkan ”Tipe MiniBus yang digunakan adalah Wagon, untuk bongkar muat barang”).
19. Memiliki *method* tampilInfo(), untuk menampilkan informasi keseluruhan dari miniBus (memanggil *method* tampilInfo() dari kelas Sedan, *method* infoBus() dari kelas Bus, dan *method* infoMiniBus()).
20. Memiliki *method* float hitungPajak(), untuk mengembalikan besar pajak dengan rumus :

* Jika tipe adalah ”**Pribadi**”, rumusnya adalah : (Sedan.hitungPajak() \* 0.05) + (Bus.hitungPajak() \* 0.03).
* Jika tipe adalah ”**Wagon**”, rumusnya adalah : (Sedan.hitungPajak() \* 0.03) + (Bus.hitungPajak() \* 0.05).
* Analisis Soal

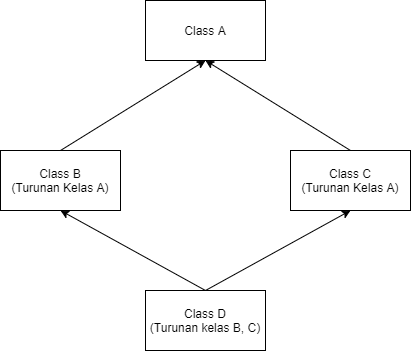
Soal ini berbeda dengan soal-soal *unguided* pada modul sebelumnya, pada *unguided* kali ini langsung mengambil studi kasus permasalahan dalam bahasa Java, yaitu *Multiple Inheritance*.

Seperti yang telah dijelaskan, bahwa bahasa pemrograman Java singkatnya tidak mendukung *Multiple Inheritance*, begitu pun pada dokumentasi Java, dan forum-forum pemrograman juga menjelaskan demikian.

*Multiple Inheritance* atau pewarisan jamak dipandang sebagai mekanisme untuk membentuk suatu sub kelas dari beberapa implementasi super kelas. Tetapi dalam pemrograman Java akan muncul beberapa masalah.

Ambiguitas

1. Timbul ketika 2 kelas induk memiliki suatu fungsi dengan nama yang sama,.
2. Kelas turunan memiliki beberapa salinan kelas dasar yang sama.



Pada contoh diagram diatas adalah salah satu masalah *multiple inheritance* dari bahasa Java, yaitu *The Diamond Problem*, adalah ketika suatu kelas sebut saja kelas A mempunyai sebuah *method* panggil() dan kelas B dan C adalah turunan dari kelas A dan didalamnya juga terdapat implementasi dari *method* panggil(), dan kelas D adalah turunan dari kelas B dan C, menggunakan *multiple inheritance* dan jika memanggil *method* panggil(), maka *compiler* tidak dapat memilih *method* panggil() mana yang harus dieksekusi.

Untuk solusi dari diagram dan program diatas disini menggunakan super class, selengkapnya akan dijelaskan pada bagian penjelasan program.

package kendaraan;

public interface Mobil {

public String NO\_PLAT = "IF 07 N";

public String MERK = "Filfimo";

public double PAJAK = 700000;

default void tampilInfo(){

System.out.println("Tampil Info Mobil");

System.out.println("No Plat : " + NO\_PLAT);

System.out.println("Merk : " + MERK);

System.out.println("Pajak : " + PAJAK);

System.out.println();

}

}

*Source* *code* *interface* mobil

*Source* *code* *interface* sedan

package kendaraan;

public interface Sedan extends Mobil {

public String FASILITAS\_KEAMANAN = "Alarm";

public String FASILITAS\_KENYAMANAN = "Auto Pilot";

public double KAPASITAS\_CC = 255.5;

default void infoSedan(){

System.out.println("Info Sedan");

System.out.println("Fasilitas Keamanan : " + FASILITAS\_KEAMANAN);

System.out.println("Fasilitas Kenyamanan : " + FASILITAS\_KENYAMANAN);

System.out.println("Kapasitas CC : " + KAPASITAS\_CC);

System.out.println();

}

@Override

default void tampilInfo(){

Mobil.super.tampilInfo();

infoSedan();

}

default double hitungPajak(){

double PAJAK\_SEDAN = PAJAK + (PAJAK + (PAJAK \* KAPASITAS\_CC \* 0.00005));

System.out.println("Total Pajak Sedan : " + PAJAK\_SEDAN);

return PAJAK\_SEDAN;

}

}

package kendaraan;

public interface Bus extends Mobil {

public int KAPASITAS\_PENUMPANG = 35;

public int KAPASITAS\_BAGASI = 50;

default void infoBus(){

System.out.println("Info Bus");

System.out.println("Kapassitas Penumpang : " + KAPASITAS\_PENUMPANG);

System.out.println("Kapasitas Bagasi : " + KAPASITAS\_BAGASI + " Kg");

System.out.println();

}

*Source* *code* *interface* bus bagian 1

*Source* *code* *interface* bus bagian 2

@Override

default void tampilInfo(){

Mobil.super.tampilInfo();

infoBus();

}

default double hitungPajak(){

double PAJAK\_BUS = PAJAK + (PAJAK \* KAPASITAS\_PENUMPANG \* KAPASITAS\_BAGASI \* 0.00005);

System.out.println("Total Pajak Bus : " + PAJAK\_BUS);

return PAJAK\_BUS;

}

}

package kendaraan;

public class MiniBus implements Sedan, Bus {

public String tipe = "Wagon";

public void infoMiniBus(){

System.out.println("Info Mini Bus");

if(tipe == "Pribadi"){

System.out.println("Tipe MiniBus yang digunakan adalah Pribadi");

} else {

System.out.println("Tipe MiniBus yang digunakan adalah Wagon, untuk bongkar muat barang");

}

System.out.println();

}

@Override

public void tampilInfo(){

Sedan.super.tampilInfo();

Bus.super.infoBus();

infoMiniBus();

}

*Source* *code* *class* MiniBus bagian 1

*Source* *code* *class* MiniBus bagian 2

@Override

public double hitungPajak(){

if (tipe == "Pribadi"){

double PAJAK\_MINIBUS\_PRIBADI;

PAJAK\_MINIBUS\_PRIBADI = (Sedan.super.hitungPajak() \* 0.05) + (Bus.super.hitungPajak()\*0.03);

System.out.println("Total Pajak MiniBus Pribadi : " + PAJAK\_MINIBUS\_PRIBADI);

return 0;

} else {

double PAJAK\_MINIBUS\_WAGON;

PAJAK\_MINIBUS\_WAGON = (Sedan.super.hitungPajak() \* 0.03) + (Bus.super.hitungPajak()\*0.05);

System.out.println("Total Pajak MiniBus Wagon : " + PAJAK\_MINIBUS\_WAGON);

return 0;

}

}

}

package kendaraan;

public class Kendaraan {

public static void main(String[] args) {

MiniBus obj = new MiniBus();

System.out.println("======== PEWARISAN JAMAK KENDARAAN ========");

System.out.println();

obj.tampilInfo();

obj.hitungPajak();

System.out.println();

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

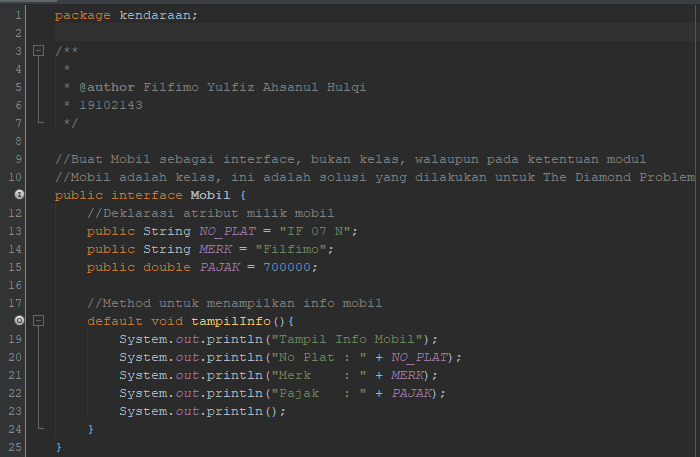
}

}

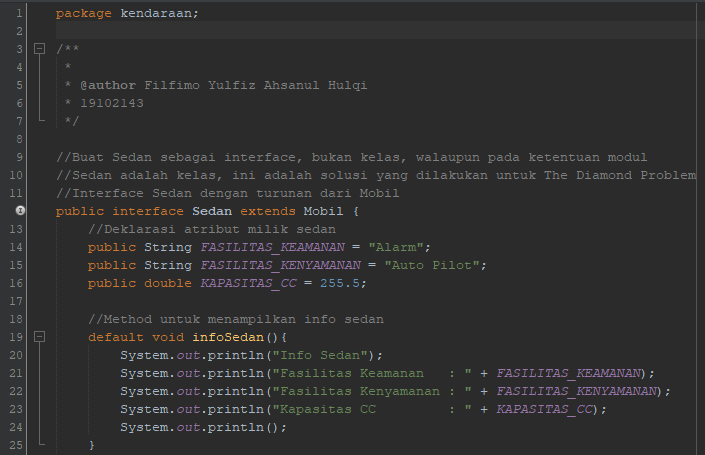
*Source code main class*

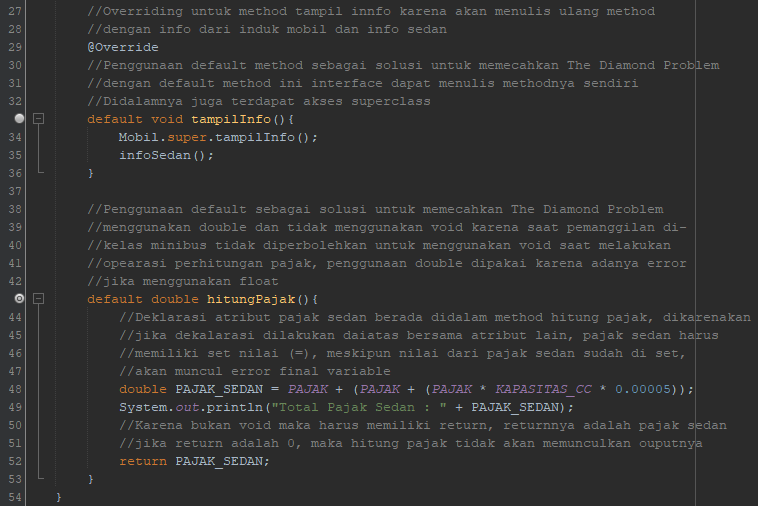
Screenshot Program

* *Interface* Mobil

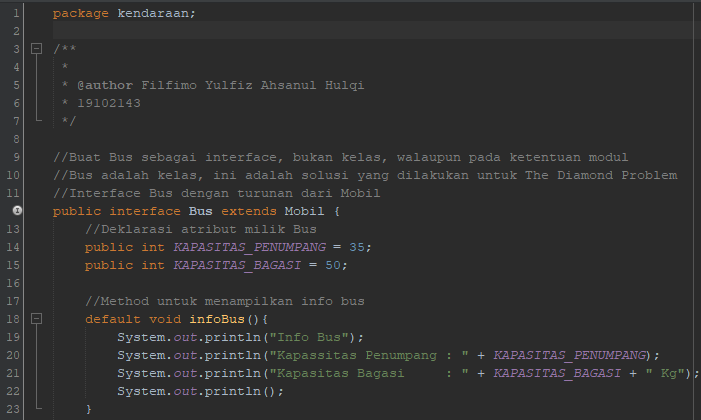


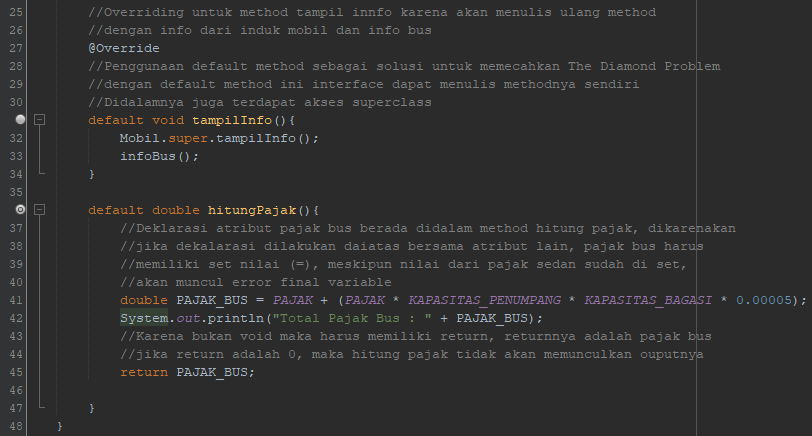
* *Interface* Sedan

**

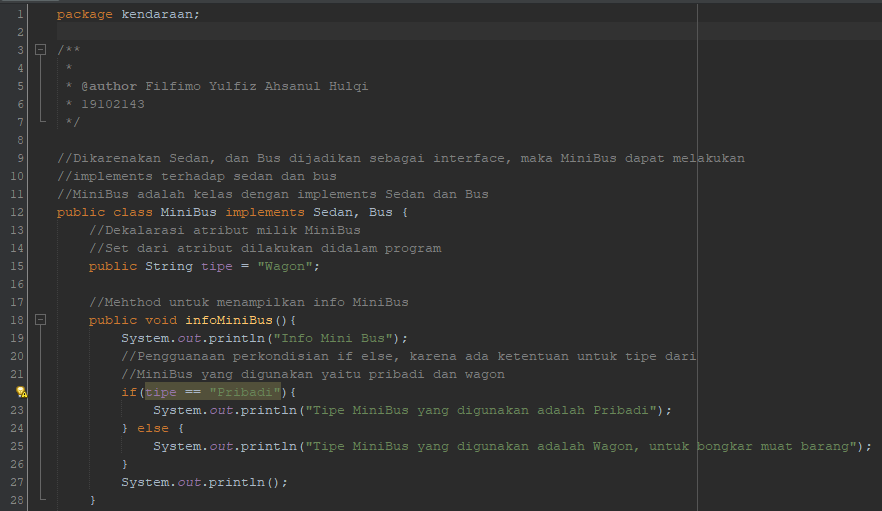
**

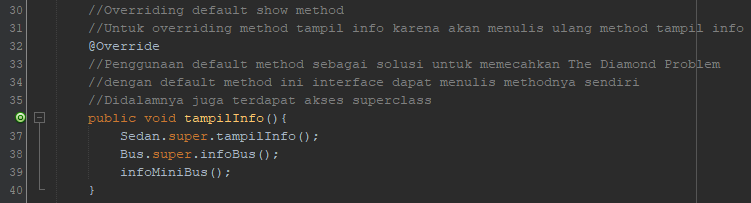
* *Interface* Bus

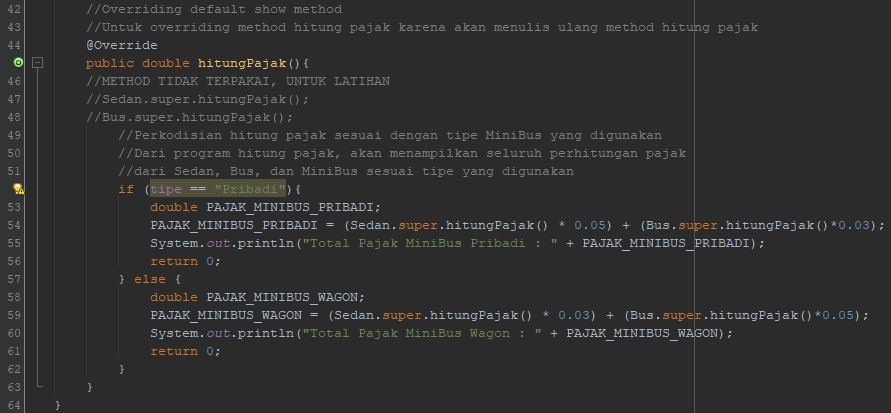
**

**

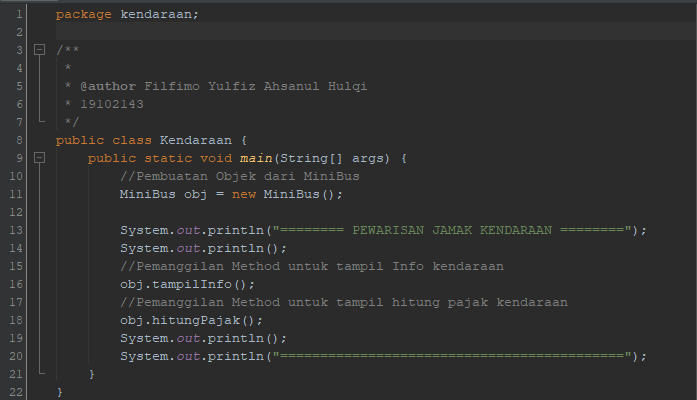
* *Class* MiniBus

**

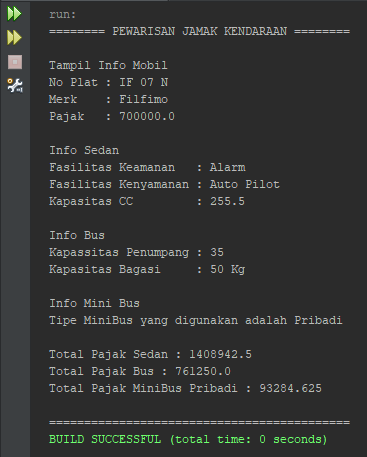
**

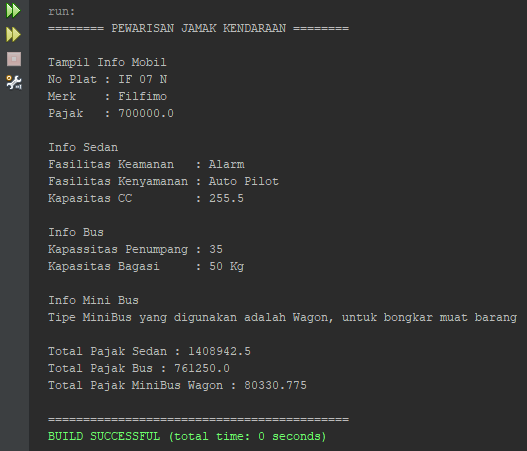
**

* *Main Class*



* Ouput





Penjelasan Program :

Untuk solusi *The Diamond Problem* adalah kita ubah ketentuan modul dari pembuatan kelas mobil, sedan, dan bus menjadi *interface* didalamnya nanti terdapat *method,* seperti yang kita tahu bahwa didalam *interface* kita tidak bisa membuat *method* tetapi hal ini dapat diatasi dengan menggunakan *default method* karena diperbolehkan dalam bahasa Java (versi 8, dan diatasnya).

Pada *interface* Mobil, deklarasikan atribut milik mobil, untuk pendeklarasian diketiga *interface* ini nanti menggunakan huruf kapital karena atribut dianggap sebagai *constant name*. selanjutnya terdapat *default* *method* tampilInfo untuk menampilkan info dari mobil.

*Pada interface* Sedan, lakukan *extends* terhadap kelas mobil, deklarasikan atribut milik sedan, selanjutnya terdapat *default method* infoSedan untuk menampilkan informasi dari sedan. *Overriding* untuk *default method* tampilInfo didalamnya terdapat pemanggilan objek milik mobil dengan menggunakan *super class*, didalam *method* tampilInfo ini memiliki info sedan dan info mobil dari induk. Selanjutnya adalah pembuatan *method* hitungPajak disini ada yang berbeda dengan *default method* sebelumnya, pada *method* hitungPajak di semua *interface* menggunakan penulisan *default double* karena untuk operasi matematika perhitungan pajak pada *class* MiniBusnanti tidak bisa memuat *void*, kemudian untuk atribut pajak sedan, dideklarasikan didalam *method default*nya sebagai *double* karena jika dideklarasikan diatas sama seperti atribut lainnya, atribut pajak sedan harus berupa *constant name* dan harus memiliki nilai, dan akan mengalami *error final variable*, dan tambahkan *return* unttuk mengembalikkan nilai, karena *method* ini sudah tidak berupa *void*.

Pada *interface* Bus, lakukan *extends* terhadap kelas mobil, deklarasikan atribut milik bus, selanjutnya terdapat *default method* infoBus untuk menampilkan informasi dari bus. *Overriding* untuk *default method* tampilInfo, didalamnya terdapat pemanggilan objek milik mobil dengan menggunakan *super class*, didalam *method* tampilInfo ini memiliki info bus dan info mobil dari induk. Selanjutnya adalah pembuatan *method* hitungPajak, untuk pembuatannya masih sama dengan *interface* sedan, hanya rumusnya saja yang berbeda.

Pada *class* MiniBus, dikarenakan Sedan dan Bus dijadikan *interface*, maka *class* MiniBus dapat melakukan *implements* atau pewarisan jamak terhadap *interface* sedan dan bus, deklarasikan atribut milik MiniBus, selanjutnya terdapat *method* info MiniBus untuk menampilkan inforrmasi dari mini bus, didalam *method* ini terdapat perkondisian *if else* terhadap tipe mini bus yang digunakan. *Overriding* untuk *default* method tampilInfo, didalamnya terdapat pemanggilan objek milik sedan dan bus menggunakan *super* *class*, didalam *method* tampilInfo ini memiliki info mini bus, info sedan, info bus, dari *interface* sedan dan bus, sebagai catatan, bahwa *interface* sedan dan bustersebut tetap dianggap sebagai induk dari *class* MiniBus. *Overriding* untuk *method* hitungPajak, seperti ketentuan pada modul bahwa *method* hitungPajak didalam *class* MiniBus adalah untuk menampilkan perhitungan pajak dari mini bus, menggunakan operasi matematika dengan rumus yang didalamnya menggunakan hasil akhir perhitungan pajak dari *interface* sedan dan bus, mengikuti ketentuan dari *interface* sedan dan bus, didalam *method* hitungPajak ini haru mempunyai *return*, tetapi nilai return di*method* ini adalah 0, rumus perhitungan pajak tadi dipisah berdasarkan perkodinsian *if else* sesuai dengan tipe mini bus yang digunakan, misalkan *method* hitung pajak dikedua *interface* tadi masih berupa *void* maka akan muncul *error void type not allowed*.

Buat *main class* untuk melakukan eksekusi program, terdapat pembuatan objek dari *class* MiniBus, pemanggilan *method* dari objekuntuk menampilkan tampil info, dan hitung pajak dari seluruh kendarann.

**BAB IV**

**Kesimpulan**

Pada Praktikum kali ini dapat diambil beberapa kesimpulan yaitu :

1. Penggunaan *super class* adalah salah satu solusi untuk memecahkan *The Diamond Problem*, *super class* sendiri untuk menunjukkan hirarki *class* yang berarti *class* dasar dari *subclass* / *class* anak.
2. *Multiple Inheritance* tidak dapat diimplementasikan lansung dalam bahasa Java.
3. *Interface* adalah solusi untuk melakukan *Multiple Inheritance* dalam bahasa Java.
4. *Interface* tidak dapat memuat *method,* hanya dapat memuat *method* kosong saja.
5. Untuk Java 8 dan terbaru, *Interface* dapat memuat *default method*, yang didalamnya bisa ditulis *behaviour*.
6. *Overriding* dapat digunakan pada *default method*.