**LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL II**

**KELAS DAN OBJEK**

****

**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.

**BAB II**

**Dasar Teori**

**Kelas dan Objek**

Objek adalah kesatuan entitas (benda), baik berwujud nyata ataupun hanya suatu sistem atau konsep yang memiliki sifat karakteristik dan fungsi. Dalam kehidupan sehari-hari sering dijumpai banyak objek dengan jenis yang sama. Contohnya sepeda Anda miliki salah satu jenis dari sepeda yang ada di dunia. Sepeda Anda adalah *instance* dari kelas yang disebut kelas sepeda.

**Objek** adalah suatu abstraksi dari suatu problem. Sedangkan **Kelas** adalah kumpulan objek yang memiliki atribut dan perilaku yang sama (karakteristik sama). Setiap objek memiliki nilai atribut/state yang unik yang membedakan dengan ibjek lain dari kelas yang sama, dan objek memiliki perilaku/*behaviour* untuk mengakses atribut/*state* yan dimilikinya.

Contoh : **Kelas Sepeda** memiliki beberapa atribut/*state* (posisi gir, kecepatan, model, dll) dan perilaku/*behaviour* yang sama (mengubah gir, mengerem, dll). **Objek Sepeda Anda** memilikki nilai atribut yang berbeda bila dibandingkan dengan objek sepeda yang lain, tetapi **Sepeda Anda** dan sepeda-sepeda yang lain memilii perillaku/*behaviour* yang sama.

**Kelas** adalah pemodelan dari objek yang berisi informasi (aturan) tentang sifat karakteristik (data) dan tingkah laku (*method*) yang dimiliki oleh objek tersebut. Kelas dapat dianalogikan sebagai struktur data dari objek. Perbedaan kelas pada pemrograman berorientasi objek dengan struktur data pada pemrograman terstruktur adalah abhwa kelas pada pemrograman berorientasi objek tidak hanya berisi data saja tetapi juga fungsi-fungsi yang mengaksesnya, sehingga data dan fungsi harus dirancang bersamaan (**kelas = struktur data + fungsi**).

**Mendefiniskan Kelas**

Elemen-elemen dasar dalam mendefinisikan kelas :

1. *Fields*/*Variable*

*Field* atau *variable* adalah implementasi dari atribut suatu objek. *Field*/*Variable* digunakan untuk menyimpan data dari objek.

Jenis *Field*/*Variable* :

* *Instance Variable*

Setiap objek memiliki salinan sendiri dan salinan tersebut memiliki nilai masing-masing.

* *Class Variable*

Suatu kelas hanya memiliki satu *variable* jenis ini dan digunakan bersama oleh semua objek dari kelas tersebut.

1. Konstruktor/*Constructor*

Konstruktor adalah *method* yang berfungsi untuk menginisialisasi *variable-variable* *instance* yang akan dimilki oleh objek. *Method* konstruktor harus memiliki nama yang sama dengan nama kelas. Konstruktor ini dipanggil pada saat proses instansiasi kelas menjadi objek.

Kegunaaan Konstruktor :

* Mengalokasiskan ruang bagi sebuah objek.
* Memberikan nilai awal terhadap anggota data suatu objek.
* Membentuk tugas-tugas umum lainya.

Perlu diketahui :

* Konstruktor tidak mempunyai nilai balik (bahkan tanpa *void*).
* Konstruktor harus diletakkan pada bagian *public*.

Karakteristik Konstruktor :

* Tidak pernah memiliki nilai balikan.
* Memiliki nama sama dengan nama kelas.

1. *Methods*

*Method* merupakan fungsi-fungsi implementasi perilaku objek untuk mengakses atribut-atributnya.

**BAB III**

**Penjelasan**

**GUIDED**

1. Hubungan antara kelas objek, atribut dan *method*.

**SUB**

package buku;

public class Buku {

public String judul;

public String pengarang;

public int jumlah;

public void setNilai(String judul, String pengarang, int jumlah){

this.judul = judul;

this.pengarang = pengarang;

this.jumlah = jumlah;

}

void Display(){

System.out.println("Judul : " + judul);

System.out.println("Pengarang : " + pengarang);

System.out.println("Jumlah : " + jumlah);

}

}

**MAIN**

package buku;

public class MainBuku {

public static void main(String[] args) {

// TODO code application logic here

Buku a = new Buku();

//Pemanggilan Method

a.setNilai("Jurassic Park", "Michael Crichton", 21);

a.Display();

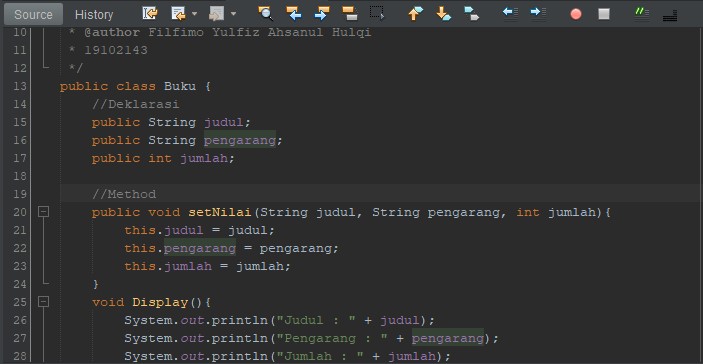
}

}

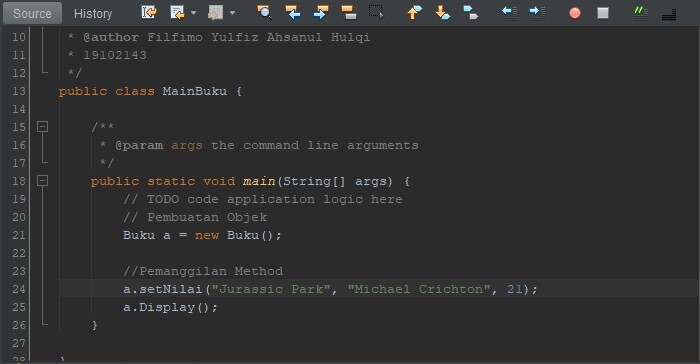
*Source code* Java

**Screenshot Program :**

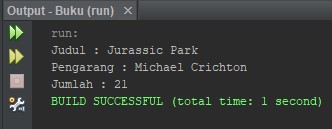
* *Sub class*



* *Main class*



* *Output*



**Penjelasan Program :**

Pada program diatas adalah contoh pemrogram kelas dan objek tanpa menggunakan konstuktor, jika langsung *running program* sesuai dengan modul, maka program akan mengalami *error* dibagian deklarasi, untuk mengatasi nya adalah dengan mengganti *private* menjadi *public*.

1. Menggunakan Konstruktor

**SUB**

package buku2;

public class Buku2 {

public String judul;

public String pengarang;

public Buku2(){

judul = "TIDAK DIKETAHUI";

pengarang = "TIDAK DIKETAHUI";

}

public Buku2(String judul, String pengarang){

this.judul = judul;

this.pengarang = pengarang;

}

void Display(){

if(judul == null && pengarang == null){

}

else{

System.out.println("Judul : " + judul);

System.out.println("Pengarang : " + pengarang);

}

}

}

**MAIN**

package buku2;

public class MainBuku2 {

public static void main(String[] args) {

// TODO code application logic here

Buku2 a,b = new Buku2();

System.out.println("Konstruktor Sedang Dijalankan....");

a = new Buku2("Bumi", "Tere Liye");

b = new Buku2();

a.Display();

b.Display();

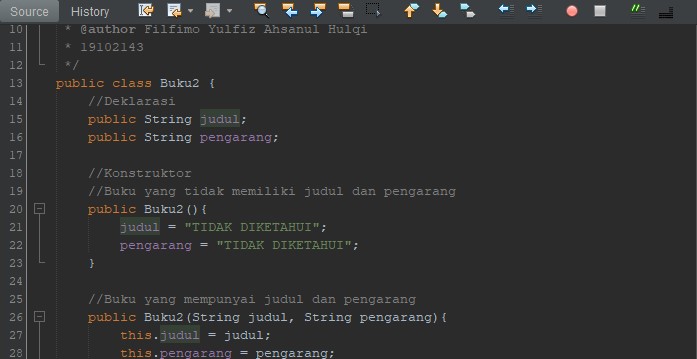
}

}

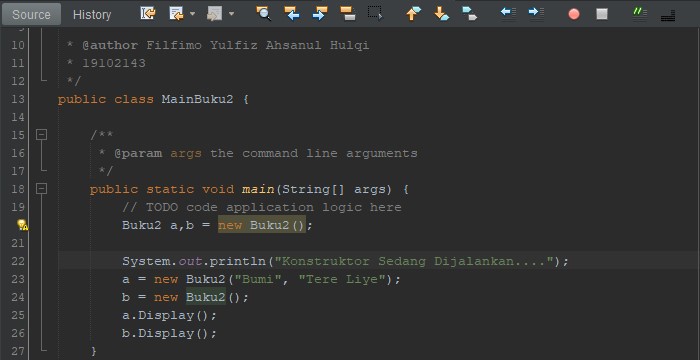
*Source code* Java

**Screenshot Program :**

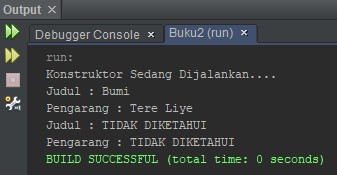
* *Sub Class*



* *Main Class*



* *Ouput*



**Penjelasan Program :**

Pada program diatas adalah contoh pemrogram kelas dan objek dan menggunaka konstruktor, jika langsung *running program* sesuai dengan modul, maka program akan mengalami *error* dibagian deklarasi dan *method* *display*, untuk mengatasi nya adalah dengan mengganti deklarasi *private* menjadi *public* dan menambahkan *else* pada *method display*.

Dapat diperhatikan juga adalah bagaimana penulisan konstruktor, penulisan konstruktor harus sama persis dengan kelas yang digunakan, juga jika nama konstruktor berbeda dengan kelas, maka konstruktor tidak dapat dijalankan.

**UNGUIDED**

1. Buatlah program dalam bahasa **Java** dimana diminta untuk mengimplementasikan kasus operasi terhadap suatu Bank dalam konsep PBO

package bank;

public class Bank {

int saldo;

Bank (int saldo){

this.saldo = saldo;

}

public void saldoAwal(int uang){

this.saldo = uang;

System.out.println("Saldo Uang Saat ini = Rp."+uang);

}

public void simpanUang(int uang){

this.saldo += uang;

System.out.println("Simpan Uang = Rp."+uang);

getSaldo();

}

public void ambilUang(int uang){

this.saldo -= uang;

System.out.println("Ambil Uang = Rp."+uang);

getSaldo();

}

public void getSaldo(){

System.out.println("Saldo saat ini = Rp."+this.saldo);

}

}

*Source code* Java *Sub Class* dari Bank

*Source code* Java *Main Class* dari Bank

package bank;

public class MainBank {

public static void main(String[] args) {

Bank bank = new Bank(100000);

bank.saldoAwal(100000);

System.out.println();

bank.simpanUang(500000);

System.out.println();

bank.ambilUang(150000);

System.out.println();

}

}

**Screenshot Program :**

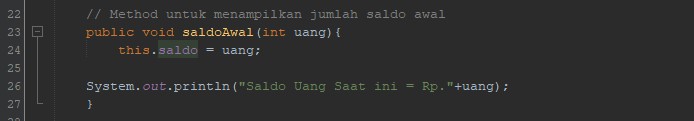
* *Sub Class*



Deklarasi Fields

Penjelasan *Souce code* :

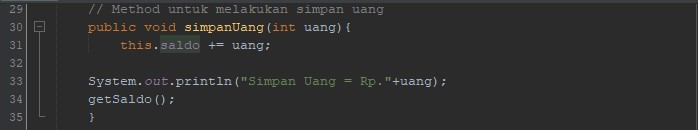
Pada *sub class* harus mendeklarasikan *fields* terlebih dahulu yaitu int saldo dan membuat sebuah konstruktor dengan paramaternya adalah saldo yang akan di *set* oleh *user* itu sendiri.



*Method* saldoAwal

Penjelasan *Souce code* :

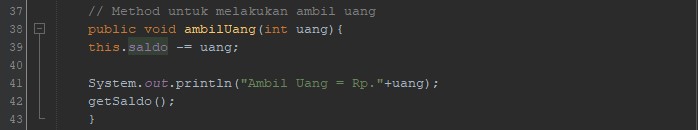
Selanjutnya disini menambahkan *method*  baru, yang tidak ada ketentuan di modul yaitu *method* saldo awal, *method* ini untuk menampilkan jumlah saldo awal.



*Method* simpanUang

Penjelasan *Souce code* :

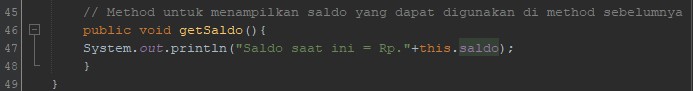
Selanjutnya disini menambahkan *method* baru yaitu *method* untuk melakukan simpan uang, sesuai dengan namanya *method* ini akan melakukan penjumlahan saldo awal dengan uang yang telah di-*set*.



*Method* ambilUang

Penjelasan *Souce code* :

Selanjutnya disini menambahkan *method* baru yaitu *method* untuk melakukan ambil uang, sesuai dengan namanya *method* ini akan melakukan pengurangan saldo dari saldo yang telah disimpan oleh *method* simpan uang.

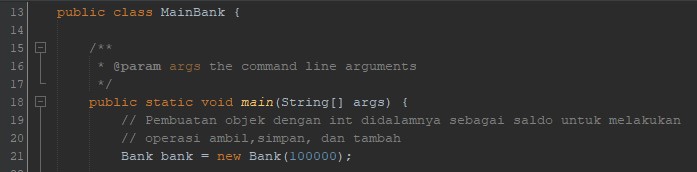


*Method* getSaldo

Penjelasan *Souce code* :

Selanjutnya adalah *method*  baru yaitu *method* untuk mengeluarkan output saldo, nama dari *method* ini adalah getSaldo, *method* ini cukup sederhana karena hanya melakukan ouput dari jumlah saldo akhir, *method* ini juga dapat digunakan pada *method* yang lain.

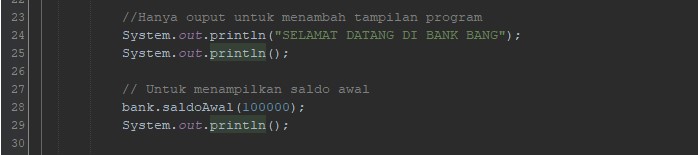
* *Main class*



Pembuatan objek

Penjelasan *Souce code* :

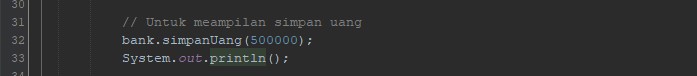
Pada *main class*, pertama deklarasikan terlebih dahulu pembuatan objek dan masukan juga nilai dari saldo bank.



Pemanggilan method saldoAwal

Penjelasan *Souce code* :

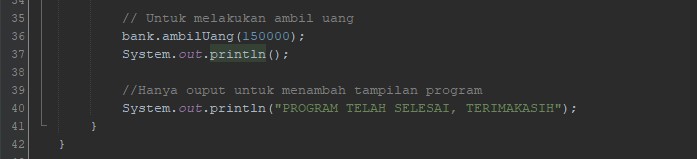
Pemanggilan fungsi *method* saldoAwal dari *sub class*, masukan juga nilai dari saldo awal.



Pemanggilan method simpanUang

Penjelasan *Souce code* :

Pemanggilan fungsi *method* simpanUang dari *sub class*, masukan juga nilai dari saldo yang akan disimpan, dan nantinya nilai yang di-*set* inilah yang akan masuk ke *sub class* untuk dilakukan operasi aritmatik.



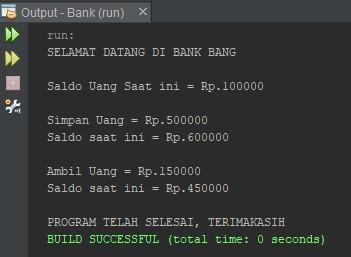
Pemanggilan method ambilUang

Penjelasan *Souce code* :

Sama seperti sebelumnya. Pemanggilan fungsi *method* ambilUang dari *sub class*, masukan juga nilai dari saldo yang akan diambil, dan nantinya nilai yang di-*set* inilah yang akan masuk ke *sub class* untuk dilakukan operasi aritmatik.

Sisanya hanyalah baris program untuk menambah tampilan output agar lebih menarik

* *Ouput Program*



Tampilan *running program*

**Penjelasan Program :**

Pada program Bank diatas adalah program dimana akan menampilkan output berupa saldo awal, simpan uang, dan ambil uang. Saldo di-*set* didalam program.

Tidak ada inpur dari *user*, kemudian dari saldo tadi akan dilakukan operasi simpan uang berupa penambahan saldo ditambah dengan uang yang akan disimpan, dan operasi ambil uang berupa pengurangan saldo dengan uang yang akan diambil, uang yang akan disimpan dan uang yang akan diambil juga tidak melalui input *user* untuk menentukan nominalnya, tetapi sudah di-*set* juga didalam program.

1. Buatlah program bahasa Java dimana diminta untuk mengimplementasikan kasus operasi terhadap suatu Titik dalam konsep PBO

package titik;

import java.util.Scanner;

import static java.lang.Math.\*;

public class Titik {

Scanner masukan = new Scanner(System.in);

double titik;

int skalar;

public float x1;

public float y1;

public float x2;

public float y2;

public Titik(float x1, float y1, float x2, float y2){

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

}

void inisialisasiTitik(){

System.out.println("-Input Data Titik-");

System.out.println("Masukkan X :");

x1=masukan.nextFloat();

System.out.println("Masukkan Y :");

y1=masukan.nextFloat();

System.out.println("Program Selesai melakukan Input");

System.out.println();

}

void tampilTitik(){

System.out.println("Berikut adalah titik X&Y");

System.out.println("Titik X adalah = " +x1);

System.out.println("Titik Y adalah = " +y1);

System.out.println("PERHATIAN : TITIK YANG DITAMPILKAN HANYA TITIK 1");

System.out.println("Program Selesai menampilkan Input");

System.out.println();

}

*Source code Sub class* bagian 1

*Source code Sub class* bagian 2

void perkalianSkalar(){

System.out.println("Perkalian Skalar");

System.out.println("Masukkan Nilai Skalar : ");

skalar = masukan.nextInt();

x1 = x1\*skalar;

y1 = y1\*skalar;

System.out.println

("Nilai X dan Y setelah dikalikan dengan Skalar : "+ x1 + " dan " + y1);

}

void pencerminanSumbuX(){

System.out.println("Pencerminan Sumbu X");

System.out.println("Hasilnya adalah :");

System.out.println("X = " + x1\*1);

System.out.println("Y = " + y1\*-1);

System.out.println("Program Selesai melakukan pencerminan Sumbu X");

System.out.println();

}

void pencerminanSumbuY(){

System.out.println("Pencerminan Sumbu Y");

System.out.println("Hasilnya adalah :");

System.out.println("X = " + x1\*-1);

System.out.println("Y = " + y1\*1);

System.out.println("Program Selesai melakukan pencerminan Sumbu Y");

System.out.println();

}

void jarak(){

System.out.println("-Input Data Titik 2-");

System.out.println("Masukkan X :");

x2=masukan.nextFloat();

System.out.println("Masukkan Y :");

y2=masukan.nextFloat();

System.out.println();

titik = (sqrt(((x2-x1)\*(x2-x1))+((y2-y1)\*(y2-y1))));

System.out.println("Jadi Jaraknya adalah " + titik);

System.out.println();

}

}

*Source code Main class* bagian 1

package titik;

import java.util.Scanner;

public class MainTitik {

public static void main(String[] args) {

Titik a = new Titik(0,0,0,0);

Scanner baca = new Scanner(System.in);

int pilih;

do{

System.out.println("MENU OPERASI TITIK");

System.out.println("1. Inisialisasi Titik");

System.out.println("2. Tampil Titik");

System.out.println("3. Perkalian Skalar Titik");

System.out.println("4. Pencerminan Terhadap Sumbu X");

System.out.println("5. Pencerminan Terhadap Sumbu Y");

System.out.println("6. Jarak Antara Dua Titik");

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

System.out.println("Masukkan Pilihan : ");

pilih = baca.nextInt();

switch(pilih){

case 1:

a.inisialisasiTitik();

break;

case 2:

a.tampilTitik();

break;

case 3:

a.perkalianSkalar();

break;

case 4:

a.pencerminanSumbuX();

break;

case 5:

a.pencerminanSumbuY();

break;

case 6:

a.jarak();

break;

case 0:

System.out.println("PROGRAM SELESAI");

System.out.println();

break;

*Source code Main class* bagian 2

default:

System.out.println("Tidak ada layanan");

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

System.out.println();

break;

}

} while (pilih != 0);

}

}

**Screenshot Program :**

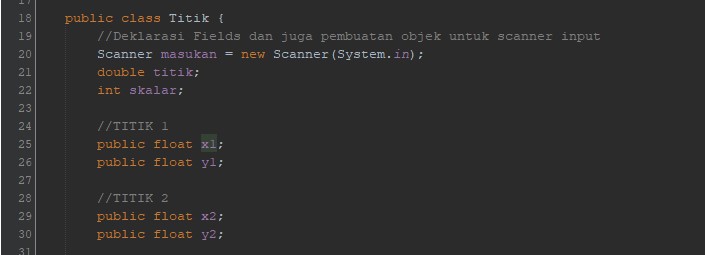
* Sub class



*Library*

Penjelasan *Souce code* :

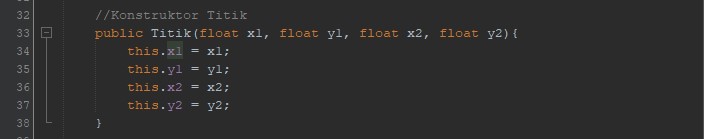
Import library yang akan digunakan yaitu *scanner* karena data akan ada input dari *user* langsung, dan *math* karena akan menggunakan operasi matematika pada *method* 6.



Deklarasi *fields* dan objek

Penjelasan *Souce code* :

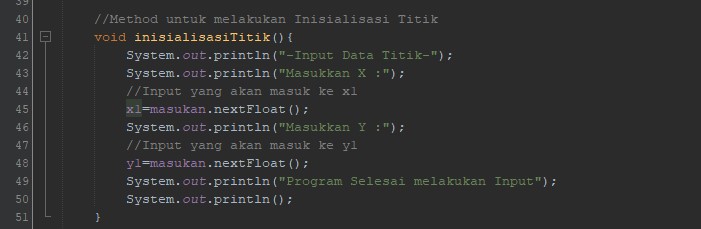
Deklarasi fields yang akan digunakan, ada juga pembuatan objek *scanner* masukan. Dan titik milik titik 1 dan titik 2, disini sedikit berbeda dimana data yang digunakan float.



Pembuatan Konstruktor

Penjelasan *Souce code* :

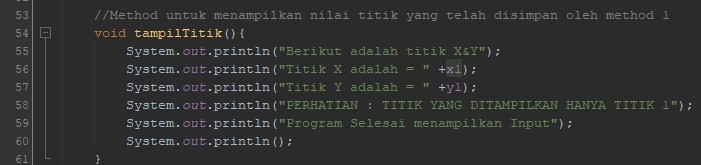
Pembuatan konstruktor titik.



*Method* inisialisasiTitik

Penjelasan *Souce code* :

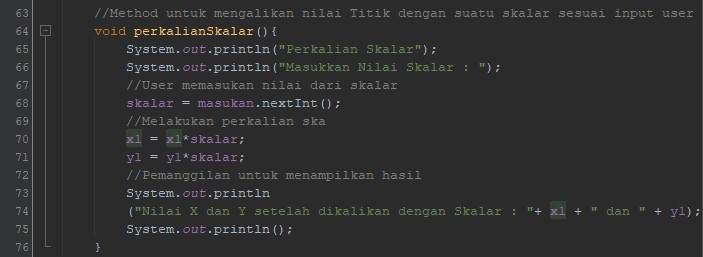
*Method* inisialisasi titik, disini *user* dapat melakukan input nilai titik secara langsung.



*Method* tampilTitik

Penjelasan *Souce code* :

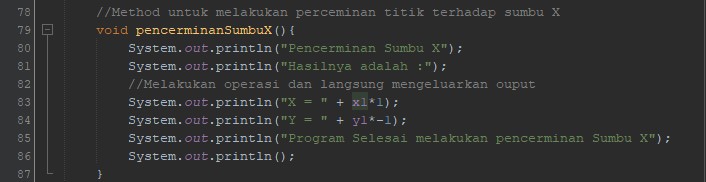
*Method* tampul titik, disini *user* dapat memanggil *ouput* untuk menampilkan titik yang telah di-*input* pada *method* inisialisasi titik tadi.



*Method* perkalianSkalar

Penjelasan *Souce code* :

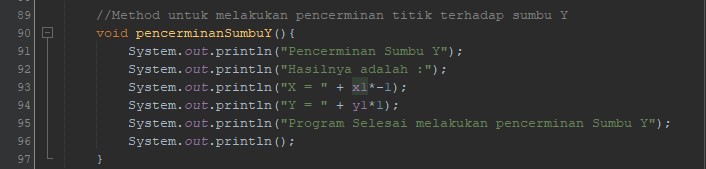
*Method* perkalian skalar, disini program akan mengalikan titik yang telah di-*inputkan* dengan skalar, dalam *method* ini juga *user* dapat melakukan *input* nilai untuk skalarnya setelah ditentukan skalar yang telah di-*input*kan barulah program akan melakukan operasi perkalian.



*Method* pencerminanSumbuX

Penjelasan *Souce code* :

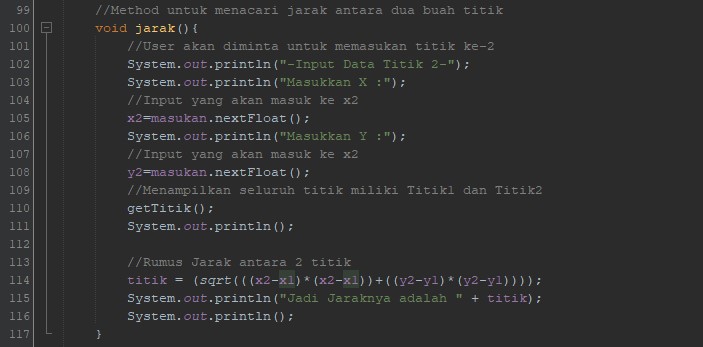
*Method* pencerminan sumbu x, disini program akan mencerminkan titik yang telah di-*input* oleh *user* pada *method* inisialisasi titik, dimana nanti nilai dari y akan berubah menjadi negatif.



*Method* pencerminanSumbuY

Penjelasan *Souce code* :

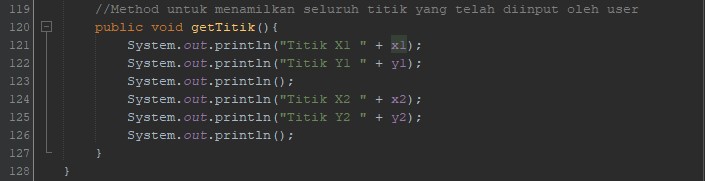
*Method* pencerminan sumbu y, disini program akan mencerminkan titik yang telah di-*input*, kebalikan dari *method* sebelumnya disini nilai yang akan menjadi negatif adalah x.



*Method* jarak

Penjelasan *Souce code* :

*Method* jarak, disini program akan mencari jarak antara dua titik, dikarenakan sebelumnya *user* hanya meng-*input* satu titik saja, didalam *method* ini *user* diminta memasukan *input* lagi untuk data titik kedua. Setelah *user* selesai melakukan inputnya akan muncul *ouput* dari titik masing-masing. Kemudian program akan mencari jarak antar kedua titik tersebut dengan rumus yang telah ditentukan pada modul



*Method* getTitik

Penjelasan *Souce code* :

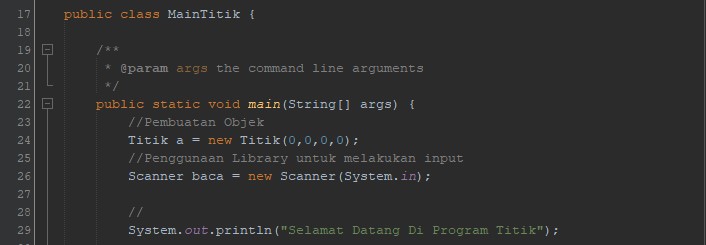
*Method* getTitik, sebenarnya ini hanya *method* tambahan yang tidak wajib untuk dibuat, tapi Saya buat karena untuk memudahkan pembacaan informasi titik pada *method* jarak.

* *Main class*



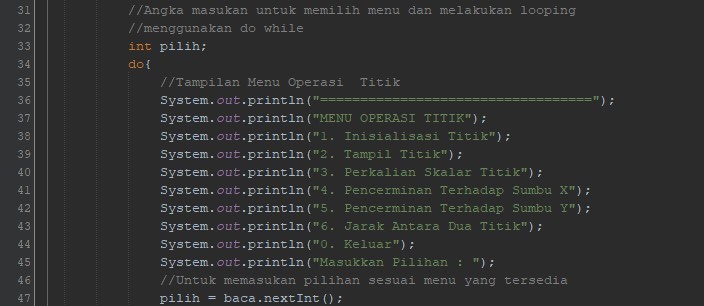
Penjelasan *Souce code* :

*Import library* yang akan digunakan yaitu *scanner* karena akan ada interaksi *input* oleh *user*.



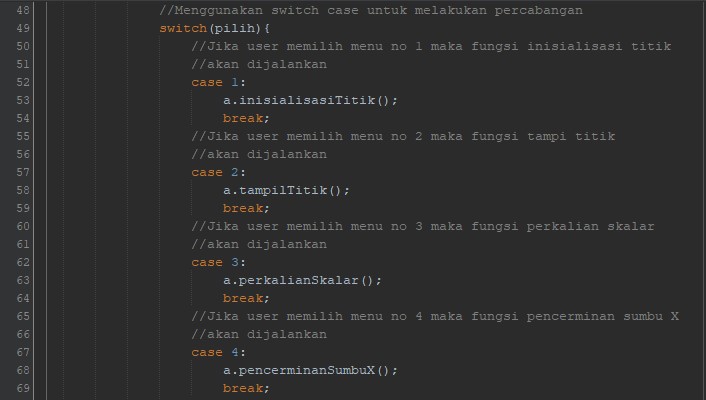
Penjelasan *Souce code* :

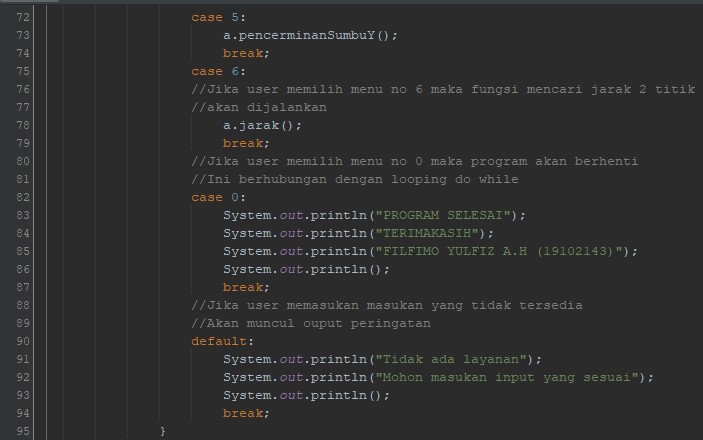
Pembuatan objek dari Konstruktor Titik yang telah dibuat, jangan lupa untuk *set* nilai konstruktor disini *set* dengan nilai 0,0,0,0 yang nantinya akan menjadi titik milik x1, y1, x2, dan y2.

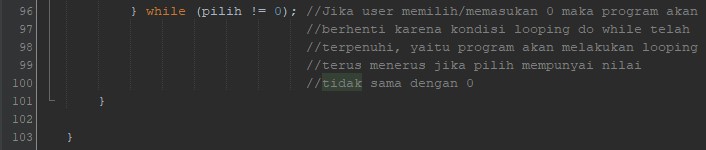


Penjelasan *Souce code* :

Untuk membuat *output* menu, juga ada deklarasi *int* pilih, dimana pilih ini akan menjadi *input* untuk memilih menu dan juga untuk melakukan *looping* kondisi *do while*.





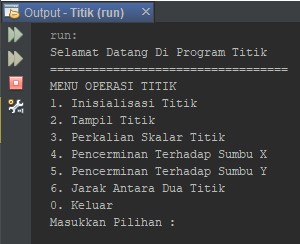


Penjelasan *Souce code* :

Setelah sebelumnya membuat banyak *method* pada *sub class* disinilah *method* tersebut akan dipanggil, juga tidak lupa menggunakan *swith case*, *switch case* sendiri adalah model lain dari percabang *if else*, dimana *input* yang *user* pilih saat memilih menu akan menentukan *case* mana yang akan dijalankan, kemudian *case* tersebut akan menjalankan *method* yang ada didalamnya.

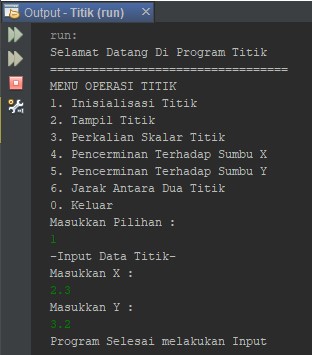
Selanjutnya adalah *while*, seperti yang telah disinggung *looping* ini adalah kondisi, dimana program akan terus *running* jika kondisi pilih tidak sama dengan nol.

* *Running Program* :



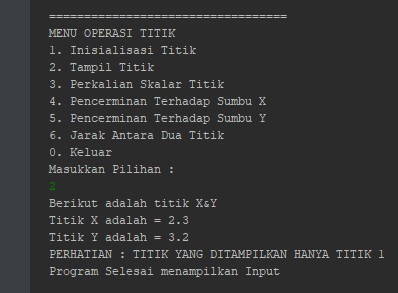
**Penjelasan *Program* :**

Tampilan menu saat *running program*.



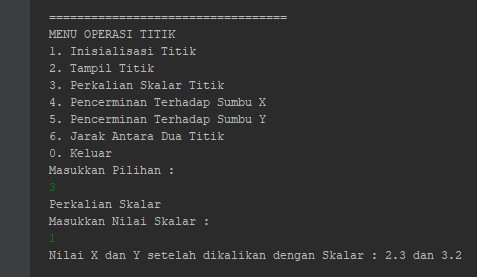
**Penjelasan *Program* :**

Jika *user* memilih menu 1, maka *user* akan diminta untuk memasukan nilai x dan y.



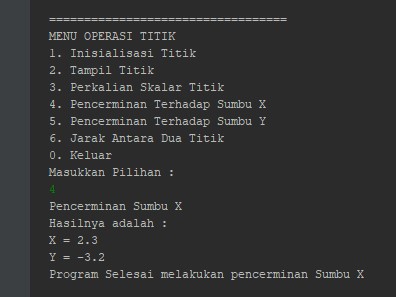
**Penjelasan *Program* :**

Jika user memilih menu 2, maka program akan melakukan *ouput* nilai dari *input*an x dan y yang telah dilakukan pada menu 1.



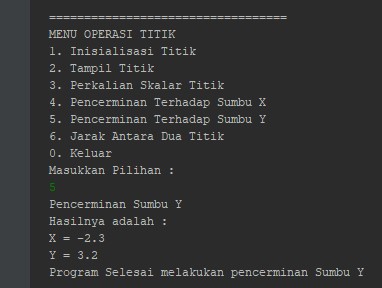
**Penjelasan *Program* :**

Jika user memilih menu 3, maka program akan melakukan perkalian skalar, *user* diminta untuk *input* nilai skalar.



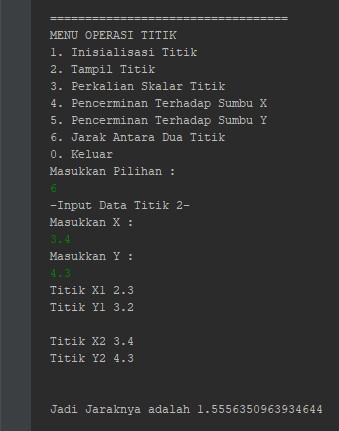
**Penjelasan *Program* :**

Jika *user* memilih menu 4, maka program akan melakukan pencerminan sumbu x, dengan titik yang telah di*input*kan oleh *user* pada menu 1.



**Penjelasan *Program* :**

Jika *user* memilih menu 5, maka program akan melakukan pencerminan sumbu y, dengan titik yang telah di*input*kan oleh *user* pada menu 1.



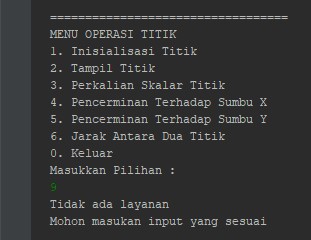
**Penjelasan *Program* :**

Jika *user* memilih menu 6, maka program akan melakukan pencarian antara dua titik, *user* akan diminta *input* nilai x dan y lagi sebagai titik 2. Kemudian program juga akan menampilkan titik 1 dan titik 2.



**Penjelasan *Program* :**

Jika *user* memilih menu 0, maka program akan otomatis berhenti dan keluar.



**Penjelasan *Program* :**

Jika *user* memilih menu yang tidak sesuai dengan menu, maka program akan mengeluarkan *ouput* peringatan, bahwa *input* yang dimasukan user tidak ada.

**BAB IV**

**Kesimpulan**

Pada praktikum kali ini dapat diambil beberapa kesimpulan, yaitu :

1. Pemrograman berorientasi objek (OOP), membuat kode program lebih mudah dibaca karena karakteristiknya.
2. Pada Java untuk membuat OOP, diperlukan *main class* dan *sub* *class*, yang nanti akan saling berhubungan satu sama lain.
3. Kelas dan Objek saling berhubungan, dan harus sangat diperhatikan saat pembuatannya baik itu *fields*, *constructor*, dan *method*.
4. Kelas dan Objek, membuat kode program menjadi lebih mudah untuk dibaca baik saat *running* progam atau saat sedang melakukan *debug*.