**LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL I**

**PENGENALAN PEMROGRAMAN BERORIENTASI 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 1**

**Tujuan Praktikum**

Mahasiswa diharapkan mampu untuk :

1. Mengenali dan memahami environment dari NetBeans.
2. Mengenali dan memahami tentang pemrograman berorientasi objek.
3. Mengenali dan memahami bahasa pemrograman Java.
4. Dapat membedakan perbedaan pemrograman terstruktur dengan pemrograman berorientasi objek.
5. Dapat membedakan perbedaan program C++ dan Java.

**BAB II**

**Dasar Teori**

Pengantar Pemrograman Berorientasi Objek

Pemrograman berorientasi objek atau OOP (*Object Oriented Programming*) merupakan teknik membuat suatu program berdasarkan objek dan apa yang bisa dilakukan objek tersebut. *Object*-*oriented* program terdiri dari objek-objek yang saling berinteraksi satu sama lain untuk menyelesaukan sebuah tugas.

OOP sering digunakan karena memudahkan programmer untuk mem-*breakdown* kode untuk di-*manage*. Breakdown berdasarkan objek-objek yang ada pada program tersebut. Dianjurkan diimplemnetasikan untuk program dengan berbagai ukuran karena lebih mudah untuk melakukan *debugging*.

Berbeda dengan pemrograman prosedural yang mengatur program dalam barisan-barisan linier yang bekerja dari atas ke bawah. Kumpulan tahapan yang dijalankan setelah yang lain berjalan. Baik untuk program kecil yang berisi sedikit baris kode. Tidak dianjurkan diimplementasikan pada program berukuran besar, karena jika terdapat *error* akan sulit untuk melakukan *debugging*.

Karakteristik OOP :

1. Abstraksi

Memungkinkan seorang *developer* memerintahkan suatu gungsi, tanpa harus mengetahui bagaimana fungsi tersebut bekerja.

1. Enkapsulasi

Prose pemaketan data objek bersama method-methodnya, berbagai objek yang berada dalam *class* tersebut dapat berdiri sendiri tanpa terpengaruh oleh yang lainnya.

1. *Inheritance*

Penciptaan kelas baru (*subclass*/kelas turunan) dengan mewarisi karakteristik dari kelas yang sudah ada (*superclass*/kelas induk), ditambah karakteristik unik kelas baru itu.

1. *Reuseability*

Dapat menggunakan kembali kelas yang sudah ada.

1. *Polymorphism*

Kemampuan suatu pesan atau data untuk diproses lebih dari satu bentuk.

Kelebihan OOP :

1. *Parallel Development*

Ketika berkerja bersama dengan tim, masing-masing programmer dapat membangun *class* sendiri. Dengan membangun *class* secara sendiri, komponen yang sudah dibentuk kemudian dapat digabungkan menjadi satu keastuan.

1. *Scalability*

Berbagai prinsip yang dimiliki OOP bertujuan untuk mempermudah kebutuhan program yang lebih luas atau rumit. Hal ini membuat jika terjadi perkembangan dari program yang sudah ada, menambahkan beberapa fungsi, *object*, atau *class* lainnya akan menjadi lebih mudah.

Kekurangan OOP :

1. Tidak efisien

Menggunakan OOP akan lebih memakan daya pada CPU yang digunakan. Oleh karena itu, sangat disarankan untuk menggunakan perangkat terbaru saat melakukan pengembangan dengan OOP.

1. Membutuhkan manajemen data yang ketat

Perlunya kontrol yang cukup ketat terhadap kode-kode tersebut. Hal ini karena OOP akan memunculkan beberapa kode-kode baru.

1. Kemungkinan duplikasi

Dengan kemudahan OOP, mengembangkan sebuah program baru dari yang setelah adaa menjadi cukup mudah. Namun hal ini justru membuat berbagai *project* yang dibuat akan terasa seperti sekedar duplikasi saja.

**BAB III**

**Penjelasan**

**GUIDED I**

1. Membandingkan Program Terstruktur dan Program Berorientasi Objek
2. Program Terstruktur C++

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

using namespace std;

void helloWorld(int jmlh\_looping){

for(int i=0;i<jmlh\_looping;i++){

cout<<i+1<<". Hello World...."<<endl;

}

}

int tambah(int bil1,int bil2){

return(bil1+bil2);

}

int main(){

int bil1=5,bil2=2;

int jmlh\_looping=5;

cout<<endl;

cout<<"-Display Hello World-"<<endl;

helloWorld(jmlh\_looping);

cout<<endl<<"-Pemanggilan Fungsi Tambah-"<<endl;

cout<<"Hasil Operasi Tambah : "<<tambah(bil1,bil2);

cout<<endl;

getch();

return 0;

}

*Source code* Struktural C++

1. Pemrograman Berorientasi Objek C++

#include <iostream>

#include <conio.h>

using namespace std;

class DemoObjectOriented{

public:

void helloWorld(int jmlh\_looping);

int tambah(int bil1,int bil2);

};

void DemoObjectOriented::helloWorld(int jmlh\_looping){

for(int i=0;i<jmlh\_looping;i++){

cout<<i+1<<". Hello World...."<<endl;

}

}

int DemoObjectOriented::tambah(int bil1, int bil2){

return bil1+bil2;

}

int main()

{

int bil1=5, bil2=2;

int jmlh\_looping=5;

DemoObjectOriented Demo;

cout<<endl;

cout<<"-Display Hello World-"<<endl;

Demo.helloWorld(jmlh\_looping);

cout<<endl<<"-Pemanggilan Fungsi Tambah-"<<endl;

cout<<"Hasil Operasi Tambah : "<<Demo.tambah(bil1,bil2);

cout<<endl;

getch();

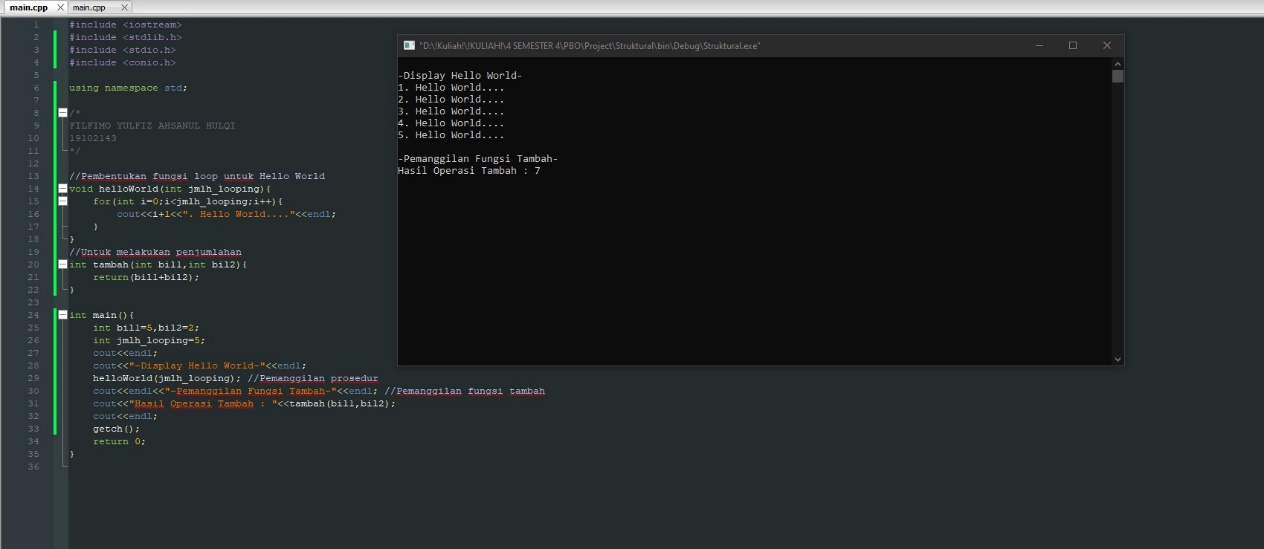
return 0;

}

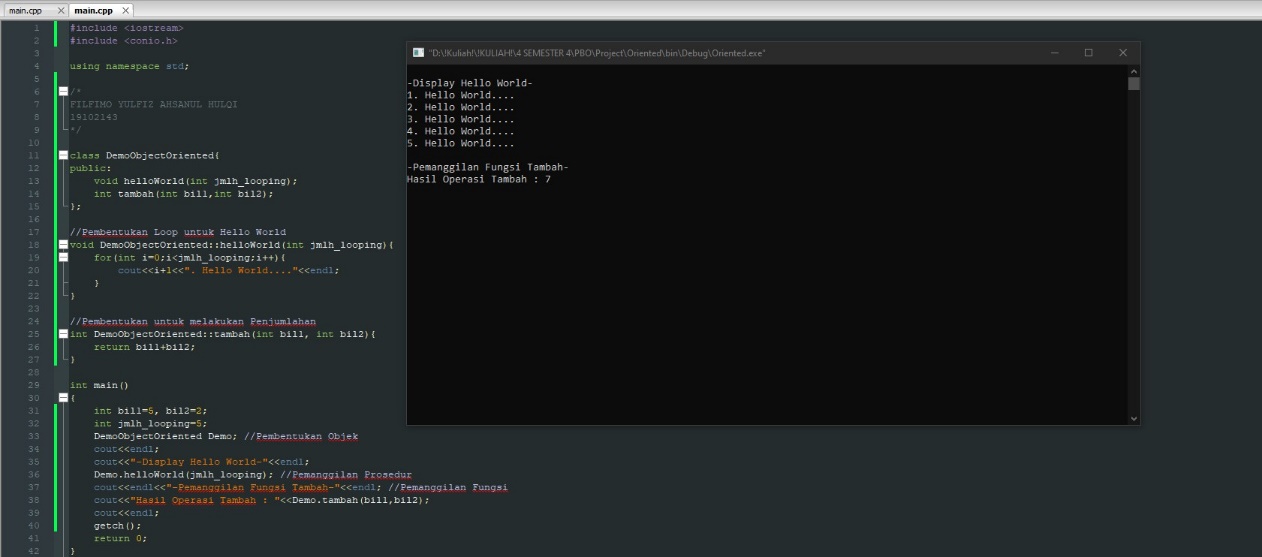
*Source code OOP* C++

Screenshot Program :

* Struktural C++



* OOP C++



1. Pemrograman Berorientasi Objek Java

**MAIN**

package demoapp;

public class MainDemoApp {

public static void main(String[] args) {

int jmlh\_looping=5;

int bil1=5;

int bil2=2;

DemoOOP Demo = new DemoOOP();

Demo.helloWorld(jmlh\_looping);

System.out.println("\n-Pemanggilan Fungsi Tambah-\n");

System.out.println("Hasil Operasi Tambah :"+Demo.tambah(bil1, bil2));

}

}

**SUB**

package demoapp;

import java.io.\*;

public class DemoOOP {

public void helloWorld(int jmlh\_looping){

System.out.println("\n-Display Hello World");

for(int i=0;i<jmlh\_looping;i++){

System.out.print(+(i+1)+". Hello World....\n");

}

}

public int tambah(int bil1, int bil2){

return(bil1+bil2);

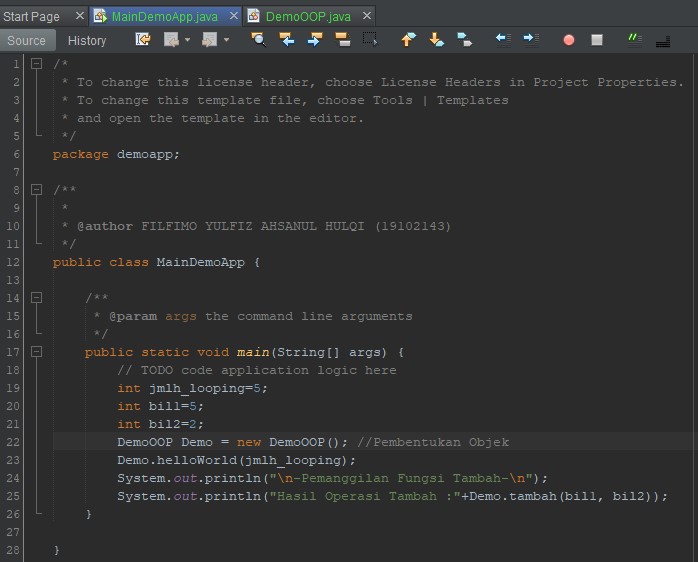
}

}

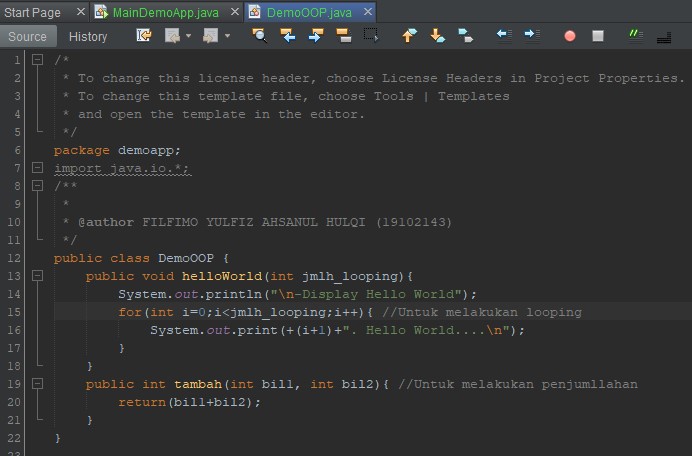
*Source code OOP* Java

Screenshot Program :

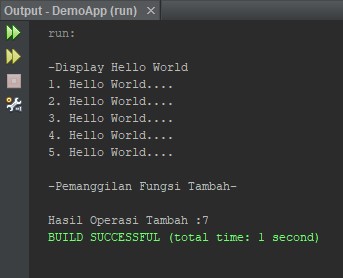
* *Main Class*



* *Sub Class*



* *Ouput* program



Penjelasan Program C++ :

Pada program diatas adalah contoh pemrogram struktural dan berorientasi objek yang diimplementasikan di C++, jika langsung *running* program sesuai modul, maka program akan mengalami *error* karena tidak adanya “*using namespace std;*”.

Penjelasan Program Java :

Pada program diatas adalah contoh pemrograman berorientasi objek yang diimplementasikan di Java. Pada Java kita harus membuat minimal 2 buah file, yang satu sebagai *main class* dan satunya sebagai *sub* *class*, jika langsung *running* program sesuai modul, tidak dapat langsung *running* karena adanya masalah, yaitu penamaan dari “jumlah looping” pada *main class* dan *sub class* berbeda. Kemudian yang perlu diperhatikan selanjutnya adalah saat pembetukan objek *sub class* pada *main class*, yang harus sesuai dengan nama file yang user *setting*.

Perbedaan Program :

Dari tampilan output secara sekilas program diatas tidak memiliki perbedaan. Perbedaan dari program diatas dapat terlihat dari *source code* nya, dapat dilihat bahwa penulisan kode jelas sangat berbeda, dimana pemrograman terstruktur tidak membuat kelas dan langsung dijalankan dari atas ke bawah secara bergantian, sedangkan pemrograman berorientasi objek membuat kelas terlebih dahulu, kelas-kelas tersebut nantinya akan dapat dipanggil/digunakan di *int main*. Sedangkan pada Java, harus membuat dua buah file sebagai *main class* dan *sub class*.

Jadi user melakukan *programming* di bagian pembentukan kelasnya saja, kemudian jika sudah siap, user cukup memanggil kelas tersebut.

**GUIDED II**

1. Membuat kelas untuk menampilkan kalimat “Selamat Datang di Praktikum PBO” di dalam C++ dan Java.

#include <iostream>

#include <conio.h>

using namespace std;

class Welcome{

public:

void display(){

cout<<"Selamat Datang Di Praktikum PBO"<<endl;

}

};

int main(){

Welcome W;

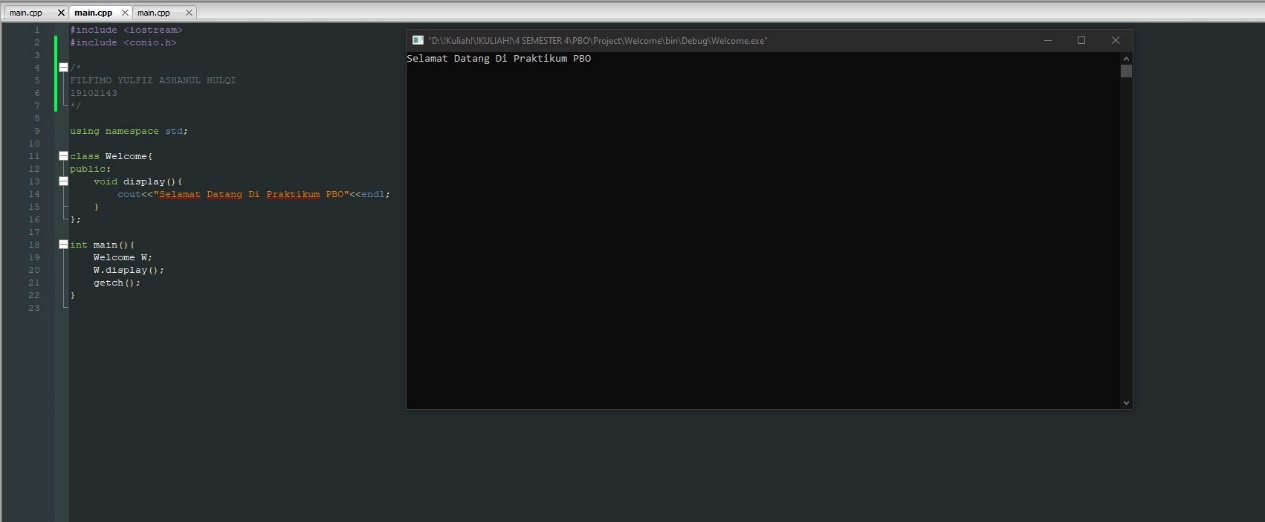
W.display();

getch();

}

*Source code* bentuk 1

Screenshot program bentuk 1



#include <iostream>

#include <conio.h>

using namespace std;

class Welcome{

public:

void display();

};

void Welcome::display(){

cout<<"Selamat Datang Di Praktikum PBO"<<endl;

}

int main(){

Welcome W;

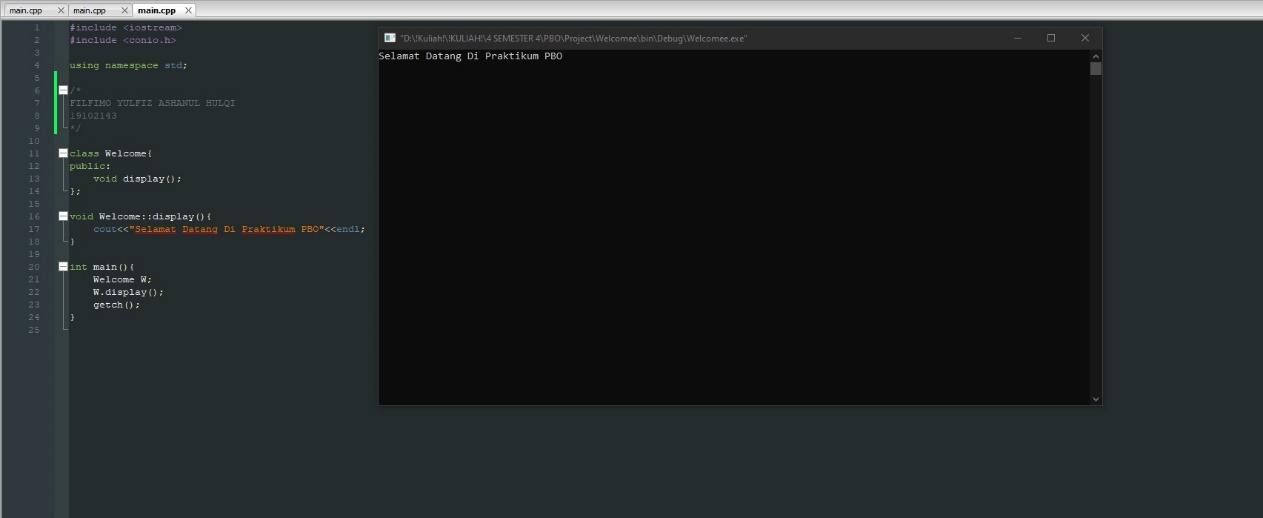
W.display();

getch();

}

*Source code* bentuk 2

Screenshot program bentuk 2



**MAIN**

package welcome;

public class MainWelcome {

public static void main(String[] args) {

// TODO code application logic here

Welcome W = new Welcome();

W.display();

}

}

**SUB**

package welcome;

public class Welcome {

public void display()

{

System.out.println("Selamat Datang di Praktikum PBO");

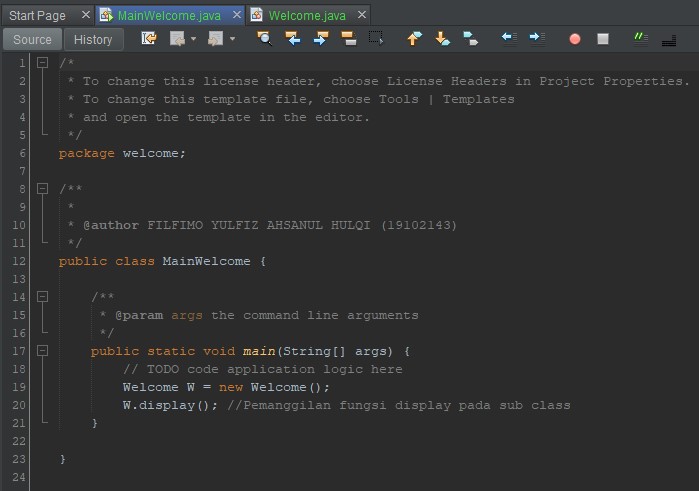
}

}

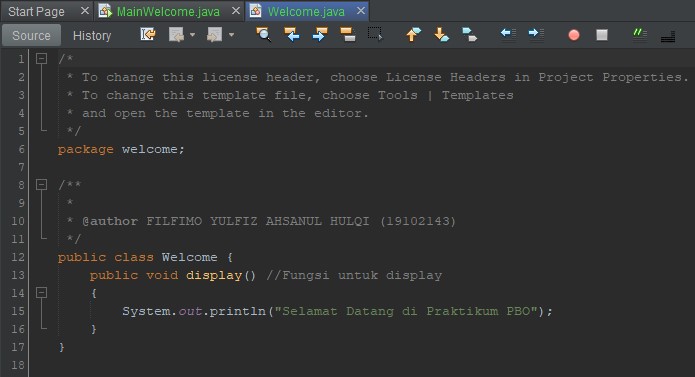
*Source code* program Java

Screenshot program Java :

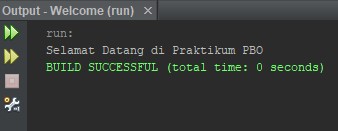
* *Main Class*



* *Sub Class*



* *Ouput* program



Penjelasan Program :

Pada Program bentuk 1 dan bentuk 2 jika langsung dijalankan sesuai dengan modul maka program tersebut tidak akan dapat *running*, karena adanya error, juga kurangnya fungsi “*using namespace std;*”. Dan fungsi “*void main*()” tidak dapat mengeluarkan ouput dan harus diganti menjadi “*int main()*”, agar program tersebut dapa *running*.

Sementara itu dalam program bentuk Java, *user* harus membuat dua buah file terlebih dahulu yaitu yang berperan sebagai *main class* (induk) dan *sub class* (anak), seperti yang sudah dijelaskan diatas, dalam *main class*, *user* hanya perlu memanggil fungsi yang telah ada / yang telah dibuat pada *sub* *class*.

Perbedaan Program :

Sebenarnya tidak ada perbedaan yang signifikan antara kedua program diatas, yang membedakan hanyalah jumlah fungsi antara program tersebut. Pada program bentuk 1 hanya menggunakan 1 fungsi *void*, sedangkan pada program bentuk 2 menggunakan 2 fungsi *void*.

Dalam program bentuk Java, *user* membuat dua buah file (*main class* dan *sub class*) dimana kedua file tersebut saling berhubungan satu sama lain.

1. Membuat fungsi *input* dan *output* dalam C++

#include <iostream>

#include <conio.h>

using namespace std;

class Mahasiswa

{

private:

char nama[20];

char nim[20];

public:

void inputData();

void display();

};

void Mahasiswa::inputData(){

cout<<"\n-Input Data Mahasiswa-"<<endl;

cout<<"Masukkan Nama Mahasiswa :";

cin>>nama;

cout<<"Masukkan NIM :";

cin>>nim;

cout<<endl;

};

void Mahasiswa::display(){

cout<<"\n-Display Data Mahasiswa-"<<endl;

cout<<"Nama : "<<nama<<endl;

cout<<"NIM : "<<nim<<endl;

};

int main(){

Mahasiswa mhs;

mhs.inputData();

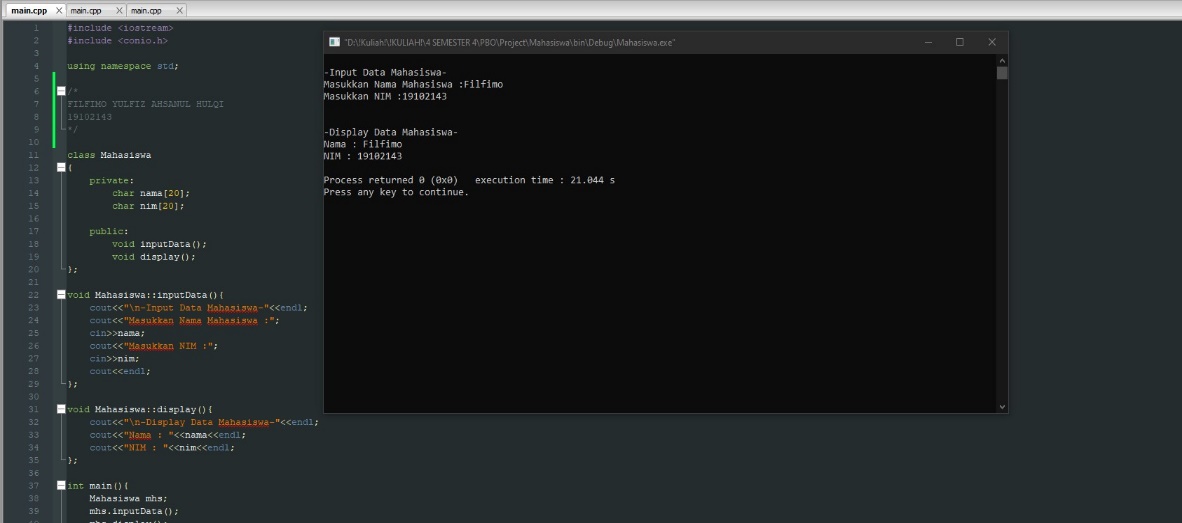
mhs.display();

getch();

}

*Source code* program i/o C++

Screenshot program



Penjelasan program :

Pada program diatas adalah OOP yang diterapkan pada C++, dapat dilihat sebelum fungsi *main*, diatasnya adalah fungsi *void* dan *class* mahasiswa, didalam *class* mahasiswa terdapat fungsi *private* dan *public*, *private* merupakan fungsi yang hanya dapat diakses secara internal oleh objek(*class* mahasiswa), sedangkan *public* adalah fungsi yang dapat diakses oleh siapa saja.

Selanjutnya ada *void* *input* data, fungsi tersebut nantinya akan menyimpan masukkan nama dan nim dari *user*. Kemudian ada *void* mahasiswa *display* fungsi ini akan menampilkan *output* data dari fungsi *input* data sebelumnya.

Dan terakhir adalah *int main*, disini *user* hanya perlu memanggil fungsi *void* yang telah dibuat diatas, yaitu fungsi *input* data, dan fungsi untuk *display*.

1. Membuat fungsi *input* dan *output* beserta *looping* (perulangan) dalam Java

**MAIN**

package mahasiswa;

public class MainMahasiswa {

public static void main(String[] args) {

// TODO code application logic here

Mahasiswa M=new Mahasiswa();

M.inputData();

M.display();

}

}

**SUB**

package mahasiswa;

import java.io.BufferedReader;

import java.io.InputStreamReader;

public class Mahasiswa {

private String Nama[] = new String[3];

private String NIM[] = new String[3];

private int i;

public void inputData()

{

BufferedReader b;

b = new BufferedReader(new InputStreamReader(System.in));

try

{

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

for (i=0;i<3;i++)

{

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

Nama[i]=b.readLine();

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

NIM[i]=b.readLine();

}

}

catch (Exception E){}

}

public void display()

{

System.out.println("");

System.out.println("-Display Data Mahasiswa-");

for(i=0;i<3;i++)

{

System.out.println("Nama : "+Nama[i]);

System.out.println("NIM : "+NIM[i]);

System.out.println("");

}

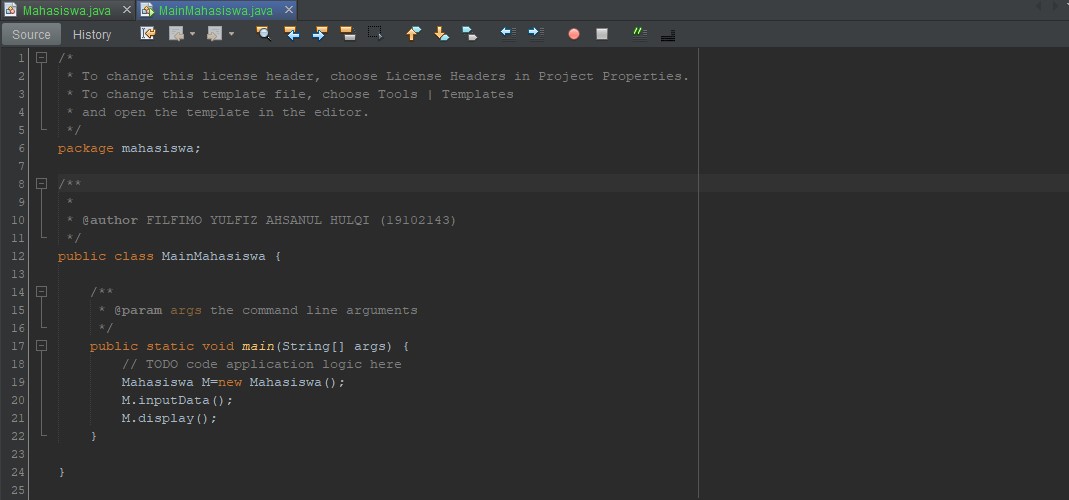
}

}

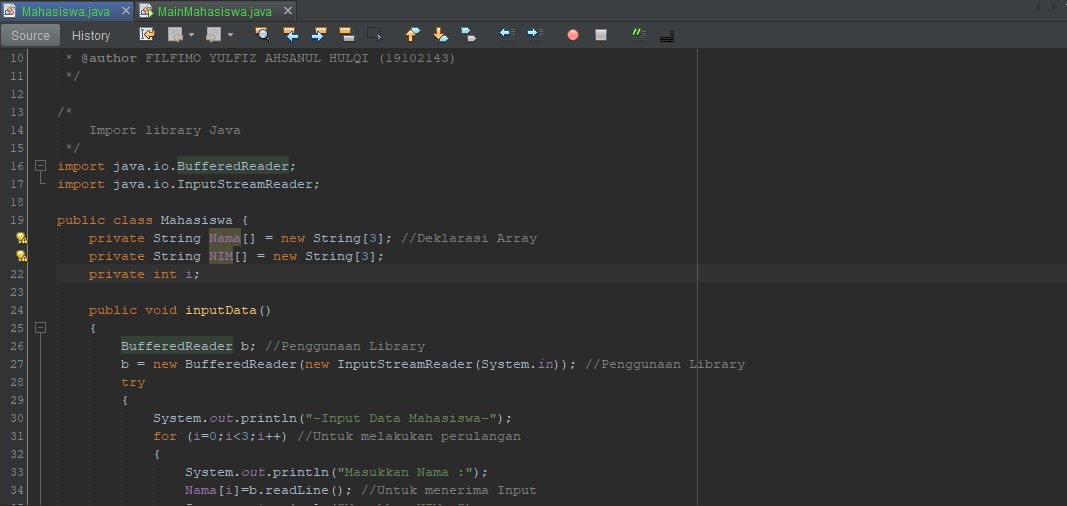
*Souce code i/o loop* Java

Screenshot Program

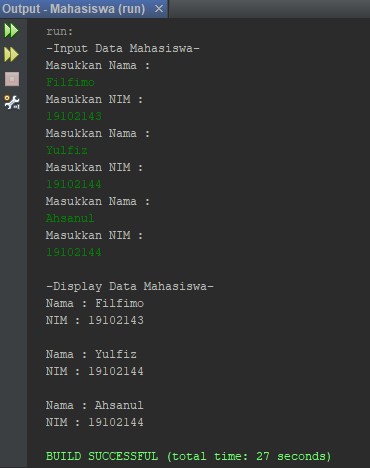
* *Main class*



* *Sub class*



* *Ouput* program



Penjelasan Program :

Pada program diatas, adalah OOP dalam Java, jika *user* menjalankan program secara langsung sesuai dengan modul, maka program tidak dapat dijalankan.

Program akan mengalami error pada *subclass* baris “*BufferedReader*” dan “*InputStreamReader”*, karena belum di import library dari “*BufferedReader*” dan “*InputStreamReader*”. Fungsi dari “*BufferedReader*” adalah membaca file input “*Stream*”.

Kemudian pada fungsi *void input* data bagian *looping* baris masukkan nama, ada kode yang kurang rapih yaitu *print* menjadi *println*, perlu diubah karena saat program *running* tampilan dari masukkan nama akan saling bertumpuk dengan tampilan *ouput* sebelumnya.

**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. Perbedaan antara pemrograman terstruktur dengan pemrograman berorientasi objek terletak dari *syntax*nya.
3. Untuk *syntax* penulisan bahasa antara C++ dan Java juga sangat berbeda, dari cara penulisan dan pembuatan pemrograman terstruktur maupun pemrograman berorientasi objek.
4. Pada Java untuk membuat OOP, diperlukan *main class* dan *sub* *class*, yang nanti akan saling berhubungan satu sama lain.