

**LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II**



NAMA : YUDHA ARTHA NUGRAHA
NIM : 193030503045
KELAS : A
**MODUL : I(DASAR PEMROGRAMAN
BERORIENTASI OBJEK)**

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020**

**LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II**



Nama : **YUDHA ARTHA NUGRAHA**
NIM : **193030503045**
Kelas : **A**
Modul : **DASAR PEMROGRAMAN BERORIENTASI**
OBJEK

Handwritten red text: "Type font" with a bracket pointing to the text above.

Komposisi	MAX	Nilai
BAB I Tujuan dan Landasan Teori	10	7
BAB II Pembahasan	60	50
BAB III Kesimpulan	20	15
Daftar Pustaka	5	3
Lampiran	5	3
Jumlah	100	78

Penilai
Asisten Praktikum


Dianh

BAB I

LANDASAN TEORI

1.1. TUJUAN

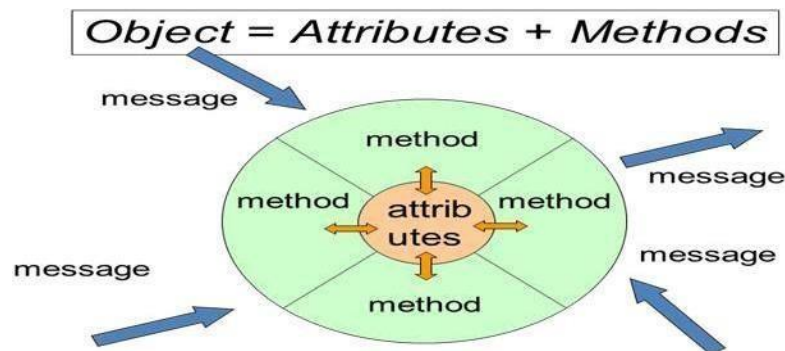
Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu:

1. memahami dasar-dasar pemrograman berorientasi obyek
2. Memahami enkapsulasi
3. membuat kelas dan objek

1.2. LANDASAN TEORI

Perbedaan pemrograman tradisional dan berorientasi objek adalah pada cara menyelesaikan suatu permasalahan. Pada pemrograman tradisional dalam memecahkan suatu masalah, masalah akan dibagi menjadi fungsi-fungsi yang lebih kecil, sedangkan pada pemrograman berorientasi objek (PBO) setiap masalah diselesaikan dengan cara dibagi ke dalam objek-objek.

Pada PBO dilakukan pembungkusan data (attributes) dan fungsi (behavior) ke paket yang disebut kelas. *Attributes* merupakan data yang menggambarkan status internal sebuah objek dan biasanya merupakan “*member variables*” pada C++, tidak dapat diakses dari luar (enkapsulasi), dan juga sebagai “*state*”. *Methods* merupakan fungsi yang mengakses status internal sebuah objek dan biasanya merupakan “*member functions*” pada C++, dapat diakses dari luar, memanipulasi atribut, dan disebut juga “*behavior*”. Berikut ini merupakan gambaran mengenai objek.



Kelas (Class) terdiri dari model objek yang memiliki atribut (data members) dan *Behaviors* (*member functions*), dan *Member functions* yaitu *Methods* yang dipanggil sebagai response terhadap pesan. Kelas didefinisikan dengan keyword **class**.

Mode Akses akses yang ada pada kelas ada tiga yaitu **private** yang merupakan *default* mode akses dan dapat diakses oleh *member functions*, **public** yang dapat diakses oleh setiap Accessible fungsi dalam program, dan **protected** yang biasanya digunakan untuk pewarisan .

Fungsi *Constructor* merupakan *member function* khusus yang menginisialisasi data members dan memiliki nama yang sama dengan nama kelas. Fungsi *Constructor* dipanggil saat membuat objek dari kelas dan tidak memiliki tipe balikan.

Member functions yang didefinisikan di luar kelas dilakukan dengan menggunakan *binary scope resolution operator* (::) yang berfungsi untuk “mengikat” nama fungsi ke nama kelas dan mengidentifikasi fungsi dari suatu kelas tertentu.

Berikut ini merupakan format dari *member functions*.

```

NilaiBalikan NamaKelas::NamaFungsi( ){

    ...

}

```

Member functions yang didefinisikan di dalam kelas tidak membutuhkan scope resolution operator dan nama kelas

1.2.1. Penjelasan dan Contoh Encapsulation (Enkapsulasi)

Encapsulation adalah sebuah konsep *Object Oriented Programming* digunakan untuk membungkus data dan fungsi, untuk menjaga tetap terjaga agar tidak adanya penyalahgunaan. Konsep *encapsulation* menyebabkan sebuah konsep OOP yang bernama “*Abstraction*” atau “*Data Hiding*”.

Encapsulation adalah sebuah teknik untuk membuat antar muka dan menyembunyikan mekanisme atau isi secara menyeluruh terhadap pengguna, dengan hal itu pengguna tidak diperbolehkan untuk mengakses data yang disembunyikan secara langsung. tapi bisa menggunakan dan memahami dengan mudah berdasarkan antar muka yang telah disediakan.

Syarat *Encapsulation*

Setelah penulis contohkan di atas, untungnya mengimplementasikan sebuah konsep *encapsulation*, dibutuhkan syarat sebagai berikut:

- Data dan fungsi yang disembunyikan harus berlabel *private* atau *protected* (jika dibutuhkan untuk hubungan antar class), agar tidak bisa diakses secara sembarang dan disalahgunakan.
- Data dan fungsi yang digunakan untuk antarmuka harus berlabel *public*.

1.2.2. Penjelasan dan Cara Pembuatan Class dan Object

Salah satu tujuan diciptakan Bahasa Pemrograman C++ adalah untuk menambahkan fitur baru, salah satunya adalah OOP (*Object Oriented Programming*) pada Bahasa Pemrograman C.

Fitur *Class* `class` adalah fitur OOP pada C++ mirip seperti Fitur *Data Structures* `struct` pada C, keduanya dapat menampung *variabel* sebagai anggota. Tapi perbedaan dari *class* dan *struct* tersebut adalah pada *Class* memiliki kemampuan lebih, seperti dimungkinkan untuk memuat *method*, *inheritance* dan lain-lain.

1.2.2.1. Pengertian Class

Class adalah salah satu dari konsep OOP yang digunakan untuk membungkus data abstraksi *procedural* sebagai deskripsi tergeneralisir atau rancangan dari sebuah *object* untuk mendefinisikan atau menggambarkan isi dan tingkah laku sebagai entitas dari *object*.

1.2.2.2. Cara Mendirikan Class

Untuk mendirikan *Class* kita membutuhkan keyword `class` yang dilanjutkan dengan pemberian nama dari deklarasi *class* tersebut. lalu dilanjutkan dengan meletakkan tanda `{` dan `}` untuk mengapit definisi dari *class*.

Class termasuk sebuah pernyataan maka dari itu akhir dari deklarasi *class* diwajibkan untuk mengakhiri *class* menggunakan tanda titik-koma `;`

Class dan *Object* merupakan fitur yang sangat membantu untuk mendirikan sebuah program besar, menjadikan sebuah code program yang ditulis oleh programmer mudah untuk dimengerti, lebih terstruktur, dan juga mudah dalam pemeliharaan program.

Terdapat banyak fasilitas yang disediakan oleh *class*, yaitu dapat menampung *member variabel, function, constructor, destructor, overloading* dan lain-lain. Diluar definisi *class* kita juga dimungkinkan untuk membuat relalasi seperti *inheritance* dan *overriding*.

1.2.2.3. Cara mendirikan Object

Mendirikan *class* seperti halnya kita membuat sebuah kerangka atau *blueprint* untuk *object*. Terdapat dua cara untuk mendirikan *object* pada *class*, yaitu:

Pertama, Mendirikan *object* di dalam deklarasi *Class*: Seperti contoh sebelumnya, bahwa kita dapat membuat *object* setelah definisi dari *class* yaitu tepat di antara tanda `}` dan `;`. Memberi *object* setelah definisi *class* tidak diharuskan, tapi dapat dilakukan dan kita bisa membuat *object* lebih dari satu dengan memisahkan masing-masing *object* dengan tanda koma `,`.

Kedua, Mendirikan *object* di luar deklarasi *Class*: Mendirikan *object* sama seperti kita

mendirikan *variabel*, yang berbeda adalah nama dari *class* akan digunakan sebagai tipe data dari *object*.

Contoh penulisan

```
1.mahasiswa anton;  
2mahasiswa budi, tono;  
3//atau  
4class mahasiswa anton;
```

Dalam mendirikan *object class* kita bebas untuk menambahkan keyword `class` atau tidak menambahkannya sebelum nama dari *class* tersebut, karena semua bentuk penulisan memiliki arti yang sama.

BAB II

PEMBAHASAN

2.1. Buatlah program berorientasi objek tentang hewan

```
#include <iostream>
using namespace std;
class kucing{
    public:
        kucing(int);
        int jkaki();
        int jekor();
        int jtelinga();
    private:
        int kaki;
        int ekor;
        int telinga;
};
kucing::kucing(int y){
    kaki=y;
    ekor=y;
    telinga=y;
}
int kucing::jkaki(){
    return kaki;
}
int kucing::jekor(){
    return ekor;
}
int kucing::jtelinga(){
    return telinga;
}
```



```

int main(){
    kucing a(4);
    kucing b(1);
    kucing c(2);
    cout<<"===== HEWAN
KUCING ====="<<endl;
    cout<<"jumlah kaki kucing adalah  :
"<<a.jkaki()<<endl;
    cout<<"jumlah ekor kucing adalah  :
"<<b.jekor()<<endl;
    cout<<"jumlah telinga kucing adalah :
"<<c.jtelinga()<<endl;
    return 0;
}

```

Pembahasan :

115 {

```

#include <iostream>
using namespace std;

```

iostream.h : Merupakan singkatan dari input output stream header yang digunakan sebagai standar input output operasi yang digunakan oleh bahasa C++.

Using Namespace Std adalah sebuah intruksi terhadap compiler untuk menggunakan semua fungsi yang terkait untuk kerangka yang sama, bisa berupa berkas, class, dan sejenisnya yang berkaitan dengan std. sehingga fungsinya ialah berguna untuk meminimalisir kesalahan akibat tidak dikenalnya fungsi cout dan cin apabila dalam penulisan secara langsung.

Pada umumnya, penulisan perintah cout dan cin sebenarnya adalah std::cout atau std::cin namun dengan adanya penggunaan Using

Namespace Std sehingga kamu tidak perlu menuliskan std:: lagi, contoh tersebut sangat singkat dan mungkin anda berfikir tidak terlalu repot.

```
class kucing{  
    public:  
        kucing(int);  
        int jkaki();  
        int jekor();  
        int jtelinga();  
    private:  
        int kaki;  
        int ekor;  
        int telinga;  
};
```

peprogram di atas merupakan program class yaitu program yang dapat menampung *member variabel, function, constructor, destructo.r*

program dalam text box diatas merupakan program class kucing yang menampung public : variaebel jkaki,jekor dan jteling yang merupakan integer dan private : kaki,ekor dan teling yang merupakan integer

```
kucing::kucing(int y){  
    kaki=y;  
    ekor=y;  
    telinga=y;  
}  
int kucing::jkaki(){  
    return kaki;  
}
```

Pada program diatas digunakan untuk mengidentifikasi kaki,ekor,telinga sebagai variable y

```
int kucing::jkaki(){  
    return kaki;  
}
```

Dimaksudkan untuk jkaki pada class kucing sebagai kaki

```
int kucing::jekor(){  
    return ekor;  
}  
int kucing::jtelinga(){  
    return telinga;  
}
```

Bertujuan untuk class kucing pada public jekor sebagai ekor

Bertujuan untuk class kucing pada public jtelinga sebagai telinga

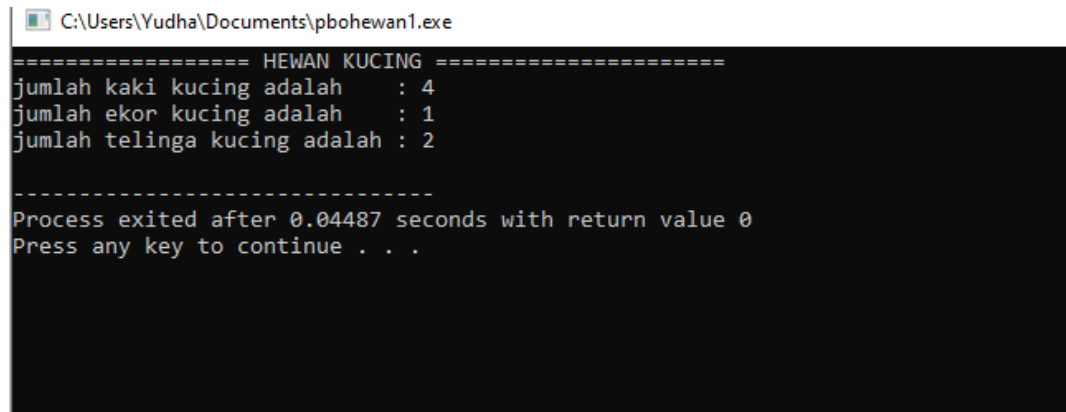
```
int main(){  
    kucing a(4);  
    kucing b(1);  
    kucing c(2);  
    cout<<"===== HEWAN  
KUCING ====="<<endl;  
    cout<<"jumlah kaki kucing adalah :  
<<a.jkaki()<<endl;  
    cout<<"jumlah ekor kucing adalah :  
<<b.jekor()<<endl;  
    cout<<"jumlah telinga kucing adalah :  
<<c.jtelinga()<<endl;  
    return 0;  
}
```

Di bagian terakhir program atau text box di atas digunakan untuk memunculkan output dari seluruh program yang dibuat pada program telah di tambah kan kucong a(4), kucing b(1) dan kucing c(2) ini dimaksudkan agar nilai dari variable telah ditetapkan

Pada program ini kita memunculkan output yaitu :

Jumlah kaki kucing	:4
Jumlah ekor kucing	:1
Jumlah telinga kucing	:2

Contoh output :



```
C:\Users\Yudha\Documents\pbohewan1.exe
==== HEWAN KUCING =====
jumlah kaki kucing adalah : 4
jumlah ekor kucing adalah : 1
jumlah telinga kucing adalah : 2
-----
Process exited after 0.04487 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1 output pbo kucing

2.2. Buatlah program berorientasi objek tentang hewan

```
#include <iostream>
#include <string>
using namespace std;

class komodo{
    public:
        komodo(int);
        int jkaki();
        int jekor();
        komodo(string);
        string jberbisa();
        string jpunah();

    private:
        int kaki;
        int ekor;
        string berbisa;
        string punah;
};

komodo::komodo(int y){
    kaki=y;
    ekor=y;
}

komodo::komodo(string y){
    berbisa=y;
    punah=y;
}

int komodo::jkaki(){
    return kaki;
}

int komodo::jekor(){
    return ekor;
}

string komodo::jberbisa(){
    return berbisa;
}
```

```

string komodo::jpunah(){
    return punah;
}

int main(){
    komodo a(4);
    komodo b(1);
    komodo c("berbisa");
    komodo d("hampir punah");
    cout<<"===== HEWAN Komodo
===== "<<endl;
    cout<<"jumlah kaki komodo adalah  :
"<<a.jkaki()<<endl;
    cout<<"jumlah ekor komodo adalah  :
"<<b.jekor()<<endl;
    cout<<"apakah hewan berbisa?      :
"<<c.jberbisa()<<endl;
    cout<<"status                      : "<<d.jpunah()<<endl;
    return 0;
}

```

Pembahasan:

```

#include <iostream>

#include <string>

using namespace std;

```

iostream : Merupakan singkatan dari input output stream header yang digunakan sebagai standar input output operasi yang digunakan oleh bahasa C++.

String : Merupakan file header yang berfungsi untuk melakukan manipulasi string. Fungsi-fungsi yang ada di string.h

Using Namespace Std adalah sebuah intruksi terhadap compiler untuk menggunakan semua fungsi yang terkait untuk kerangka yang sama, bisa berupa berkas, class, dan sejenisnya yang berkaitan dengan std. sehingga fungsinya ialah berguna untuk meminimalisir kesalahan akibat tidak dikenalnya fungsi cout dan cin apabila dalam penulisan secara langsung. Pada umumnya, penulisan perintah cout dan cin sebenarnya adalah std::cout atau std::cin namun dengan adanya penggunaan

Using Namespace Std sehingga kamu tidak perlu menuliskan std:: lagi, contoh tersebut sangat singkat dan mungkin anda berfikir tidak terlalu repot.

```
class komodo{
    public:
        komodo(int);
        int jkaki();
        int jekor();
        komodo(string);
        string jberbisa();
        string jpunah();

    private:
        int kaki;
        int ekor;
        string berbisa;
        string punah;
};
```

peprogram di atas merupakan program class yaitu program yang dapat menampung *member variabel, function, constructor, desctructo.r*

program class komodo di dalam program public memiliki konstruktor int dan string. Yang mana jkaki dan jekor sebagai integer dan jberbisa dan jpunah sebagai string

di bagian private ada 2 string dan dua integer yaitu kaki dan ekor sebagai integer dan berbisa dan punah sebagai string

```

komodo::komodo(int y){
    kaki=y;
    ekor=y;
}

komodo::komodo(string y){
    berbisa=y;
    punah=y;
}

int komodo::jkaki(){
    return kaki;
}

int komodo::jekor(){
    return ekor;
}

string komodo::jberbisa(){
    return berbisa;
}

```

Di sana terdapat 2 program yang menyatakan berbisa dan punah sebagai y demikian pula kaki dan ekor sebagai y

```

int komodo::jkaki(){
    return kaki;
}

```

Untuk mengidentifikasi jkaki sebagai kaki dalam bentuk integer begitu pula yang lain

Untuk mengidentifikasi jekor sebagai ekor dalam bentuk integer

Untuk mengidentifikasi jberbisa sebagai kaki dalam bentuk string


```

string komodo::jpunah(){
    return punah;
}

int main(){
    komodo a(4);
    komodo b(1);
    komodo c("berbisa");
    komodo d("hampir punah");
    cout<<"===== HEWAN Komodo
===== "<<endl;
    cout<<"jumlah kaki komodo adalah  :
"<<a.jkaki()<<endl;
    cout<<"jumlah ekor komodo adalah  :
"<<b.jekor()<<endl;
    cout<<"apakah hewan berbisa?      :
"<<c.jberbisa()<<endl;
    cout<<"status                      : "<<d.jpunah()<<endl;
    return 0;
}

```

Program di atas untuk menghasilkan output dari keseluruhan program yang dibuat

```

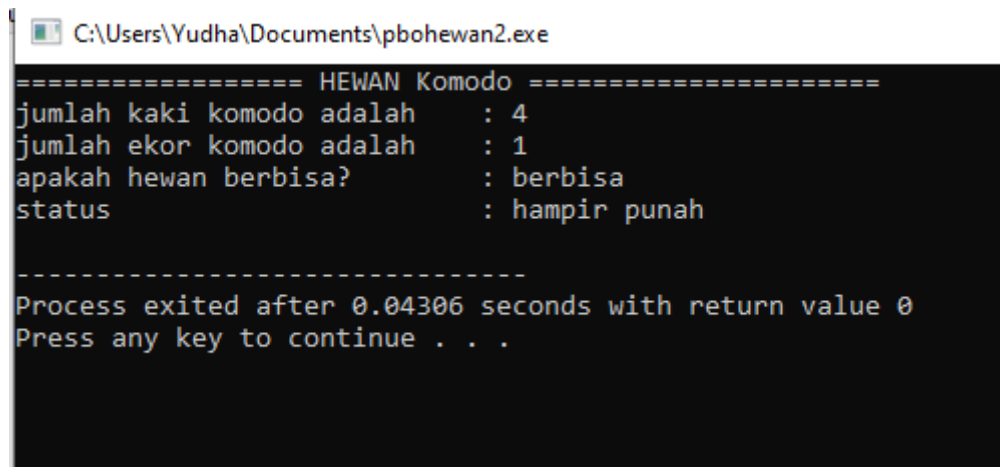
string komodo::jpunah(){
    return punah;
}

```

Untuk mengidentifikasi jpunah sebagai punah

Dalam int main terdapat komodo a(4), komodo b(1);, komodo c("berbisa"); komodo d("hampir punah"); sebagai variable yang telah diinputkan hal yang diinginkan setelah program di run akan menghasilkan output

Contoh output



```
C:\Users\Yudha\Documents\pbohewan2.exe
===== HEWAN Komodo =====
jumlah kaki komodo adalah      : 4
jumlah ekor komodo adalah      : 1
apakah hewan berbisa?         : berbisa
status                         : hampir punah

-----
Process exited after 0.04306 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.2 output pbo komodo

KESIMPULAN

Pemrograman Berorientasi Objek (PBO) atau dalam bahasa Inggris disebut Object-Oriented Programming adalah pemrograman yang berorientasikan kepada objek. PBO bukan merupakan salah satu jenis bahasa pemrograman, tetapi merupakan sebuah paradigma (cara berpikir) baru dalam pembuatan sebuah program. PBO menitikberatkan pada identifikasi objek-objek yang terlibat dalam pemrograman dan bagaimana objek-objek tersebut berinteraksi untuk menyelesaikan suatu tugas atau proses dari program tersebut.

Dari apa yang telah kita pelajari mengenai access specifier di artikel sebelumnya, encapsulation adalah dimana kemampuan kita untuk mengorganisasi kode kita dengan pemahaman menggunakan access specifier dimana teknik ini akan mengarahkan anda ke sebuah teknik Abstraction, dimana kita harus menyembunyikan data yang harus disembunyikan agar tidak disalahgunakan dan hanya membagikan sedikit data sebagai antar muka yang memang layak diberikan.

fitur Class adalah fitur OOP pada C++ mirip seperti Fitur Data Structures `struct` pada C, keduanya dapat menampung variabel sebagai anggota. Tapi perbedaan dari class dan struct tersebut adalah pada Class memiliki kemampuan lebih, seperti dimungkinkan untuk memuat method, inheritance dan lain-lain. Mendirikan class seperti halnya kita membuat sebuah kerangka atau blueprint untuk object. Setelah kita berhasil mendirikan object, object tersebut akan sepenuhnya memiliki bentuk berdasarkan rancangan pada class yang dipakai. Untuk mengakses Object dan member dari object tersebut kita membutuhkan "Member Access Operator" yang diletakkan di antara nama object dan nama member.

DAFTAR PUSTAKA

Unknow.2019. Fungsi Using Namespace Std di Dev C++ dan Contoh Tanpa Using Namespace Std dan Menggunakan Using Namespace Std.
<https://www.anakit.id/2019/09/fungsi-using-namespace-std-di-dev-c-dan.html> (diakses 9 april 2020 pukul 07.02)

Saputro,tedy tri.2018. Pemrograman Berorientasi Objek Dengan C++.
<https://embeddednesia.com/v1/pemrograman-berbasis-objek-dengan-c-bagian-1/> (diakses 9 april 2020 pukul 08.16)

Damas.2019. Contoh Program Sederhana C++ Menggunakan Class dan Object.
<https://kodedasar.com/class-cpp/> (diakses 9 april 2020 pukul 08.40)

Fajar.2019. Penjelasan dan Contoh Encapsulation (Enkapsulasi).
<https://www.belajarcpp.com/tutorial/cpp/encapsulation/> (diakses 9 april 2020 pukul 09.00)

Fajar.2018. Penjelasan dan Cara Pembuatan Class dan Object.
<https://www.belajarcpp.com/tutorial/cpp/class/> (diakses 9 april 2020 pukul 09.00)

diperhatikan lagi!!

LAMPIRAN

```
C:\Users\Yudha\Documents\pbohewan1.exe
===== HEWAN KUCING =====
jumlah kaki kucing adalah      : 4
jumlah ekor kucing adalah      : 1
jumlah telinga kucing adalah   : 2
-----
Process exited after 0.04487 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.1 output pbo kucing

31.5

```
C:\Users\Yudha\Documents\pbohewan2.exe
===== HEWAN Komodo =====
jumlah kaki komodo adalah      : 4
jumlah ekor komodo adalah      : 1
apakah hewan berbisa?          : berbisa
status                          : hampir punah
-----
Process exited after 0.04306 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.2 output pbo komodo

