

LAPORAN HASIL PRAKTIKUM ALGORITMA DAN PEMROGRAMAN II



NAMA : Ahmad Daffa Fahrezi
NIM : 193010503008
KELAS : A
MODUL : IV (RELASI KELAS)

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2020

**LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN II**



Nama : Ahmad Daffa Fahrezi
NIM : 193010503008
Kelas : A
Modul : IV (RELASI KELAS)

Komposisi	MAX	Nilai
BAB I Tujuan dan Landasan Teori	10	8
BAB II Pembahasan	60	40
BAB III Kesimpulan	20	8
Daftar Pustaka	5	5
Lampiran	5	5
Jumlah	100	

Penilai
Asisten Praktikum

Diana

BAB I

TUJUAN DAN LANDASAN TEORI

1.1. TUJUAN

1. Setelah menyelesaikan modul ini, mahasiswa diharapkan mampu membuat berbagai jenis relasi antara kelas.

1.2. LANDASAN TEORI

Jenis relasi antar kelas terdiri dari pewarisan, agregasi, asosiasi. Pewarisan merupakan hubungan antar satu kelas dengan kelas dengan kelas yang lain dalam suatu hirarki kelas induk dan kelas turunan. Superclass (“kelas dasar” atau “kelas induk”) merupakan kelas yang lebih general dalam relasi “is – a”. Subclass (“kelas turunan” atau “kelas anak”) merupakan kelas yang lebih spesifik dalam relasi “is – a”. objek yang dikelompokkan dalam sub kelas memiliki atribut dan perilaku kelas induk, dan juga atribut dan perilaku tambahan. Kita mengatakan subclass “mewarisi” suatu superclass (atau juga bisa dikatakan sebuah subclass “turunan dari” suatu superclass).

Agregasi merupakan hubungan antar kelas yang menyatukan suatu kelas merupakan **bagian** dari kelas yang lain atau hubungan antar kelas yang menyatakan suatu kelas **memiliki** kelas lain [sebagai atribut]. Agregasi merupakan relasi “has a”. Contoh :

- Mobil memiliki mesin.
- Fakultas terdiri atas jurusan.
- Rumah memiliki dapur.

Mobil, fakultas dan rumah dinyatakan sebagai kelas agregat. Mesin, jurusan dan dapur merupakan kelas penyusun.

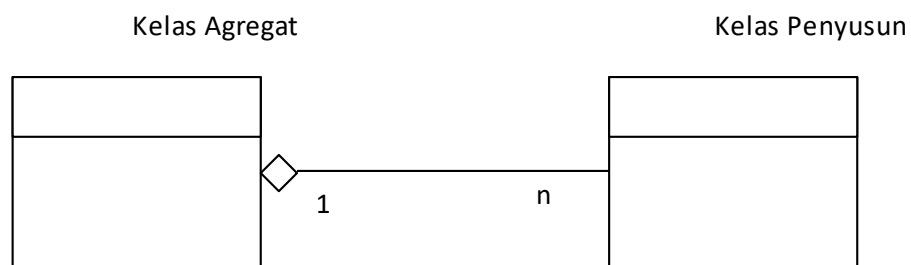
Kardinalitas menyatakan jumlah objek penyusunan yang terlibat dalam pembentukan relasi agregasi. Suatu kelas agregat dibentuk dari beberapa (disimbolkan dengan *) objek kelas penyusun.

Asosiasi menggambarkan hubungan struktural antar kelas. Setiap kelas memiliki kedudukan yang sama (tidak merupakan bagian dari kelas lain). Pada

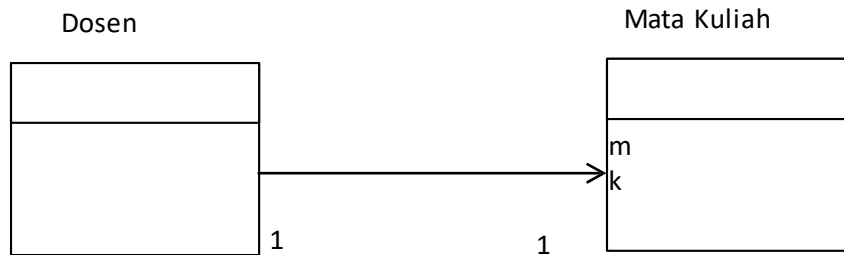
saat merancang relasi antar kelas, ada dua hal yang perlu diperhatikan yaitu berapa objek yang terlibat dan masing-masing kelas yang berelasi dan apakah relasi tersebut bersifat wajib (mandatory) atau opsional.

Agregasi adalah relasi antara dua buah objek dengan mengatakan bahwa satu objek memiliki atau mengandung atau berisi objek yang lain. Misalnya, sebuah mobil memiliki mesin, roda, body; sebuah rumah memiliki dapur, kamar mandi, kamar makan. Apabila suatu objek tertentu tersusun atas objek-objek lain, maka objek tersebut dikatakan sebagai objek agregat atau objek komposit (aggregate, composite object). Relasi ini sering disebut juga dengan relasi 'has-a' atau relasi 'whole-part'.

Dalam diagram UML, relasi agregat ini digambarkan dengan simbol diamond. Simbol ini menunjukkan adanya inklusi struktural sebuah objek terhadap objek yang lain. Hal ini diwujudkan dengan membuat kelas Agregat yang memiliki atribut yang bertipe kelas Penyusun.



Relasi asosiasi menyatakan suatu hubungan struktural antara beberapa objek yang menggambarkan objek dari suatu kelas dihubungkan dengan objek lain. Menunjukkan variabel dalam suatu kelas yang menyimpan rujukan bertipe kelas lain. Diagram berikut menunjukkan relasi asosiasi antara kelas Dosen dan kelas Matakuliah, dengan arah panah asosiasi dari kelas Dosen menuju kelas Matakuliah yang menunjukkan bahwa Dosen menyimpan rujukan ke Matakuliah.



Nama yang berada di anak panah (mk) merupakan role name yang kelak akan menjadi atribut dari kelas `Dosen`. Pada saat kita akan merancang relasi kelas, ada beberapa hal yang perlu diperhatikan, yaitu :

- Ada berapa objek dari masing-masing kelas yang terlibat dalam relasi.
- Apakah relasi tersebut bersifat wajib (mandatory) atau optional.

Dalam implementasi, asosiasi secara sintaks tidak memiliki perbedaan dengan implementasi agregasi, kecuali asosiasi bersifat dua arah.

BAB II

PEMBAHASAN

1. PROGRAM BUAH AGREGASI

```
//agregasi  
  
#include <iostream>  
  
#include <stdlib.h>  
  
using namespace std;
```

Program yang pertama ini adalah agregasi. Program ini merupakan program yang menghasilkan output berupa hasil buah yang terdapat pada satu pohon.

Pada program pertama disini menggunakan dua header yaitu `#include <iostream.h>` yang digunakan untuk mengaktifkan fungsi *cout* yang berguna untuk menampilkan output berupa teks yang diinput oleh *user* yang diapit dengan tanda `<<` “<< dan diakhiri dengan *endl* dan `#include <stdlib.h>` yang digunakan untuk menjabarkan beberapa fungsi umum dan macro termasuk manajemen memori dinamis, menjalin komunikasi dengan perangkat sekitar, membuat bilangan secara random, aritmatika bilangan integer, pencarian, pengurutan dan pengkonversian.

```
class BuahPohon  
{  
private:  
float Pohon1;  
public:  
{  
Pohon1 = 10;  
}
```

```
BuahPohon(float Buah1)
{
    Buah1 = Pohon1;
}

void JumlahPohon()
{
    cout<< "[" << Pohon1 << "]" << endl;
}

};
```

Preprocessor merupakan instruksi pada C++ yang digunakan untuk memberikan suatu perintah pada compiler.

Setelah memasukkan preprocessornya, kita membuat class yang berfungsi sebagai enkapsulasi yang menggunakan mode akses private dan public. Dalam C++, mode akses private ini sendiri merupakan mode akses yang dimana variabel dari class itu hanya bisa diakses oleh member functionnya saja. Sedangkan mode akses public merupakan mode akses yang dimana variabel dari class itu bisa diakses oleh class itu sendiri maupun oleh class-class turunannya, juga bisa langsung dieksekusi oleh program. Class di atas diberi judul class BuahPohon, dimana kelas inilah yang merupakan kelas penyusun dalam program.

```
class TotalPohon
{
private:
    float pohon;
    BuahPohon JumlahPohon;
public:
```

```

TotalPohon(float p)
{
    pohon = p;
}

TotalPohon(float pohon1, float p):JumlahPohon(pohon1)
{
    pohon = p;
}

void cetakBuah()
{
    cout<< "Total Pohon Mangga : " << pohon << endl;
    cout<< "Banyak Buah yang ada di pohon    : ";JumlahPohon.JumlahPohon();
}

};

```

Class yang selanjutnya diberi judul class TotalPohon. Class inilah yang merupakan kelas agregasi pada program ini. Pada mode akses privatenya, terdapat variabel pohon (tipe datanya float yang digunakan untuk memberi nilai bilangan decimal) dan variabel Buahpohon JumlahPohon.

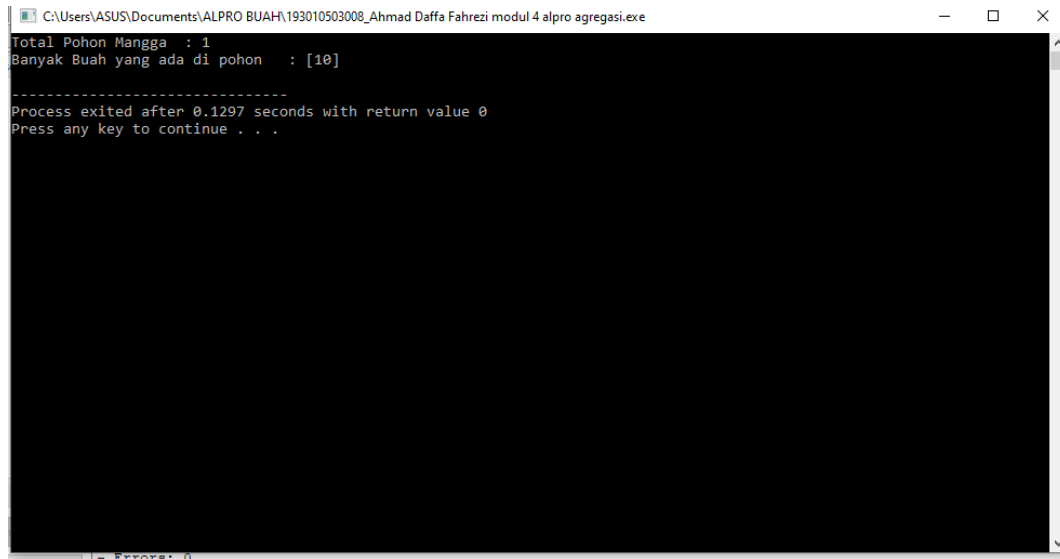
Pada program ini juga digunakan **cout** yang berfungsi untuk menampilkan data ke dalam output (cetak pada layar). Tanda '<<' merupakan operator yang fungsinya sebagai penghubung antara stream dengan kalimat inputan. **endl** merupakan manipulator yang digunakan untuk menyisipkan karakter mengatur pindah baris pada program. Tanda ';' artinya menyatakan bahwa akhir dari sebuah pernyataan. Tanda '{' dan '}' berfungsi untuk menandakan awal dan akhir sebuah definisi fungsi dan program.


```
int main()
{
    TotalPohon buah1(1);
    buah1.cetakBuah();
}
```

Bagian program di atas merupakan main program (bagian utama program). Hal itu ditandai dengan penggunaan `int main() {` yang menandakan awal dari program dan tanda `}` sebagai tanda berakhirnya program. *int main()* merupakan menandakan awalan dari bagian utama program dan menandakan berakhirnya program.

Pada bagian inilah nilai-nilai yang akan digunakan dalam proses perhitungan pohon dan buah diinputkan. Sehingga, saat program dijalankan hasil perhitungan akan langsung ditampilkan dan user tidak perlu lagi memasukkan nilai-nilai tersebut.

Hasil outputnya :



```
C:\Users\ASUS\Documents\ALPRO BUAH\193010503008_Ahmad Daffa Fahrezi modul 4 alpro agregasi.exe
Total Pohon Mangga : 1
Banyak Buah yang ada di pohon : [10]
-----
Process exited after 0.1297 seconds with return value 0
Press any key to continue . . .
Errors: 0
```

Gambar 2.1. output agregasi.

2. PROGRAM BUAH ASOSIASI.

```
// ASOSIASI

#include <iostream>

#include <stdlib.h>

using namespace std;
```

Program yang kedua ini adalah asosiasi. Program ini merupakan program yang menghasilkan output berupa harga buah perkilogram.

Pada program ini menggunakan dua header yaitu `#include <iostream>` yang digunakan untuk mengaktifkan fungsi *cout* yang berguna untuk menampilkan output berupa teks yang diinput oleh *user* yang diapit dengan tanda `<<` “<< dan diakhiri dengan *endl* dan `#include <stdlib.h>` yang digunakan untuk menjabarkan beberapa fungsi umum dan macro termasuk manajemen memori dinamis, menjalin komunikasi dengan perangkat sekitar, membuat bilangan secara random, aritmatika bilangan integer, pencarian, pengurutan dan pengkonversian.

```
class JenisBuah
{
    private:
        char* jeBu;
        char* namaBu;
        int HGBu;
    public:
        JenisBuah(char* JB, char* nm, int HG)
        {
            jeBu = JB; namaBu = nm; HGBu = HG;
```

```

    }

    void cetakJB()

    {

        cout << jeBu << " : " << namaBu << " : " << HGBu << endl;

    }

};

```

Bagian program di atas yaitu, sebuah class yang diberi nama class JenisBuah. Pada mode akses privatenya, terdapat variabel jeBu dan namaBu yang tipe datanya char, juga variabel HGBu dengan tipe data integer.

Pada mode akses publicnya, terdapat variabel jeBu = JB; namaBu = nm; HGBu = HG;. Variabel yang selanjutnya adalah cetakJB yang tipe datanya void. Fungsi ini digunakan untuk menampilkan jenis buah, nama buah dan harga buah.

Pada program ini juga digunakan **cout** yang berfungsi untuk menampilkan data ke dalam output (cetak pada layar). Tanda '<<' merupakan operator yang fungsinya sebagai penghubung antara stream dengan kalimat inputan. **endl** merupakan manipulator yang digunakan untuk menyisipkan karakter mengatur pindah baris pada program. Tanda ';' artinya menyatakan bahwa akhir dari sebuah pernyataan. Tanda '{' dan '}' berfungsi untuk menandakan awal dan akhir sebuah definisi fungsi dan program.

```

class Buah
{
    private:
    char* NPP;
    char* Nama;
    JenisBuah* Jb;
    int indeks;

    public:

```

```

Buah(char* noPeg, char* nm)
{
    NPP = noPeg; Nama = nm; indeks = 0;
}

void tambahJB(JenisBuah jenisbuah)
{
    Jb[indeks] = jenisbuah; indeks++;
}

void cetakInformasi()
{
    cout << Nama << " MENJUAL : \n" << endl;
    for (int i = 0; i < 2; i++)
        Jb[i].cetakJB();
}
};

```

Pada bagian program di atas, yaitu Class yang selanjutnya diberi nama class Buah. Pada mode akses privatenya, terdapat variabel NPP dan nama yang tipe datanya char. Pada mode akses publicnya, terdapat variabel :

- Buah, yang memiliki parameter noPeg dan nm yang tipe datanya char. Pada variabel ini juga dilakukan pendeklarasian terhadap NPP menjadi noPeg, Nama menjadi nm dan nilai indeks yang diisi dengan nilai 0.
- tambahJB, yang memiliki tipe data void. Variabel ini digunakan untuk memasukkan mata kuliah. Pada variabel ini digunakan fungsi increment pada nilai indeks, yang dilakukan untuk penambahan satu terhadap nilai suatu variabel.
- cetakInformasi yang memiliki tipe data void. Variabel ini digunakan untuk mencetak nama dosen dengan diikuti kata “Mengajar”. Untuk nilai i=0 dan i kurang dari 2 maka lakukan increment terhadap variabel i, lalu nilai i

dimasukkan ke dalam mk sebagai nilai indeks dan diikuti pemanggilan pada fungsi cetakJB.

Pada program ini juga digunakan **cout** yang berfungsi untuk menampilkan data ke dalam output (cetak pada layar). Tanda '<<' merupakan operator yang fungsinya sebagai penghubung antara stream dengan kalimat inputan. **endl** merupakan manipulator yang digunakan untuk menyisipkan karakter mengatur pindah baris pada program. Tanda ';' artinya menyatakan bahwa akhir dari sebuah pernyataan. Tanda '{' dan '}' berfungsi untuk menandakan awal dan akhir sebuah definisi fungsi dan program.

```
int main()
{
    Buah dd("", " TOKO PACARKU ");
    JenisBuah dd1(" LANGSAT\t ", "per Kg\t", 15000);
    JenisBuah dd2(" KELENGKENG\t ", "per Kg\t", 15000);
    dd.tambahJB(dd1); dd.tambahJB(dd2); dd.cetakInformasi();
}
```

Bagian program di atas merupakan main program (bagian utama program). Hal itu ditandai dengan penggunaan `int main() {` yang menandakan awal dari program dan tanda `}` sebagai tanda berakhirnya program. *int main()* merupakan menandakan awalan dari bagian utama program dan menandakan berakhirnya program. Pada bagian inilah nama toko, per kg, nama buah dan jumlah harga diinputkan. Pada coding program di atas ditambahkan pemanggilan terhadap fungsi :

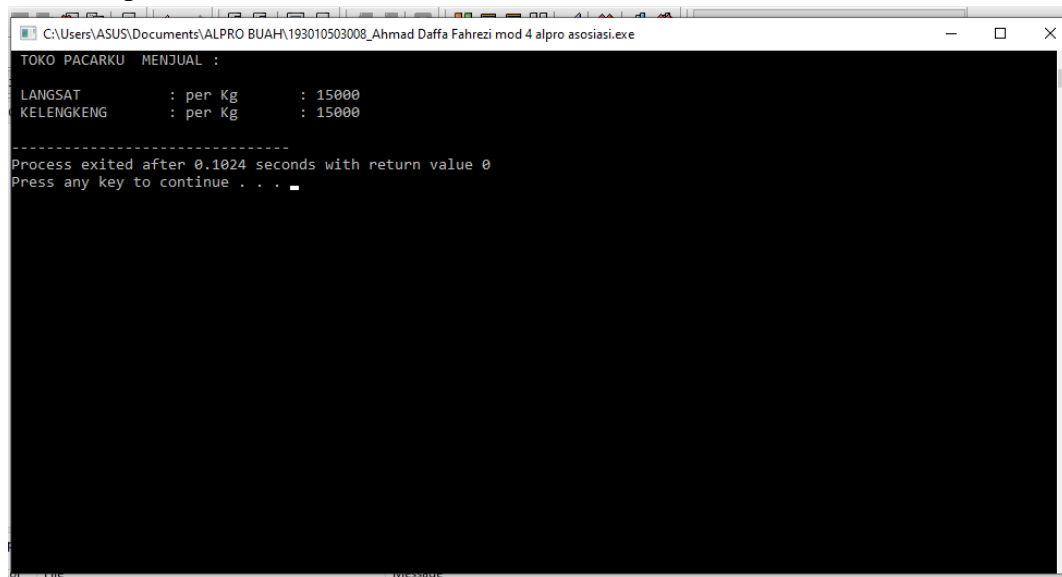
```
dd.cetakInformasi();
dd.tambahJB(dd1);
dd.tambahJB(dd2);
```

Hal ini dilakukan agar informasi tentang buah toko dan harga per kg dari program dijalankan. Program inilah yang disebut memiliki relasi kelas asosiasi.

Dimana setiap kelas pada program memiliki kedudukan yang sama (tidak merupakan bagian dari kelas yang lainnya).

Juga jangan lupa untuk menambahkan fungsi `getch` dan `return 0;` pada coding program, agar saat dibuild tidak terdapat *warning*. *Getch* digunakan untuk membaca karakter dengan sifat/jenis karakter yang dimasukkan tanpa perlu diakhiri dengan menekan tombol enter. Karakter akan langsung ditampilkan pada layar. *Return* digunakan karena pada main program menggunakan tipe data integer. Tanda ‘`}`’ menandakan berakhirnya program.

Hasil output :



```
C:\Users\ASUS\Documents\ALPRO BUAH\193010503008_Ahmad Daffa Fahrezi mod 4 alpro asosiasi.exe
TOKO PACARKU MENJUAL :
LANGSAT      : per Kg      : 15000
KELENGKENG   : per Kg      : 15000
-----
Process exited after 0.1024 seconds with return value 0
Press any key to continue . . .
```

Gambar 2.2. output asosiasi.

BAB III

KESIMPULAN

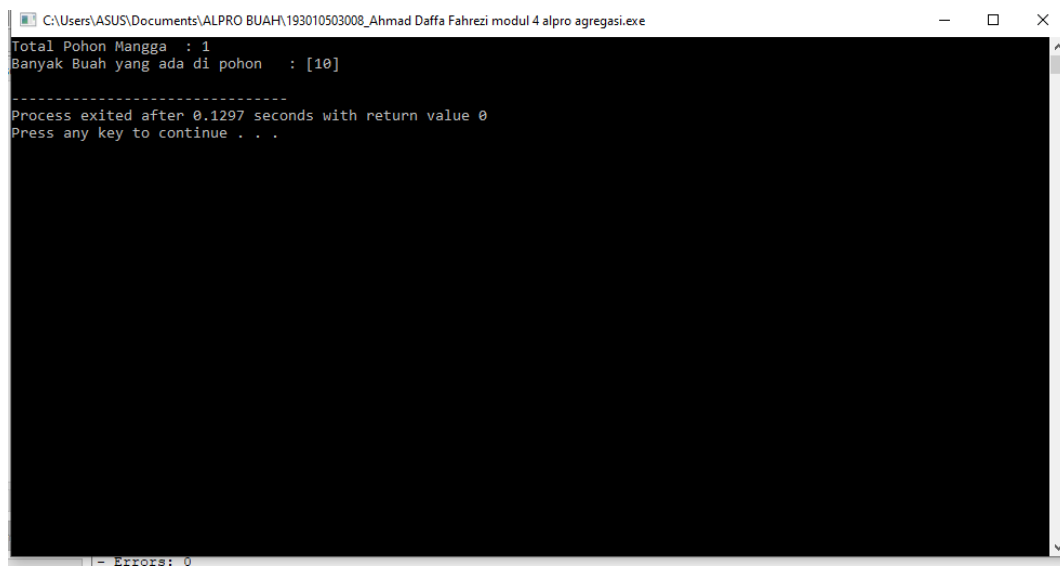
Dari modul ini dapat di ambil kesimpulan bahwa jenis relasi antar kelas terdiri dari pewarisan, agregasi, asosiasi adalah sebagai berikut:

- Pewarisan merupakan hubungan antar satu kelas dengan kelas lain dalam suatu hirarki kelas induk dan kelas turunan. Superclass (“kelas dasar” atau “kelas induk”) merupakan kelas yang lebih general dalam relasi “is-a”. subclass (“kelas turunan” atau “kelas anak”) merupakan kelas ang lebih sepesifik dalam relasi “is-a”. Objek yang dikelompokan dalam sub kelas memiliki atribut dan perilaku kelas induk, dan juga atribut dan perilaku tambahan. Kita mengatakan subcalass “mewarisi” suatu superclass (atau juga bisa dikatakan sebuah subclass “turunan dari” suatu superclass).
- Agregasi adalah hubungan antar kelas yang menyatakan bahwa suatu kelas merupakan bagian dari kelas yang lain. Di dalam konsep agregasi berlaku relasi “has a”. Contoh dari agregasi : pohon memiliki ranting. Pohon disebut sebagai kelas agregat, sedangkan ranting merupakan kelas penyusunnya. Kardinalitas menyatakan jumlah objek penyusun yang terlibat dalam pembentukan relasi agregasi. Suatu kelas agregat terdiri dari beberapa objek kelas penyusun da dilambangkan dengan tanda *.
- Asosiasi menggambarkan hubungan struktural antar kelas, dimana setiap kelas memiliki kedudukan yang sama (tidak merupakan bagian dari kelas yang lain). Contoh dari asosiasi : dosen dan mata kuliah. Dosen bukan merupakan turunan dari mata kuliah dan juga bukan merupakan bagian dari mata kuliah.
- Ada dua hal yang perlu kita perhatikan saat kita akan merancang relasi antar kelas, yaitu :
 1. Berapa jumlah objek yang terlibat dari masing-masing kelas yang berelasi.
 2. Apakah relasi bersifat wajib (mandatory) atau opsional.

DAFTAR PUSTAKA

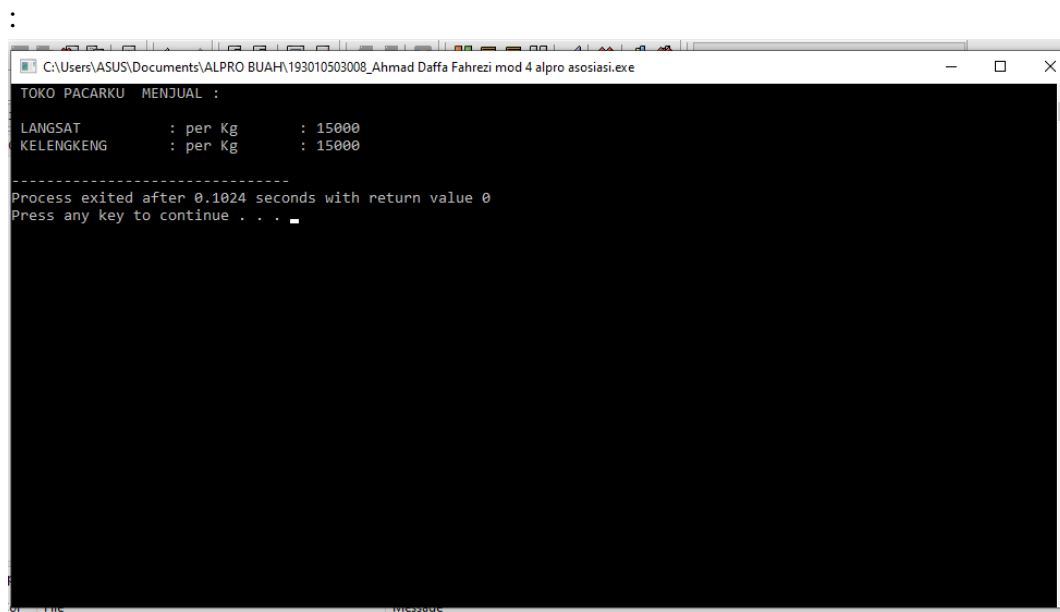
- Tim Dosen Algoritma Pemrograman II. 2020. *Modul Praktikum Algoritma Pemrograman II*. Universitas Palngka Raya : UPR. Fakultas Teknik.
- Turalakoujingkai Ciamis. 2011. *Relasi kelas pada c*.
<http://blogspot.com/2011/10/relasi-kelas -pada-c.html>. Diakses pada 28 April 2020 jam 11.00 WIB.

LAMPIRAN



```
C:\Users\ASUS\Documents\ALPRO BUAH\193010503008_Ahmad Daffa Fahrezi modul 4 alpro agregasi.exe
Total Pohon Mangga : 1
Banyak Buah yang ada di pohon : [10]
-----
Process exited after 0.1297 seconds with return value 0
Press any key to continue . . .
Errors: 0
```

Gambar 2.1. output agregasi.



```
C:\Users\ASUS\Documents\ALPRO BUAH\193010503008_Ahmad Daffa Fahrezi mod 4 alpro asosiasi.exe
TOKO PACARKU MENJUAL :
LANGSAT      : per Kg      : 15000
KELENGKENG   : per Kg      : 15000
-----
Process exited after 0.1024 seconds with return value 0
Press any key to continue . . .
Errors: 0
```

Gambar 2.2. output asosiasi.