STMIK ANTAR BANGSA

Pertemuan 12

OOP (Object Oriented Programming) Class C++

Lukman Nulhakim, M.Kom

STMIK ANTAR BANGSA

lukman@antarbangsa.ac.id

Pokok Bahasan

STMIK ANTAR BANGSA

1. Definisi OOP
2. Class
3. Program Class

Definis OOP

OOP adalah pemrograman yang menitikberatkan kepada objek-objek untuk menyelesaikan tugas atau proses dari program tersebut.

Sedangkan penitik beratkan ini dimaksudkan adanya interaksi pengiriman nilai, pesan atau pernyataan antar objek.

Kemudian objek yang merespon hasil dari interaksi tersebut akan membentuk suatu tindakan atau aksi

Jenis OOP

STMIK ANTAR BANGSA

Class

Calss merupakan gambaran atau abstraksi karakter dan sifat dari suatu objek. *Class* juga dapat mendefinisikan ciri dan perilaku objek tersebut.

Object

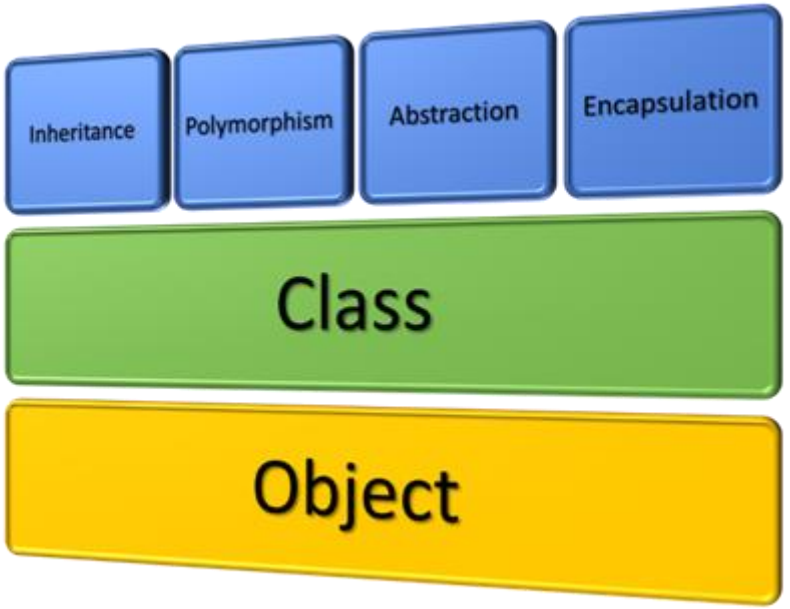
Object (objek) adalah suatu data atau entitas yang berwujud maupun tidak berwujud, memiliki sifat (karakteristik) tertentu sesuai dengan kondisi atau status dari penggunaannya. Data atau entitas di dalam pemrograman dapat disebut dengan blok fungsi.

Contoh pensil adalah suatu objek yang memiliki *attribute* (karakter) jenis, warna, panjang dan lain-lain.

Methode

Metoda merupakan tata cara objek tersebut diperlakukan, atau penggunaan atau manfaat dari objek tersebut.

Pensil juga memiliki *methode* (perilaku) seperti diruncingkan, digunakan dan lain-lain.



Format Class

STMIK ANTAR BANGSA

Bentuk umum dari kelas:

```
class class_ name
{
    private:
        data element_class;
        method;

    public:c
        data element_class;
        method;
        protype function;
};
Object Declaration;
```

Contoh deklarasi :

```
class motor
{
        char merk [50];
        float harga ;
        int stok ;
};
motor sport;
```

Nama class

Nama Anggota Data

Pendefinisian Objek

Pada sebuah kelas, item-item di dalamnya bisa bersifat private atau public. Secara default, semua item di dalam kelas bersifat **private.** Jadi tanpa menuliskan kata kunci **private**, semua item di dalam kelas sudah private.

A. Public pada kelas

Public (*public*) menyatakan bahwa deklarasi variabel atau item-item yang ada di dalam kelas dapat diakses dari luar kelas.

Contoh-1

```
//Penggunaan public pada class
#include <iostream.h>
#include <conio.h>

garis( )
{
 cout<<"================================\n";
}

class siswa
{
  public :
   char nis[9],nama[20];
   float nilai;
};
```

```
main( )
{
  clrscr( );
  siswa sekolah;
  garis( );cout<<endl;
  cout<<"\t Program Nilai Siswa"<<endl
     <<"\t ------------------"<<endl;
  cout<<" Input NIS        = ";cin>>sekolah.nis;
  cout<<" Input Nama Siswa  = ";cin>>sekolah.nama;
  cout<<" Input Nilai Akhir = ";cin>>sekolah.nilai;
  clrscr( );
  garis( );cout<<endl;
  cout<<"\t Nilai Siswa"<<endl
     <<"\t ------------"<<endl<<endl
     <<" NIS        = "<<sekolah.nis<<endl
     <<" Nama Siswa  = "<<sekolah.nama<<endl
     <<" Nilai Akhir = "<<sekolah.nilai<<endl;
  garis( );
  getch( );
}
```

B. Private pada Kelas

Private digunakan pada kelas untuk memproteksi anggota-anggota tertentunya agar tidak dapat diakses dari luar kelas secara langsung.

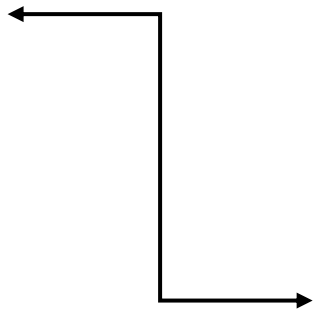
Contoh-2

```
//Penggunaan private pada class
#include <conio.h>
#include <iostream.h>
#define pi 3.14

class tabung
{
  private :
   int j,t;
   float v,k;
  public :
   tabung( );
   void keluaran( );
};
```

```
void main( )
{
 clrscr( );
  tabung s;

  s.keluaran( );
  getch( );
}
```



```
tabung :: tabung( )
{
 cout<<"\n Menghitung Tabung"<<endl
      <<" ----------------"<<endl<<endl;
  cout<<" Masukan Jari-jari = ";cin>>j;
  cout<<" Masukan Tinggi    = ";cin>>t;
  v=(pi*j*j)*t;
  k=(2*(pi*2*j))+t;
}
```

```
void tabung :: keluaran( )
{
   cout<<endl
       <<" Volume Tabung     = "<<v<<endl
       <<" Keliling Tabung   = "<<k<<endl;
}
```

Konstruktor

Konstruktor (*constructor*) merupakan suatu fungsi dari anggota suatu kelas yang memiliki nama yang sama dengan nama kelas fungsi itu berada. Konstruktor ini digunakan untuk mengalokasikan ruang untuk suatu objek dan untuk memberikan nilai awal.

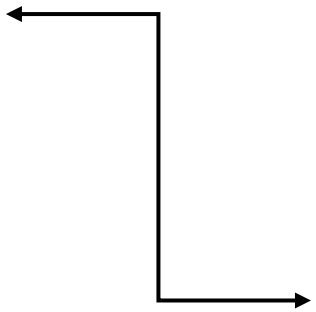
Contoh-3

```
//Konstruktor
#include <conio.h>
#include <iostream.h>

class bilangan
{
  private :

   int bulat;
   double nyata;
  public :
   bilangan( ); //konstruktor
     void info( );
};
```

```
void main( )
{
  clrscr( );
  bilangan a;
  a.info( );
  bilangan b;
  b.info( );
  getch( );
}
```



```
bilangan :: bilangan( )
{
  cout<<"\n Konstruktor dijalankan ... "<<endl;
  bulat = 5.2;
  nyata = 3.6;
}
void bilangan :: info( )
{
   cout<<"\n Jenis Bilangan: "<<endl
       <<" Bulat = "<<bulat<<endl
       <<" Nyata = "<<nyata<<endl;
}
```

```
void main( )
{
  clrscr( );
  motor sport("Honda CBR",500,30500000);
  motor matic("Honda Vario",125,14500000);
  sport.keterangan( );
  matic.keterangan( );
  getch( );
}
```

```
motor :: motor(char *nama, float cc, long hrg)
{
  merk = new char[25];//merubah data karakter menjadi string
  strcpy(merk,nama);
  cc_mesin=cc;
  harga=hrg;
}

motor :: ~motor( )
{
 delete [ ] merk;//menghapus memori karakter pd merk
}
void motor :: keterangan( )
{
 cout<<"\n Informasi Motor:"<<endl

    <<" Merk     = "<<merk<<endl
    <<" CC Mesin = "<<cc_mesin<<endl
    <<" Harga    = "<<harga<<endl<<endl;
}
```

Karakterisitk OOP

Di dalam penggunaan konsep pemrograman yang berbasis objek atau yang disebut Object Oriented Pemrograman (OOP), haruslah memiliki karakteirstik. Adapun karakteristik tersebut adalah memiliki sifat turunan atau pewarisan (*Inheritance*), satu nama memiliki banayak sifat atau perilaku (*Polymorphism*), pembungkusan sifat dari objek yang berbeda (*Encapsulation*). Berikut akan dijelaskan karakteristik OOP tersebut:

Pewarisan (Inheritance)

Pewarisan (Inheritance) : mendefinisikan suatu kelas dan kemudian menggunakannya untuk membangun hirarki kelas turunan, yang mana masing-masing turunan mewarisi semua akses kode maupun data kelas dasarnya.

- Konsep
  - ✓ Suatu kelas dapat diciptakan berdasarkan kelas lain.
  - ✓ Kelas baru ini mempunyai sifat-sifat yang sama dengan kelas pembentuknya, ditambah sifat-sifat khusus lainnya.
  - ✓ Dengan pewarisan kita dapat menciptakan kelas baru yang mempunyai sifat sama dengan kelas lain tanpa harus menulis ulang bagian-bagian yang sama.
  - ✓ Pewarisan merupakan unsur penting dalam pemrograman berorientasi objek dan merupakan blok bangunan dasar pertama penggunaan kode ulang (*code reuse*).

Pembatasan Inheritance

- ✓ Data dan fungsi yang dapat diwariskan hanya yang bersifat *public* dan *protected*.
- ✓ Untuk data dan fungsi private tetap tidak dapat diwariskan. Hal ini disebabkan sifat private yang hanya dapat diakses dari dalam kelas saja.

➢ Sifat Inheritance

- ✓ Sifat pewarisan ini menyebabkan kelas-kelas dalam pemrograman berorientasi objek membentuk hirarki kelas mulai dari kelas dasar, kelas turunan pertama, kelas turunan kedua dan seterusnya.

Tanpa inheritance, kelas merupakan sebuah unit yang berdiri sendiri. Inheritance akan membentuk suatu konsep dimana jika konsep yang diatas berubah maka perubahan akan juga berlaku pada yang ada dibawahnya. Inherate sangat mirip dengan hubungan orang tua dengan anak. Manakala suatu kelas menerima warisan, semua anggota data dan fungsi juga akan menerima warisan, walalupun tidak semuanya akan dapat di akses oleh anggota fungsi dari kelas.

Di dalam C++ penentuan akses pada inheritance ada tiga macam, yaitu :

1. Public
   Penentuan akses berbasis public menyebabkan anggota dari public dari sebuah kelas utama akan menjadi anggota public kelas turunan dan menyebabkan juga anggota protect kelas utama menjadi anggota protect kelas turunan, namun untuk anggota kelas private tetap pada private kelas utama.
2. Private
   Penentu akses berbasis private menyebabkan anggota dari anggota public dari kelas utama akan menjadi anggota protect kelas turunan, dan menyebabkan anggota dari kelas utama menjadi anggota protect kelas turunan, namun untuk anggota kelas private tetap pada private kelas utama.
3. Protected
   Penentu akses berbasis protect menyebabkan anggota dari anggota protect dan public dari kelas utama akan menjadi anggota private dari kelas turunan. Anggota private dari kelas utama selalu menjadi anggota private kelas utama.

Polimorphisme *(Polymorphism)*

Polimorphisme (Polymorphism) : memberikan satu aksi untuk satu nama yang dipakai bersama pada satu hirarki kelas, yang mana masing-masing kelas hirarki menerapkan cara yang sesuai dengan dirinya. Poly (Many) + morph (Shape) = POLYMORPHISM: banyak bentuk (*many shapes*).

- Polymorphism sangat berhubungan dg konsep Inheritance.
- Implementasi polymorphism dg 2 cara:
  - ✓ Overloading function: penggunaan kembali nama fungsi yang sama tapi dengan argumen yang berbeda.
  - ✓ Overriding function: sebuah fungsi dalam class turunan yang memiliki nama, return type dan argumen function yang sama dengan fungsi dalam class induk.

- Konsep
  - ✓ Polimorfisme merupakan fitur pemrograman berorientasi objek yang penting setelah pengkapsulan dan pewarisan. Polimorfisme berasal dari bahasa Yunani, *poly*(banyak) dan *morphos* (bentuk).
  - ✓ Polimorfisme menggambarkan kemampuan kode C++ berperilaku berbeda tergantung situasi pada waktu **run** (program berjalan).
  - ✓ Konstruksi ini memungkinkan untuk mengadakan ikatan dinamis (juga disebut ikatan tunda, atau ikatan akhir).
  - ✓ Kalau fungsi-fungsi dari suatu kelas dasar didefinisikan ulang atau ditindih pada kelas turunan, maka objek-objek yang dihasilkan hirarki kelas berupa objek polimorfik.
  - ✓ Polimorfik artinya mempunyai banyak bentuk atau punya kemampuan untuk mendefinisi banyak bentuk.

Encapsulation

Ciri penting lainnya dari OOP adalah *encapsulation*. Encapsulation adalah sebuah proses dimana tidak ada akses langsung ke data yang diberikan, bahkan *hidden*. Jika ingin mendapat data, maka harus berinteraksi dengan objek yang bertanggung jawab atas dara tersebut. Berikut cirri dari encapsulation:

1. Variabel dan method dalam suatu obyek dibungkus agar terlindungi
2. Untuk mengakses, variabel dan method yang sudah dibungkus tadi perlu interface
3. Setelah variabel dan method dibungkus, hak akses terhadapnya dapat ditentukan.
4. Konsep pembungkusan ini pada dasarnya merupakan perluasan dari tipe data struktur

Dua hal dalam enkapsulasi :

1. Information hiding
2. Menyediakan perantara (method) untuk mengakses data

Pada intinya, encapsulation adalah pemisahan antara bagian private dan public pada sebuah objek. Atau, dapat dipandang sebagai pemisahan antara interface (bagian private) dan implementation (bagian public).

Objek-objek lain yang hendak berinteraksi dengan objek ini akan mengirimkan sebuah pesan (message) dan objek ini akan mengerjakan sesuatu dan mengirimkan pesan balik sebagai jawaban jika diperlukan.

Keuntungan utama dari encapsulation tentu saja adalah penyembunyian implementasi (implementation hiding). Dengan implementation hiding, kita dapat memperbaiki bagaimana objek kita bekerja tanpa harus khawatir bagaimana menginformasikan perubahan tersebut ke objek-objek yang lain. Selama kita tidak merubah interface dari objek kita, objek-objek yang lain akan tetap dapat menggunkan objek kita.

STMIK ANTAR BANGSA

THANK FOR ATTENTION

Lukman Nulhakim, M.Kom

STMIK ANTAR BANGSA

lukman@antarbangsa.ac.id