PEMROGRAMAN LANJUT

04 | Encapsulation

Team Teaching Pemrograman Lanjut

Fakultas Ilmu Komputer UB

Enkapsulasi

- Pengkapsulan informasi
  - **Mengelompokkan** ke dalam 1 bundle
    - Membuat **class**
  - **Menyembunyikan** sebagian informasi
    - Membuat **private**

Keuntungan

- Mencegah pengaksesan bagian data field (variabel) dalam objek secara langsung
  - Yang berhak mengakses dan memanipulasi adalah method objek
- Mencegah kecelakaan
  - Karena adanya perlindungan (enkapsulasi) di sekitar data field maupun method dari suatu class, dan menyembunyikan detail implementasi dari objek

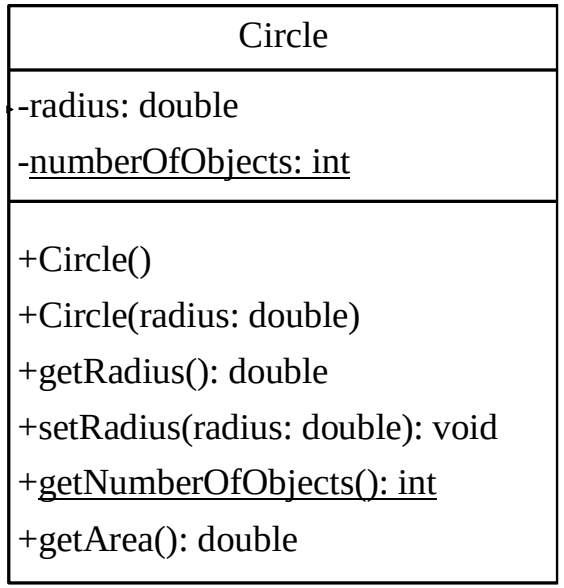
Modifier

- private
  - Data atau method hanya bisa diakses oleh class tersebut
  - Secara default anggota class (data atau method) memiliki modifier private
- public
  - Dapat diakses oleh method apapun dari class manapun
- protected
  - Dapat diakses oleh class tersebut dan turunannya (derived class)

Contoh Enkapsulasi Data Field

Tanda (-) menandakan modifier private

Garis bawah menandakan static



Radius lingkaran (default 1.0)

Jumlah objek lingkaran yang dibuat

Default constructor dari objek Circle

Constructor objek Circle dengan parameter radius

Mengembalikan nilai radius lingkaran

Mengatur nilai radius baru

Mengembalikan jumlah lingkaran yang dibuat

Mengembalikan luas lingkaran

Contoh Enkapsulasi (1)


```
class MyClass
{
        public: //access from anywhere
                int x;
        private:        //only access from within a class
                int y;
        protected:      //access from within a class ,or derived class
                int z;
};

void main()
{
        MyClass CopyClass;
        CopyClass.x = 1;        //OK, Public Access.
        CopyClass.y = 2;        //Error! Y isn't a member of MyClass
        CopyClass.z = 3;        //Error! Z isn't a member of MyClass
}
```

Contoh Enkapsulasi (2)

```
#include <iostream>
// Declaration of the Box class.

class Box
{
        private:
                int height, width, depth;     // private data members.
        public:
                Box(int, int, int);    // constructor function.
                ~Box();                        // destructor function.
                int volume();          // member function (compute volume).
};
```

Contoh Enkapsulasi (2) lanj.


```
// Definition of the Box class.
Box::Box( int ht, int wd, int dp )        // The constructor function.
{
    height = ht;
    width = wd;
    depth = dp;
}

Box::~Box()                               // The destructor function.   Use of scope
resolution '::'
{
    // does nothing
}

int Box::volume()                         // Member function to compute the Box's
volume.
{
    return  height * width * depth;
}
```

Contoh Enkapsulasi (2) lanj.


```
// The main() function.
int main()
{
    // Construct a Box object.
    Box thisbox(7, 8, 9);                    // actual values

    // Compute and display the object's volume.
    int volume = thisbox.volume();
    std::cout << "output volume is :  " << volume;

    return 0;
}
```

Mengapa Harus Private?

- Melindungi data
- Kemudahan perawatan
- Validasi