IPB University
Bogor Indonesia

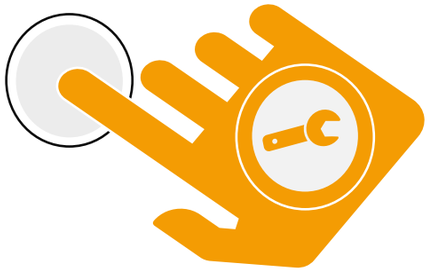
KOM120C -- BAHASA PEMROGRAMAN

Pemrograman Berorientasi Objek

- Prinsip Dasar OOP
- Object and Class
- Struct versus Class

Tim Pengajar Bahasa Pemrograman IPB University

PRINSIP DASAR OOP

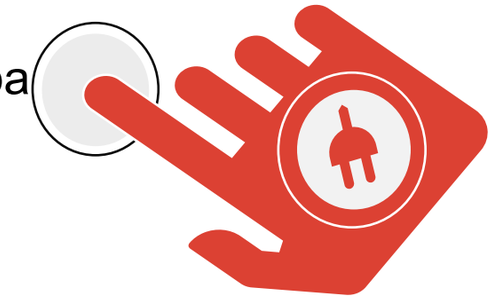


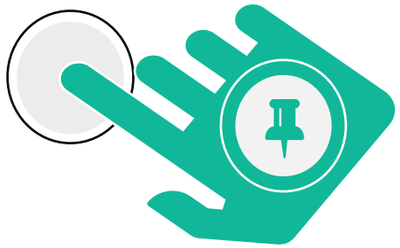
ENCAPSULATION

- Membungkus prosedur dan data dalam satu objek.
- OOP memodelkan objek yang ada di dunia nyata ke dalam software objek dalam pemrograman.
- Implementasi dalam bentuk **Class**.
- Berfungsi sebagai ADT (*Abstract Data Type*)

POLYMORPHISM

Tiap objek tahu siapa dirinya





INHERITANCE

- Pewarisan sifat objek
- Mengembangkan class baru dari class yang sudah ada.

Contoh Problem

Menjumlahkan dua bilangan

Paradigma Prosedural

Berfikir *algorithmic* → *imperative programming* (prosedural):

- Input. Apa inputnya? : dua bilangan bulat, mis. a dan b
- Output. Apa outputnya? : sebuah bilangan hasil, mis. sum
- Proses. Bagaimana langkah-langkah perhitungannya? : sum=a+b

Solusi → dalam bentuk algoritme:

```
read(a,b)
sum=a+b
write(sum)
```

Kode program:

```
#include <iostream>
int main() {
    int a,b;
    std::cin >> a >> b;
    int sum=a+b;
    std::cout << sum << '\n';
    return 0;
}
```

Contoh Problem

Menjumlahkan dua bilangan

Paradigma OOP

Saya harus membuat “benda” (objek) yang:

- memiliki dua variabel bilangan bulat

→ data / atribut

- dapat dimasukkan (menerima) dua nilai
- dapat dimasukkan (menerima) nilai yang pertama
- dapat dimasukkan (menerima) nilai yang kedua
- dapat memberikan hasil penjumlahan dua nilai

→ prosedur / behaviour / methods

Enkapsulasi adalah membungkus data/atribut dan prosedur ke dalam sebuah objek

Membuat Objek dalam Pemrograman

Objek diimplementasikan dalam bentuk class.

Struktur class dalam C++:

```
class <class-name> {
    // data/atribut
    ...
    ...
    // prosedur/fungsi
    ...
    ...
}
```

Setiap anggota atau elemen dari class jenis akses (*access modifier*), yang menunjukkan bisa tidaknya elemen tersebut diakses suatu class:

- **private**
  hanya dapat diakses di dalam class itu sendiri
- **protected**
  dapat diakses oleh class itu sendiri beserta turunannya
- **public**
  dapat diakses di luar class

By default, akses terhadap elemen class adalah private.

Class myClass

Contoh problem menjumlahkan dua bilangan

Saya harus membuat “benda” (objek) yang:

- memiliki dua variabel bilangan bulat

- dapat dimasukkan (menerima) dua nilai
- dapat dimasukkan (menerima) nilai yang pertama
- dapat dimasukkan (menerima) nilai yang kedua
- dapat memberikan hasil penjumlahan dua nilai

```
class myClass {
    private:
        int a,b;
    public:
        void set(int p1, int p2)
            { a=p1; b=p2; }
        void setA(int p) { a=p; }
        void setB(int p) { b=p; }
        int sum() { return a+b; }
};
```

Menguji Class

Contoh program menjumlahkan dua bilangan

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

class myClass
{
    private:
        int a,b;
    public:
        void set(int p1, int p2) { a=p1; b=p2; }
        void setA(int p) { a=p; }
        void setB(int p) { b=p; }
        int sum() { return a+b; }
};

// driver
int main()
{
    myClass myObj;   // myClass sebagai ADT
    myObj.set(5,10);
    cout << myObj.sum() << endl;
    myObj.setB(3);
    cout << myObj.sum() << endl;
    return 0;
}
```

Create object/instance → instansiasi

Memberikan nilai 5 dan 10 ke dalam objek

Mengganti nilai kedua dari objek myObj

Constructor

```
int main()
{
    myClass myObj;
    myObj.set(5,10);
    // ...
    return 0;
}
```

Pada saat dibuat objek myObj, berapa nilai data a dan b ?

Perlukah dibuat inisialisasi nilai a dan b pada saat dibuat objek myObj pertama kali? → Constructor

Apakah bisa dibuat objek myObj sekaligus memberi nilai atributnya ? → Constructor

Constructor adalah fungsi yang otomatis diakses pada saat instansiasi.
Nama fungsi constructor sama dengan nama class.
Sering menggunakan konsep *function overloading*.

Class myClass

Memiliki constructor

```
class myClass
{
    private:
        int a,b;
    public:
        myClass() { a=b=0; }  // default constructor
        myClass(int p1, int p2) { a=p1; b=p2; }
        void set(int p1, int p2) { a=p1; b=p2; }
        void setA(int p) { a=p; }
        void setB(int p) { b=p; }
        int sum() { return a+b; }
};

int main()
{
    myClass myObj1;     // membuat objek myObj
    myClass myObj2(5,10);
    cout << myObj1.sum() << endl;
    cout << myObj2.sum() << endl;
    return 0;
}
```

Membuat Objek

Automatic and Dynamic (Pointer)

```
// Create automatic storage object
myClass obj1;
obj1.set(5,10);

// Create dynamic storage object
myClass* obj2=new myClass();
obj2->set(5,10);

// Use constructor
myClass obj3(5,10);
myClass* obj4=new myClass(5,10);
```

STRUCT versus CLASS

Struct hanya berisi data/atribut

```
struct Bilangan
{
      int a;
      int b;
};
```

Class berisi data/atribut dan/atau prosedur

```
class Bilangan
{
      int a;
      int b;
      public:
          Bilangan() { a=b=0; }
          // dst.
};
```

Setiap elemen struct adalah public.

Setiap elemen class bisa private, protected, public.

Struct tidak dapat memiliki turunan (inheritance).

Class dapat memiliki turunan (inheritance)

Struct dan Class, keduanya berfungsi sebagai ADT (Abctract Data Type).

STRUCT versus CLASS

Struct dapat diimplementasikan menggunakan Class. Itu kenapa di Java tidak ada Struct.

```
struct Bilangan
{
    int a;
    int b;
};

int main()
{
    struct Bilangan obj;
    obj.a=5;
    obj.b=10;
    // ...
    // ...
    return 0;
}
```

```
class Bilangan
{
    int a;
    int b;
};

int main()
{
    Bilangan obj;
    obj.a=5;
    obj.b=10;
    // ...
    // ...
    return 0;
}
```

ERROR.
Mengapa?

STRUCT versus CLASS

Struct dapat diimplementasikan menggunakan Class. Itu kenapa di Java tidak ada Struct.

```
struct Bilangan
{
    int a;
    int b;
};

int main()
{
    struct Bilangan obj;
    obj.a=5;
    obj.b=10;
    // ...
    // ...
    return 0;
}
```

```
class Bilangan {
public:
    int a;
    int b;
};

int main()
{
    Bilangan obj;
    obj.a=5;
    obj.b=10;
    // ...
    // ...
    return 0;
}
```

Class sudah identic dengan Struct

BENAR

Tahapan OOP

Tahapan yang umum dilakukan untuk menyelesaikan persoalan menggunakan OOP

Object Design

Merancang objek (*attribute and behaviour*)
Menyusun hubungan antar objek (inheritance, polymorphism)

Create Class

Implementasi objek dalam pemrograman

Driver

Menyusun instruksi untuk memecahkan persoalan dengan menggunakan class yang sudah dibuat. Disini tempat untuk membuat instance (object).

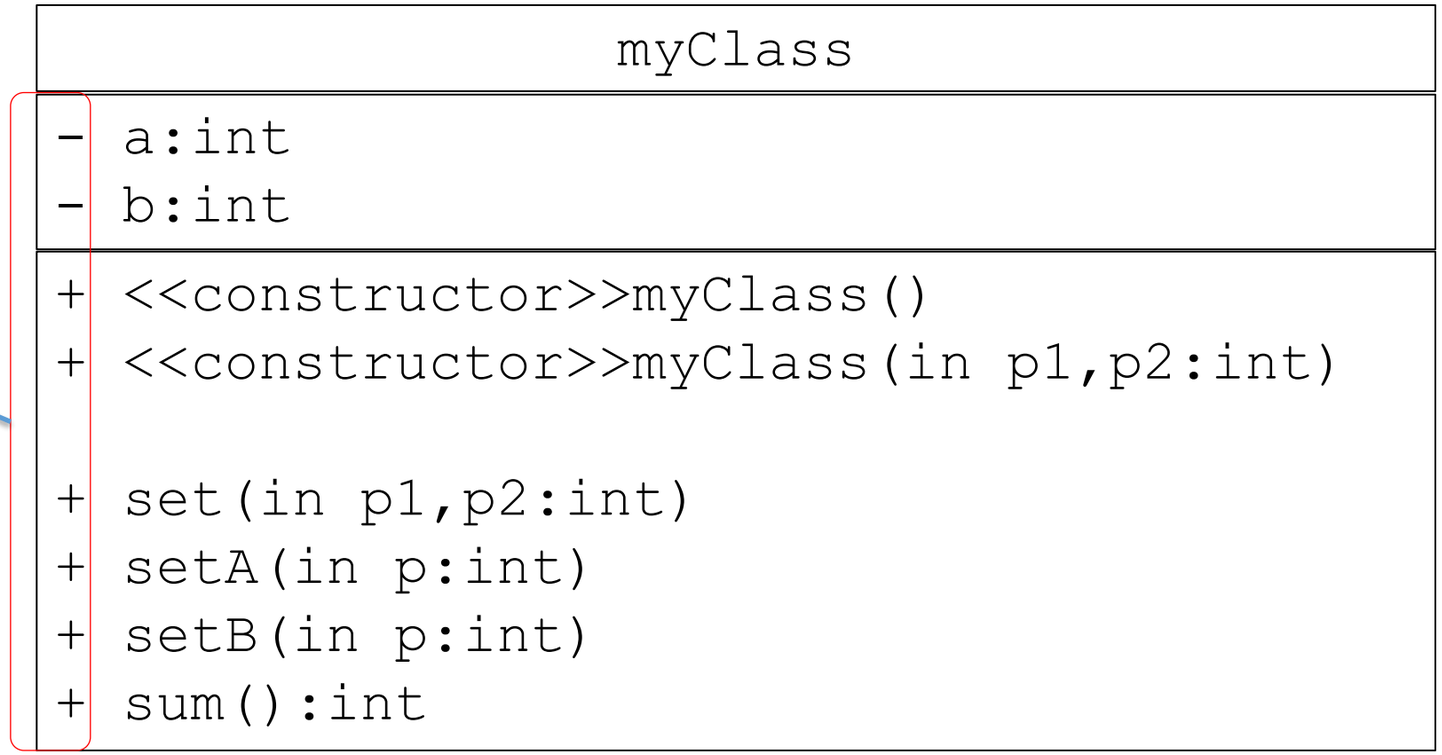
Testing

Menguji program dengan menggunakan kasus data.

Object Design

Merancang objek, salah satunya menggunakan presentasi UML (*Unified Modeling Language*)

Access modifier
+ : public
- : private
: protected



Latihan → Diskusi Kelas

Rancanglah suatu pendekatan OOP untuk menyelesaikan persoalan berikut:

Buat program untuk membuat dan mengolah nilai sebuah counter. Nilai counter dapat dinaikkan satu satuan, diturunkan satu satuan, dan ditampilkan nilainya. Nilai awal counter dapat dibuat dengan nilai tertentu. Pengolahan terhadap counter tersebut menggunakan kode operasi seperti berikut:

- 0 n membuat counter baru dengan nilai awal n
- 1 menaikkan counter satu satuan
- -1 menurunkan counter satu satuan
- 9 menampilkan nilai counter
- -9 akhir dari input data

Contoh input:

```
0  5
1
1
9
-1
9
0  0
-1
1
9
-9
```

Contoh output:

```
7
6
1
```