

Aplikasi Rancangan Orientasi Object

Tujuan :

- Menjelaskan aplikasi OOD dengan OOPL
- Menjelaskan evaluasi sintaks dan karakteristik bahasa pemrograman

Pemilihan Bahasa Pemrograman Berorientasi Object.

Kriteria evaluasi berdasarkan dukungan terhadap:

- Class, object
- Generalisation, Specialization: inheritance, dan pemecahan konflik pada nama
- Whole-part
- Attribute: dukungan terhadap Instance Connection (bagian dari object), visibility, dan kendala
- Fungsi: dukungan terhadap Message Connection (interaksi object), visibility, dan dynamic binding.

C++

Mendukung OOA dan OOD

- C++ Class : anggota Class terdiri dari private, protected, public, atau friends
- Variabel dan fungsi disebut Class member
- Class member khusus untuk menciptakan object (Constructor) dan menghapus object (Destructor)
- Deklarasi object : static atau dynamic

Pengaruh Bahasa Terhadap Pengembangan Orientasi Object

- Sintaks pada bahasa pemrograman object oriented harus dapat merepresentasikan semantik domain masalah. Hal tersebut dikarenakan:
 - Representasi uniform: pengembangan object-oriented harus mempunyai representasi uniform yang stabil dari waktu ke waktu. Hal ini berlangsung dari OOA ke OOD dan sampai ke OOP.
 - Reusability: Penggunaan ulang hasil OOA, yang kemudian juga diterapkan pada hasil OOD, dan OOP.
 - Maintainability: Pemilihan bahasa lebih didasarkan pada bahasa mana yang lebih baik dalam menangkap semantik domain masalah.

Contoh definisi Class dan penciptaan Object

```
typedef int      CompletionStatus;
typedef float    SensorReading;
typedef int      SensorAddress;

class Sensor
{
public:
    enum OperatingState (Off,StandBy,Monitor);
    //create a sensor
    Sensor();
    //copy creation
    Sensor (const Sensor & aSensor, int Address)
    //destructor
    ~Sensor();
    ....
    ....
protected:
    //Attributes
    char*      ModelNumber;
    char*      Manufacturer;
    int         InitSequence;
    UnitConversion Conversion;
    float       Interval;
    ...
}
```

```

...
private:
    //Class variable identifying Object type
    static char* theObjectType = "Sensor";

    SensorReading    Sample();
    CompletionStat us SetModelNumber;
    ....
}

//static declaration using 1st constructor
Sensor IntruderSensor;

//dynamic declaration using 2nd constructor
Sensor* DoorSensor = new Sensor();

```

Generalisasi-Spesialisasi

- mendukung inheritance tunggal maupun jamak
- derived class dapat mendecare inheritance menjadi public atau private
- class member dapat dideclare sebagai static (satu copy member statik), dan digunakan oleh seluruh object.

```

}
C++ Attribute, Fungsi
- Attribute merupakan variabel (member data)
- Visibility: private, protected, public, atau friend
- Class variabel dapat dideclare sebagai static
- Instance connection diimplementasikan dengan memakai pointer
- Fungsi merupakan member function
- Constructor dan Destructor harus mempunyai nama yang sama dengan Class
- Message connection merupakan function call
- Visibility : public, protected, dalam Class (private), friend.

```

Implementasi Fungsi MonitorForAlarmCondition.

```

Void Sensor::MonitorForAlarmCondition ()
{
    float          SensorReading = 0.0;
    AlarmDevice* WarningAlarmDevice = NULL;
    AlarmEvent*   WarningAlarmEvent = NULL;
    //precondition: State = StandBy
    if (StateOf () != StandBy)
        ReportError("Invalid State");
    else
    {
        .....
    }
}

```

Contoh mendefinisi Class spesialisasi:

```

Class CriticalSensor : public Sensor
{
    public:
        CriticalSensor();
        CriticalSensor (const CriticalSensor);
        ~CriticalSensor();

    protected:
        float Tolerance;

    private:
        CompletionStatus    SetTolerance ()
}

```

C++ Whole-Part

- merupakan nested object
- mendukung definisi Class dalam Class

```

class Sensor
{
    public:
        Building BuildingAttachedTo();
    protected:
        Building AssociateBuilding;
}

```

Object Pascal

Object Pascal mendukung OOA dan OOD.

Object Pascal – Class, Object

- Class, Object merupakan object type dan object
- Definisi Class mirip dengan definisi "record" d Pascal

Contoh definisi Class dan penciptaan Object:

```

Const
    Len255 = 255;
type
    OperatingState = (Off,StandBy,Monitor);
    Str255=packed array [1...Len255] of char;

    Sensor = object
        TheObjectType      :Str255;
        ModelNumber        :Str255;
        Manufacturer        :Str255;
        InitSequence        :Integer;
        .....

        constructor Create;
        destructor Destroy;

        procedure ConvertTo;

```

```

        procedure SetModelNumber;
    end;
var
    IntruderSensor :Sensor; {static allocation}
    WindowSensor :^Sensor;{dynamic allocation}
...
begin
    IntruderSensor.Create;
    New (WindowSensor.Create);
    .....
end.

```

Object Pascal – Generalisasi-Spesialisasi

- Generalisasi-spesialisasi merupakan parent-child
- Mendukung inheritance tunggal
- Mendukung encapsulation data dan proses dalam object

Contoh Class spesialisasi

```

type
    CriticalSensor(Sensor) = object
        Tolerance:Real;
        Constructor Create;
        Destructor Create;
        .....
    end;

```

Object Pascal – Whole-Part

- Whole-Part merupakan embedded pointer
- Struktur Whole-Part diimplementasikan dengan memasukkan variabel pointer object dalam object lain

Contoh implementasi struktur Whole-Part dengan menambah atribut dan fungsi ke setiap class.

```

type
    Sensor = object
        AssociateBuilding :Building;
        Function BuildingAttachedTo :Building;

    Building = object
        Set :BuildingSensor;
        .....
        function EmergencyNumberOf :Str255;
        function StreetAddressFor :Str255;
        .....
    end;

```

Object Pascal – Attribute, Fungsi

- Attribute merupakan variabel
- Instance Connection merupakan kumpulan pointer
- Fungsi merupakan prosedur atau fungsi

- Message Connection merupakan procedure call atau function call

Contoh:

```
Sensor.MonitorForAlarmCondition ();
```

```

var
    SensorReading :Real;
    WarningAlarmDevice :^AlarmDevice;
    WarningAlarmEvent :^AlarmEvent;

begin
    if (State<>StandBy) then
        ReportError ("InvalidState");
    else begin
        WaitForMonitor ();
        while (State = Monitor) do
            begin
                delay (Interval);
                SensorReading := Sample ();

                Value:=ConversionConvert(SensorReading);
                ....
                ....
                WarningAlarmDevice^.Destroy;
                WarningAlarmEvent^.Destroy;
            end;
end;

```