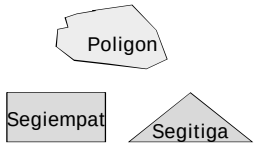
Bahasa Pemrograman

Kuliah 10 :: Implementasi Class

Julio Adisantoso

Konsep Pewarisan

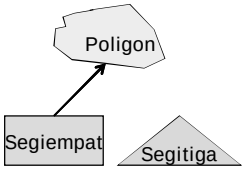



```
class Poligon {
   int n;
   vector <double> x, y;
   public:
      void set(...);
};
```

```
class Segiempat {
   int n;
   vector <double> x, y;
   public:
     void set(...);
     double luas();
};
```

```
class Segitiga {
   int n;
   vector <double> x, y;
   public:
     void set(...);
     double luas();
};
```

Konsep Pewarisan :: class Segiempat

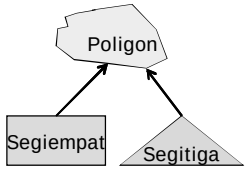



```
class Poligon {
   protected:
      int n;
      vector <double> x, y;
   public:
      void set(...);
};
```

```
class Segiempat:
public Poligon {
   public:
     double luas();
};
```

```
class Segiempat {
protected:
   int n;
   vector <double> x, y;
public:
   void set(...);
   double luas();
};
```

Konsep Pewarisan :: class Segitiga

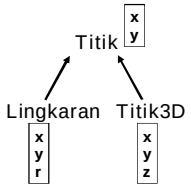



```
class Poligon {
   protected:
      int n;
      vector <double> x, y;
   public:
      void set(...);
};
```

```
class Segitiga:
public Poligon {
   public:
     double luas();
};
```

```
class Segitiga {
protected:
   int n;
   vector <double> x, y;
public:
   void set(...);
   double luas();
};
```

Konsep Pewarisan :: class Titik




```
class Titik {
protected:
  int x, y;
public:
  void set (int a, int b);
};
```

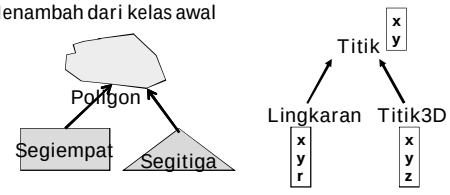
```
class Lingkaran :
public Titik {
   double r;
};
```

```
class Titik3D :
public Titik {
   int z;
};
```

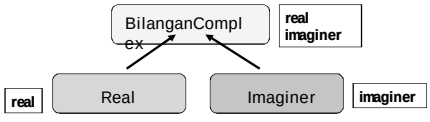
Slide 4

Konsep Pewarisan

- Menambah dari kelas awal



- Spesialisasi dari kelas awal



Slide 5

Mengapa Pewarisan ?

Pewarisan adalah mekanisme untuk

- Mengembangkan class baru dari class yang sudah ada
- Mendefinisikan class baru untuk menambah dan spesialisasi dari class yang sudah ada

Harus memahami istilah base class dan derived class !

Slide 6

Mendefinisikan pewarisan

- Sintaks:

```
class <DerivedClass> : <access-level> <BaseClass>
```

dimana <access-level> menunjukkan tipe pewarisan
  - private (by default)
  - public

- Setiap class dapat berfungsi sebagai Base Class, sehingga Derived Class dapat dibuat sebagai Base Class

Slide 7

Penurunan Class


```
class Titik {
  protected:
    int x, y;
  public:
    void set (int a, int b);
};
```

```
class Titik3D : public Titik {
   private:
      double z;
   ...
};
```

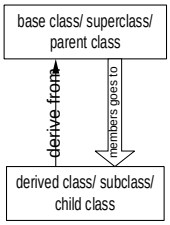
```
class Bola : public Titik3D {
  private:
     double r;
  ...
};
```

Titik adalah baseclass dari **Titik3D**, sedangkan **Titik3S** adalah baseclass dari **Bola**

Slide 8

Akses anggota class

Pada prinsipnya, setiap anggota base class diturunkan ke derived class. Hanya tingkatan aksesnya yang berbeda, tergantung jenis penurunannya. Perkecualian: contructor, destructor, fungsi anggota operator=(), dan friend.



Jenis Penurunan

Kontrol akses Untuk anggota

	private	protected	public
private	-	-	-
protected	private	protected	protected
public	private	protected	public

Slide 9

Aturan constructor untuk turunan

Default constructor dan destructor dari base class selalu dipanggil ketika suatu obyek baru dari class turunannya dibuat atau di-destroy.

```
class A {
  public:
  A() {
   cout<<"A:default\n"; }
  A(int a) {
   cout<<"A:parameter\n"; }
};
```

```
class B : public A {
  public:
  B(int a) {
    cout<<"B\n"; }
};
```

B x(1); →

output:

```
A:default
B
```

Slide 10

Aturan constructor untuk turunan

Kita dapat menentukan constructor dari base class yang digunakan untuk class turunannya.

```
class A {
  public:
  A() {
   cout<<"A:default\n"; }
  A(int a) {
   cout<<"A:parameter\n"; }
};
```

```
class C : public A {
  public:
  C (int a) : A(a) {
    cout<<"C"<<endl; }
};
```

C x(1); →

output:

```
A:paramet
er
C
```

Slide 11

Overriding

Class turunan dapat meng-override fungsi yang telah didefisinikan oleh base class. Dengan overriding,

- Fungsi atau method dalam class turunan memiliki signature yang identik dengan method pada base class.
- Derive class mengimplementasikan method versinya sendiri.

```
class A {
  protected:
  int x, y;
  public:
    void print() {
      cout<<"A"<<endl; }
};
```

```
class B : public A {
  public:
    void print() {
        cout<<"B"<<endl; }
};
```

Slide 12

Akses terhadap fungsi anggota

```
class Titik {
  protected:
    int x, y;
  public:
    void set(int a, int b)
      {x=a; y=b;}
    void foo ();
    void print();
};
```

```
class Lingkaran : public Titik {
  private:  double r;
  public:
  void set (int a, int b, double c)
    {
      Titik :: set(a, b);
      r = c; }
  void print();
};
```

```
Titik A;
A.set(30,50);  // base class
A.print(); // base class
```

```
Lingkaran C;
C.set(10,10,100);  // class Lingkaran
C.foo ();  // dari class Titik
C.print(); // dari class Lingkaran
```

Slide 13