PEMROGRAMAN BERORIENTASI OBJEK

Inheritance dan Polimorfisme

Telkom University

Budhi Irawan, S.Si, M.T

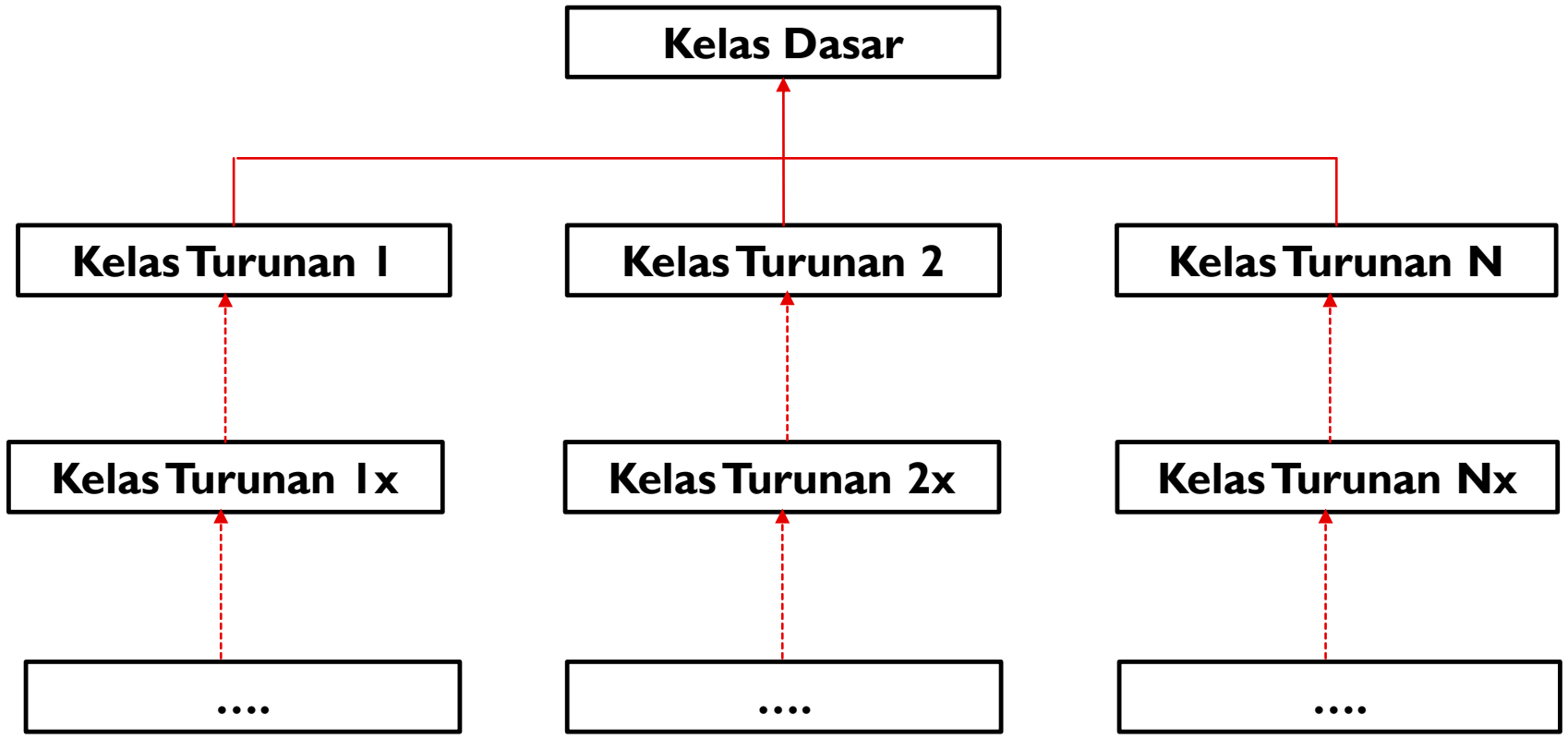
PENDAHULUAN

- Salah satu ciri dari PBO adalah kemampuan suatu objek atau kelas untuk ***mewariskan sifat-sifat*** yang terdapat didalamnya ke kelas turunannya dan Hail ini dinamakan dengan ***proses pewarisan (inheritance).***
- Konsep ini kelihatannya mudah, tapi sebenarnya banyak detail tersembunyi yang harus diperhatikan.
- Walaupun demikian hal yang paling utama ditekankan dalam proses ***penurunan*** sebuah ***kelas*** adalah pemberian ***hak akses*** terhadap ***kelas-kelas turunan***.

KELAS DASAR & KELAS TURUNAN

- ***Kelas dasar*** adalah sebuah ***kelas*** yang dijadikan sebagai ***induk*** dari ***kelas lain***.
- ***Kelas dasar*** ini lebih dikenal dengan istilah dalam bahasa asing adalah ***base class***.
- Adapun ***kelas baru*** yang merupakan hasil dari proses ***penurunan*** tersebut disebut dengan ***kelas turunan*** atau sering juga disebut dengan ***derived class***.

KELAS DASAR & KELAS TURUNAN



MEMBUAT KELAS TURUNAN

- Untuk membuat sebuah ***kelas turunan*** sebenarnya sama seperti pada saat kita membuat ***kelas-kelas*** biasa, perbedaanya hanya disini terdapat pendefinisian nama ***kelas*** yang dijadikan sebagai ***kelas dasar***
- Berikut ini adalah bentuk umum dari penulisan ***kelas turunan*** :

```
Class nama_kelas_turunan: hak_akses nama_kelas_dasar {
// data dan fungsi-fungsi
…
};
```

MEMBUAT KELAS TURUNAN

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

// Membuat kelas dasar dengan nama INDUK
class INDUK {
  int X;
public:
  void SetX(int XX) {
    X = XX;
  }
  int GetX() {
    return X;
  }
};

// Membuat kelas turunan dengan nama TURUNAN
class TURUNAN: public INDUK {
  int Y;
public:
  void SetY(int YY) {
    Y = YY;
  }
  int GetY() {
    return Y;
  }
};

// Fungsi utama
int main() {

  // Melakukan instansiasi terhadap kelas INDUK
  INDUK A;

  // Memanggil fungsi-fungsi milik kelas INDUK dari kelas INDUK
  A.SetX(12);

  cout<<"Nilai X yang dipanggil "
      <<"dari kelas INDUK: ";
  cout<<A.GetX()<<endl;

  // Melakukan instansiasi terhadap kelas TURUNAN
  TURUNAN B;

  // Melakukan pemanggilan fungsi-fungsi yang terdapat pada kelas TURUNAN
  B.SetY(40);

  cout<<"Nilai Y yang terdapat "
      <<"pada kelas TURUNAN : ";
  cout<<B.GetY()<<endl;

  // Memanggil fungsi-fungsi milik kelas INDUK dari kelas TURUNAN
  B.SetX(35);

  cout<<"Nilai X yang dipanggil "
      <<"dari kelas TURUNAN : ";
```

MEMBUAT KELAS TURUNAN

- Dari hasil diatas dapat kita simpulkan bahwa ***kelas TURUNAN*** mewarisi sifat-sifat yang diwakili oleh ***kelas INDUK***.
- Sebagai bukti pernyataan ini, diatas kita dapat menggunakan ***fungsi setX()*** dan ***GetX()*** didalam ***objek B*** yang merupakan ***instance*** dari ***kelas TURUNAN***, padahal fungsi-fungsi tersebut adalah milik ***kelas INDUK***.

MEMBUAT KELAS TURUNAN

- Hal ini menunjukan bahwa setelah proses ***penurunan kelas*** diatas, ***kelas INDUK*** akan mewariskan ***fungsi SetX()*** dan ***GetX()*** kepada ***kelas TURUNAN***
- Ini berarti bahwa sekarang ***kelas TURUNAN*** juga telah mempunyai fungsi bernama ***SetX()*** dan ***GetX()***
- Dalam program, hal inilah yang dinamakan dengan ***konsep pewarisan sifat objek***.

MEMBUAT KELAS TURUNAN

```
#include <iostream>

using namespace std;

// Membuat kelas dasar dengan nama INDUK
class INDUK {
  int X;
public:
  void SetX(int XX) {
    X = XX;
  }
  int GetX() {
    return X;
  }
};

// Membuat kelas turunan dari kelas INDUK dengan nama TURUNAN1
class TURUNAN1: public INDUK {
  int Y;
public:
  void SetY(int YY) {
    Y = YY;
  }
  int GetY() {
    return Y;
  }
};

// Membuat kelas turunan dari kelas TURUNAN1 dengan nama TURUNAN2
class TURUNAN2: public TURUNAN1 {
  int Z;
public:
  void SetZ(int ZZ) {
    Z = ZZ;
  }
  int GetZ() {
    return Z;
  }
};

// Fungsi utama
int main() {

  // Melakukan instansiasi terhadap kelas TURUNAN2
  TURUNAN2 A;

  // Menentukan nilai X, Y, dan Z melalui kelas TURUNAN2
  A.SetX(20);
  A.SetY(30);
  A.SetZ(40);

  // Menampilkan nilai X, Y, dan Z melalui kelas TURUNAN2
  cout<<"Nilai X yang dipanggil "
      <<"dari kelas TURUNAN2: ";
  cout<<A.GetX()<<endl;
```

MEMBUAT KELAS TURUNAN

- Hasil diatas menunjukan bahwa kelas ***TURUNAN2*** akan memiliki fungsi-fungsi yang dimiliki oleh ***kelas induknya (kelas TURUNAN1),*** yaitu ***fungsi SetY()*** dan ***GetY()***.
- Namun perlu diperhatikan, disini ***kelas TURUNAN1*** adalah ***turunan dari kelas INDUK,*** sehingga secara otomatis ***kelas TURUNAN2*** juga akan memiliki fungsi-fungsi yang dimiliki oleh ***kelas INDUK*** yaitu ***fungsi SetX()*** dan ***GetX()***.
- Walaupun demikian, sebuah ***kelas turunan*** tetap dapat mewarisi data atau ***fungsi-fungsi*** yang bersifat ***private*** dari ***kelas induknya.***

HAK AKSES PADA PROSES PEWARISAN

- Dalam proses ***penurunan*** sebuah ***kelas***, kita harus benar-benar pandai dalam memberikan ***hak akses data*** atau ***fungsi*** terhadap ***kelas turunan***.
- Berikut ini hal-hal yang perlu diketahui dalam membuat sebuah ***kelas turunan*** adalah :

HAK AKSES PADA PROSES PEWARISAN

- Apabila ***kelas diturunkan*** dengan ***hak akses public*** dari ***kelas induknya***, maka :
  1. ***Bagian public*** yang terdapat pada ***kelas induk tetap*** akan menjadi ***bagian public*** pada ***kelas turunannya***
  2. ***Bagian protected*** yang terdapat pada ***kelas induk tetap*** akan menjadi ***bagian protected*** pada ***kelas turunannya***
  3. ***Bagian private*** yang terdapat pada ***kelas induk tetap*** tidak dapat diakses oleh ***kelas turunannya***

HAK AKSES PADA PROSES PEWARISAN

- Apabila ***kelas diturunkan*** dengan ***hak akses private*** dari ***kelas induknya***, maka :
  1. ***Bagian public*** yang terdapat pada ***kelas induk tetap*** akan menjadi ***bagian private*** pada ***kelas turunannya***
  2. ***Bagian protected*** yang terdapat pada ***kelas induk tetap*** akan menjadi ***bagian private*** pada ***kelas turunannya***
  3. ***Bagian private*** yang terdapat pada ***kelas induk tetap*** tidak dapat diakses oleh ***kelas turunannya***

HAK AKSES PADA PROSES PEWARISAN

```
#include <iostream>

using namespace std;

// Membuat kelas dasar dengan nama INDUK
class INDUK {
  int X;
public:
  void SetX(int XX) {
    X = XX;
  }
  int GetX() {
    return X;
  }
};

// Membuat kelas TURUNAN1 yang diturunkan sebagai public dari kelas INDUK
class TURUNAN1: public INDUK {
  int Y;
public:
  void SetY(int YY) {
    Y = YY;
  }
  int GetY() {
    return Y;
  }
};

// Membuat kelas TURUNAN2 yang diturunkan sebagai public dari kelas TURUNAN1
class TURUNAN2: public TURUNAN1 {
  int Z;
public:
  void SetZ(int ZZ) {
    Z = ZZ;
  }
  int GetZ() {
    return Z;
  }
};

// Fungsi utama
int main() {

  // Melakukan instansiasi terhadap kelas TURUNAN2
  TURUNAN2 A;

  // Menentukan nilai X, Y, dan Z melalui kelas TURUNAN2
  A.SetX(20);
  A.SetY(30);
  A.SetZ(40);

  // Menampilkan nilai X, Y, dan Z melalui kelas TURUNAN2
  cout<<"Nilai X yang dipanggil "
      <<"dari kelas TURUNAN2: ";
  cout<<A.GetX()<<endl;
  cout<<"Nilai Y yang dipanggil "
      <<"dari kelas TURUNAN2: ";
  cout<<A.GetY()<<endl;
  cout<<"Nilai Z yang dipanggil "
      <<"dari kelas TURUNAN2: ";
  cout<<A.GetZ()<<endl;
```

HAK AKSES PADA PROSES PEWARISAN

- Apabila kita analisis pada program diatas, ***bagian public*** dari ***kelas induk (yaitu fungsi SetX()*** dan ***GetX()*** tetap akn menjadi ***public*** pada ***kelas TURUNAN1***, sedangkan ***variabel X*** yang terdapat pada ***bagian private*** tetap tidak dapat diakses oleh ***kelas TURUNAN1***
- Sampai disini berarti ***kelas TURUNAN1*** telah mempunyai ***empat buah fungsi*** yang bersifat ***public (yaitu fungsi SetY(), GetY(), SetX() dan GetX())***.

HAK AKSES PADA PROSES PEWARISAN

- Selanjutnya ***kelas TURUNAN1*** akan diturunkan lagi secara ***public*** terhadap ***kelas TURUNAN2***, hal ini menyebabkan bagian ***public*** yang terdapat pada ***kelas TURUNAN1*** akan diwariskan semuanya kepada ***kelas TURUNAN2*** sehingga ***kelas TURUNAN2*** akan mempunyai ***enam buah fungsi*** yang bersifat ***public (yaitu fungsi SetZ(), GetZ(), SetY(), GetY(), SetX() dan GetX())***.
- Itulah sebabnya pada program diatas kita dapat mengkses fungsi-fungsi yang terdapat pada ***kelas INDUK*** dan ***kelas TURUNAN1*** melalui suatu ***objek (instance)*** dari ***kelas TURUNAN2***.

FUNGSI VIRTUAL

- ***Fungsi Virtual*** adalah ***fungsi*** yang mendukung adanya ***polymorphic function***, artinya ***fungsi*** tersebut dapat didefinisikan ulang pada ***kelas-kelas turunannya***.
- ***Fungsi virtual*** ini biasanya terdapat pada ***kelas-kelas dasar***, tetapi bisa juga dideklarasikan pada ***kelas-kelas turunan*** yang akan dijadikan sebagai ***kelas dasar*** bagi ***kelas-kelas lainnya***.

FUNGSI VIRTUAL

- Dalam C++, untuk mendefinisikan fungsi sebagai fungsi virtual adalah dengan menggunakan keyword virtual, yaitu dengan menempatkan didepan pendeklarasian fungsi tersebut.
- Pendefinsian fungsi virtual yang terdapat pada kelas dasar biasanya tidak begitu berarti artinya kode-kode yang terdapat didalamnya masih bersifat umum.