POLYMORPHISM

Oleh:

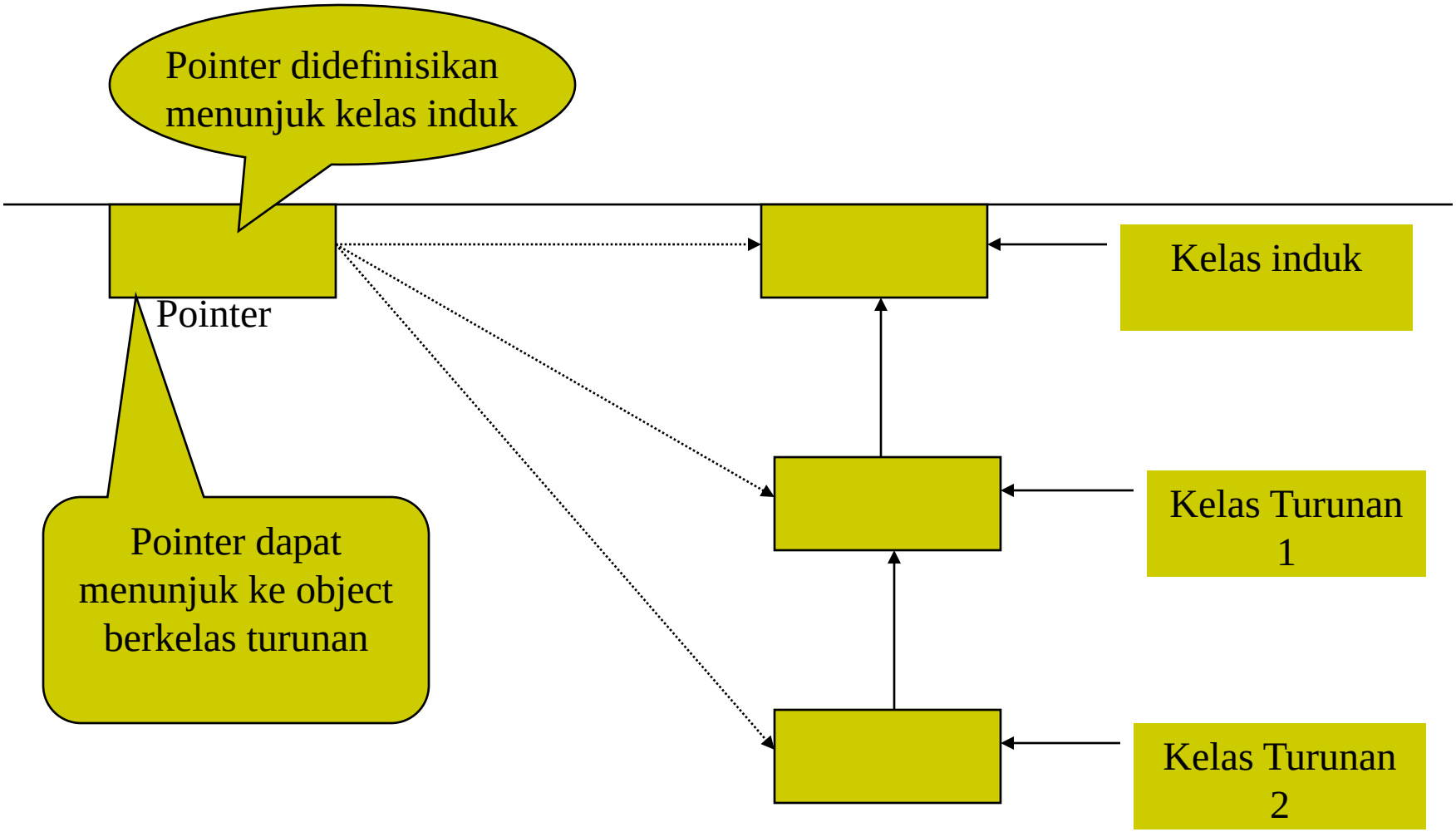
Rasim

Ilkom- FPMIPA

Universitas Pendidikan Indonesia

Fungsi Virtual

- Dasar dari Polymorphism
- Suatu fungsi anggota dari suatu kelas dapat dijadikan sebagai kelas virtual, dengan cara: mendeklarasikan fungsi tsb pada kelas turunan dan suatu pointer mennjuk kelas induk.
- Pointer dapat memilih object yang tepat pada saat fungsi anggota tsb dipanggil via pointer


Pointer didefinisikan menunjuk kelas induk
Pointer
Pointer dapat menunjuk ke object berkelas turunan
Kelas induk
Kelas Turunan 1
Kelas Turunan 2

```
Class Mahluk{
    public:
        void info(){
            cout <<“Informasi() pada mahluk ….” << endl;
        }
        virtual void keterangan(){
            cout << “Keterangan () pada mahluk … “ << endl;
        }
};
Class Mamalia: public Mahluk{
    public:
        void info(){
            cout << “Informasi () pada Mamalia….” << endl;
        }
        void keterangan(){
            cout<<“Keterangan() pada Mamalia …” <<endl;
        }
};
```

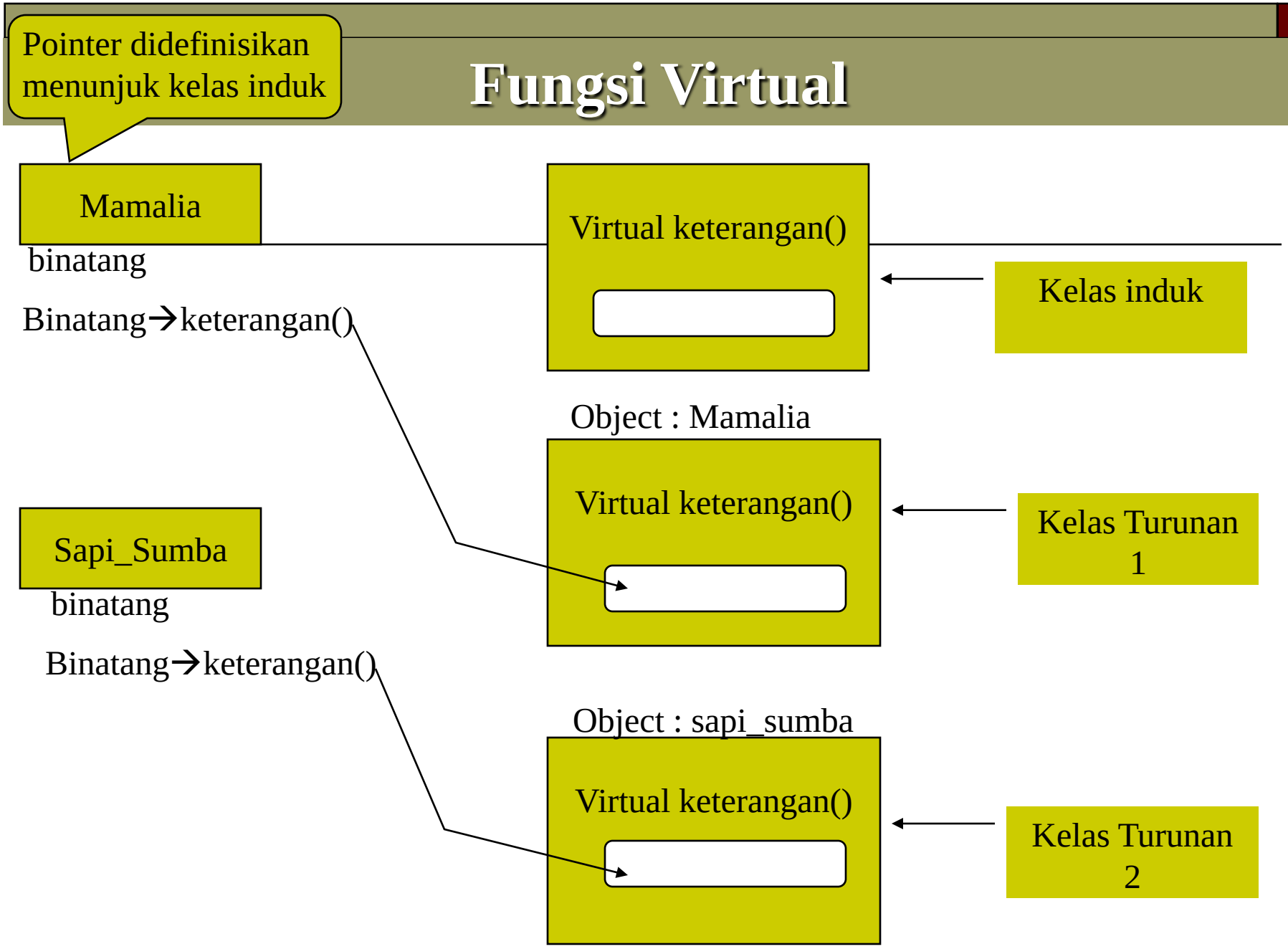
```
Class Sapi: public Mamalia{
      public:
            void info(){
                  cout <<“Informasi() pada Sapi ….” << endl;
            }
            void keterangan(){
                  cout << “Keterangan () pada Sapi … “ << endl;
            }
};
Void  Main() {
       Mamalia mamalia;
       Sapi sapi_sumba;
       Mahluk *binatang;

       binatang=&mamalia;
      binatang→info();
      binatang→keterangan();
     cout<<“----------------------”<<endl;
```

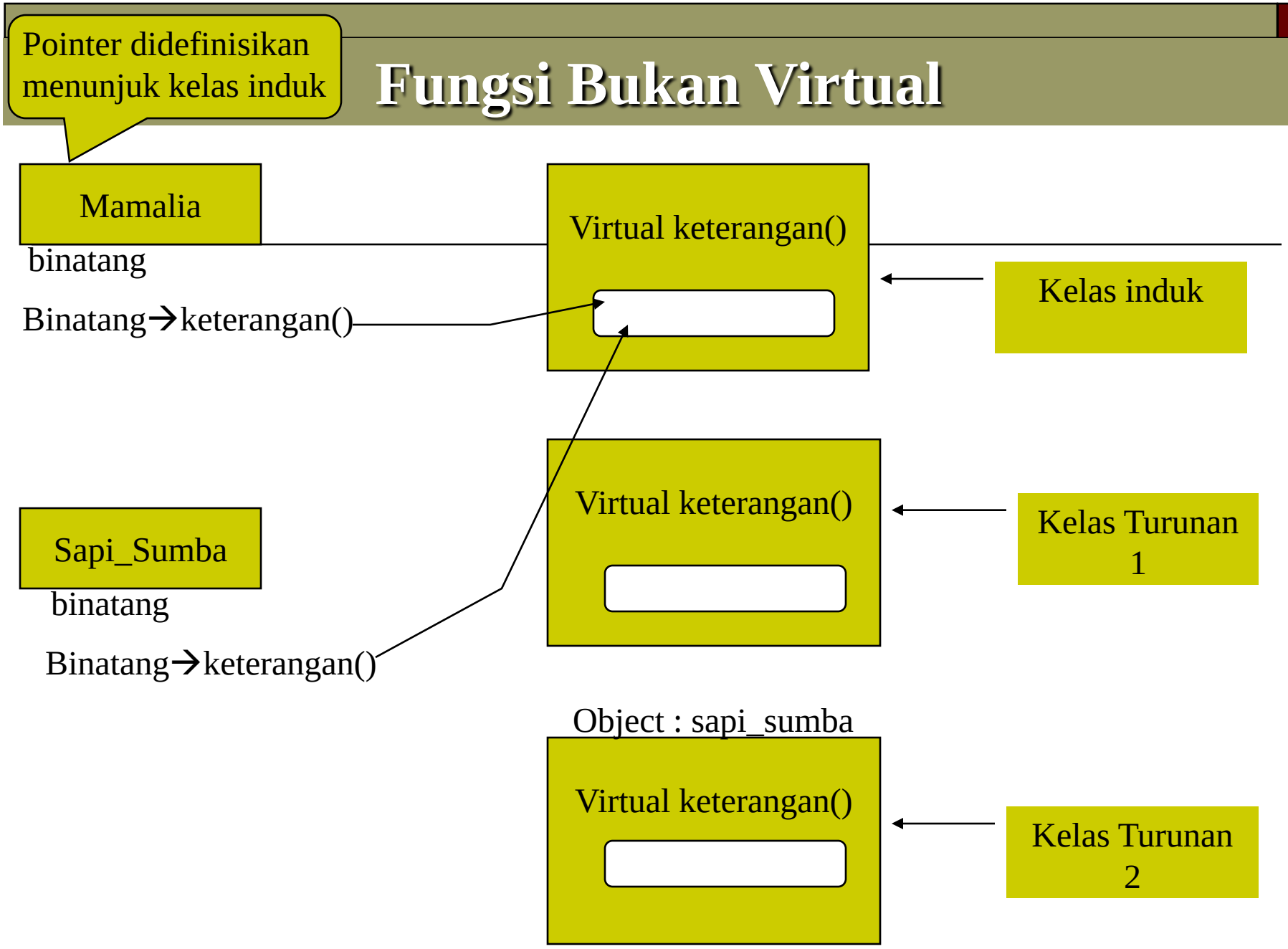
```
Binatang = &sapi_sumba;
Binatang→info();
Binatang→keterangan();


};
```

Fungsi Virtual



Fungsi Bukan Virtual



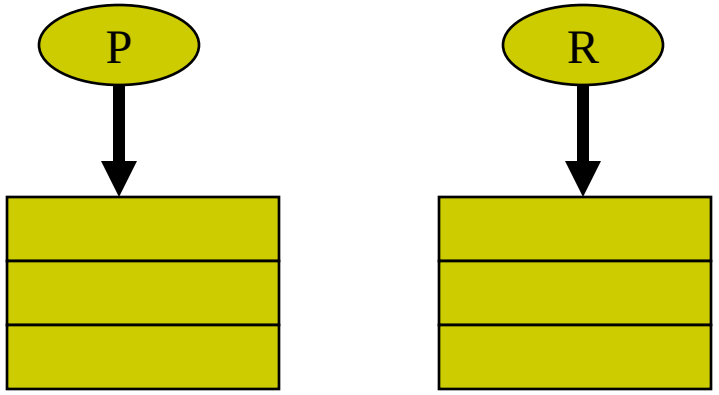
Selayang Polymorphism

- Akibat dari pewarisan, dapat menimbulkan polymorphism
- Artinya: pada saat yang sama sebuah reference dapat mengacu ke object yang berbeda kelasnya (namun harus se-keturunan)
- Jika dideklarasikan:
  - P : Poligon;
  - R : Rectangle; // Rectangle adalah poligon
  - T : Triangle; // Triangle adalah poligon

Selayang Polymorphism

- Polymorphism dapat beroperasi pada 2 aras:
  - Saat kompilasi
  - Saat Eksekusi
- Overloading fungsi dan operator adalah polymorphism saat kompilasi
- Polymorphism pada C++ adalah polymorphism saat eksekusi (late binding/ dynamic binding)
  - Bentuk ini dapat menangani dua atau lebih bentuk object sesuai dengan lingkungan object bersangkutan

- Karena kecocokan type antara induk dan turunannya, maka
  - P = R; // valid
  - P = T; // valid

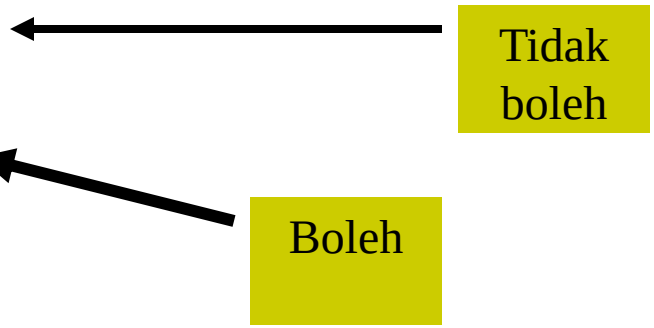


Fungsi Virtual Murni & Class Abstrak

- Fungsi virtual yang tidak dipakai dalam program karena fungsi tersebut dipakai oleh objek yang ditunjuk
- Dari contoh virtual void keterangan() pada class mahluk.
- Oleh sebab itu dapat disingkirkan dengan cara:
  - Virtual void keterangan()=0;
- Fungsi virtual murni biasanya dipakai sebagai kelas abstrak

Kelas Abstrak

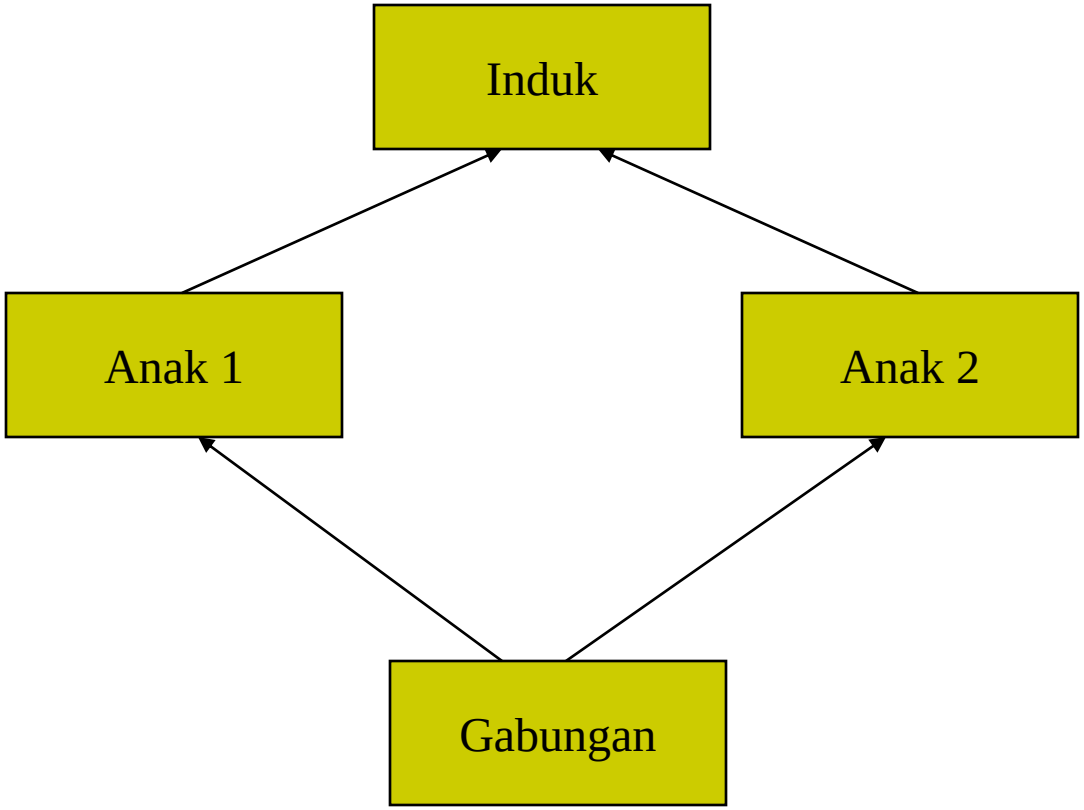
- Adalah kelas yang dideklarasikan idak untuk menciptakan object.
- Cirinya:
  - Mengandung paling tidak sebuah fungsi virtual murni
- Sehingga:
  - Mahluk binatang
  - Mahluk *binatang



```
Class Mahluk{
    public:
        virtual void keterangan(){
            cout << "Keterangan () pada mahluk … " << endl;
        }
};
Class Mamalia: public Mahluk{
    public:
        void keterangan(){
            cout<<"Keterangan() pada Mamalia …" <<endl;
        }
};
Class Sapi: public Mamalia{
    public:
        void keterangan(){
            cout << "Keterangan () pada Sapi … " << endl;
        }
};
```

```
Void  Main() {
        Mamalia mamalia;
        Sapi sapi_sumba;
        Mahluk *binatang;
        binatang=&mamalia;
        binatang→info();
        binatang→keterangan();
        binatang = &sapi_sumba;
        binatang→info();
        binatang→keterangan();
};
```

Kelas Dasar Virtual



Deklarasi kelas tersebut adalah

```
Class induk{
      protected:
            char nama[25];
};
Class anak1: virtual public induk{

};
Class anak2: virtual public induk{

};

Class gabungan: public anak1, public anak2{
    public:
        void info_nama(){
              cout <<"Nama Induk : " << nama << endl;
        }
}
```

Tambahkan kata virtual

Tambahkan kata virtual

Nama didapat dari anak1 dan anak2, karena itu menjadi salah

Destruktor Virtual

- Fungsi yang dapat dijadikan virtual adalah:
  - Fungsi anggota
  - Destruktor
- Destruktor virtual dipakai kalau kelas perlu untuk menghapus object dari kelas turunan berdasarkan pointer yang menunjuk ke kelas induk

```
Class trah_keluarga{
      protected:
          char *nama_keluarga;
      public:
           trah_keluarga(char *nama){
                      nama= new char[strlen(nama)+1];
                       strcpy(nama_keluarga, nama);
           }
            virtual ~trah_keluarga(){
                    cout<<“Destruktor di trah keluarga ….” << endl;
                    delete [ ] nama_keluarga;
            }
            virtual void info() = 0;
};
```

```
Class keturunan: public trah_keluarga{
            char *nama_depan;
      public:
            keturunan(char *nama_awal, char *nama_kel): trah_keluarga(nama_kel){
                nama_depan= new char[strlen(nama_depan)+1];
                strcpy(nama_depan, nama_awal);
              }
        ~keturunan(){
              cout<<“Destruktor di keturunan …” <<endl;
              delete [ ] nama_depan;
        }
        void info(){
               cout<<“Nama depan  : “ << nama_depan << endl;
       }
};
Void main(){
        trah_keluarga *anak_pertama= new keturunan(“Umar”, “Bakri”);
        anak_pertama→info();
        delete anak_pertama
};
```