PEMROGRAMAN LANJUT

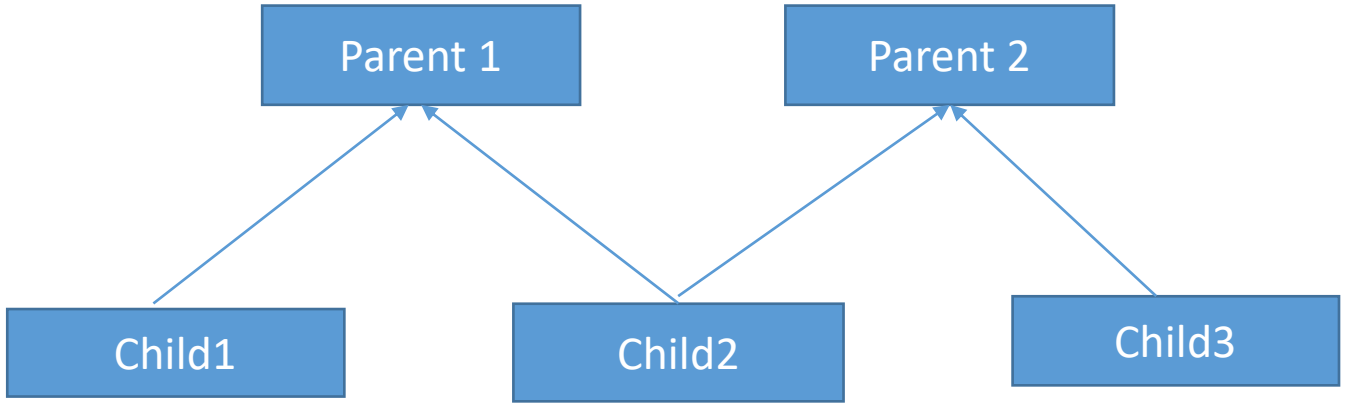
05 |Multiple Inheritance in C++

Team Teaching Pemrograman Lanjut

Fakultas Ilmu Komputer UB

Multiple Inheritance

- Pewarisan Ganda (Multiple Inheritance) adalah sebuah class yang diturunkan dari beberapa parent class.
- Klas turunan tersebut dapat mewarisi karakteristik yang dimiliki oleh class yang menjadi parent nya.
- Beberapa bentuk hirarki class yang merupakan contoh pewarisan ganda (*multiple inheritance*) dapat dilihat seperti di bawah ini:



C++ Syntax for multiple inheritance

- Setiap parent class yang mewarisi dipisahkan dengan tanda koma, dan masing-masing parent class harus didahului dengan jenis akses (private, protected, public).

class Child **: jenis_akses** Parent1**, jenis_akses** Parent2

{ <Anggota_klas> };

Contoh Multiple Inheritance


```
#include <iostream>
using namespace std;
class Pegawai
{         protected:
                  string nip;
                  string nama;
          public:
                  void setDataPegawai(string nip,string nama){
                           this->nip=nip;
                           this->nama=nama;
                  }
                  void tampilDataPegawai(){
                           cout<<"NIP:"<<this->nip<<endl;
                           cout<<"Nama:"<<this->nama<<endl;
                  }
};
```

```
class Pendidikan
{          protected:
                    string tingkat;
                    string universitas;
           public:
                    void setDataPendidikan(string tingkat,string universitas){
                              this->tingkat=tingkat;
                              this->universitas=universitas;
                    }
                    void tampilDataPendidikan(){
                              cout<<"tingkat:"<<this->tingkat<<endl;
                              cout<<"universitas:"<<this->universitas<<endl;
                    }
};
```

```
class Peneliti : public Pegawai, public Pendidikan
{          private:
                    int publikasi;
           public:
                    void setDataPeneliti(int publikasi){
                              this->publikasi=publikasi;
                    }
                    void tampilDataPeneliti(){
                              cout<<"jumlah publikasi:"<<this->publikasi<<endl;
                    }
};
```

```
int main(){
            Peneliti peneliti;
            peneliti.setDataPegawai("0011","adi");
            peneliti.setDataPendidikan("sarjana","universitas brawijaya");
            peneliti.setDataPeneliti(10);
            peneliti.tampilDataPegawai();
            peneliti.tampilDataPendidikan();
            peneliti.tampilDataPeneliti();
}
```

Konstruktor dan Destructor child class

- Konstruktor untuk child class perlu dibuat untuk melakukan inisialisasi class itu sendiri dan parent class.
- Jika konstruktor parent class memiliki parameter (argument), maka parameter konstruktor parent class tersebut harus dimasukkan dalam konstruktor child class
- Destruktor child class dijalankan dengan urutan berbalikan dengan urutan konstruktor, yaitu destruktor child class dijalankan dahulu baru destruktor parent class
- Apabila child class memiliki lebih dari satu parent class, setelah menjalankan destruktor child class kemudian destruktor parent class terakhir dijalankan sampai destruktor klas induk pertama dijalankan (sesuai dengan urutan terbalik dari penurunan parent class di dalam definisi child class tersebut).

Contoh


```
#include <iostream>
using namespace std;
class Pegawai
{          protected:
                    string nip;
                    string nama;
           public:
                    Pegawai(){cout<<"konstruktor Pegawai dijalankan"<<endl;}
                    ~Pegawai(){cout<<"destruktor Pegawai dijalankan"<<endl;}
                    void setDataPegawai(string nip,string nama){
                              this->nip=nip;
                              this->nama=nama;
                    }
                    void tampilDataPegawai(){
                              cout<<"NIP:"<<this->nip<<endl;
                              cout<<"Nama:"<<this->nama<<endl;
                    }
};
```

```
class Pendidikan
{          protected:
                    string tingkat;
                    string universitas;
           public:
                    Pendidikan(){
                              cout<<"konstruktor Pendidikan dijalankan"<<endl;
                    }
                    ~Pendidikan(){
                              cout<<"destruktor Pendidikan dijalankan"<<endl;
                    }
                    void setDataPendidikan(string tingkat,string universitas){
                              this->tingkat=tingkat;
                              this->universitas=universitas;
                    }
                    void tampilDataPendidikan(){
                              cout<<"tingkat:"<<this->tingkat<<endl;
                              cout<<"universitas:"<<this->universitas<<endl;
                    }
};
```

```
class Peneliti : public Pegawai, public Pendidikan
{       private:
                int publikasi;
        public:
                Peneliti(){
                        cout<<"konstruktor Peneliti dijalankan"<<endl;
                }
                ~Peneliti(){
                        cout<<"destruktor Peneliti dijalankan"<<endl;
                }
                void setDataPeneliti(int publikasi){
                        this->publikasi=publikasi;
                }
                void tampilDataPeneliti(){
                        cout<<"jumlah publikasi:"<<this->publikasi<<endl;
                }
};
```

```
int main(){
            Peneliti peneliti;
            peneliti.setDataPegawai("0011","adi");
            peneliti.setDataPendidikan("sarjana","universitas brawijaya");
            peneliti.setDataPeneliti(10);
            peneliti.tampilDataPegawai();
            peneliti.tampilDataPendidikan();
            peneliti.tampilDataPeneliti();
}
```