DOKUMENTASI

PROGRAM SHAPE

Pemrograman Lanjut

Oleh :

Deddy Van Hauten / 1400910039

Felicia / 1400910022

Poh Steven Sean Davin / 1400910044

PROGRAM STUDI HUMAN-COMPUTER INTERACTION

FAKULTAS ILMU HAYATI

UNIVERSITAS SURYA

SERPONG

2015

Inheritance

Inheritance adalah suatu konsep pemrograman yang sering kita gunakan dalam OOP. Inheritance berarti pewarisan dalam Bahasa Indonesia, yang berarti pada kode inheritance, kita akan menggunakan lagi kode yang ada pada kelas induk kemudian diturunkan ke kelas – kelas turunan.

Salah satu keuntungan dari penggunaan Inheritance adalah programmer tidak perlu mengetik coding berulang kali karena dapat mengambil code dari kelas induk yang kemudian dapat digunakan berulang kali.

Sifat Inheritance ini kami gunakan dalam class rectangle, circle dan square untuk mengakses variabel public dan protected class shape seperti variabel luas dan keliling.

Berikut potongan code dari kelompok kami yang menggunakan konsep Inheritance :

```
class Shape
{
        public:
                //tampilluas();
                string jenisShape;
                string getJenisShape();
                float getLuas();
                float getKeliling();
                virtual void hitungLuas()=0; //pure virtual function
                virtual void hitungKeliling()=0;
                virtual void printDetails()=0;
                //Untuk sorting:
                static bool sortByLuas(Shape *a, Shape *b);
                static bool sortByKeliling(Shape *a, Shape *b);
        protected:
                float luas;
                float keliling;
};

class Rectangle : public Shape
```

```
{
        public:
                Rectangle();
                //void loadFromFile(char *filename);
                //void tampilluas();
                //void tampilkeliling();
                //void tambah(int a, int b);
                //void hapus(int a, int b);
                //void saveToFile(char *filename);
                Rectangle(int panjangBaru, int lebarBaru);
                void hitungLuas();
                void hitungKeliling();
                void printDetails();
                int getpanjang();
                int getlebar();
        protected:
                int panjang;
                int lebar;
};
```

Ada beberapa alasan code kami menggunakan inheritance, antara lain :

1. Agar tidak menuliskan code yang sama berulang – ulang.
2. Coding lebih bersih karena tidak banyak code yang diulang.
3. Karena code singkat waktu compile dan running juga akan menurun.
4. Agar data yang disimpan dalam class shape bisa diakses class lainnya yang merupakan turunannya

Virtual Function:

Fungsi atau metode yang perilakunya dapat diganti dalam sebuah kelas inheritance dengan fungsi yang memiliki signature yang sama. Konsep ini merupakan bagian penting dari bagian polymorphism pemrograman berorientasi objek (OOP). Fungsi ini dideklarasikan menggunakan kata kunci virtual. Virtual function bertujuan agar tidak ada static linkage untuk fungsi ini.

Sifat virtual function ini kami gunakan dalam class shape untuk menampilkan nilai keliling dan luas.

Berikut potongan code dari kelompok kami yang menggunakan konsep Virtual Function :

```
class Shape
{
	public:
		//tampilluas();
		string jenisShape;
		string getJenisShape();
		float getLuas();
		float getKeliling();
		virtual void hitungLuas()=0; //pure virtual function
		virtual void hitungKeliling()=0;
		virtual void printDetails()=0;
		//Untuk sorting:
		static bool sortByLuas(Shape *a, Shape *b);
		static bool sortByKeliling(Shape *a, Shape *b);
	protected:
		float luas;
		float keliling;
};
```

Kelas dasar abstract

Kelas dasar abstrak adalah tipe nominatif sistem yang tidak dapat dipakai langsung. Jenis abstrak juga dikenal sebagai jenis eksistensial. Seringkali, jenis abstrak akan memiliki satu atau lebih implementasi yang disediakan secara terpisah. Mereka adalah kelas yang hanya dapat digunakan sebagai kelas dasar, dan dengan demikian diperbolehkan untuk memiliki fungsi anggota virtual tanpa definisi (dikenal sebagai fungsi virtual murni). Perintahnya adalah untuk mengganti definisi mereka dengan = 0 (dan tanda sama dan nol):

Penggunaan kelas dasar abstract dalam code kami adalah pada kelas shape, untuk mengembalikan nilai dari luas setiap bentuk (circle/rectangle/square)

Berikut penggalan kode program kami yang menggunakan kelas dasar abstract.

```
class Shape
{
	public:
		//tampilluas();
		string jenisShape;
		string getJenisShape();
		float getLuas();
		float getKeliling();
		virtual void hitungLuas()=0; //pure virtual function
		virtual void hitungKeliling()=0;
		virtual void printDetails()=0;
		//Untuk sorting:
		static bool sortByLuas(Shape *a, Shape *b);
		static bool sortByKeliling(Shape *a, Shape *b);
	protected:
		float luas;
		float keliling;
};
```

Error Checking

Error checking adalah cara yang digunakan untuk melaporkan dan menangani kesalahan logika dan kesalahan runtime dengan menggunakan pengecualian.

Error checking ini kami gunakan dalam mengecek input yang digunakan user, untuk mengecek apakah input berupa angka atau tidak kami menggunakan cin.fail() sedangkan untuk error yang lain kami menggunakan try and catch.

Berikut potongan code dari kelompok kami yang menggunakan konsep Error Checking :

```
void tampilpersegipanjang()
{
	loop :
	int input;
	header();
	cout << "MENU TAMPILKAN BENTUK PERSEGI PANJANG" << endl;
	cout << "1. Urutkan Berdasarkan Luas" << endl;
	cout << "2. Urutkan Berdasarkan Keliling" << endl;
	cout << "3. Kembali ke Menu Tampilkan Bentuk" << endl;
	cout << "Silahkan Masukkan pilihan Anda (1-3): " << endl;
```

```
        cin >> input;
        if (cin.fail())
        {
    cout <<"Maaf input yang anda masukkan harus berupa angka, silahkan masukan kembali"<<endl;
    cin.clear ();
    cin.ignore();
        goto loop;
  } else {
     switch(input)
     {
        case 1:luaspersegipanjang(); break;
        case 2:kelilingpersegipanjang(); break;
        case 3:tampilbentuk(); break;
        default : {
        cout<<"Maaf input anda salah, masukkan angka 1/2/3"<<endl;
                        goto loop;
        }
     }
  }
}
void tambahlingkaran()
{
        header();
        cout<<"Masukkan jari-jari : ";
        try
        {
                cin>>input;
                if (cin.fail())
                {
                        cout <<"Maaf input yang anda masukkan harus berupa angka, silahkan masukan kembali"<<endl;
                        cin.clear ();
                        cin.ignore();
                        tambahlingkaran();
                } else
                {
                        if (input>0) shapes.push_back(new Circle(input)); else throw"Input tidak boleh negatif";
                        luaslingkaran();
                }
        }
        catch (const char* e)
        {
                cerr << e << endl;
                cout << endl;
                tambahlingkaran();
        }
```

```
}
```