Pertemuan 4

Pemrograman Berbasis Obyek (PBO)

(Overloading dan Overriding)

A. Tujuan

1. Mahasiswa memahami konsep overloading
2. Mahasiswa memahami konsep overriding

B. Dasar Teori

Overloading adalah suatu program dimana beberapa method dapat mempunyai nama yang sama, akan tetapi mempunyai fungsionalitas yang berbeda. Contoh penggunaan overloading dilihat dibawah ini:

```
objek(int t1) //Terdapat 1 parameter, untuk menggambar titik
objek(int t1,int t2) //Terdapat 2 parameter, untuk menggambar Garis
objek(int t1,int t2,int t3) //Terdapat 3 parameter, untuk menggambar Segitiga
objek(int t1,int t2,int t3,int t4) //Terdapat 1 parameter, untuk menggambar Persegi
```

Overloading ini dapat terjadi pada class yang sama atau pada suatu parent class dan subclass-nya. Overloading mempunyai ciri-ciri sebagai berikut:

1. Nama method harus sama
2. Daftar parameter harus berbeda
3. Return type boleh sama atau berbeda

Overriding adalah sebuah situasi dimana member function class turunan menimpa member function milik parent class. Ini bisa terjadi jika terdapat nama function yang sama di child class dan juga parent class.

```
#include <iostream>
using namespace std;
class Mobil {
      public:
      string cekInfo() {
            return "Ini berasal dari class Mobil”;
      }
};
```

```
class Garuda: public Mobil {
      public:
      string cekInfo() {
            return "Ini berasal dari class Garuda”;
      }
};
int main()
{     Garuda viar;
      cout << viar.cekInfo() << endl;
      return 0;
}
```

Pada kode program tersebut, class **Mobil** diturunkan kepada class **Garuda**. Perhatikan bahwa di dalam kedua class terdapat member function **cekInfo()** dengan hak akses **public**. Di dalam function main, dilakukan meng-instansiasi class **Garuda** ke dalam variabel **viar**, lalu menjalankan perintah **cout << viar.cekInfo()**.Pada program tersebut ternyata function cekInfo() milik class Garuda-lah yang dijalankan. Dalam kasus ini sudah terjadi **function overriding**, dimana member function cekInfo() milik class Garuda menimpa member function cekInfo() milik class Mobil. Overriding menyebabkan kita tidak bisa mengakses member function milik parent class.

Untuk melakukan akses pada function milik parent class dapat dilakukan melalui operator khusus yang disebut sebagai **scope resolution operator**. Operator ini ditulis dengan tanda titik dua, dua kali “**::**“. Berikut contoh penggunaannya:

```
class Mobil {
      public:
      string cekInfo() {
            return "Ini berasal dari class Mobil”;
      }
};
class Garuda: public Mobil {
      public:
      string cekInfo() { return "Ini berasal dari class Garuda”; }
};
int main()
{     Garuda viar;
      cout << viar.cekInfo() << endl;
      cout << innoba.Mobil::cekInfo() << endl;
      return 0;
}
```