MATERI 5 – POLIMORPHISM & ABSTRACT CLASS

PEMROGRAMAN LANJUT S1 TI

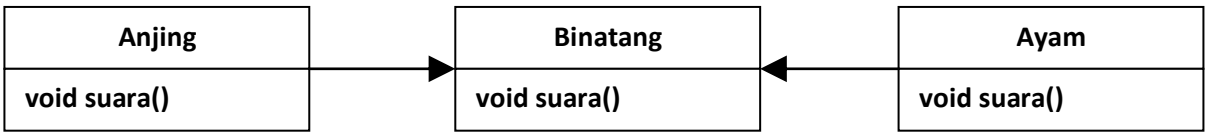
STEVI EMA WIJAYANTI

POLIMORFISME

DEFINISI POLIMORFISME

Polimorfisme adalah kemampuan suatu objek untuk mengungkap banyak hal melalui satu cara yang sama. Seperti konsep ***override*** yang sudah pernah dipelajari, konsep ini mendukung proses polimorfisme. Polimorfisme merupakan salah satu esensial dalam konsep OOP karena mengijinkan kelas induk untuk mendefinisikan sebuah method bersifat general (bersifat umum) untuk semua kelas turunannya dan selanjutnya kelas-kelas turunan dapat memperbaharui implementasi dari method tersebut secara lebih spesifik sesuai dengan karakteristik masing-masing.

Contoh :



Seperti contoh diatas :

Class Binatang merupakan BASE CLASS (Class induk dari class Anjing dan Ayam)

Di dalam class Binatang, Anjing dan Ayam sama-sama memiliki method **suara()**.

Jika dituliskan dalam kode program maka akan menghasilkan program seperti berikut :

```
class Binatang {
    public void suara(){
        Console.WriteLine("tidak terdefinisi");
    }
}

class Anjing:Binatang{
    //override function
    public void suara(){
        Console.WriteLine("Anjing menggonggong");
    }
}

class Ayam:Binatang {
    //override function
    public void suara(){
        Console.WriteLine("Ayam berkokok");
    }
}
```

```
public class Latihan1 {
    static void main(string[] args) {
        Binatang binatang;

        Anjing anjing = new Anjing();
        Ayam ayam = new Ayam();

        binatang = ayam;
        binatang.suara();

        binatang = anjing;
        binatang.suara();
    }
}
```

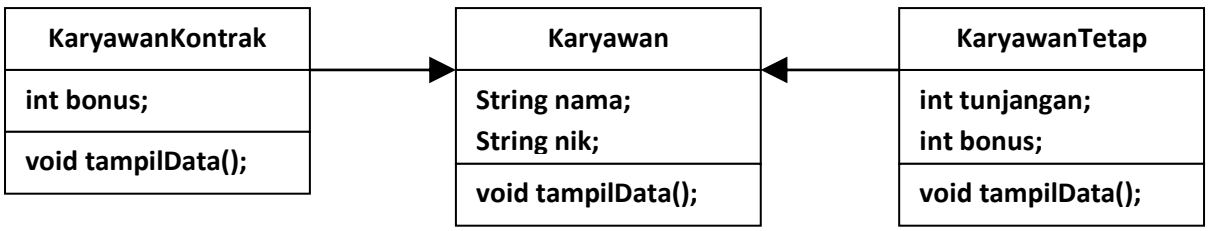
Maka output yang dihasilkan adalah :

```
Ayam berkokok
Anjing menggonggong
```

Penjelasan dari listing program diatas :

Seperti yang dapat dilihat dari listing program diatas, kita mendeklarasikan variabel **binatang** dengan tipe **Binatang**. Pada tahap tersebut kita belum mengetahui apakah **binatang** merupakan **Anjing** atau **Ayam.** Namun pada saat **binatang** mengacu pada obyek di kelas **Anjing** dan kemudian memanggil method **suara()** melalui referensi tersebut, maka method yang akan dieksekusi adalah method **suara()** yang ada pada kelas **Anjing** bukan dari kelas **Binatang**. Ini artinya bahwa method yang akan dipanggil oleh **binatang** akan tergantung dari obyek yang sedang ditunjuk. Begitu pula yang terjadi pada kelas **Ayam**. Hal seperti inilah yang disebut dengan konsep **POLIMORFISME**.

Contoh lain :



```
class Karyawan {
    protected string nama,nik;
    protected int gapok;

    public Karyawan() {
        nama = "Stevi Ema W.";
        nik = "12345";
        gapok = 1000000;
    }

    public void tampilData(){
        Console.WriteLine("Nama Karyawan : " + nama);
        Console.WriteLine("NIK Karyawan : " + nik);
        Console.WriteLine("Gaji Pokok : " + gapok);
    }
}

class KaryawanTetap:Karyawan {
    int tunj,bonus;

    public KaryawanTetap() {
        tunj = 500000;
        bonus = 350000;
    }

    public void tampilData(){
        Console.WriteLine("Nama Karyawan : " + nama);
        Console.WriteLine("NIK Karyawan : " + nik);
        Console.WriteLine("Gaji Pokok : " + gapok);
        Console.WriteLine("Tunjangan : " + tunj);
        Console.WriteLine("Bonus : " + bonus);
    }
}

class KaryawanKontrak:Karyawan{
    int bonus;

    public KaryawanKontrak() {
        bonus = 250000;
    }

    public void tampilData(){
        Console.WriteLine("Nama Karyawan : " + nama);
        Console.WriteLine("NIK Karyawan : " + nik);
        Console.WriteLine("Gaji Pokok : " + gapok);
        Console.WriteLine("Bonus : " + bonus);
    }
}
```

```
public class Latihan2 {
    public static void main(String[] args) {
        Karyawan karyawan = new Karyawan;

        KaryawanTetap tetap = new KaryawanTetap();
        KaryawanKontrak kontrak = new KaryawanKontrak();

        karyawan = tetap;
        karyawan.tampilData();

        System.out.println(" ");

        karyawan = kontrak;
        karyawan.tampilData();
    }
}
```

Output dari program diatas adalah :

```
Nama Karyawan : Stevi Ema W.
NIK Karyawan : 12345
Gaji Pokok : 1000000
Tunjangan : 500000
Bonus : 350000

Nama Karyawan : Stevi Ema W.
NIK Karyawan : 12345
Gaji Pokok : 1000000
Bonus : 250000
```

ABSTRACT CLASS

Pada pemrograman C#, apabila kita ingin mendefinisikan sebuah kelas induk (**BASE CLASS**) yang memiliki deklarasi method namun tidak memerlukan implementasi sama sekali. Fungsi-fungsi tersebut selanjutnya baru akan diimplementasikan oleh kelas-kelas turunannya (**DERIVED CLASS**). Fungsi seperti ini disebut dengan **fungsi abstrak (abtract function)**. Untuk mendeklarasikan suatu fungsi menjadi fungsi abstrak, kita perlu menambahkan kata kunci **abstract** didepan deklarasi fungsi.

Contoh penulisan **abtrak function :**

```
abstract  tipe  NamaFungsi (daftar-parameter) ;
```

Kelas yang memiliki satu atau lebih fungsi abstrak harus dideklarasikan juga sebagai kelas abstrak. Berikut ini adalah contoh pendeklarasian kelas abstrak :

```
abstract  class  NamaKelas {
        //pernyataan;
}
```

Kelas abstrak merupakan suatu bentuk khusus dari kelas dimana kelas tersebut tidak dapat diinstansiasi dan digunakan hanya untuk diturunkan ke dalam bentuk kelas konkret atau kelas abstrak berikutnya. Dengan kata lain tidak diperbolehkan untuk membentuk obyek dari kelas abstrak. Meskipun demikian, masih diperbolehkan untuk mendeklarasikan variabel refensi ke kelas abstrak. Variabel referensi tersebutlah yang akan digunakan untuk menunjuk/mengacu ke obyek-obyek dari kelas turunan.

Contoh program :

```
//class abstract
abstract class Binatang {
    //method abstract --> tidak diimplementasikan
    public abstract void berjalan();
    public abstract void bersuara();
}

class Kucing : Binatang{
    //implementasi abstract method pada class turunan
    public void berjalan(){
        Console.WriteLine("Kucing berjalan dengan 4 kaki");
    }

    public void bersuara(){
        Console.WriteLine("kucing bersuara dengan mengeong");
    }
}

class Ayam : Binatang {
    //implementasi abstract method pada class turunan
    public void bersuara(){
        Console.WriteLine("ayam bersuara dengan berkokok");
    }

    public void berjalan(){
        Console.WriteLine("Ayam berjalan dengan 2 kaki");
    }
}
```

```
public class Latihan {
    public static void main(String[] args) {
        Binatang hewan;

        Kucing tom = new Kucing();
        Ayam chicken = new Ayam();

        hewan = tom;
        hewan.berjalan();
        hewan.bersuara();

        hewan = chicken;
        hewan.bersuara();
        hewan.berjalan();
    }
}
```

Output dari program tersebut adalah :

```
Kucing berjalan dengan 4 kaki
kucing bersuara dengan mengeong
ayam bersuara dengan berkokok
Ayam berjalan dengan 2 kaki
```

Pada kode program di atas dideklarasikan kelas **Binatang** yang abstrak, dimana didalamnya dideklarasikan dua buah method, yaitu **berjalan()** dan **bersuara()**. Fungsi didalam kelas **Binatang** tidak didefinisikan implementasi **fungsi** dengan harapan bahwa obyek turunan hewan yang spesifik, seperti **Kucing** dan **Ayam** akan menyediakan implementasinya yang spesifik dengan kondisi masing-masing hewan. Pada contoh diatas, **fungsi berjalan()** dan **bersuara()** dalam kelas Binatang diimplementasikan oleh kelas **Ayam** dan **Kucing**.