POLIMORFISME (POLYMORPHISM)

DOSEN : SULISTYOWATI, ST., M.KOM.

- Polymorphism berasal dari bahasa Yunani yang berarti banyak bentuk

- Konsep ini memungkinkan memungkinkan digunakannya suatu interface yang sama untuk memerintah obyek agar melakukan aksi atau tindakan yang mungkin secara prinsip sama namun secara proses berbeda

- Dalam konsep yang lebih umum sering kali polymorphism disebut dalam istilah satu interface banyak aksi

Contoh yang konkrit dalam dunia nyata yaitu mobil.

- ✓Mobil yang ada dipasaran terdiri atas berbagai tipe dan berbagai merk, namun semuanya memiliki interface kemudi yang sama, seperti: stir, tongkat transmisi, pedal gas dan rem
- ✓Jika seseorang dapat mengemudikan satu jenis mobil saja dari satu merk tertentu, maka orang itu akan dapat mengemudikan hampir semua jenis mobil yang ada, karena semua mobil tersebut menggunakan interface yang sama
- ✓Harus diperhatikan disini bahwa interface yang sama tidak berarti cara kerjanya juga sama
- ✓Misal pedal gas pedal gas, jika ditekan maka kecepatan mobil akan meningkat. Tapi bagaimana proses peningkatan kecepatan ini dapat berbeda-beda untuk setiap jenis mobil

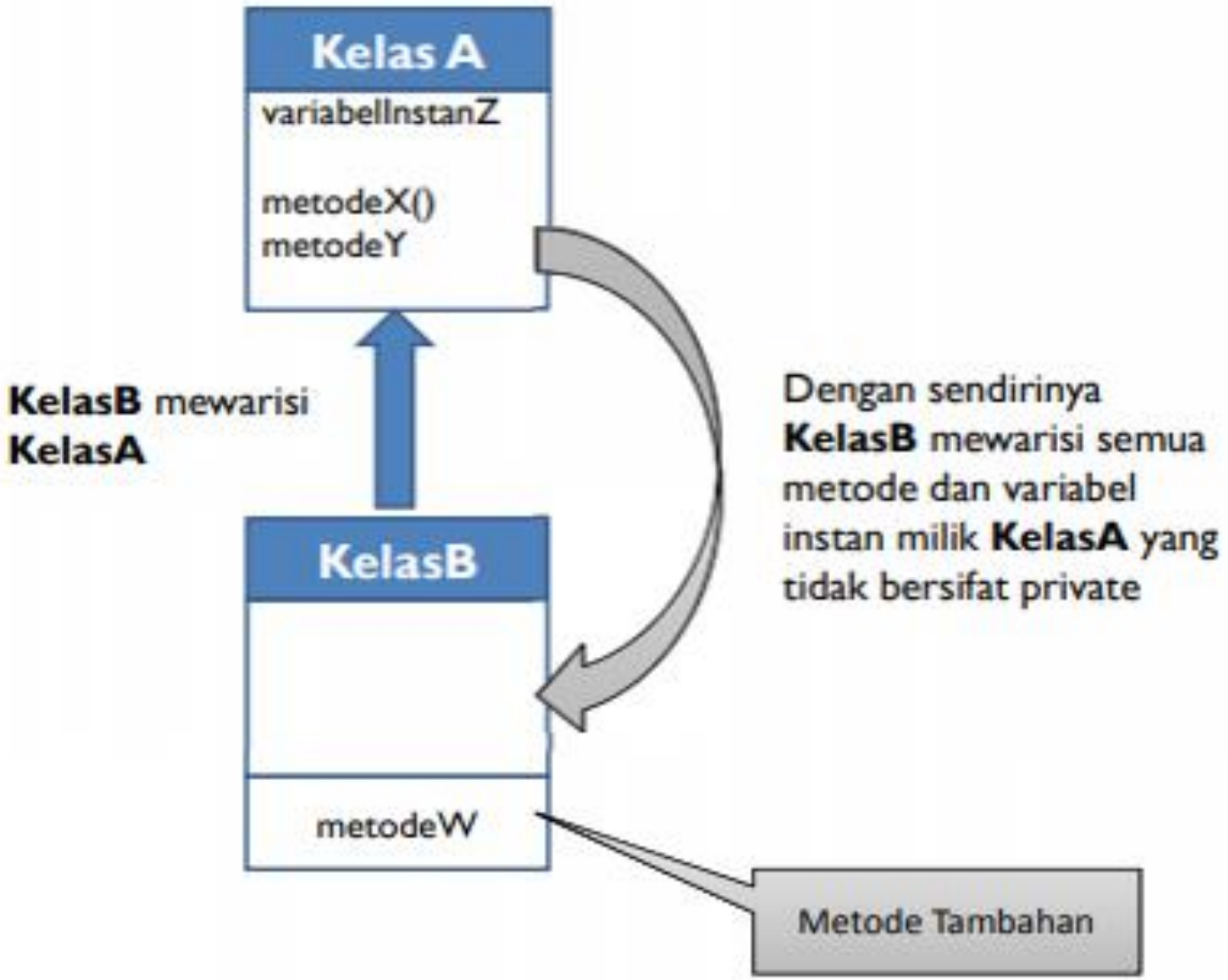
Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi obyek-obyek apa yang dapat melakukan pemecahan pemecahan masalah masalah tersebut.

- Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya.

- Misal manager tersebut ingin memperoleh data dari bagian administrasi, maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya

- Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui obyek petugas administrasi

- Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar obyek-obyek yang ada karena setiap obyek memiliki deskripsi tugasnya sendiri


Kelas A
variabelInstanZ
metodeX()
metodeY
KelasB mewarisi KelasA
Dengan sendirinya KelasB mewarisi semua metode dan variabel instan milik KelasA yang tidak bersifat private
KelasB
metodeW
Metode Tambahan

Virtual Method Invocation (VMI)

- Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan Overriding
- Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap Overriding method pada subclass, dimana yang seharusnya dipanggil adalah overridden method

Parent.java

```
class Parent
{
    int x = 5;
    public void Info()
    {
            System.out.println ("Ini class Parent");
    }
}
```

Child.java

```
class Child extends Parent
{
    int x = 10;
    public void Info()
    {
        System.out.println ("Ini class Child");
    }
}
```

Test.java

```
public class Test
{
    public static void main(String args[])
    {
        Parent tes = new Child();
        System.out.println ("Nilai x = " + tes.x);
        tes.Info()
     }
}
```

Hasil dari running program :

Nilai x = 10

Ini class Child

Polymorphic arguments

- Polymorphic arguments adalah tipe suatu parameter yang menerima suatu nilai yang bertipe subclass-nya
- Berikut contoh dari polymorphics arguments:

Pegawai.java

```
class Pegawai
{
    .....
}
```

Manajer.java

```
class Manajer extends Pegawai
{
    .....
}
```

Tes.java

```
public class Tes
{
   public static void Proses(Pegawai peg)
   {
        .....
   }
   public static void main(String args[])
   {
      Manajer man = new Manajer();
      Proses(man);
   }
}
```

Pernyataan instanceof

- Pernyataan instanceof sangat berguna untuk mengetahui tipe asal dari suatu polymorphic arguments
- Untuk lebih jelasnya, misalnya dari contoh program sebelumnya, kita sedikit membuat modifikasi pada class Tes dan ditambah sebuah class baru Kurir, seperti yang tampak dibawah ini :

```
class Kurir extends Pegawai
{
    .....
}
```
Kurir.java

```
public class Tes
{
   public static void Proses(Pegawai peg)
   {
        if (peg instanceof Manajer)
        {
            ... lakukan tugas-tugas manajer ...
        }
       else if (peg instanceof Kurir)
       {
            ... lakukan tugas-tugas kurir ...
       }
            else
            {
                ... lakukan tugas-tugas lainnya...
             }
    }
   public static void main(String args[])
   {
       Manajer man = new Manajer();
       Kurir kur = new Kurir();
       Proses(man);
       Proses(kur);
  }
}
```
Tes.java

Overloading

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

a. Parameter titik, untuk menggambar titik Gambar(int t1)
b. Parameter titik, untuk menggambar garis Gambar(int t1,int t2)
c. Parameter titik, untuk menggambar segitiga Gambar(int t1,int t2,int t3)
d. Parameter titik, untuk menggambar persegi empat Gambar(int t1,int t2,int t3,int t4)

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

    a. Nama method harus sama

    b. Daftar parameter harus berbeda

    c. Return type boleh sama, juga boleh berbeda

Overloading Construktor

- Kita dapat menyembunyikan information dari suatu class sehingga anggota-anggota class tersebut tidak dapat diakses dari luar
- Adapun caranya adalah cukup dengan memberikan akses kontrol private ketika mendeklarasikan suatu atribut atau method.

Contoh : private int nrp;

```
public class Siswa
{
    private int nrp;
    public void setNrp(int n)
    {
        nrp=n;
    }
}
```

- Konstruktor (konstruktor) adalah suatu method yang pertama kali dijalankan pada saat pembuatan suatu obyek.
- Konstruktor mempunyai ciri yaitu:
  - Mempunyai nama yang sama dengan nama class
  - Tidak mempunyai return type (seperti void, int, double, dan lainlain)

Suatu class dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama

```
public class Siswa
{
    private int nrp;
    private String nama;
    public Siswa(String m)
    {
        nrp=0;
        nama="";
    }
    public Siswa(int n, String m)
    {
        nrp=n;
        nama=m;
    }
}
```

Overriding

- Overriding adalah suatu keadaan dimana method pada subclass menolak method pada parent class-nya.
- Overriding mempunyai ciri-ciri sebagai berikut :

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

Berikut ini contoh terjadinya Overriding dimana method Info() pada class Child meng-override method Info() pada class parent:

Parent.java

```
class Parent
{
    public void Info()
    {
            System.out.println ("Ini class Parent");
    }
}
```

Child.java

```
class Child extends Parent
{
    Public void Info()
    {
            System.out.println ("Ini class Child");
    }
}
```

Method yang terkena override (overridden method) diharuskan tidak boleh mempunyai modifier yang lebih luas aksesnya dari method yang meng-override override (Overriding Overriding method)