

# Pemrograman I

# Literatur

- H.M Deitel, P.J Deitel, **Small Java How to Program**-sixth Edition, Pearson Prentice Hall, 2005
- Elliot B. Koffman, Paul A.T. Wolfgang, **Objects, Abstraction, Data Structures and Design Using Java**, John Wiley & Sons.Inc, 2005
- Ian F. Darwin, **Java Cookbook**, O'Reilly, 2001
- Mark Allen Weiss, **Data Structures & Algorithm Analysis in Java**, Addison-Wesley, 1999
- Moh.Sjukani, **Algoritma & Struktur Data dengan C, C++ dan Java**, Mitra Wacana Media, Agustus 2005
- Rangsang Purnama, **Tuntunan Pemrograman Java jilid- 1**, Prestasi Pustaka Publisher, Januari 2003
- Rangsang Purnama, **Tuntunan Pemrograman Java jilid- 2**, Prestasi Pustaka Publisher, Juli 2003
- Rangsang Purnama, **Tuntunan Pemrograman Java jilid 3**, Prestasi Pustaka Publisher, Maret 2003
- Ariesto Hadi Sutopo, Fajar Masya, **Pemrograman Berorientasi Objek dengan Java**, Graha Ilmu, 2005
- Indrajani, Martin, **Pemrograman Berorientasi Objek dengan Java**, Elex Media Komputindo, 2004
- Melvin Antonius, Damian Bayu Imam Santoso, Carneles, **Membuat Animasi dengan Java**, Elex Media Komputindo, 2004

# Materi **\*praktikum**

1. **Pengertian Java**
2. **Setup /Instalasi Java**
3. **Version Control (Git)**
4. **Anatomi aplikasi Java**
5. **Classpath**
6. **Variabel dan Tipe Data**
7. **Operator**
8. **Control Flow (If.... Else, For/While)**
9. **Class & Object**
10. **Method**
11. **Exception**
12. **Konsep OOP**
13. **Inheritance**
14. **Encapsulation**
15. **Polymorphism**
16. **Abstract Class & Interface**
17. **Composition & Aggregation**
18. **Studi Kasus & Presentasi**

# **INHERITANCE & ENCAPSULATION**

# Inheritance

- Inheritance biasa juga disebut pewarisan
- Inheritance atau pewarisan adalah proses penciptaan kelas baru dengan mewarisi karakteristik kelas yang telah ada, ditambah karakteristik unik kelas baru
- Inheritance atau pewarisan adalah mekanisme yang memungkinkan kelas mewarisi fungsionalitas kelas yang ada
- Untuk menciptakan kelas baru, hanya perlu menspesifikasikan cara kelas itu berbeda dari kelas yang telah ada
- Dengan pewarisan dimungkinkan menciptakan klasifikasi berhirarki

# Inheritance

- Dengan pewarisan, dapat menciptakan class umum yang mendefinisikan perilaku umum dari item-item yang saling berhubungan
- Class yang diwarisi disebut superclass sedangkan kelas yang mewarisi disebut subclass
- Subclass mewarisi semua metode dan variabel superclass
- Superclass secara otomatis memberi perilakunya ke subclass pewaris
- Subclass merupakan gabungan seluruh fitur dari superclass-superclass di hirarki ditambah fitur kepunyaan sendiri

# Inheritance

- Contoh :
  - Kucing.Besar → Superclass
    - Singa → Subclass
    - Harimau → Subclass
- Struktur pewarisan dalam konteks pemrograman berorientasi yang memungkinkan kita mewariskan data/atribut serta metoda/fungsi yang dimiliki oleh kelas induk (superkelas) ke kelas-kelas dibawah (subkelas) yang terkait menurut hierarki pewarisan

# Contoh 01 : menggunakan abstract

- Super Kelas : “Kucing Besar”
- Mempunyai 2 Sub Kelas, yaitu :
  - Sub Kelas : “Singa”
  - Sub Kelas : “Harimau”
- Kelas “Singa” dan “Harimau” mewarisi data/atribut serta metoda/fungsi yang dimiliki oleh Kelas “Kucing Besar” yang dispesifikasi visibilitynya sebagai **protected**
- Kata kunci **abstract**, berarti memuat definisi-definisi data/atribut serta metoda/fungsi, sedangkan implementasinya masing-masing akan didefinisikan di kelas-kelas turunannya; implementasi metode/fungsi akan dilakukan di SubKelas



# Contoh 01 : menggunakan abstract

- Kata kunci **protected** memungkinkan kita mewariskan baik data/atribut nama serta metoda/fungsi makan() dari superkelas kucing besar ke kelas-kelas dibawahnya (singa, Harimau), sehingga kelas-kelas dibawahnya tidak memerlukan definisi untuk data/atribut nama maupun metoda/fungsi makan()
- Metode makan() didefinisikan sebagai abstract sedangkan untuk implementasinya menggunakan kelas singa dan harimau
- Misal : **class** Singa **extends** KucingBesar
  - Berguna untuk memberitahu interpreter Java bahwa kelas Singa merupakan turunan dari kelas KucingBesar

# Contoh 01 : class KucingBesar

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;

public abstract class KucingBesar
{
    protected static String nama;
    protected abstract void Makan();
}
```

# Contoh 01 : class Singa

```
//Perintah Inheritance
//Kelas turunannya
package Latihan03.Praktikum04.bin;

class Singa extends KucingBesar
{
    //Konstruktor untuk class Singa
    public Singa(String namaSinga)
    {
        //Mengisi pengubah yang diwarisi oleh kelas abstract
        Singa.nama = namaSinga;
    }
    //implementasi metode yang diwarisi dari kelas abstract
    public void Makan()
    {
        System.out.println("Singa makan daging");
        System.out.println();
    }
}
```

# Contoh 01 : class Harimau

```
//Perintah Inheritance
//Kelas turunannya
package Latihan03.Praktikum04.bin;

class Harimau extends KucingBesar
{
    //Konstruktor untuk class Harimau
    public Harimau(String namaHarimau)
    {
        //Mengisi pengubah yang diwarisi oleh kelas abstract
        Harimau.nama = namaHarimau;
    }
    //implementasi metode yang diwarisi dari kelas abstract
    public void Makan()
    {
        System.out.println("Harimau makan daging dan minum susu");
        System.out.println();
    }
}
```

# Contoh 01 : class TesKucingBesar

```
//Perintah Inheritance
//Kelas turunannya
package Latihan03.Praktikum04.bin;
```

```
public class TesKucingBesar
{
    public static void main(String[] Xx)
    {
        Singa mySinga = new Singa("SIMBABA SURAI");
        System.out.println("Singa merupakan " +Singa.nama);
        mySinga.Makan();

        Harimau myHarimau = new Harimau("HARIMAU SUMATERA");
        System.out.println("Harimau merupakan " +Harimau.nama);
        myHarimau.Makan();
    }
}
```

```
E:\Data Sep2010\06 Kuliah Gasal 1314\04 Pe
i01>java -cp bin Latihan03.Praktikum04.bin
Singa merupakan SIMBABA SURAI
Singa makan daging
Harimau merupakan HARIMAU SUMATERA
Harimau makan daging dan minum susu
```

# Contoh 02 : menggunakan inputan

- Super Kelas : “Kucing”
- Mempunyai 2 Sub Kelas, yaitu :
  - Sub Kelas : “KucingMakan”
  - Sub Kelas : “JmlAnakKucing”
- Kelas “KucingMakan” dan “JmlAnakKucing” mewarisi data/atribut serta metoda/fungsi yang dimiliki oleh Kelas “Kucing” yang dispesifikasi visibilitynya sebagai **protected**
- Menggunakan 2 jenis inputan, yaitu : bilangan bulat dan string
- Diimplementasikan dengan menggunakan 2 buah metoda/fungsi yang berbeda, yaitu : metoda/fungsi **inputDataInteger()**, **inputDataString()** dan **Tulis()**

# Contoh 02 : menggunakan inputan

- Kelas **Kucing** mewariskan semua data/attribut yang dimilikinya (Nama, Umur, Berat)
- Memiliki setter/setting untuk masing-masing data/ atribut (setNama, setUmur, setBerat) yang mengakses ke dalam kelas **Kucing**
- Kelas KucingMakan memiliki atribut Makanan
- Kelas JmlAnakKucing memiliki atribut Makanan dan Jumlah Anak
- Kelas KucingMakan dan JmlAnakKucing memiliki atribut yang sama yaitu Makanan
- Tetapi ke dua kelas (KucingMakan, JmlAnakKucing) memiliki visibility bersifat **private**
- Untuk ke dua kelas (KucingMakan, JmlAnakKucing) menempati lokasi yang berbeda di memori komputer

# Contoh 02 : class Kucing

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Kucing
{
    protected String NAMA;
    protected int UMUR;
    protected int BERAT;
    public void setName(String Nama)
    {
        NAMA = Nama;
    }
    public void setUmur(int Umur)
    {
        UMUR = Umur;
    }
}
```



# Contoh 02 : class Kucing

```
public void setBerat(int Berat)
{
    BERAT = Berat;
}
public void Tulis()
{
    System.out.println("Nama Kucing Kesayangan saya : " +this.NAMA);
    System.out.println("Umur Kucing Kesayangan saya : " +this.UMUR);
    System.out.println("Berat Kucing Kesayangan saya : " +this.BERAT);
}
protected static int inputDataInteger()
{
    BufferedReader BFR = new BufferedReader(new InputStreamReader (System.in))
    String angkaInput = null;
    try
    {
        angkaInput = BFR.readLine();
    }
}
```

# Contoh 02 : class Kucing

```
catch(IOException e)
{
    e.printStackTrace();
}
int Data = Integer.valueOf(angkaInput).intValue();
return Data;
}
protected static String inputDataString()
{
    final BufferedReader BFR = new BufferedReader(new InputStreamReader (System
String Input = null;
try
{
    Input = BFR.readLine();
}
catch (final IOException e)
{
    e.printStackTrace();
}
    final String Data = String.valueOf(Input);
    return Data;
}
```

## Contoh 02 : class KucingMakan

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;

class KucingMakan extends Kucing
{
    private String MAKANAN;
    public String getMakanan()
    {
        return MAKANAN;
    }
    public void setMakanan(String Makanan)
    {
        MAKANAN = Makanan;
    }
}
```

## Contoh 02 : c JmlAnakKucin

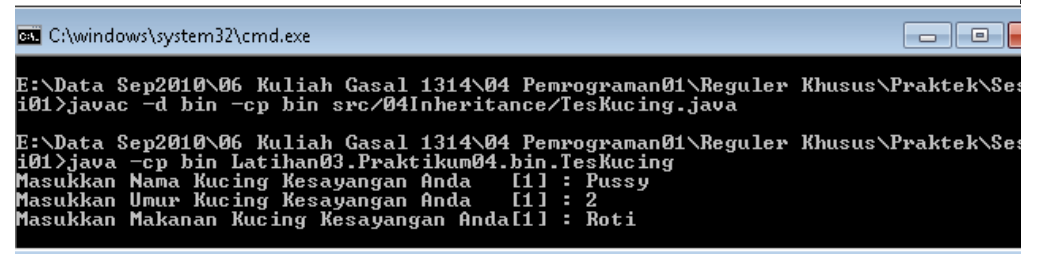
```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;
class JmlAnakKucing extends Kucing {
    private String MAKANAN;
    private int JumlahAnak;
    public String getMakanan() {
        return MAKANAN;
    }
    public void setMakanan(String Makanan)
    {
        MAKANAN = Makanan;
    }
    public int getJumlahAnak()
    {
        return JumlahAnak;
    }
    public void setJumlahAnak(int JumlahAnak)
    {
        this.JumlahAnak = JumlahAnak;
    }
}
```

# Contoh 02 : class TesKucing

```
//Perintah Inheritance
//Kelas turunannya
package Latihan03.Praktikum04.bin;

public class TesKucing
{
    private static int Jumlah;
    public static void main(String[] Xx)
    {
        Jumlah = 1;

        //Kucing Makan
        KucingMakan KM = new KucingMakan();
        System.out.print("Masukkan Nama Kucing Kesayangan Anda    [" + Jumlah + "] : ");
        KM.setNama(KucingMakan.inputDataString());
        System.out.print("Masukkan Umur Kucing Kesayangan Anda    [" + Jumlah + "] : ");
        KM.setUmur(KucingMakan.inputDataInteger());
        System.out.print("Masukkan Makanan Kucing Kesayangan Anda[" + Jumlah + "] : ");
        KM.setMakanan(KucingMakan.inputDataString());
        Jumlah++;
    }
}
```



```
C:\windows\system32\cmd.exe

E:\Data Sep2010\06 Kuliah Gasal 1314\04 Pemrograman01\Reguler Khusus\Praktek\Ses
i01>javac -d bin -cp bin src\04Inheritance\TesKucing.java

E:\Data Sep2010\06 Kuliah Gasal 1314\04 Pemrograman01\Reguler Khusus\Praktek\Ses
i01>java -cp bin Latihan03.Praktikum04.bin.TesKucing
Masukkan Nama Kucing Kesayangan Anda    [1] : Pussy
Masukkan Umur Kucing Kesayangan Anda    [1] : 2
Masukkan Makanan Kucing Kesayangan Anda[1] : Roti
```

# Contoh 02 : class TesKucing

```
//JmlAnakKucing
JmlAnakKucing JAK = new JmlAnakKucing();
System.out.print("Masukkan Nama Kucing Kesayangan Anda    [" +Jumlah +"] : ");
JAK.setNama(KucingMakan.inputDataString());
System.out.print("Masukkan Umur Kucing Kesayangan Anda    [" +Jumlah +"] : ");
JAK.setUmur(KucingMakan.inputDataInteger());
System.out.print("Masukkan Berat Kucing Kesayangan Anda    [" +Jumlah +"] : ");
JAK.setBerat(KucingMakan.inputDataInteger());
System.out.print("Masukkan Makanan Kucing Kesayangan Anda[" +Jumlah +"] : ");
JAK.setMakanan(KucingMakan.inputDataString());
System.out.print("Masukkan Jumlah Anak Kucing Anda        [" +Jumlah +"] : ");
JAK.setJumlahAnak(JmlAnakKucing.inputDataInteger());

//Menampilkan Data Kucing
System.out.println("\n\n DATA KUCING        ");
KM.Tulis();
System.out.println("Makanan Kucing Anda : " +KM.getMakanan());
JAK.Tulis();
System.out.println("Makanan Kucing Anda : " +JAK.getMakanan());
System.out.println("Jumlah Anak Kucing : " +JAK.getJumlahAnak());
}
```

# Contoh 03 : menggunakan superclass

- Kelas **Sukulnd** mewariskan semua data/attribut yang dimilikinya (Nama, Alamat, Hobby, Pekerjaan)
- SubKelas Sunda, Sumba, Batak, Jawa menggunakan super untuk mengambil super kelas
- Kelas Jawa menggunakan metode overloading dengan parameter berupa pecahan (metode1) dan parameter berupa bilangan bulat (metode2)

# Contoh 03 : class SukuInd

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;
public class SukuInd
{
    protected String NAMA;
    protected String ALAMAT;
    protected String HOBBY;
    protected String PEKERJAAN;

    public SukuInd(String Nama, String Alamat, String Hobby, String Pekerjaan)
    {
        super();
        NAMA      = Nama;
        ALAMAT    = Alamat;
        HOBBY     = Hobby;
        PEKERJAAN = Pekerjaan;
    }
    public String getNAMA()    {
        return NAMA;
    }
    public String getALAMAT() {
        return ALAMAT;
    }
    public String getHOBBY()  {
        return HOBBY;
    }
    public String getPEKERJAAN() {
        return PEKERJAAN;
    }
}
```



# Contoh 03 : class Sunda

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;

public class Sunda extends SukuInd
{
    private double GAJI;
    public Sunda(String Nama, String Alamat, String Hobby, String Pekerjaan)
    {
        super(Nama, Alamat, Hobby, Pekerjaan);
    }
    public double getGAJI()
    {
        return GAJI;
    }
    public void setGAJI(double Gaji)
    {
        GAJI = Gaji;
    }
}
```

# Contoh 03 : class Sumba

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;

public class Sumba extends SukuInd
{
    private double GAJI;
    public Sumba(String Nama, String Alamat, String Hobby, String Pekerjaan)
    {
        super(Nama, Alamat, Hobby, Pekerjaan);
    }
    public double getGAJI()
    {
        return GAJI;
    }
    public void setGAJI(double Gaji)
    {
        GAJI = Gaji;
    }
}
```

# Contoh 03 : class Batak

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;

public class Batak extends SukuInd
{
    private double GAJI;
    public Batak(String Nama, String Alamat, String Hobby, String Pekerjaan)
    {
        super(Nama, Alamat, Hobby, Pekerjaan);
    }
    public double getGAJI()
    {
        return GAJI;
    }
    public void setGAJI(double Gaji)
    {
        GAJI = Gaji;
    }
}
```

# Contoh 03 : class Jawa

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;
public class Jawa extends SukuInd {
    private double GAJI;
    public Jawa(String Nama, String Alamat, String Hobby, String Pekerjaan) {
        super(Nama, Alamat, Hobby, Pekerjaan);
    }
    public double getGAJI() {
        return GAJI;
    }
    public void setGAJI(double Gaji) {
        GAJI = Gaji;
    }
    //Metode Overloading-Metode yang apabila parameternya berupa pecahan (versi 1/Pecahan)
    public String THR(double persen) {
        if(persen > 500000)
            return "Gaji ke 13 Terlalu Besar !!!!";
        else
            return "Gaji ke 13 sebesar " +persen*GAJI;
    }
    //Metode yang apabila parameternya berupa bil Bulat (versi 2/BilBul)
    public String THR(long persen) {
        if(persen > 500000)
            return "Gaji ke 13 Terlalu Besar !!!!";
        else
            return "Gaji ke 13 sebesar " +(persen*GAJI/100);
    }
}
```

# Contoh 03 : class TesSuku

```
//Perintah Inheritance -- Kelas turunannya
package Latihan03.Praktikum04.bin;
public class TesSuku {
    public static void main(String[] Xx) {
        SukuInd SI = new SukuInd("Tiara", "Jakarta", "Nonton", "Direktur");
        System.out.println("Nama      : " +SI.getNAMA());
        System.out.println("Alamat    : " +SI.getALAMAT());
        System.out.println("Hobby     : " +SI.getHOBBY());
        System.out.println("Pekerjaan : " +SI.getPEKERJAAN());
        System.out.println();
        System.out.println();
        Sunda SD = new Sunda("Sekar Arum", "Bogor", "Menyanyi", "Manager");
        System.out.println("Nama      : " +SD.getNAMA());
        System.out.println("Alamat    : " +SD.getALAMAT());
        System.out.println("Hobby     : " +SD.getHOBBY());
        System.out.println("Pekerjaan : " +SD.getPEKERJAAN());
        System.out.println();
        System.out.println();
        Batak BT = new Batak("Cinthya Sinaga", "Medan", "Menari", "Marketing");
        System.out.println("Nama      : " +BT.getNAMA());
        System.out.println("Alamat    : " +BT.getALAMAT());
        System.out.println("Hobby     : " +BT.getHOBBY());
        System.out.println("Pekerjaan : " +BT.getPEKERJAAN());
        System.out.println();
        System.out.println();
    }
}
```

# Contoh 03 : class TesSuku

```
Sumba SB = new Sumba("Fanny", "Depok", "Baca", "Operator");
System.out.println("Nama      : " +SB.getNAMA());
System.out.println("Alamat    : " +SB.getALAMAT());
System.out.println("Hobby     : " +SB.getHOBBY());
System.out.println("Pekerjaan : " +SB.getPEKERJAAN());
System.out.println("Gaji      : Rp.  " +SB.getGAJI());
System.out.println();
System.out.println();
//Bilangan Bulat
Jawa JWa = new Jawa("Shakila", "Jepara", "Melukis", "Sekretaris");
System.out.println("Nama      : " +JWa.getNAMA());
System.out.println("Alamat    : " +JWa.getALAMAT());
System.out.println("Hobby     : " +JWa.getHOBBY());
System.out.println("Pekerjaan : " +JWa.getPEKERJAAN());
System.out.println("Gaji      : Rp.  " +JWa.getGAJI());
System.out.println("Keterangan: " +JWa.THR(30000));
System.out.println();
System.out.println();
//Bilangan Pecahan
Jawa JWb = new Jawa("Diana", "Trenggalek", "Membatik", "HRD");
System.out.println("Nama      : " +JWb.getNAMA());
System.out.println("Alamat    : " +JWb.getALAMAT());
System.out.println("Hobby     : " +JWb.getHOBBY());
System.out.println("Pekerjaan : " +JWb.getPEKERJAAN());
System.out.println("Gaji      : Rp.  " +JWb.getGAJI());
System.out.println("Keterangan: " +JWb.THR(0.75));
System.out.println();
System.out.println();
}
}
```

## Contoh 04: menggunakan Attribut dan Method

- Setiap penduduk yang telah bekerja pasti mendapatkan upah/pendapatan
- Pendapatan yang diperoleh dikenakan pajak berdasarkan aturan pajak
- Aturan pajaknya adalah sebagai berikut :
  - $\leq 50.000.000 \rightarrow$  pajak = 5%
  - 50.000.001 s.d. 250.000.000  $\rightarrow$  pajak = 15%
- Harus mengetahui di propinsi mana penduduk tersebut tinggal

# Contoh 04: menggunakan Attribut dan Method

- **Analisa :**
  - Kelas **Pajak** mempunyai attribut **Pendapatan, Propinsi**
  - Kelas **Pajak** mempunyai turunan kelas **PajakProp** dan **PajakApply**
  - Kelas **PajakProp** memiliki semua attribut dari kelas **Pajak** yang menurunkannya
  - Kelas **PajakProp** untuk membuat objek, menentukan nilai variabel dari kelas **Pajak** dan memanggil metode yang terdapat hitung Pajak serta mencetak hasil pada layar
  - Kelas **PajakProp** mewarisi metode dari kelas **Pajak**, yaitu **HitungPajak** dan memiliki metode **HitungPjkProp**
  - Metode **HitungPjkProp** digunakan untuk menghitung pajak yang harus dibayar berdasarkan pajak umum dengan pengurangan **Rp. 50.000,-**



# Attribut dan Metode Kelas Pajak

## Attribut

Penghasilan  
Propinsi

## Keterangan

Penghasilan per tahun  
Lokasi tempat tinggal

## Metode

HitungPajak

## Keterangan

Menghitung Pajak

# Attribut dan Metode Kelas PajakProp

## Attribut

-

## Keterangan

-

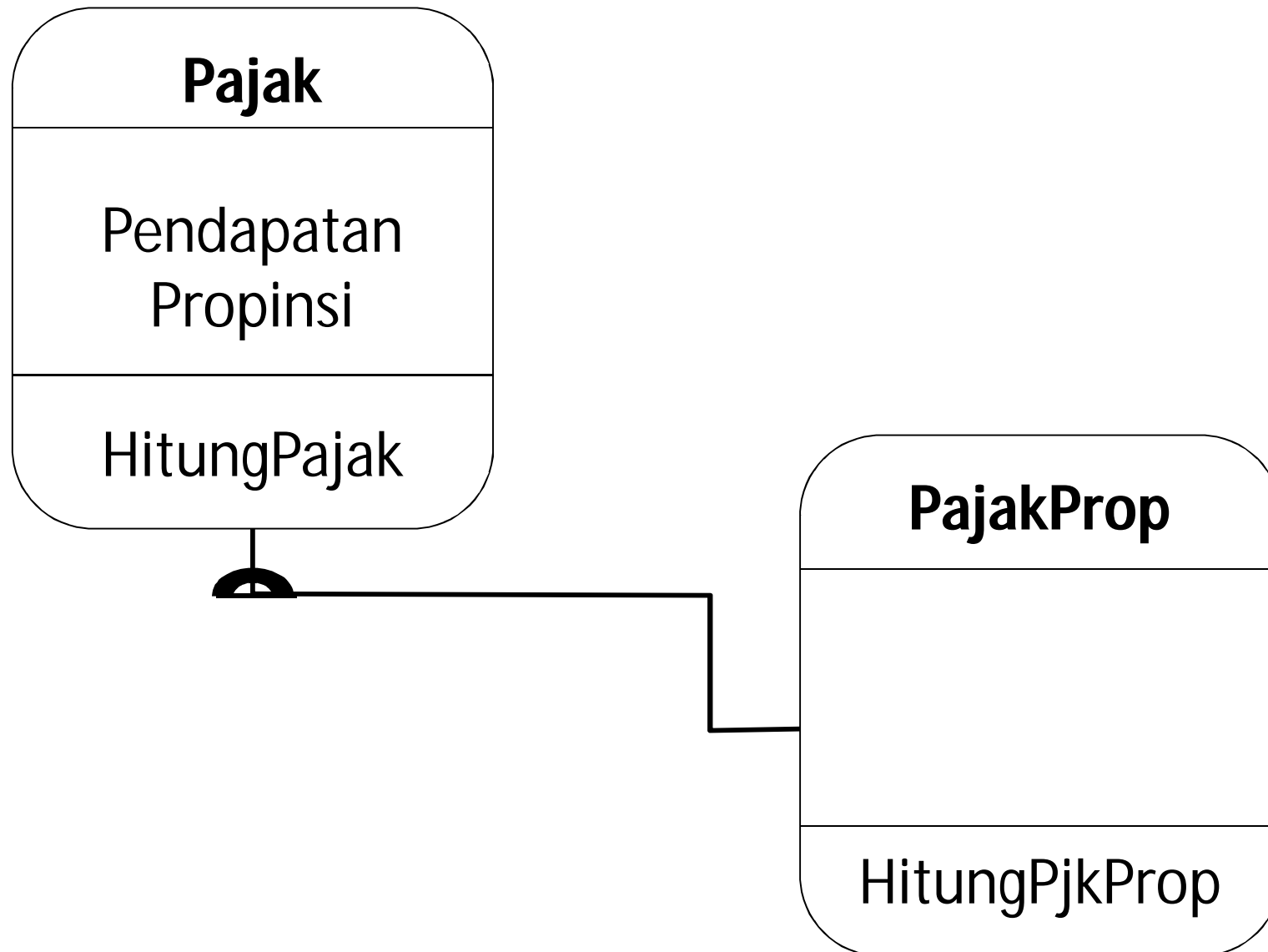
## Metode

HitungPjkProp

## Keterangan

Menghitung Pajak  
yang berlokasi di prop.

# Attribut dan Metode Kelas Pajak & PajakProp



```
package Latihan03.Praktikum04.bin;

class Pajak
{
    long Pendapatan;
    String Propinsi;
    String Kota;
    public double HitungPajak()
    {
        double PajakUmum=0;
        if(Pendapatan <5000000001)
        {
            PajakUmum = Pendapatan * 0.025;
        }
        else
        {
            PajakUmum = Pendapatan * 0.25;
        }
        return PajakUmum;
    }
}
```

## Contoh 04: class Pajak

# Contoh 04 : class PajakProp

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;

class PajakProp extends Pajak
{
    public double HitungPjkProp(double PajakUmum)
    {
        double PajakProp = PajakUmum = 75000;
        return PajakProp;
    }
}
```

# Contoh 04 : class PajakApply

```
package Latihan03.Praktikum04.bin;
class PajakApply
{
    public static void main(String[] Xx)
    {
        Pajak PJK = new Pajak(); //Membuat Obyek
        PJK.Pendapatan = 6000000; //Memberikan Nilai
        PJK.Propinsi = "Jawa Timur";
        PJK.Kota = "Mojosari-Mojokerto";
        double PajakAnda = PJK.HitungPajak(); //Menghitung Pajak

        PajakProp PJK01 = new PajakProp();
        double PajakAndaProp = PJK01.HitungPjkProp(PajakAnda);

        //Menampilkan Data Pajak
        System.out.println("Besar Pajak Anda sebesar Rp. " +PajakAnda);
        System.out.println("Anda Berada di Propinsi " +PJK.Propinsi);
        System.out.println("Anda di Kota " +PJK.Kota);
        System.out.println("Potongan Pajak Anda sebesar Rp." +PajakAndaProp);
    }
}
```

```
E:\Data Sep2010\06 Kuliah Gasal 1314\04 Pemrograman01\Reguler Khusus\Praktek\Sesi01>javac -d bin -cp bin src\04Inheritance\PajakApply.java
E:\Data Sep2010\06 Kuliah Gasal 1314\04 Pemrograman01\Reguler Khusus\Praktek\Sesi01>java -cp bin Latihan03.Praktikum04.bin.PajakApply
Besar Pajak Anda sebesar Rp. 150000.0
Anda Berada di Propinsi Jawa Timur
Anda di Kota Mojosari-Mojokerto
Potongan Pajak Anda sebesar Rp.75000.0
```

# Contoh 05: menggunakan Constructor

- Kelas Mobil menyatakan semua jenis mobil
- Kelas Mobil (super kelas) mempunyai jenis Jeep (sub kelas)
- Kelas Mobil mempunyai atribut jenis dengan tipe data String dan constructor
- Kelas Jeep mempunyai karakteristik (memiliki tiga argumen) Nama, NoPolisi, Kecepatan
- Kelas Jeep memiliki metode Display
- Pendefinisian objek dilakukan dalam kelas MobilApply
- Objek Mobil1, Mobil2 dan Mobil3 adalah anggota dari kelas Jeep dengan nilai masing-masing atribut
- Constructor dari superkelas Mobil diwariskan kepada objek Mobil1, Mobil2 dan Mobil3

# Contoh 05 : class Mobil

```
//Perintah Inheritance
package Latihan03.Praktikum04.bin;

class Mobil
{
    private String Jenis; //Merupakan variabel superkelas

    //constructor superkelas
    public Mobil(String aTipe)
    {
        Jenis = new String(aTipe);
    }
    public String JenisMobil()
    {
        return "Merupakan Jenis Mobil = " +Jenis;
    }
}
```

# Contoh 05 : class Jeep

```
class Jeep extends Mobil
{
    String Nama;           //Variabel kelas
    String NoPolisi;
    int Kecepatan;

    //constructor Kelas
    public Jeep(String aNama, String aNoPolisi, int aKecepatan)
    {
        super("Jeep");
        Nama = aNama;
        NoPolisi = aNoPolisi;
        Kecepatan = aKecepatan;
    }

    //Menampilkan Informasi
    public void Display()
    {
        System.out.println ("Nama Mobil           = " + this.Nama);
        System.out.println ("Nomor Mobil          = " + this.NoPolisi);
        System.out.println ("Kecapatan Mobil       = " + this.Kecepatan);
    }
}
```



# Contoh 05 : class MobilApply

Nomor Mobil	= B-6663-TOC
Kecapatan Mobil	= 150
Nama Mobil	= Jeep Tentara
Nomor Mobil	= B-63-ATO
Kecapatan Mobil	= 180
Nama Mobil	= Jeep Gunung
Nomor Mobil	= B-66-CAN
Kecapatan Mobil	= 200
Nama Mobil	= Jeep Pantai
Nomor Mobil	= B-6663-TOC

```
public class MobilApply
{
    public static void main (String[] args)
    {
        //Mendefinisikan objek dalam kelas
        Jeep Mobil1 = new Jeep("Jeep Tentara", "B-63-ATO", 180);
        Jeep Mobil2 = new Jeep("Jeep Gunung", "B-66-CAN", 200);
        Jeep Mobil3 = new Jeep("Jeep Pantai", "B-6663-TOC", 150);

        //Menampilkan informasi
        Mobil1.JenisMobil(); Mobil1.Display();
        System.out.println();
        Mobil2.JenisMobil(); Mobil2.Display();
        System.out.println();
        Mobil3.JenisMobil(); Mobil3.Display();
    }
}
```

# Contoh 06: dengan Perluasan Kelas

- Suatu perkuliahan yang melibatkan Dosen, Mahasiswa
  - Dosen memiliki jabatan : Asisten Ahli, Lektor, Lektor Kepala dan Guru Besar
- Gaji yang diterima oleh Dosen berdasarkan perhitungan honor tiap sks pengajarannya
- Mahasiswa memiliki Indeks Prestasi berdasarkan nilai Ujian

## Contoh 06: dengan Perluasan Kelas

- **Analisa :**
  - Kelas **Orang** mempunyai atribut Nama, Umur, Alamat, Status dan Nomor Telepon
  - Metode yang dimiliki oleh kelas **Orang** adalah setData untuk memberikan nilai variabel dan Diplay untuk menampilkan pada layar.
  - Kelas **Orang** memiliki turunan kelas **Mahasiswa** dan **Dosen**
  - Kelas **Mahasiswa** memiliki atribut dari kelas **Orang** yang menurunkannya tambahannya adalah atribut Nim dan IPK
  - Kelas **Dosen** memiliki atribut dari kelas **Orang** yang menurunkannya tambahannya adalah atribut NIDN, JJA (Jenjang Jabatan Akademik) dan Gaji
  - Kelas **Mahasiswa** dan **Dosen** mewarisi metode dari kelas **Orang**, yaitu setData, getData dan Display

# Attribut dan Metode Kelas Orang

## **Attribut**

Nama

Umur

Alamat

Status

NoTlp

## **Keterangan**

Nama Orang

Umur Orang

Alamat Orang

Status Orang

Nomor Telepon

## **Metode**

setData

getData

Display

## **Keterangan**

Setting Data

Mengambil Data

Menampilkan Informasi

## Contoh 06 : class Orang

```
class Orang    {
    //atribut
    private String Nama = "";
    private int Umur;
    private String Alamat = "";
    private String Status = "";
    private String NoTlp = "";

    //metode - constructor
    public Orang(String NM, int UMR, String ALMT, String STS, String NT)
        Nama = NM;
        Umur = UMR;
        Alamat = ALMT;
        Status = STS;
        NoTlp = NT;          }

    //Modifier
    public void setNama(String NM)      {
        this.Nama = NM;                }
    public void setUmur(int UMR)        {
        this.Umur = UMR;                }
    public void setAlamat(String ALMT)  {
        this.Alatamat = ALMT;          }
    public void setStatus(String STS)   {
        this.Status = STS;              }
```

# Contoh 06 : class Orang (cont)

```
public void setNoTlp(String NT)    {
    this.NoTlp = NT;              }

//Accessor
public String getNama()           {
    return Nama;                  }
public int getUmur()              {
    return Umur;                  }
public String getAlamat()         {
    return Alamat;                }
public String getStatus()         {
    return Status;                }
public String getNoTlp()          {
    return NoTlp;                 }
public String Display()           {
    return "Nama " +Nama+"- Umurnya " +Umur +"- Alamat " +Alamat+ "- Status " +Status +
    "- Nomer Telpannya " + NoTlp;
}
}
```

# Contoh 06 : class Dosen

```
class Dosen extends Orang
{
    //atribut untuk dosen
    private String NIDN = "";
    private String JJA = "";
    private float Honor;

    //metode constructor
    public Dosen(String NM, int UMR, String ALMT, String STS, String NT, String ND, String JA, float HN)
    {
        super(NM, UMR, ALMT, STS, NT);
        NIDN = ND;
        JJA = JA;
        Honor= HN;
    }

    //Modifier
    public void setNID(String ND)
    {
        this.NIDN = ND;
    }
    public void setJJA(String JA)
    {
        this.JJA = JA;
    }
}
```

# Contoh 06 : class Dosen

```
public void setHonor(float HN)
{
    this.Honor = HN;
}

//Accessor
public String getNIDN()
{
    return NIDN;
}
public String getJJA()
{
    return JJA;
}
public float getHonor()
{
    return Honor;
}
public String Display()
{
    return super.Display() +"NIDN " +NIDN+"- JJA " +JJA+ "- Gaji " +Honor;
}
}
```



# Contoh 06 : class Mahasiswa

```
class Mahasiswa extends Orang
{
    //atribut untuk Mahasiswa
    private String NIM = "";
    private float IPK;

    //metode constructor
    public Mahasiswa(String NM, int UMR, String ALMT, String STS, String NT, String NI, float IP)
    {
        super(NM, UMR, ALMT, STS, NT);
        NIM = NI;
        IPK= IP;
    }
}
```

# Contoh 06 : class Mahasiswa

```
//Modifier
public void setNIM(String NI)
{
    this.NIM = NI;
}
public void setIPK(float IP)
{
    this.IPK = IP;
}

//Accessor
public String getNIM()
{
    return NIM;
}
public float getIPK()
{
    return IPK;
}
public String Display()
{
    return super.Display() +"NIM " +NIM+"- IPK " +IPK;
}
}
```

# Contoh 06 : class KuliahApply

```
public class KuliahApply
{
    public static void main(String[] args)
    {
        Dosen DOS = new Dosen("Candra Nursari", 45, "Depok", "Menikah", "007", "03", "Lektor Kapala", 1500000);
        Mahasiswa MH = new Mahasiswa("Nadya Risti", 20, "Jakarta", "Mahasiswa", "451010", "2010", 3);

        System.out.println(DOS.Display());
        System.out.println(MH.Display());

    }
}
```