MODUL

PEMROGRAMAN BERORIENTASI OBJEK

Disusun oleh :

PULUT SURYATI

SUMIYATUN

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER

AKAKOM

YOGYAKARTA

2019

MODUL 7
PEWARISAN

CAPAIANPEMBELAJARAN

1. Dapat mendefinisikan kelas turunan, Dapat membuat Objek turunan, Dapat mengakses atribut turunan
2. Dapat menerapkan konstruktor pada sub kelas dalam pewarisan.

KEBUTUHAN ALAT/BAHAN/SOFTWARE

Alat : Komputer, Viewer
Software : Java, TextPad/Netbean

DASAR TEORI

Pendahuluan

Pewarisan, merupakan salah satu bentuk reuse software dalam kelas baru yang dibuat dapat memanggil member dari kelas dan menambahkan member baru atau memodifikasi kemampuannya. Deng Pewarisan, hal ini dapat meningkatkan kemungkinan implementasi dan pemeliharaan sistem secara efektif.

Kelas yang ada disebut **superclass** sedangkan kelas baru disebut **subclass. (**bahasa pemrograman C++ untuk superclass sebagai **base class** dan subclass sebagai **derived class**). Setiap subclass dapat menjadi superclass untuk subclass yang akan datang. Subclass dapat menambahkan atribut/field dan metode sendiri. Oleh karena itu, subclass lebih spesifik dan representasikan kelompok object lebih spesialis dibandingkan superclass. Behavior (Operasi/Metode) dari superclass dapat dimodifikasi oleh subclass, sehingga mempunyai operasi yang tepat, hal ini menggapa inheritan kadang-kadang di refer sebagai *specialization*.

Superclass langsung merupakan superclass dari subclass yang secara explisit diturunkan. Sedangkan superclass tidak langsung merupakan kelas diatas superclass lansung didalam hirarki class.

Pendefinisian hubungan pewarisan diantara jelas, dalam java hirarki kelas dimulai dengan kelas **Object** (dalam package java.lang), selanjutnya untuk perwarisan dideklarasikan dengan keyword **extends.** Java mendukung konsep **single inheritance,** yaitu setiap kelas diturunkan dari tepat satu (1) superclass lansung.

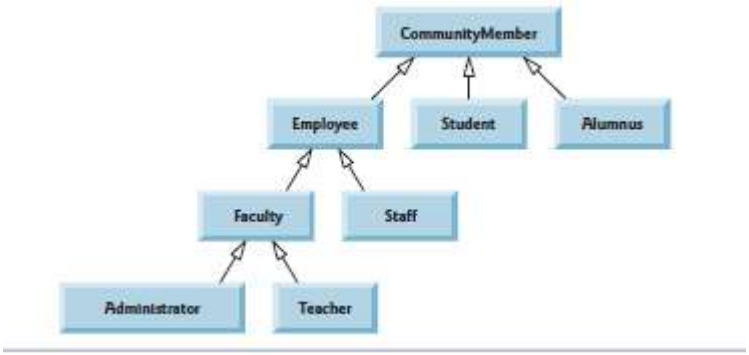
Kelas Object, yang didefinisikan pada paket java.lang, mendefiniskan dan mengimplementasi perilaku umum dari semua kelas. dalam platform Java, banyak kelas diturunkan langsung dari Object, kelas lain turunan dari kelas lainnya, dan seterusnya, sehingga membentuk hirarki kelasseperti berikut.

Contoh Superclass dan Subclass

Superclass	**Subclass**
Student	GraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle, Sphere, Cube
Loan	CarLoan, HomeImprovementLoan, MortgageLoan
Employee	Faculty, Staff
BankAccount	CheckingAccount, SavingsAccount

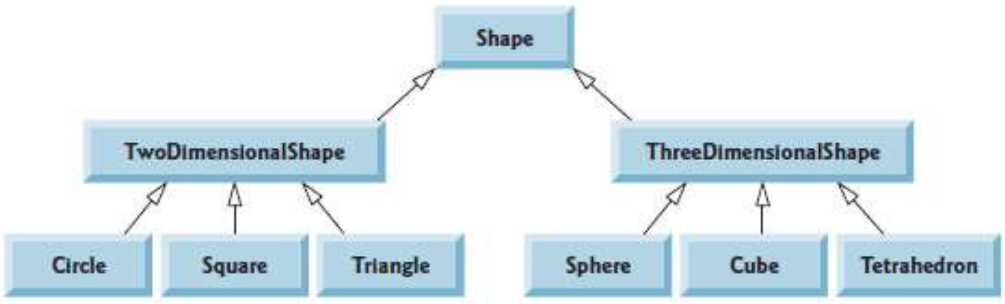
Contoh

hirarki kelas turunan untuk Sivitas Akademik Perguruan Tinggi



Gambar 1. Hirarki pewarisan dari Sivitas akademik Perguruan Tinggi

Hiraki kelas Shape



Gambar 2. Hirarki kelas Shape

PRAKTIK

1. Mendefinisikan Superclass

   Membuat Kelas Pegawai

```
public class Pegawai {
    private String nama;
    private String jabatan;
    private int gaji;
    public Pegawai(String nama, String jabatan){
       this.setNama(nama);
       this.setJabatan(jabatan);
    }
    public String getJabatan() {
       return jabatan;
    }
    public void setJabatan(String jabatan) {
       this.jabatan = jabatan;
    }
    public long getGaji() {
       return gaji;
    }
    public void setGaji(int gaji) {
       this.gaji= gaji;
    }
    public String getNama() {
       return nama;
    }
    public void setNama(String nama) {
       this.nama = nama;
    }
    public void cetakPegawai(){
    System.out.println("");
    System.out.println("Nama Pegawai:"+this.nama);
    System.out.println("Jabatan:"+this.jabatan);
    }
}
```

Membuat kelas baru dengan nama Kelas Dosen turunan dari kelas Pegawai

```
public class Dosen extends Pegawai {
    private String mtkDiampu;
    public Dosen(String nama,String jabatan,String mtkDiampu){
       this(nama, Jabatan);
       this.setmtkDiampu(mtkDiampu);
    }
    public String getMtkDiampu(){
       return this.mtkDiampu;
    }
    public void setMtkDiampu(){
       this.mtkDiampu=mtkDiampu;
    }
```

2. Membuat Objek turunan,
   a. Membuat objek dari kelas Pegawai dan Kelas Dosen

```
public class TestPewarisan1{
public static void main(String[] args) {
  //Membuat objek Pegawai => superclass
  Pegawai dataPeg=new Pegawai("Budi", "Dosen");
  dataPeg.cetakPegawai();
  //Membuat objek Dosen => subclass
   Dosen dataDosen=new Dosen(("Budi", "Dosen","PBO");
   dataDosen.cetakPegawai();
}
}
```

   b. Kelas Pegawai otomatis merupakan turunan dari kelas Object dari package java.lang Memanggil member dari kelas Object

```
public class TestPewarisan2{
public static void main(String[] args) {
  //Membuat objek Pegawai => superclass
  Pegawai dataPeg=new Pegawai("Budi", "Dosen");
  dataPeg.cetakPegawai();
  dataPeg.toString();
  //Membuat objek Dosen => subclass
   Dosen dataDosen=new Dosen(("Budi", "Dosen","PBO");
   dataDosen.cetakPegawai();
   dataDosen.toString();
}
}
```

3. Penerapan konstruktor pada sub kelas dalam pewarisan

```
public class Pegawai {
    private String nama;
    private String jabatan;
    private int gaji;
```

```
    public Pegawai(){
       this.setNama("Agus");
       this.setJabatan("Teknisi");
    }
    public Pegawai(int gaji){
       this();
       this.setGaji(gaji);
    }

    public Pegawai(String nama, String jabatan){
       this.setNama(nama);
       this.setJabatan(jabatan);
    }
    public String getJabatan() {
       return jabatan;
    }
    public void setJabatan(String jabatan) {
       this.jabatan = jabatan;
    }
    public long getGaji() {
       return gaji;
    }
    public void setGaji(int gaji) {
       this.gaji= gaji;
    }
    public String getNama() {
       return nama;
    }
    public void setNama(String nama) {
       this.nama = nama;
    }
    public void cetakPegawai(){
    System.out.println("");
    System.out.println("Nama Pegawai:"+this.nama);
    System.out.println("Jabatan:"+this.jabatan);
    }
}
```

Konstruktor dalam subclass

```
public class Dosen extends Pegawai {
    private String mtkDiampu;
  public Dosen(int gaji){
       super();
       this.setmtkDiampu(gaji);
    }

  public Dosen(String nama,String jabatan,String mtkDiampu){
       this(nama, Jabatan);
       this.setmtkDiampu(mtkDiampu);
    }
    public String getMtkDiampu(){
       return this.mtkDiampu;
    }
    public void setMtkDiampu(){
       this.mtkDiampu=mtkDiampu;
```

```
    }
```

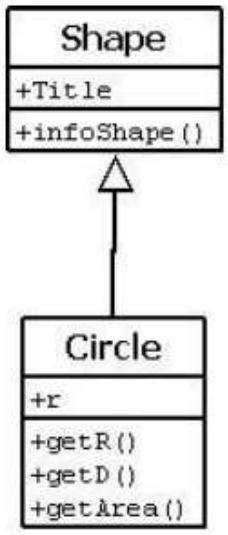
Pemanggilan object subclass dengan alternatif konstruktor

```
public class TestPewarisan3{
public static void main(String[] args) {
  //Membuat objek Pegawai => superclass
  Pegawai dataPeg1=new Pegawai();
  dataPeg1.toString();
  Pegawai dataPeg2=new Pegawai(3000000);
  dataPeg2.toString();
  Pegawai dataPeg3=new Pegawai("Budi", "Dosen");
  dataPeg3.toString();

  //Membuat objek Dosen => subclass
   Dosen dataDosen1=new Dosen(4000000);
   dataDosen.toString();
   Dosen dataDosen1=new Dosen(("Budi", "Dosen","PBO");
   dataDosen.toString();
}
}
```

LATIHAN

1. Gambarkan dengan Kelas Diagram hubungan antara kelas Pegawai dengan Kelas Dosen
2. Implementasikan dalam program Gambar kelas Diagram dibawah ini, lengkapi dengan konstruktor, metode asesor, metode mutator dan berikan pengaturan hak akses pada atribut dan metode yang dimiliki



TUGAS

1. Buatlah overloading kontruktor, pada kelas Circle yang mengakses kontruktor milik kelas Shape dan konstruktor lain dalam kelas Circle
2. Buatlah objek dari kelas Shape dan Circle

REFERENSI

1. Dietel P., Dietel H., 2015, *Java How to Program Tenth edition*, Deitel & Associates, Inc., Prentice Hall, New Jersey.