Pertemuan 12 :

Implementasi Polimorfisme dengan Interface

Tessy Badriyah, SKom. MT.

http://lecturer.eepis-its.edu/~tessy

Topik

- Konsep Dasar Polimorfisme
- Interfaces
- Interface References
- Interface Inheritance
- Operator instanceof

Konsep Dasar Polimorfisme

- Polimorfisme artinya penyamaran dimana suatu bentuk dapat memiliki lebih dari satu bentuk
- Dalam pemrograman berbasis obyek, polimorfisme adalah kemampuan untuk mempunyai beberapa bentuk class yang berbeda.
- Contoh polimorfisme yang dibahas disini adalah implementasi konsep dasar polimorfisme dengan pembuatan interface

Definisi Interface

- Interface merupakan unit pemrograman yang berisi deklarasi method dan konstanta yang diperlukan
- Deskripsi dari desain awal hanya memiliki konstanta dan method tanpa implementasi
- Dengan interface dimungkinkan suatu implementasi multiple inheritance

Deklarasi Interface

- Deklarasi Interface

```
intfModifier interface namaInterface {
 varModifier1 tipedata1 varName1 = value1;
 varModifier2 type2 varName2 = value2;
 varModifier3 tipedata3 varName3 = value3;
  ........
 varModifierN tipedataN varNameN = valueN;
 mthModifier1 tipeReturn1 namaMethod1(params1);
 mthModifier2 tipeReturn2 namaMethod2(params2);
 ...........
 mthModifierN tipeReturnN namaMethodN(paramsN);
}
```

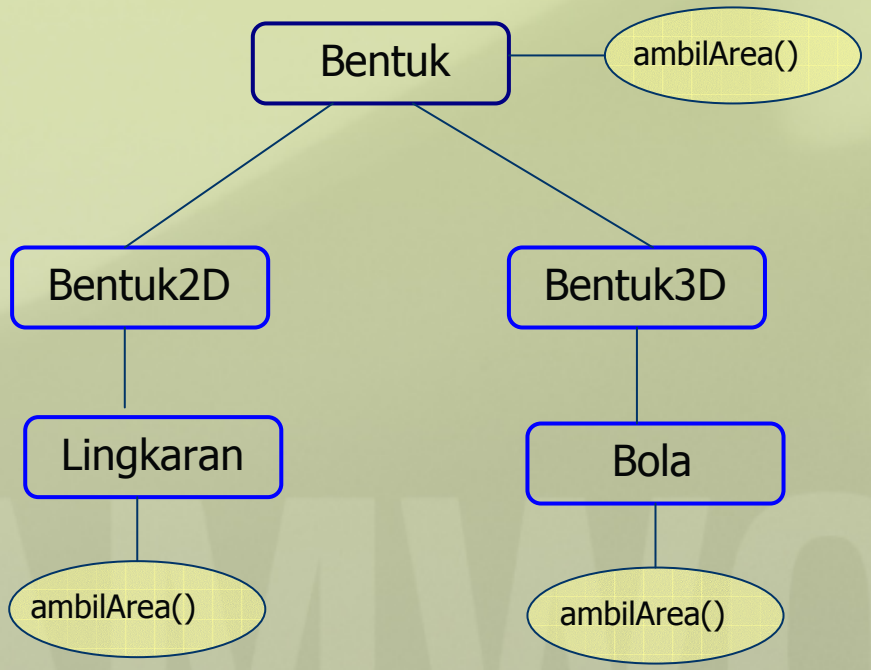
Implementasi Interface

- Implementasi Interface

```
clsModifiers class namaClass extends namaSuper
   implements intfList
{
   // implementation
   ..........
}
```

Contoh penggunaan interface

- Desain awal dibuat interface dari bentuk dua dimensi (Bentuk2D) dan bentuk tiga dimensi (Bentuk3D) dari class titik tiga dimensi (Titik3D)



Contoh Interface Sederhana (Percobaan 1)

```
interface Bahan {
  String perunggu = "perunggu";
  String emas = "emas";
  String marmer = "marmer";
  String perak = "perak";
  String kayu = "kayu";
}

abstract class BahanObject {
  String bahan;
}

class Bola extends BahanObject {
  Bola(String bahan) {
    this.bahan = bahan;
  }
}

class Koin extends BahanObject {
  Koin(String bahan) {
    this.bahan = bahan;
  }
}
```

```
class Cincin extends BahanObject {
  Cincin(String bahan) {
    this.bahan = bahan;
  }
}

class BahanObjects {
  public static void main(String args[]) {
    Bola bola = new Bola(Bahan.kayu);
    Koin koin = new Koin(Bahan.perak);
    Cincin cincin = new Cincin(Bahan.emas);
    System.out.println(bola.bahan);
    System.out.println(koin.bahan);
    System.out.println(cincin.bahan);
  }
}
```

Hasil output:

```
kayu
perak
emas
```

Mereferensi Variabel Interface (Percobaan 2)

Mereferensi Variabel Interface

intfRef.varName

intfRef.mthName(args)

```
interface A {
  void tampil(String s);
}

class C1 implements A {
  public void tampil(String s) {
    System.out.println("C1: " + s);
  }
}

class C2 implements A {
  public void tampil(String s) {
    System.out.println("C2: " + s);
  }
}

class C3 implements A {
  public void tampil(String s) {
    System.out.println("C3: " + s);
  }
}
```

```
class InterfaceReferenceVariable {
public static void main(String args[]) {
  A a;
  a = new C1();
  a.tampil("String 1");
  a = new C2();
  a.tampil("String 2");
  a = new C3();
  a.tampil("String 3");
 }
}
```

Hasil output:

C1: String 1

C2: String 2

C3: String 3

Interface Inheritance (Percobaan 3)

Inheritance pada Interface

```
intfModifier interface intfname extends intfList {
  // interface body
}
```

```
interface J {
  int j = 200;
  int j1();
}

interface K {
  double k1();
}

interface L extends J, K {
  boolean l1();
}

class I implements L {
  public int j1() {
    return 4;
  }
  public double k1() {
    return 6.8;
  }
  public boolean l1() {
    return true;
  }
}
```

```
class InterfaceInheritance {

  public static void main(String args[]) {
    I i = new I();
    System.out.println(i.j);
    System.out.println(i.j1());
    System.out.println(i.k1());
    System.out.println(i.l1());
  }
}
```

Hasil output :

```
200
4
6.8
true
```

Interface Inheritance

- Semua method pada interface secara implicit bersifat public
- Suatu interface dan interface yang mewarisinya (interface inheritance) harus memiliki kesesuaian return type untuk method dengan nama yang sama

Percobaan 4

- Program berikut ini akan error jika di-compile karena pada interface dan interface yang mewarisi suatu interface (interface inheritance) terdapat ketidaksesuaian return type pada method dengan nama yang sama

```
interface L1 {
  void f();
  void g();
}

interface L2 extends L1 {
  void f();
  int g();
}

class CompileError {
  public static void main(String args[]) {
    // ini tidak akan bisa ditampilkan
    // karena program akan error jika di-compile
    System.out.println(“Error jika di-compile");
  }
```

Operator instanceof

Operator instanceof

namaVar instanceof tipe

- Operator instanceof digunakan untuk memeriksa apakah beberapa obyek merupakan instance dari suatu class yang spesifik
- Contoh : Class Ikan terdiri dari IkanAirTawar dan IkanAirAsin. Yang termasuk golongan IkanAirTawar adalah ikan Tombro. Sedangkan yang termasuk golongan IkanAirAsin adalah Salmon dan Tuna.

Percobaan 5

- Pada percobaan 5 ini akan diimplementasikan Class Ikan yang terdiri dari subClass IkanAirTawar dan IkanAirAsin. Yang termasuk ikan air asin adalah Tuna dan Salmon. Dan Tombro untuk ikan air tawar. Program akan menampilkan ikan yang termasuk ikan air asin menggunakan instanceof

Percobaan 5

InstanceOfClass.java *

```
abstract class Ikan {
  abstract void tampil();
}

abstract class IkanAirTawar extends Ikan {
}

abstract class IkanAirAsin extends Ikan {
}

class Tombro extends IkanAirTawar {
  void tampil() {
    System.out.println("Tombro");
  }
}

class Salmon extends IkanAirAsin {
  void tampil() {
    System.out.println("Salmon");
  }
}
```

```
C:\j2sdk1.4.1_01\bin\java.exe  -classpath "C:
Salmon
Tuna
Finished executing
```

InstanceOfClass.java *

```
class Tuna extends IkanAirAsin {
  void tampil() {
    System.out.println("Tuna");
  }
}

class InstanceofClass {
  public final static int NUMIKAN = 4;

  public static void main(String args[]) {

    // array Ikan
    Ikan ikan[] = new Ikan[NUMIKAN];

    // Pembuatan Object
    ikan[0] = new Tombro();
    ikan[1] = new Salmon();
    ikan[2] = new Tuna();
    ikan[3] = new Tombro();

    // penggunaan instanceof
    for (int i = 0; i < NUMIKAN; i++) {
      Ikan ikanku = ikan[i];
      if (ikanku instanceof IkanAirAsin) {
        ikanku.tampil();

      }
    }
  }
}
```

Percobaan 6

- Pada percobaan 6 ini akan diimplementasikan class Mamalia yang terdiri dari Beruang, Gajah, Kuda, dan Singa. Masing-masing obyek mamalia mengimplementasikan interface Kendaraan, dimana hanya obyek Gajah dan Kuda saja yang memiliki method tunggangan karena obyek tersebut dapat ditunggangi.

Percobaan 6

InstanceofInterface.java

```
interface Kendaraan {
  void tunggangan ();
}

abstract class Mamalia {
}

class Beruang extends Mamalia {
}

class Gajah extends Mamalia implements Kendaraan {
  public void tunggangan() {
    System.out.println("Gajah dapat ditunggangi");
  }
}

class Kuda extends Mamalia implements Kendaraan {
  public void tunggangan() {
    System.out.println("Kuda dapat ditunggangi");
  }
}

class Singa extends Mamalia {
}
```

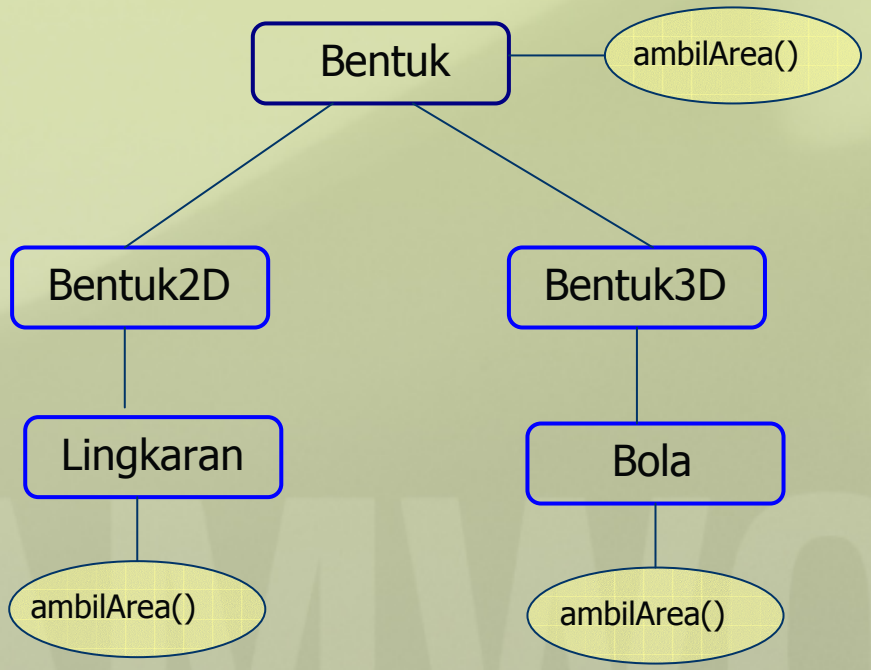
InstanceofInterface.java

```
class InstanceofInterface {
  public final static int NUMMAMALIA = 4;

  public static void main(String args[]) {

    // Array mamalia
    Mamalia mamalia[] = new Mamalia[NUMMAMALIA];

    // Obyek
    mamalia[0] = new Beruang();
    mamalia[1] = new Gajah();
    mamalia[2] = new Kuda();
    mamalia[3] = new Singa();

    // Penggunaan instanceof
    for (int i = 0; i < NUMMAMALIA; i++) {
      if (mamalia[i] instanceof Kendaraan) {
        Kendaraan v = (Kendaraan)mamalia[i];
        v.tunggangan();
      }
    }
  }
}
```

```
C:\j2sdk1.4.1_01\bin\java.exe  -classpath "C:
Gajah dapat ditunggangi
Kuda dapat ditunggangi
Finished executing
```

Percobaan 7

- Pada percobaan ini akan diimplementasikan hirarki seperti berikut :



Percobaan 7

Titik3D.java

```
interface Bentuk2D {
  double getLuas();
}

interface Bentuk3D {
  double getVolume();
}

class Titik3D {
  double x, y, z;

  Titik3D(double x, double y, double z) {
    this.x = x;
    this.y = y;
    this.z = z;
  }
}
abstract class Bentuk {
  abstract void tampil();
}
```

Titik3D.java

```
class Lingkaran extends Bentuk
implements Bentuk2D {
  Titik3D pusat, p; // p titik pd Lingkaran

  Lingkaran(Titik3D pusat, Titik3D p) {
    this.pusat = pusat;
    this.p = p;
  }

  public void tampil() {
    System.out.println("Lingkaran");
  }

  public double getLuas() {
    double dx = pusat.x - p.x;
    double dy = pusat.y - p.y;
    double d = dx * dx + dy * dy;
    double radius = Math.sqrt(d);
    return Math.PI * radius * radius;
  }
}
```

Percobaan 7

Titik3D.java *

```
class Bola extends Bentuk
implements Bentuk3D {
  Titik3D pusat;
  double radius;

  Bola(Titik3D pusat, double radius) {
    this.pusat = pusat;
    this.radius = radius;
  }

  public void tampil() {
    System.out.println("Bola");
  }

  public double getVolume() {
    return 4 * Math.PI * radius * radius * radius / 3;
  }
}

class Bentuk3 {

  public static void main(String args[]) {

    Lingkaran c = new Lingkaran(new Titik3D(0, 0, 0), new
      Titik3D(1, 0, 0));
    c.tampil();
    System.out.println(c.getLuas());
    Bola s = new Bola(new Titik3D(0, 0, 0), 1);
    s.tampil();
    System.out.println(s.getVolume());
  }
}
```

```
C:\j2sdk1.4.1_01\bin\
Lingkaran
3.141592653589793
Bola
4.1887902047863905
Finished executing
```

TUGAS

Tugas

1. Terdapat Class Binatang yang terdiri dari subClass BinatangLucu dan BinatangSeram. Yang termasuk binatang lucu adalah Kucing dan Anjing. Dan Singa untuk binatang seram. Program akan menampilkan binatang yang termasuk binatang lucu menggunakan instanceof (lihat percobaan 5)
2. Akan diimplementasikan class MakhlukHidup yang terdiri dari Manusia, Binatang dan Tumbuhan. Masing-masing obyek mamalia mengimplementasikan interface Properti, dimana hanya obyek Manusia saja yang memiliki method punya_akal karena hanya obyek manusia satu-satunya makhluk hidup yang memiliki akal. (lihat percobaan 6)
3. Lihat percobaan 7 untuk mengimplementasikan Bentuk2D berupa segiempat dan Bentuk3D berupa kotak, dimana rumus luas segiempat = panjang * lebar, sedangkan volume kotak = panjang * lebar * tinggi

selesai

5 Mei 2007