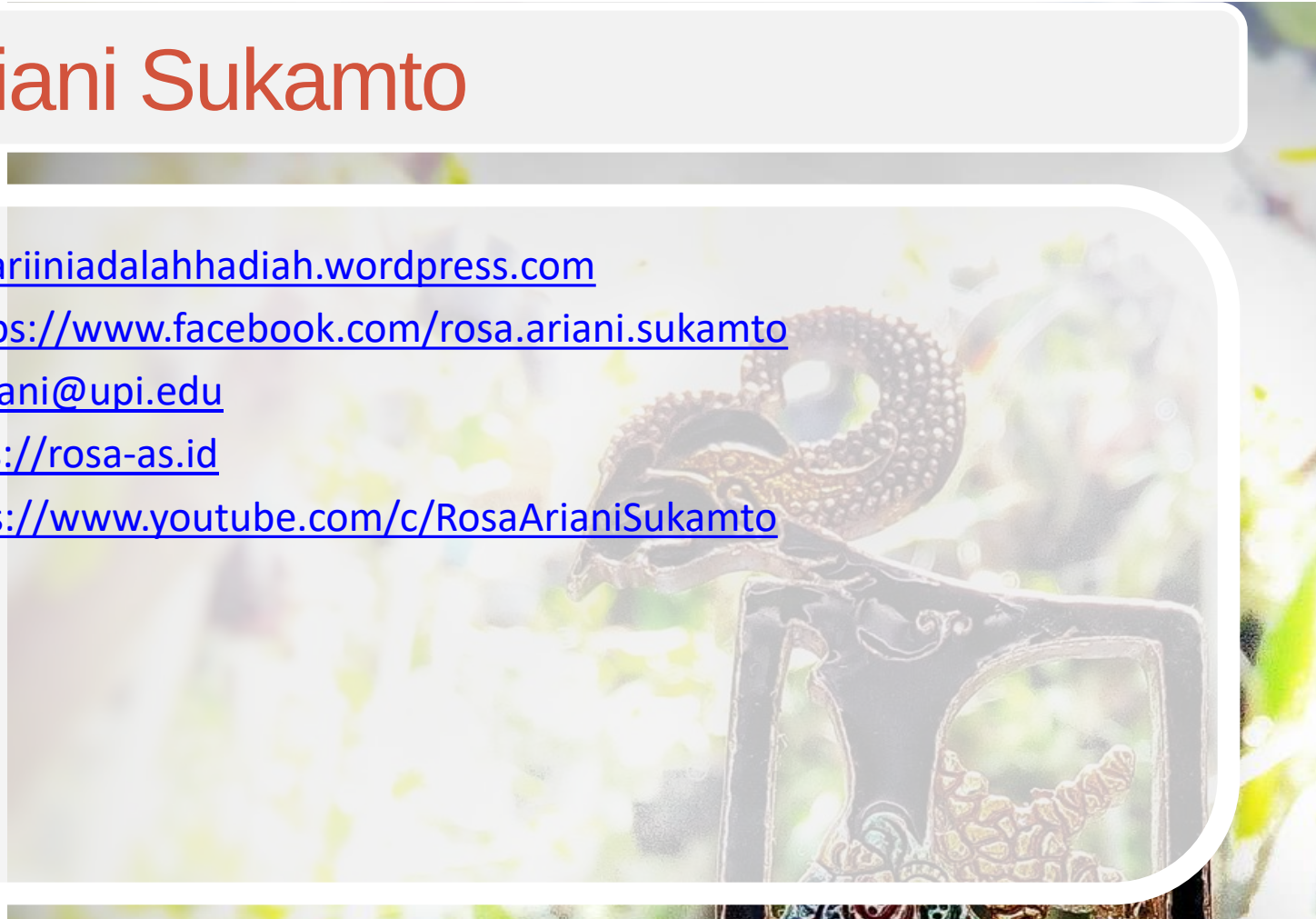
DESAIN DAN PEMROGRAMAN BERORIENTASI OBJEK

Interface dan Kelas Abstrak

Rosa A. S.

Rosa Ariani Sukamto



- Blog: http://hariiniadalahhadiah.wordpress.com
- Facebook: https://www.facebook.com/rosa.ariani.sukamto
- Email: rosa.ariani@upi.edu
- Website: https://rosa-as.id
- Youtube: https://www.youtube.com/c/RosaArianiSukamto

Interface

- Bahasa pemrograman C++ dan Python tidak mengenal *interface* karena mendukung *multiple inheritance*
- Digunakan untuk mengelompokkan
- Tidak memiliki atribut
- Hanya deklarasi method yang wajib diimplementasikan di kelas anak (kelas yang mengimplementasikan)
- Didukung PHP5 keatas dan Java

Implementasi Interface - PHP (1)

```php
<?php

interface BangunDatar{

    function luas($s1, $s2);

    function keliling($s1, $s2);
}

?>
```

```php
<?php

class Persegi implements BangunDatar{

    function __construct(){
    }

    function luas($s1, $s2){
        return ($s1 * $s2);
    }

    function keliling($s1, $s2){
        return (2 * ($s1 + $s2));
    }

    function __destruct(){
    }
}
?>
```

Implementasi Interface - PHP (2)

```php
<?php

class SegitigaSiku implements
    BangunDatar{
    function __construct(){
    }

    function luas($s1, $s2){
        return (0.5 * $s1 * $s2);
    }

    function keliling($s1, $s2){
        return ($s1 + $s2
            + sqrt($s1, $s2));
    }

    function __destruct(){
    }
}
?>
```

```php
<?php


  $opersegi = new Persegi();
  $osegitiga = new SegitigaSiku();

  echo $opersegi->luas(5, 8)."<br/>";
 echo $osegitiga->luas(5, 8)."<br/>";



?>
```

Implementasi Interface - Java (1)

```
interface BangunDatar{

    public double luas(int s1, int s2);

    public double keliling(int s1, int
    s2);


}
```

```
class Persegi implements BangunDatar{

    Persegi(){
    }

    public double luas(int s1, int s2){
        return (s1 * s2);
    }

    public double keliling(int s1, int
     s2){
        return (2 * (s1 + s2));
    }

}
```

Implementasi Interface - Java (2)

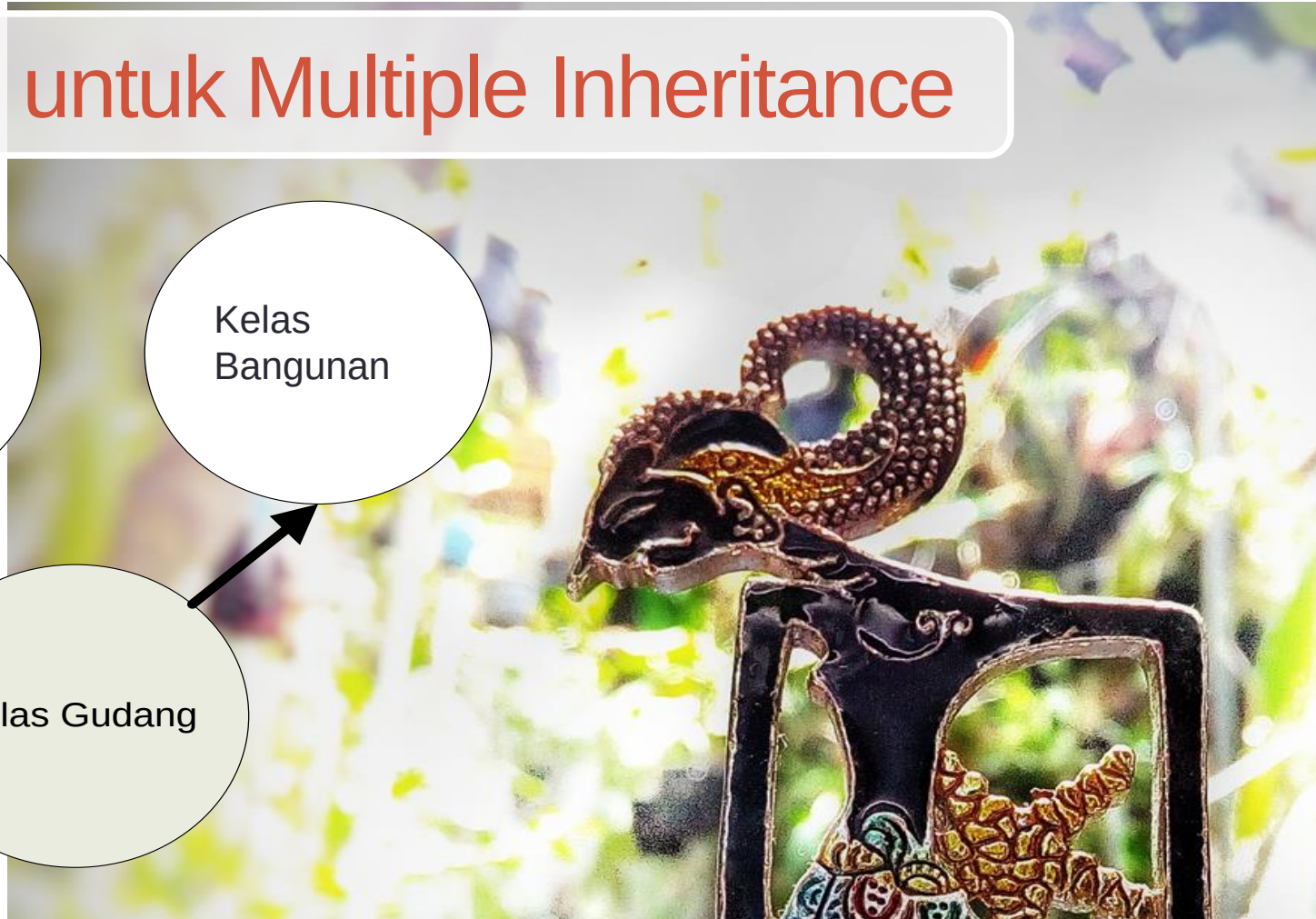
```
class SegitigaSiku implements
    BangunDatar{

    SegitigaSiku(){
    }

    public double luas(int s1, int s2){
        return (0.5 * s1 * s2);
    }

    public double keliling(int s1, int
    s2){
        return (s1 + s2
            + Math.sqrt((
        s1*s1) + (s2*s2)));
    }
}
```

```
class Main{

  public static void main(String[]
    args){
      Persegi opersegi;
      SegitigaSiku osegitiga;

      opersegi = new Persegi();
      osegitiga = new SegitigaSiku();

      System.out.println(
        opersegi.luas(5, 8));
      System.out.println(
        osegitiga.luas(5, 8));

  }

}
```

Kenapa interface ada?

```
class Anak extends
Ortu1 implements
Ortu2, Ortu3,
Ortu4, ..., OrtuN
```

Interface untuk Multiple Inheritance



Implementasi Interface Multiple Inheritance- PHP(1)

```php
<?php
interface Aset{
      function setKodeAset($kodeAset);

      function setJenisAset(
          $jenisAset);

      function setNilaiAset(
          $nilaiAset);

      function getKodeAset();

      function getJenisAset();

      function getNilaiAset();

}

?>
```

```php
<?php
class Bangunan{
      private $kodeBangunan;
      private $pemilikBangunan;

      function __construct(){
      }
      function setKodeBangunan(
        $kodeBangunan){
        $this->kodeBangunan = $kodeBangunan;
      }
      function getKodeBangunan(){
        return $this->kodeBangunan;
      }
      //set get
      function __destruct(){
      }
}
?>
```

Implementasi interface multiple inheritance- PHP5 (2)

```php
<?php
class Gudang extends Bangunan implements
    Aset{
      var $kodeAset;
      var $jenisAset;
      var $nilaiAset;
      var $kodeGudang;
      var $namaGudang;
      var $alamatGudang;
      function __construct(){
      }
      function setKodeGudang(
        $kodeGudang){
        $this->kodeGudang = $kodeGudang;
      }
      function getKodeAset(){
        return $this->kodeAset;
      }
      //set get
      function __destruct(){
      }
}
?>
```

```php
<?php
   $obangunan = new Bangunan();
   $ogudang = new Gudang();

   $obangunan->setKodeBangunan(1);
   $obangunan->setPemilikBangunan("PT Bahagia
     Sejahtera");
    echo $obangunan->getKodeBangunan()
      ."<br/>";
echo $obangunan->getPemilikBangunan()
      ."<br/>";
    // juga untuk gudang

?>
```

Implementasi Interface Multiple Inheritance- Java (1)

```
interface Aset{


    public void setKodeAset(int
  kodeAset);

    public void setJenisAset(
        int jenisAset);

    public void setNilaiAset(
        double nilaiAset);

    public int getKodeAset();

    public int getJenisAset();

    public double getNilaiAset();


}
```

```
class Bangunan{
      private int kodeBangunan;
      private String pemilikBangunan;

      Bangunan(){
      }
      void setKodeBangunan(int
    kodeBangunan){
        this.kodeBangunan =
    kodeBangunan;
      }
      int getKodeBangunan(){
        return this.kodeBangunan;
      }

      //set get
}
```

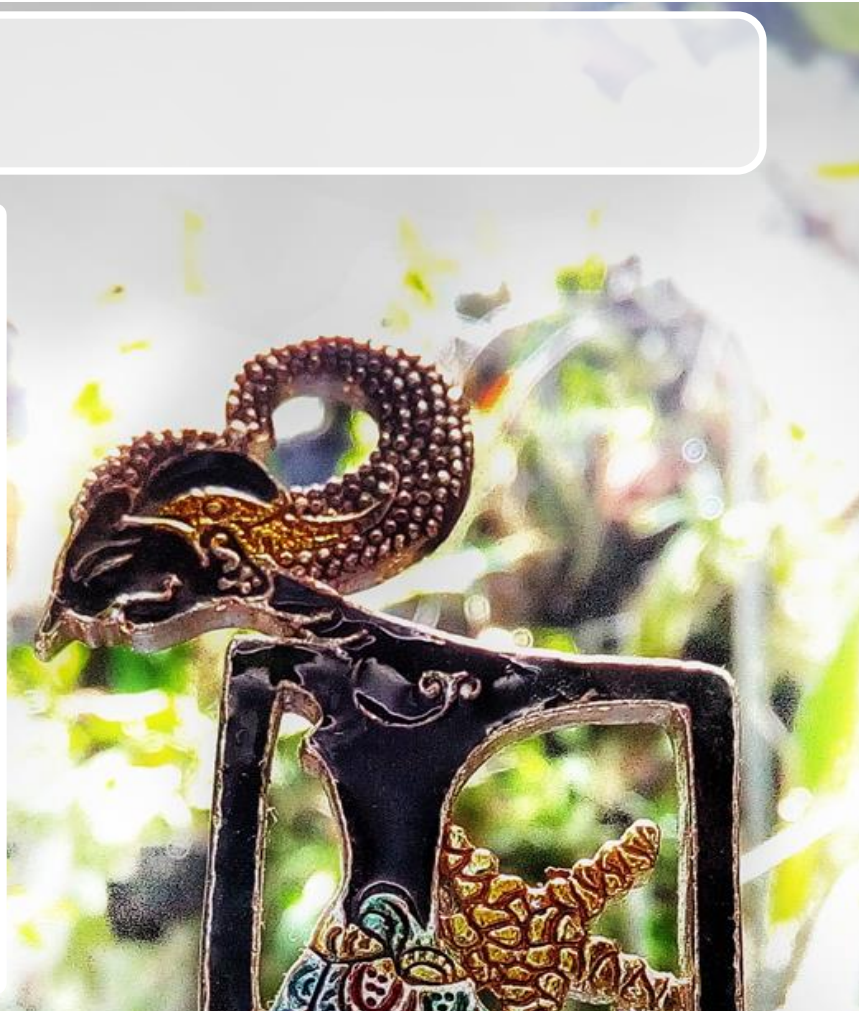
Implementasi Interface Multiple Inheritance - Java (2)

```
class Gudang extends Bangunan implements
    Aset{
     private int kodeAset;
     private int jenisAset;
     private double nilaiAset;
     private int kodeGudang;
     private String namaGudang;
     private String alamatGudang;
     Gudang(){
     }

     void setKodeGudang(int kodeGudang){
       this.kodeGudang = kodeGudang;
     }
     public int getKodeAset(){
       return this.kodeAset;
     }

     //set get

}
```

```
class Main{
 public static void main(String[] args){
  Bangunan obangunan = new Bangunan();
  Gudang ogudang = new Gudang();

  obangunan.setKodeBangunan(1);
  obangunan.setPemilikBangunan("PT Bahagia
    Sejahtera");
System.out.println(
    obangunan.getKodeBangunan ());
System.out.println(
    obangunan.getPemilikBangunan

   //lakukan juga untuk gudang
  }
}
```

Kelas Abstrak

- Hanya pada **Java** dan **Python**
- Seperti gabungan interface/multiple inheritance dan inheritance
- Kelas Abstrak dipilih jika ada metode pada kelas abstrak yang isinya hanya bisa diisi di kelas anak (metode abstrak)



Implementasi Kelas Abstrak - Java (1)

```
abstract class Titik{
/*kelas yang digunakan untuk
    mengimpelementasikan sebuah tipe titik*/
  private int x; /*koordinat x*/
  private int y; /*koordinat y*/

  Titik(){
    /*konstruktor*/
    x = 0;
    y = 0;
  }

  Titik(int xp, int yp){
    /*konstruktor*/
    x = xp;
    y = yp;
  }

  public void setX(int xp){
    /*mengeset nilai koordinat x*/
    x = xp;
  }

  public int getX(){
  /*mengembalikan nilai koordinat x*/
    return x;
  }

  public void setY(int yp){
    /*mengeset nilai koordinat y*/
    y = yp;
  }

  public int getY(){
  /*mengembalikan nilai koordinat y*/
    return y;
  }

  public abstract void printTitik();

}
```

Implementasi Kelas Abstrak - Java (2)

```
class Titik3D extends Titik{
  /*kelas turunan kelas Titik*/

  private int z; /*koordinat z*/

  Titik3D(){
    /*konstruktor*/
    z = 0;
  }

  Titik3D(int xp, int yp, int zp){

    /*konstruktor*/
    setX(xp);
    setY(yp);
    z = zp;
  }

  public void setZ(int zp){
    /*mengeset nilai koordinat z*/
    z = zp;
  }

  public int getZ(){
  /*mengembalikan nilai koordinat z*/
    return z;
  }
  public void printTitik(){
    /*menampilkan nilai koordinat titik*/
    System.out.println("nilai X : " +
    getX());
    System.out.println("nilai Y : " +
    getY());
    System.out.println("nilai Z : " +
    getZ());
  }

}
```

Implementasi Kelas Abstrak - Java (3)

```
class CobaTitik3DP{
/*metode main untuk mengetes kelas Titik dan kelas Titik3D*/

  public static void main(String[] args) {
    Titik3D t = new Titik3D(0, 0, 7);

    t.setX(28);
    t.setY(1);

    t.printTitik();

    Titik t1;
    System.out.println("==================");
    t1 = t;

    t1.printTitik();

  }
}
```

Referensi Kelas Abstrak ke Kelas Anak

Kelas Abstrak pada Python

- Menggunakan prinsip **Overriding** dimana metode abstrak dari kelas Ortu diisi dengan sebuah baris untuk mencetak sesuatu ke layar dan implementasi metode di kelas Anak adalah prosesnya.

Kapan?

- Interface
- ortu tidak punya atribut, hanya memiliki deklarasi metode
- Tidak dapat diinstansiasi
- Implements (slot bisa banyak)
- Kelas abstrak
- kelas ortu punya atribut dan memiliki deklarasi metode
- Tidak dapat diinstansiasi
- Dapat direferensikan ke anak
- extends (slot hanya 1)
- Kelas Biasa
- kelas ortu punya atribut dan tanpa deklarasi metode
- Dapat diinstansiasi
- extends (slot hanya 1)

Buatlah desain dari beberapa deskripsi berikut (menggunakan konsep inheritance atau interface) dan implementasikan desain pada bahasa pemrograman Java!

Manusia :
- no ktp
- nama
- alamat

Sifat manusia :
- kode
- nama

Kejujuran :
- kode
- nama
- reward (int)
- tahun

Kecurangan :
- kode
- nama
- punishment (int)
- tahun

AkibatDiMasaDepan :

- **tahun**
- **kode**
- **nama**
- **baik buruk(boolean)**

Gedung :

- **kode**
- **jenis**

Rumah :

- **noktppemilik**
- **alamat**
- **ukuranbangunan**
- **ukuran tanah**

Main:Merupakan proses pencarian dengan masukan berupa nama sifat manusia. Setelah sifat manusia dimasukkan maka akan keluar data dari sifat manusia, termasuk kejujuran atau kecurangan, dan akibat di masa depannya. Kelas yang diisi datanya hanya yang terkait dengan sifat manusia, kejujuran, kecurangan, dan akibat di masa depannya. Boleh menggunakan hardcode, asalkan fungsi yang diminta berjalan baik.

Contoh Soal UTS

Daftar Pustaka