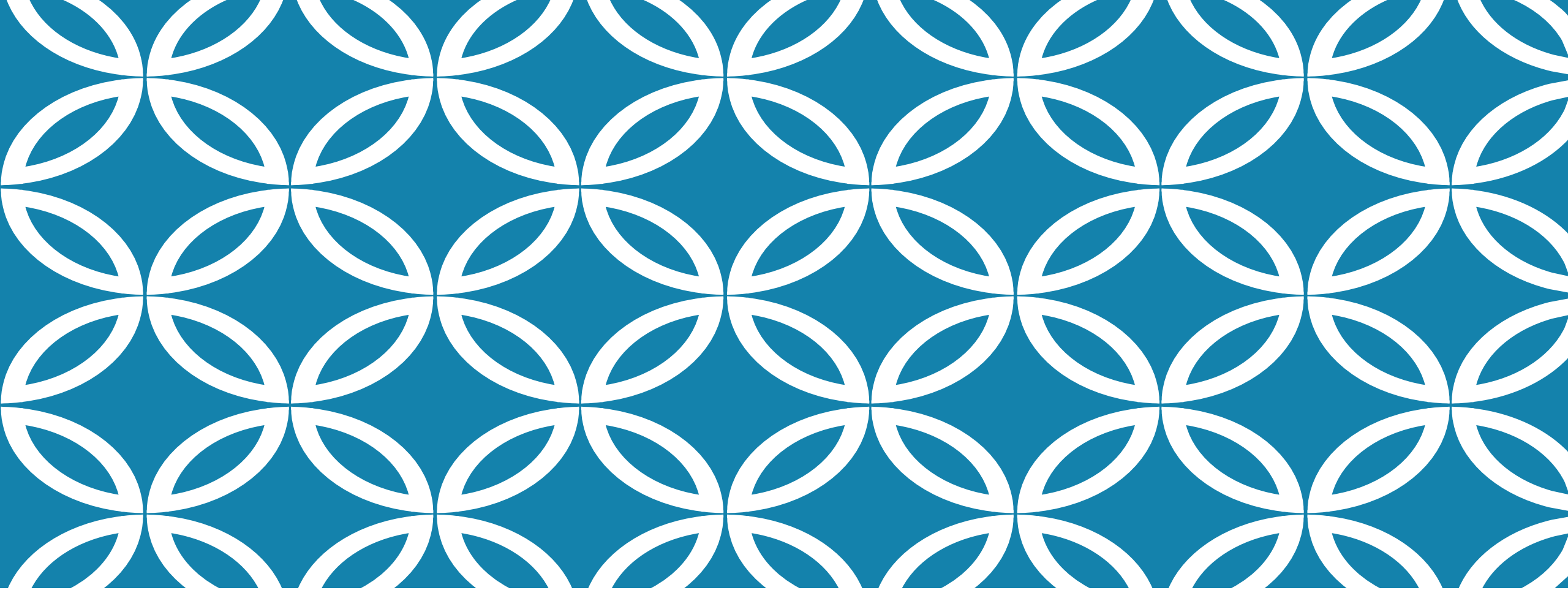


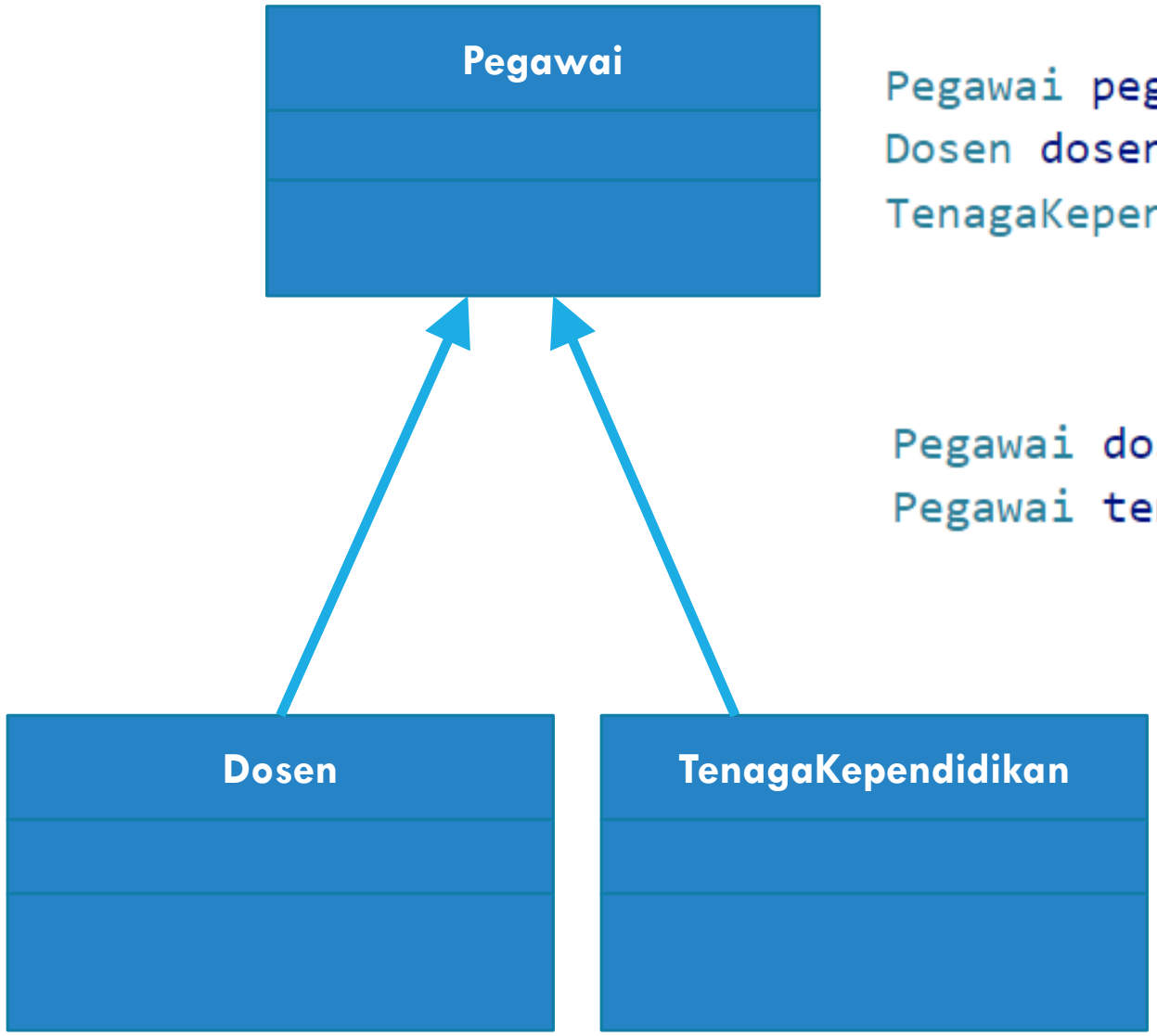
POLYMORPHISM

POLYMORPHISM

1. Heterogenous Collection
2. Object Casting
3. Polymorphic Arguments
4. InstanceOf

POLYMORPHISM

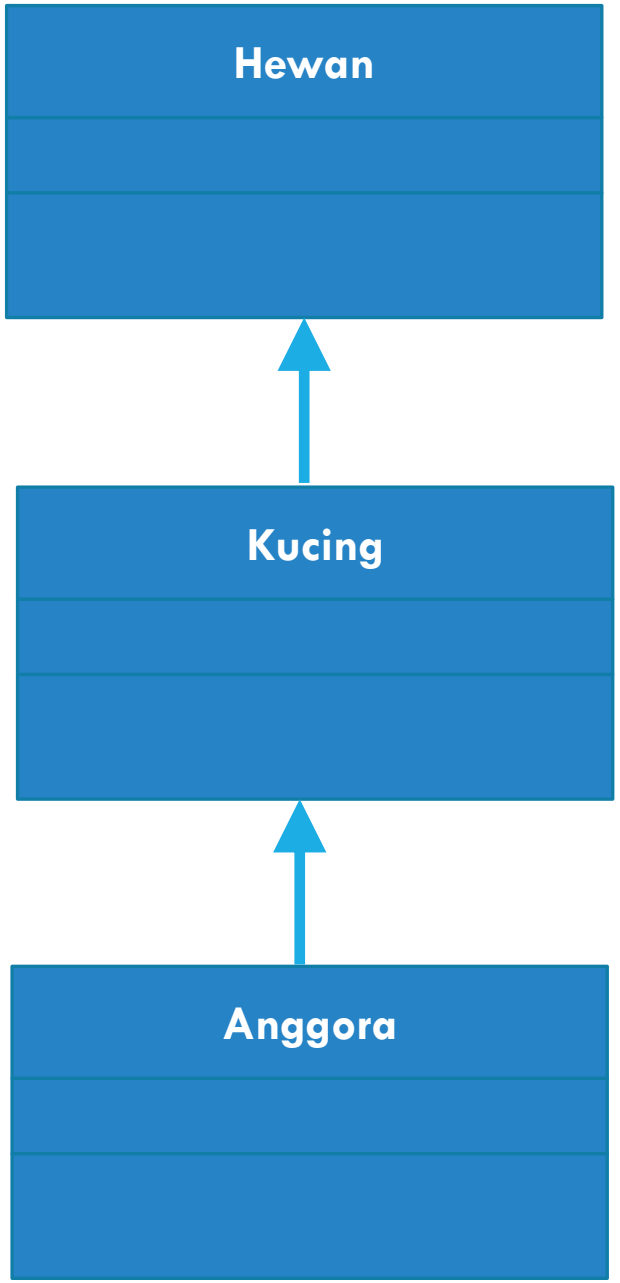
- Polymorphism: poly (banyak), morph (bentuk)
- Polimorfisme → konsep dalam OOP yang memperbolehkan sebuah aksi diimplementasikan secara berbeda
- Polimorfisme →konsep dalam OOP yang memperbolehkan sebuah objek untuk memiliki banyak bentuk




```
Pegawai pegawai1 = new Pegawai();
Dosen dosen1 = new Dosen();
TenagaKependidikan tendik1 = new TenagaKependidikan();
```

```
Pegawai dosen1 = new Dosen();
Pegawai tendik1 = new TenagaKependidikan();
```

dosen1 merupakan instance dari class Dosen tapi dikenali sebagai object bertipe Pegawai




```
Anggora anggora1 = new Anggora();
Kucing anggora2 = new Anggora();
Hewan anggora3 = new Anggora();
```

CLASS *OBJECT*

Class Object merupakan root dari semua hirarki class. Artinya, semua object memiliki superclass dengan nama Object. Oleh karena itu, semua instance dapat dikenali sebagai Object

```
Object kucing1 = new Kucing();
Object dosen1 = new Dosen();
```

HETEROGENOUS COLLECTION

- Dalam Java, setiap elemen dalam array harus memiliki tipe yang sama
- Konsep polimorfisme memungkinkan elemen dalam array bersifat heterogen

```
ArrayList<Pegawai> daftarPegawai = new ArrayList<Pegawai>();
daftarPegawai.add(new Dosen());
daftarPegawai.add(new TenagaKependidikan());
daftarPegawai.add(new Dosen());

Ular ular1 = new Ular();
Lebah lebah1 = new Lebah();
Kucing kucing1 = new Kucing();
Hewan[] daftarHewan = {ular1, lebah1, kucing1};


Object[] objects = {dosen1, ular1, tendik1, lebah1};
```

OBJECT CASTING

- Typecasting merupakan konversi variable dari suatu tipe data menjadi tipe data lainnya
- Typecasting juga dapat dilakukan terhadap object
- Terdapat 2 jenis object casting: upcasting & downcasting

UPCASTING

- Dilakukan untuk mengubah child object menjadi parent object
- Dapat dilakukan secara implisit

```
Pegawai dosen1 = new Dosen();
Pegawai dosen2 = (Pegawai) new Dosen();

TenagaKependidikan tendik1 = new TenagaKependidikan();
Pegawai pegawai = tendik1;
```

eksplisit

- dosen1 merupakan instance dari class Dosen, tetapi setelah proses upcasting akan dikenali sebagai object bertipe Pegawai
- tendik1 merupakan instance dari class TenagaKependidikan, tetapi setelah upcasting akan dikenali sebagai object bertipe Pegawai

DOWNCASTING

- Dilakukan untuk mengubah parent object menjadi child object
- Tidak dapat dilakukan secara implisit

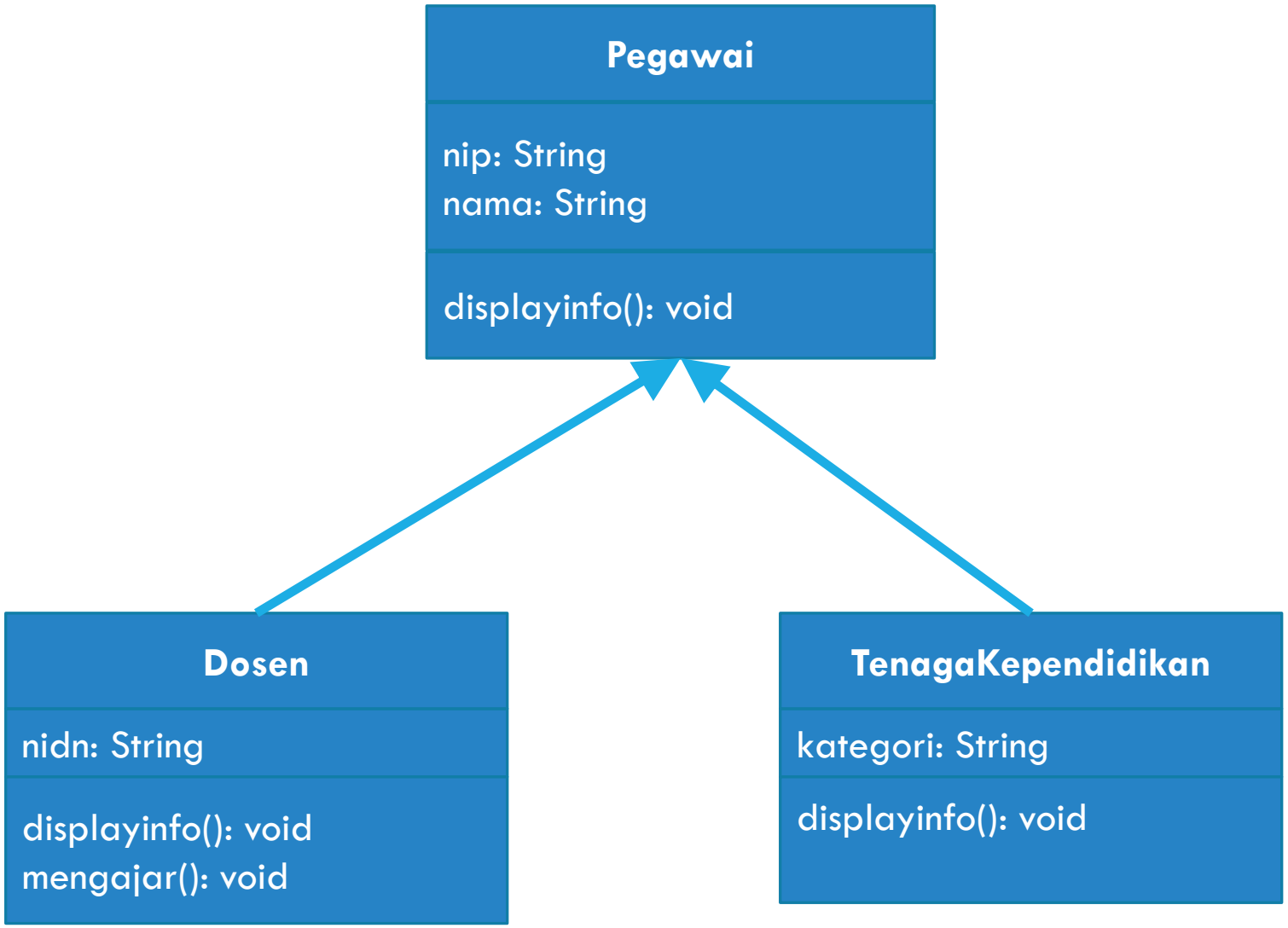
```
Hewan hewan1 = new Ular();
Hewan hewan2 = new Lebah();
```

Upcasting

```
Ular ular1 = (Ular) hewan1;
Lebah lebah1 = (Lebah) hewan2;
```

Downcasting

- Jika hewan2 didowncast menjadi object bertipe Ular maka akan muncul Java.Lang.ClassCastException


Pegawai
nip: String
nama: String
displayinfo(): void
Dosen
nidn: String
displayinfo(): void
mengajar(): void
TenagaKependidikan
kategori: String
displayinfo(): void

```
Dosen dosen1 = new Dosen("19940201", "Widia, S.Kom. M.Kom", "199402");

Pegawai pegawai1 = dosen1;

System.out.println(pegawai1.nip);
System.out.println(pegawai1.nama);
System.out.println(pegawai1.nidn);
pegawai1.mengajar();
```

- dosen1 merupakan instance dari class Dosen, tetapi setelah proses upcasting akan dikenali sebagai object bertipe Pegawai
- Oleh karna itu, dosen1 tidak akan dapat mengakses atribut yang dideklarasikan di class Dosen (misalnya nidn) maupun memanggil method yang hanya ada di class Dosen (misalnya mengajar())

```
Dosen dosen1 = new Dosen("19940201", "Widia, S.Kom. M.Kom", "199402");

Pegawai pegawai1 = dosen1;

System.out.println(pegawai1.nip);
System.out.println(pegawai1.nama);
pegawai1.displayInfo();
```

- Method displayInfo() dapat dikenali karena ada di class Pegawai, tapi karena terdapat overriding maka yang akan dieksekusi adalah method displayInfo() pada class Dosen

POLYMORPHIC ARGUMENTS

- Konsep polimorfisme juga memungkinkan parameter dari suatu method menerima argument dengan berbagai bentuk object
- Misalkan, terdapat method train() dengan parameter pegawai bertipe Pegawai. Method ini akan dapat menerima argument bertipe Pegawai atau class lain yang merupakan subclass dari Pegawai
- Proses ini juga termasuk upcasting

```
public static void train(Pegawai pegawai){
    System.out.println("Memberikan pelatihan untuk pegawai");
    pegawai.displayInfo();
}
```

```java
public static void main(String[] args) {
    Dosen dosen1 = new Dosen("19940201", "Widia, S.Kom. M.Kom", "199402");
    TenagaKependidikan tendik1 = new TenagaKependidikan("19750301", "Aida, A.Md.", "Tenaga Administrasi");

    train(dosen1);
    train(tendik1);
}

public static void train(Pegawai pegawai){
    System.out.println("Memberikan pelatihan untuk pegawai");
    pegawai.displayInfo();
}
```

Method train() dapat dipanggil dengan argument dosen1 bertipe Dosen maupun tendik1 bertipe TenagaKependidikan karena dilakukan upcasting ke class Pegawai

INSTANCEOF

- Keyword instanceOf digunakan untuk mengidentifikasi apakah sebuah object merupakan instance dari suatu class/interface

```
Hewan hewan1 = new Ular();
Hewan hewan2 = new Lebah();

System.out.println(hewan1 instanceof Ular);     //true
System.out.println(hewan2 instanceof Ular);     //false
```

```java
public static void train(Pegawai pegawai){
    pegawai.displayInfo();
    System.out.println("Mengenalkan lingkungan kampus");
    System.out.println("Menginfokan SOP/Juknis");

    if (pegawai instanceof Dosen) {
        System.out.println("Memberikan pelatihan pedagogik");
    }
}
```

POLYMORPHIC ARGUMENTS WITH INTERFACE

```
public interface Printable {
    void print();
}
```

```
public class Printer {
    public void cetak(Printable printable) {
        printable.print();
    }
}
```

CLASS WEBPAGE

```java
public class Webpage implements Printable {
    private String url;
    private String title;
    private String content;

    @Override
    public void print() {
        System.out.println("URL: " + url);
        System.out.println("Title: " + title);

        String plainText = content.replaceAll("<[^>]*>", "");
        System.out.println(plainText);
    }
}
```

CLASS DOCUMENT

```java
public class Document implements Printable {
    private String fileName;
    private String content;

    @Override
    public void print() {
        System.out.println("Mencetak dokumen: " + content);
    }
}
```

CLASS DOCUMENT

```
public class Demo {
    Run | Debug
    public static void main(String[] args) {
        Printable webpage1 = new Webpage();
        Printable doc1 = new Document();

        Printer printer = new Printer();
        printer.cetak(webpage1);
        printer.cetak(doc1);
    }
}
```