Polimorfisme
Overriding
Overloading

Pertemuan 8

Polimorfisme

- Poly artinya banyak, morfisme artinya bentuk.
- Polimorfisme (bahasa inggris polymorphism) adalah sebuah prinsip dalam biologi di mana oraganisme atau spesias dapat memiliki banyak bentuk atau tahapan (stages). (Link)
- Prinsip ini juga diadopsi pada pemrograman berorientasikan objek.
- Sehingga kita dapat definisikan sebagai berikut:
  - Polimorfisme dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak “bentuk” method yang berbeda-beda meskipun namanya sama.
  - “Bentuk” di sini dapat kita artikan: isinya berbeda, parameternya berbeda, dan tipe datanya berbeda.

Polimorfisme pada Java ada dua macam:

- Static Polymorphism (Polimorfisme statis);
- Dynamic Polymorphism (Polimorfisme dinamis).

Beda dari keduanya terletak pada cara membuat polimorfismenya.

Polimorfisme statis menggunakan **method overloading** sedangkan polimorfisme dinamis menggunakan **method overriding**.

Misal: Manager adalah Employee

```
public class Employee {
        public String nama;
        public String gaji;

        void infoNama(){
                System.out.println("Nama" + nama);
        }
}

public class Manajer extends Employee {
        public String departemen;
}
```

Contoh

- Employee emp = new Manager();
- Reference variabel dari emp adalah Employee.
- Bentuk emp adalah Manager.

Perbedaan Method Overloading dengan Method Overriding

Overloading

- Method overloading terjadi pada **sebuah class** yang memiliki **nama method yang sama** tapi memiliki **parameter dan tipe data yang berbeda**.

- Kata kunci yang perlu kamu ingat:
  - “Dalam satu class”
  - “Nama method sama”
  - “Tipe data dan parameter beda”
- Itulah **method Overloading**.

Contoh (1)

- Contohnya misalkan kita memiliki class Lingkaran.java. Pada class ini terdapat method luas().
- Method luas() ini bisa saja memiliki parameter yang berbeda.
- Misalnya kita ingin menghitung luas berdasarkan jari-jari (radius) atau diameter.

```
class Lingkaran {

    // method menghitung luas dengan jari-jari
    float luas(float r){
        return (float) (Math.PI * r * r);
    }

    // method menghitung luas dengan diameter
    double luas(double d){
        return (double) (1/4 * Math.PI * d);
    }

}
```

- Class Lingkaran memiliki dua method yang namanya sama, yakni luas().
- Tapi parameter dan tipe datanya berbeda..
- ..dan juga isi atau rumus di dalamnya berbeda.
- Inilah yang disebut **polimorfisme satatis**.

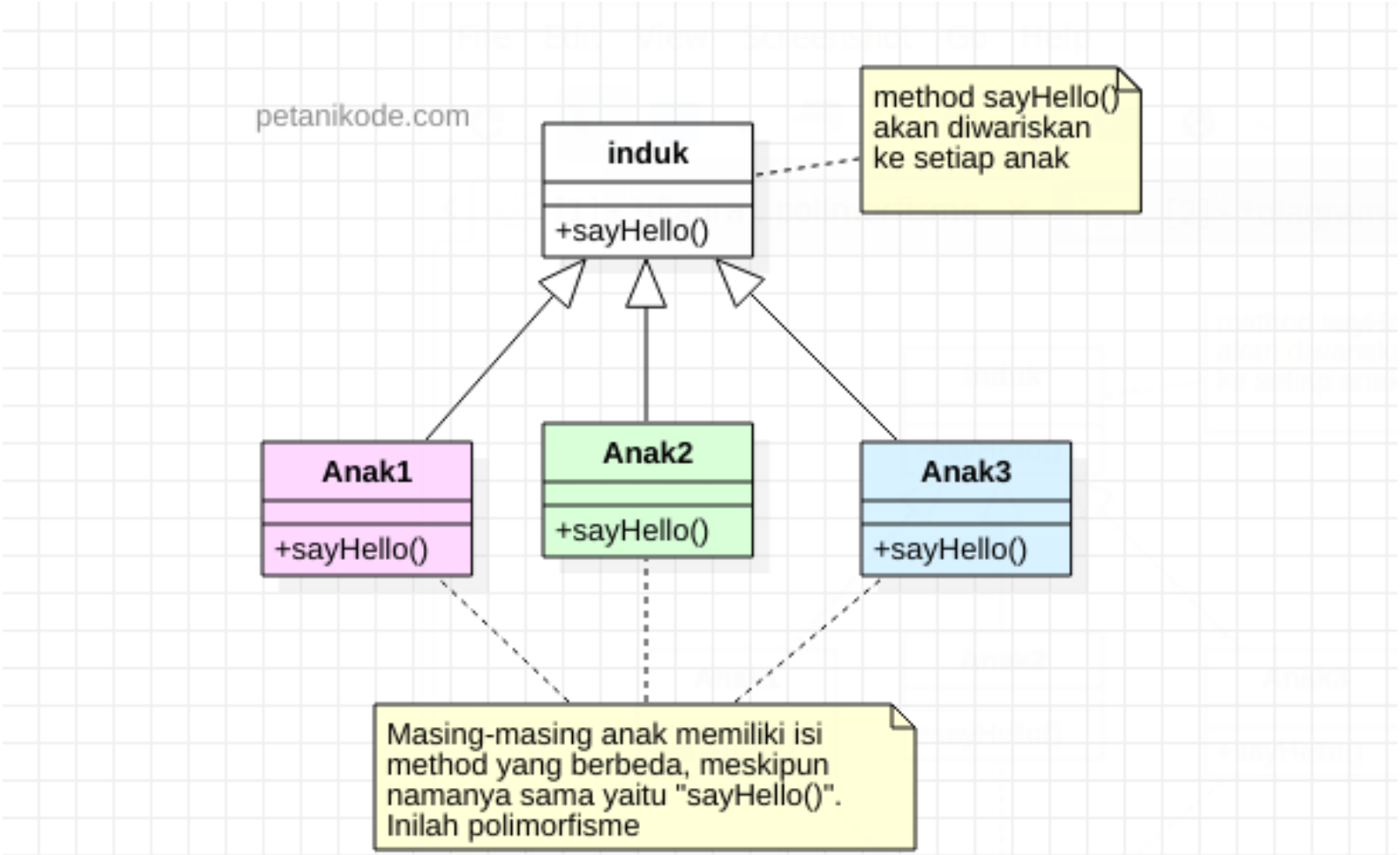
Contoh (2)

```
Class RoundingValue {

      private double defaultGrade = 100.3;

      public RoundingValue(){}

      /*
      * First Method
      */
      public double roubdingUp () {
            System.out.println("Excecuting first method");
            Return (double) Math.ceil(defaultGrade);
      }

      /*
      * Second Method - Its have different parameter with first method
      */
      public double roubdingUp (double numberGrade) {
            System.out.println("Excecuting second method");
            Return (double) Math.ceil(numberGrade);
      }
```

```
/*
* Main Method - First Call
*/
public static void main String[] args() {
      RoundingValue roundingValue = new RoundingValue();

      System.out.print("Value from first method : ");
      System.out.println(""+ roundingValue.roubdingUp() );


      System.out.print("Value from first method : ");
      System.out.print(""+ roundingValue.roubdingUp(222.2));

}
```

Overriding

- Plimorfisme dinamis biasanya terjadi saat kita menggunakan pewarisan (inheritance) dan implementasi interface.
- Pada pewarisan, kita bisa mewariskan atribut dan method dari class induk ke class anak.
- Class anak akan memiliki nama method yang sama dengan class induk dan anak yang lainnya.
- Di sinilah akan terjadi polimorfisme
- **Nama method sama.**
- **Isi / Body method berbeda.**
- **Biasanya berbeda class. interface dan implement.**



Sumber: https://www.petanikode.com/java-oop-polimorfisme/

Contoh

First Class

```
Class OverRiding  {

        public OverRiding(){}

        /*
        * First Method
        */
        public void firstMethod ( ) {
               System.out.println("Method on class OverRiding is calling
        }

}
```

Second Class

```
Class SubOverRiding extends OverRiding {

        /*
        * Override First Method on class OverRiding
        */
        public void firstMethod () {
              System.out.println("This method override ");
              System.out.println("Method on class SubOverRiding in callin
        }

        /*
        * Main Method
        */
        Public static void main (String [ ] args) {
               new OverRiding().firstMethod();

               new SubOverRiding().firstMethod();;
        }

}
```