**Pengertian Polymorphism**

* Poly -> banyak
* Morphism -> bentuk

Polymorphism merupakan sebuah konsep dalam object-oriented programming (OOP), di mana class dapat memiliki method dengan *nama yang sama* namun *implementasinya (isinya) berbeda*.

Kata kunci dari Polymorphism -> “*Berbeda-beda tapi Namanya sama*”

**Jenis-jenis Polymorphism**

* *adhoc*: overriding, overloading
* Method overloading terjadi pada sebuah class, yang memiliki nama method yang sama tapi memiliki parameter yang berbeda.
* Overriding terjadi antara parent class dengan subclass, yang nama method dan parameter nya sama

|  |  |  |
| --- | --- | --- |
|  | Overriding | Overloading |
| Relasi | Antara parent class dengan child class | Dalam satu class |
| Parameter | sama | berbeda |
| Nama | sama | sama |
| Return Value | sama | Boleh sama, boleh berbeda |

* *parameterized*: disebut juga generic, jadi kita bisa memasukkan berbagai Objek. Salah satu contoh pada ArrayList<T>, <T> melambangkan parameter generic dimana bisa diisikan Objek apa saja.

**Contoh penerapan Polymorphism**

**Oveloading**

**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);

}

}

**Overriding**

Misalkan ada SuperClass Binatang yang memiliki method suaraBinatang().

SubClass nya bisa saja Anjing, Kucing, Singa, dan binatang lainnya. Masing-masing dari SubClass tersebut pasti punya suara yang berbeda-beda. Contohnya Anjing menggonggong, kucing meng-ngeong, singa mengaum.

Kesimpulannya, walaupun mereka memiliki method yang sama (suara) tapi masing-masing memiliki suara yang berbeda.

**//SuperClass  
public** **class** Binatang {

**public** **void** suaraBinatang(){

System.***out***.println("Binatang bersuara");

}

}

//SubClass  
**public** **class** Anjing **extends** Binatang{

**public** **void** suaraBinatang(){

System.***out***.println("Anjing mengatakan: woof wooof");

}

}

//SubClass

**public** **class** Kucing **extends** Binatang{

**public** **void** suaraBinatang(){

System.***out***.println("Kucing mengatakan: miaw miaw");

}

}

//SubClass

**public** **class** Singa **extends** Binatang{

**public** **void** suaraBinatang(){

System.***out***.println("Singa mengatakan: roaar rooaaarr");

}

}

//Main Class  
**public** **class** Main{

**public** **static** **void** main(String[] args) {

Binatang binatang = **new** Binatang(); //membuat object Binatang

Binatang anjing = **new** Anjing(); //membuat object anjing

Binatang kucing = **new** Kucing(); //membuat object kucing

//memanggil method

binatang.suaraBinatang();

anjing.suaraBinatang();

kucing.suaraBinatang();

}

}

**Parameterized**

**public** **class** Box<Template> {

//Template -> tipe data generic (nama bebas)

//t -> variable

**private** Template t;

**public** Box() {

// **TODO** Auto-generated constructor stub

}

**public** **void** add(Template t){

**this**.t = t;

}

**public** Template get(){

**return** t;

}

**public** **static** **void** main(String [] args){

Box<Integer> integerBox = **new** Box<Integer>();

Box<String> stringBox = **new** Box<>();

integerBox.add(10);

stringBox.add("Hello world");

System.***out***.println("Print integer: "+integerBox.get());

System.***out***.println("Print string: "+stringBox.get());

}

}

**Abstract**

* Abstract class: tidak bisa diinstantiate langsung (constructornya tidak bisa langsung dipanggil). Harus di extends ke class lain untuk di instantiate
* Abstract method: hanya nama method, tidak perlu diberikan implementasi

Contoh:

//abstract class

//abstract class boleh memiliki abstract method dan method biasa

**public** **abstract** **class** Animal {

**public** Animal() {

}

//abstract method

**public** **abstract** **void** suaraBinatang();

}

**public** **class** Dog **extends** Animal {

**public** Dog() {

}

@Override

//implementasi di subclass

**public** **void** suaraBinatang() {

System.***out***.println("Woof!");

}

}

**public** **class** Main {

**public** Main() {

// instansiasi

Animal b = **new** Dog();

b.suaraBinatang();

}

**public** **static** **void** main(String[] args) {

**new** Main();

}

}

Untuk belajar lebih lagi silahkan buka link dibawah ini

Referensi:

<https://www.petanikode.com/java-oop-polimorfisme/>

<https://www.termasmedia.com/pemrograman/java/250-memahami-konsep-polimorfisme-polymorphism-di-java.html>

<https://www.w3schools.com/java/java_polymorphism.asp>