# Relatie overerven abstractie en polymorfisme les samenvatting

### Voorbeeld

### Te gebruiken bij:

-

### Eigenschappen:

-

### Info

#### recap

Itereren de verandering binnen de loop.

While-loop heeft enkel een conditie.

For-loop heeft initialisatie en update.

#### Collection

Array is vaste lijst met objecten

#### Collectie

List is gesorteerd en bevat alle items.

Set is gesorteerd en bevat geen duplicaten.

Map is geïndexeerde verzameling van objecten.

#### Klasse en relaties:

* Klasse is een blauwdruk.
  + Hond heeft benen (has-a) (compositie)
  + Hond is een dier (is-a) (inheritance)
  + Hond speelt met een bal “losse objecten met connectie” (uses-a) (dependence)

##### Composition (has-a):

Class Leg {

}

Class Log {

Private final list<leg> legs;

This.legs = legs;

}

##### inheritance (is-a):

class Animal {

}

Class Dog extends Animal {

}

##### dependence (uses-a):

class Ball {

}

Class Dog {

Void PlayWith(Ball Ball){ // instantie (geen nieuw object)

System.println(“play with the ball”);

}

}

#### Overerven (is-a)

* 1 extentie per classe (niveau)
* Max aantal over ervingen 60 voorkeur zo rond 5.
* Voorkomt duplicate code
* Juiste verantwoordelijkheden op het juiste abstractie niveau
* Data objecten (real-life objecten)
  + Klant is een person
  + Prospect is een person
  + Person heeft een naam en adres
* Serviceobjecten(generieke functionaliteit
  + Authenticatie Services voor authenticatie mechanisme.
* Final (open/close precipe)class kan je niet extenden
* Aanpassing moet tot het hoogste niveau aanpassen worden aangepast.
* “Tightly coupled”.
* Geen acces control public = public.

#### Composition(has-a) over inheritance(is-a)

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving

* This.person = person icm final.
* Hoeft niet alle aanpassingen tot hoogste niveau enkel de class zelf.

###### Super() call in constructor

* Zonder parameters wordt default aan gehouden.
* Met parameter verplicht om volledig uit te typen.

#### Abstracte klasse

* Abstracte methode kan alleen bestaan in een abstracte klasse.
* Kan niet worden construct.
* Kan niet op zichzelf bestaan.
* Bevat details die normaal herhaald worden en in de code staan.

#### polymorfisme

In de code worden twee klassen gedefinieerd: Animal en Dog. De klasse Dog is een subklasse van de klasse Animal. Hierdoor heeft Dog toegang tot alle eigenschappen en methoden van Animal en kan het deze overschrijven of aanpassen indien nodig.

De klasse Animal heeft een methode genaamd makeSound(). Deze methode print simpelweg "Geluid maken" naar de console.

De klasse Dog overschrijft de makeSound()-methode van de klasse Animal. Het voegt zijn eigen implementatie toe door "Woof woof!" naar de console te printen.

Dit betekent dat wanneer je een object van de klasse Dog maakt en de makeSound()-methode aanroept, de versie van de methode in de klasse Dog wordt uitgevoerd en "Woof woof!" wordt afgedrukt.

Bijvoorbeeld:

java

Animal animal = new Animal();

animal.makeSound(); // Output: "Geluid maken"

Dog dog = new Dog();

dog.makeSound(); // Output: "Woof woof!"

* Zelfde constructor elke klasse een eigen antwoord.
* Methode wordt wel gebruikte in super klasse maar staat niet in de klassen en wordt dan ook overgeorven.

Afbeelding met tekst, schermopname, Lettertype, ontwerp

Automatisch gegenereerde beschrijving