Java Programming - Deel 2

# Overerving

///DRAFTeen vorm van code-herbruik, uitbreiding van een bestaande klasse

2 klassen met dezelfde attributen => 1 superklasse met 2 subklassen waarbij de superklasse de gemeenschappelijke attributen bevat

Square (subklasse) extens Rectangle (superklasse) (klassenhiërarchie), subklasse erft over van/is een soort van superklasse.

een subklasse erft eigenschappen en methoden over van superklasse, kan zelf eigenschappen en methoden toevoegen en kan deze overriden( (één superklasse per klasse/één keer extends), die superklasse kan wel subklasse zijn van een andere superklasse enz

overriden = eigenschap met dezelfde naam en datatype, je krijgt bijgevolg de overschreven eigenschapswaarde

DON’T do this met eigenschappen, wel met methoden (bad practice)

//TOEGANGSNIVEAU DIAGRAM

alle public en protected methoden van de superklasse kunnen we gebruiken vanuit de subklasse, een subklasse is een uitbreiding van de superklasse (alle niet private eigenschappen en methoden van de superklasse zijn ook beschikbaar in de subklasse).

override (overschrijven/verbergen) is wanneer we in een subklasse een andere implementatie geven van een bestaande methode in de superklasse (dezelfde signatuur = naam + parameters + return type)

!elke methode uit een superklasse is standaard overschrijfbaar in een subklasse

public class Square extends Rectangle {  
 private int side;

//met deze annotatie in de subklasse om de compiler te laten nakijken of deze methode wel degelijk een methode uit zijn superklasse overschrijft  
 @Override  
 public int getArea(){  
 return side \* 2;  
 }  
}

om aan te duiden dat de methode uit de superklasse gebruikt moet worden

public void setSide(int side){  
 super.setWidth(side);  
 super.setHeight(side);  
}

Constructors

Indien geen geschreven => default

//default contructor van normale klasse, in dit geval superklasse  
public Rectangle(){  
}

//default constructor van subklasse  
public Square(){  
 super();  
}

Indien een subklasse gebruikt maakt van een constructor van de superklasse, moet deze met de juiste parameters opgeroepen worden in de subklasse

public Square(int side){  
 super(this, this); //altijd eerste statement  
 this.side = side;  
}

Final klassen

Kunnen geen subklassen hebben met als nut: beveiliging, ontwerp, snelheid

public final class GeenSubKlassen{}

Abstracte klassen

HIervan kunnen geen objecten gemaakt worden, dienen als superklasse van andere klassen4

public abstract class GeenObjecten{}

public abstract void abstracteMethode(); // geen implementatie!

deze bevatten (eventueel) abstracte methoden dewelke verplicht geïmplementeerd dienen te worden door de niet abstracte subklassen. Een subklasse roept altijd de constructor van de superklasse op omdat die een uitbreiding is van de superklasse!

OBJECT De klasse object is de superklasse van alle klassen, alle klassen erven over van de klasse Object. Geen attributen, enkel methoden. enkele vb.:

String toString()

boolean equals(Object o) --> vergelijkt 2 objecten (standaard zijn ze gelijk als ze naar hetzelfde object in het geheugen refereren. Als we deze overriden moeten deze samen met de methode hashcode vervangen.

int hashcode()

We bepalen zelf de eigenschappen wanneer 2 objecten gelijk zijn, als deze gelijk zijn moeten ze ook dezelfde hashcodeteruggeven:

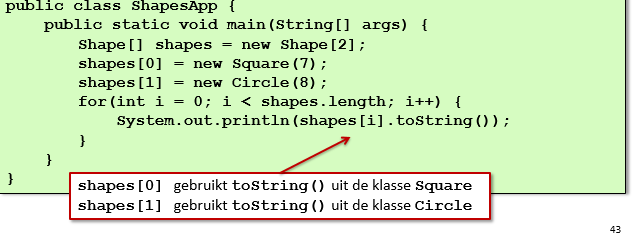
public int hashCode(){  
return x ^ y ^ height ^ width;}

Class getClass()

instanceof (kijkt na of een object een instantie is van de klasse, superklasse of interface

POLYMORFISME

Meerdere vormen/verschijningsvormen. In code betekent dit dat objecten van klasse die afgeleid is van een superklasse beschouwd en behandeld kunnen worden als objecten van die superklasse.

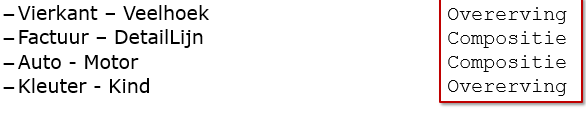


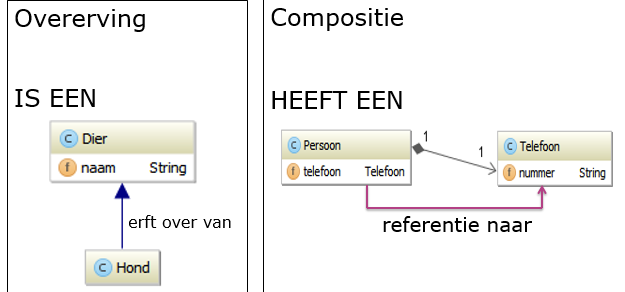
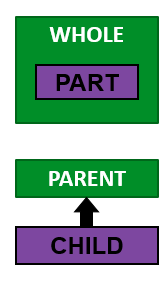
Late binding = uit welke klasse de methode gebruikt wordt, wordt at runtime bepaald. Polymorfisme geldt niet voor klasse (static) methoden of eigenschappen.

ASSOCIATIES tussen objecten

COMPOSITIE: een object bevat als onderdeel een ander object relatie: ... heeft een ...$

OVERERVING: een object is een afgeleide van een ander object relatie: ... is een ... (instanceof om te testen);





superklassen en subklassen: elke subklasse kan zelf ook een superklasse worden, een subklasse is specifieker dan zijn superklasse --> specialisatie

elke klasse kan slecht één directe superklasse hebben -> single inheritance

klassenhiërarchie begint in Java bij de klasse Object