Les 12 lesprogramma

# Opgave Personenlijst implementatie

Gegeven onderstaande code en het klassendiagram dat je in de voorbereiding gemaakt hebt.

public class Persoon {  
 protected String naam, voornaam;  
  
 public Persoon(String naam, String voornaam) {  
 this.naam = naam;  
 this.voornaam = voornaam;  
 }  
  
 public String getNaam() {  
 return naam;  
 }  
  
 public void setNaam(String naam) {  
 this.naam = naam;  
 }  
  
 public String getVoornaam() {  
 return voornaam;  
 }  
  
 public void setVoornaam(String voornaam) {  
 this.voornaam = voornaam;  
 }  
}

public class Docent extends Persoon {  
 protected String code;  
   
 public Docent(String naam, String voornaam, String code) {  
 super(naam, voornaam);  
 this.code = code;  
 }  
  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
}

public class Student extends Persoon {  
 protected int nummer;  
 protected Docent SLBer;  
   
 public Student(String naam, String voornaam, int nummer, Docent SLBer) {  
 super(naam, voornaam);  
 this.nummer = nummer;  
 this.SLBer = SLBer;  
 }  
  
 public int getNummer() {  
 return nummer;  
 }  
  
 public void setNummer(int nummer) {  
 this.nummer = nummer;  
 }  
  
 public Docent getSLBer() {  
 return SLBer;  
 }  
  
 public void setSLBer(Docent sLBer) {  
 SLBer = sLBer;  
 }  
}

## Opgave A

In de klasse Student is een veld opgenomen met de naam SLBer en het type Docent. Dit veld wijst naar de docent die de SLBer is van de betreffende student. Nu zijn er bij ICA ook SLB’ers die geen docent zijn, dus we kunnen dit ontwerp niet handhaven. Kun je het type van het veld SLBer nu beter veranderen naar:

1. String of
2. Persoon?

Beargumenteer beide opties.

## Opgave B

Implementeer in elke klasse de methode toString (zie boek pagina 343 en 344). Deze methode moet de naam en de waarde van elk veld uit de klasse teruggeven als String. De toString uit Docent en Student moet ook de naam en de waarde van elk veld uit de klasse Persoon teruggeven.

Genereer deze methoden met eclipse ( Source > Generate toString() ) en test de methoden in het hoofdprogramma.

## Opgave C

Gegeven het onderstaande programma:

import java.util.ArrayList;  
  
public class PersonenLijst {  
 private ArrayList<Persoon> lijst;  
   
 public PersonenLijst() {  
 lijst = new ArrayList<>();   
 }  
   
 public ArrayList<Student> getSLBStudenten(Docent SLBer) {  
 }  
   
 public static void main(String[] args) {  
 PersonenLijst p = new PersonenLijst();

Docent piet=new Docent("Piet","Jansen","jnsnp");

Student marie=new Student("Marie","Pieters",31415,piet);

Student jan=new Student("Jan","de Vries",92653,null); // nog geen SLB’er

p.lijst.add(piet);

p.lijst.add(marie);

p.lijst.add(jan);

// voeg zelf nog enkele docenten en studenten toe

ArrayList<Student> studentenVanPiet=p.getSLBStudenten(piet);  
   
 }  
}

Implementeer de methode getSLBStudenten (regel 10). Deze methode krijgt een instantie van Docent mee en retourneert een lijst van alle SLB-studenten. Test deze methode in main door de voorbeeldcode hierboven verder uit te breiden en de arraylist aan het einde te printen.

Hint: bij de implementatie kun je gebruik maken van de instanceof operator (boek pagina 351) en van casten (boek pagina 324).

# Opgave Personenlijst zonder cast

Zoals op pagina 325 beschreven wordt is casten meestal onwenselijk. Door de klasse Persoon abstract te maken en een abstracte methode getSLBer te maken kunnen we van deze cast afkomen.

## Opgave A

Implementeer deze oplossing en omschrijf zo exact mogelijk wat de consequentie is van deze beslissing.

## Opgave B

Is deze oplossing wenselijk, of kunnen we beter gebruik maken van een cast. Beargumenteer het antwoord.