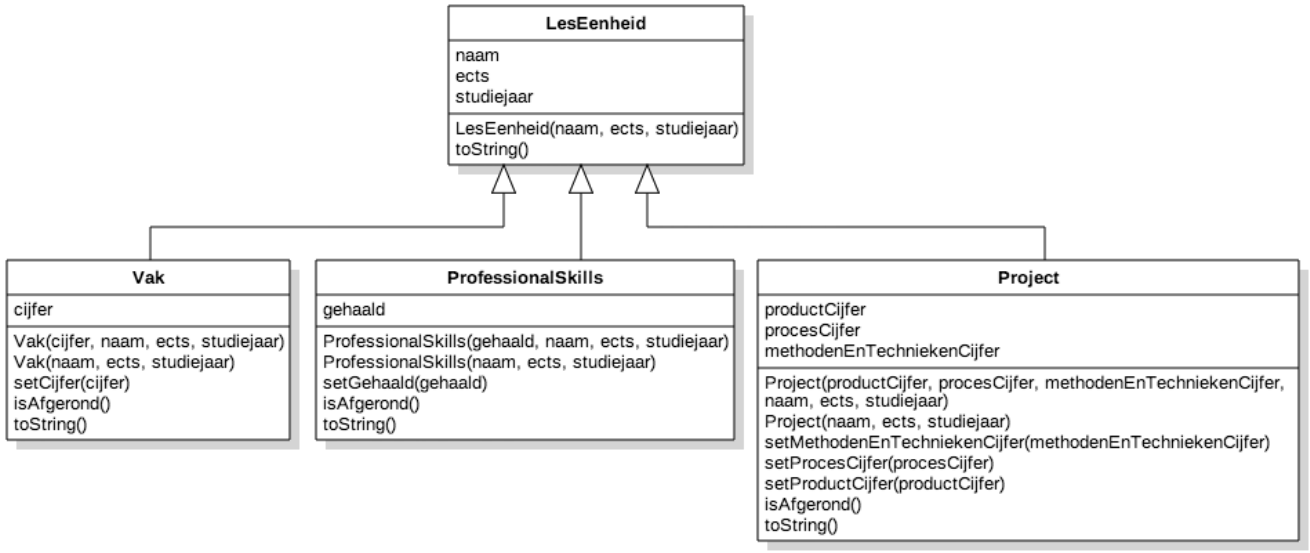
OOP1 – Practicumopdracht 4

LesEenheid

Inleiding

Voor deze opdracht maak je vier klassen die met elkaar samen werken. Het UML class diagram van de vier klassen ziet er als volgt uit (sommige elementen, zoals type van attributen zijn bewust weggelaten, die moet je zelf verzinnen):



N.B. De testcode voor deze opdracht is ook los te vinden als .txt file op de VLO.

Opdracht A

In deze opdracht gaan we inheritance toepassen, door te werken met een superclass en met subclasses.

We gaan vier klassen maken die als volgt met elkaar te maken hebben: **`LesEenheid`** is een superclass en **`Vak`**, **`ProfessionalSkills`** en **`Project`** zijn daar subclasses van.

Klasse LesEenheid

- Deze klasse heeft de volgende eigenschappen
  - naam – naam van de leseenheid.
  - ects – aantal ects dat je met deze leseenheid kan halen. Ects staat voor European Credits, een synoniem voor studiepunten.
  - studiejaar – in welk studiejaar (1, 2, 3 of 4) deze leseenheid wordt gegeven.
- Deze klasse heeft de volgende methoden:
  - Constructor – heeft drie argumenten en initialiseert hiermee de attributen.

- Methode **`String toString()`** – geeft een String representatie terug van een **`LesEenheid`**. Daarin worden alle attributen meegenomen. De String die de methode teruggeeft moet er als volgt uitzien (voorbeeld):
  `Algemeen, 3 ects, studiejaar 2`

- Test deze klasse door in de Main een nieuwe leseenheid aan te maken en deze vervolgens te printen. Je kunt hierbij gebruik maken van de volgende code:

```
LesEenheid lesEenheid = new LesEenheid("Algemeen", 3 , 2);
System.out.println(lesEenheid);
```

Klasse Vak

- Deze klasse is een afgeleide klasse (subklasse) van **`LesEenheid`**.
- Deze klasse heeft –naast de eigenschappen van **`LesEenheid`** – de volgende eigenschap:
  - cijfer – cijfer dat voor dit vak is gehaald.
- Deze klasse heeft de volgende methoden:
  - Constructor – heeft vier argumenten en initialiseert hiermee de attributen. Let goed op de volgorde van de attributen (zie Class Diagram)
    Maak hierbij gebruik van de constructor van **`LesEenheid`**. Schrijf zo min mogelijk dubbele code!
  - Een setter voor het attribuut cijfer.
  - Methode **`String toString()`** – geeft een String representatie terug van een vak. Daarin worden dus alle attributen meegenomen, ook die van **`LesEenheid`**.
    Maak hierbij gebruik van de **`toString`** methode van **`LesEenheid`**. Schrijf zo min mogelijk dubbele code! De String die de methode teruggeeft moet er als volgt uitzien (voorbeeld):
    `OOP1, 3 ects, studiejaar 1, cijfer 7.8`

  N.B. De tweede constructor en de methode **`isAfgerond()`** schrijf je pas bij opdracht B.
- Test deze klasse door in de Main een nieuw vak aan te maken en deze vervolgens te printen. Je kunt hierbij gebruik maken van de volgende code:

```
Vak vak = new Vak(7.8, "OOP1", 3 , 1);
System.out.println(vak);
```

Klasse ProfessionalSkills

- Deze klasse is een afgeleide klasse (subklasse) van **`LesEenheid`**.
- Deze klasse heeft –naast de eigenschappen van **`LesEenheid`** – de volgende eigenschap:
  - gehaald – geeft aan of wel of niet is gehaald.
- Deze klasse heeft de volgende methoden:
  - Constructor – heeft vier argumenten en initialiseert hiermee de attributen. Let goed op de volgorde van de attributen (zie Class Diagram). Maak hierbij handig gebruik van de constructor van **`LesEenheid`**. Schrijf zo min mogelijk dubbele code!
  - Een setter voor het attribuut gehaald.
  - Methode **`String toString()`** – geeft een String representatie terug van een professional skills vak. Daarin worden dus alle attributen meegenomen, ook die van **`LesEenheid`**. Maak hierbij handig gebruik van de **`toString`** methode van **`LesEenheid`**. Schrijf zo min mogelijk dubbele code! De String die de methode teruggeeft moet er als volgt uitzien (voorbeeld):

```
Personal Skills, 2 ects, studiejaar 1, niet gehaald
```

N.B. De tweede constructor en de methode **`isAfgerond()`** schrijf je pas bij opdracht B.

- Test deze klasse door in de Main een nieuw professional skills vak aan te maken en deze vervolgens te printen. Je kunt hierbij gebruik maken van de volgende code:

```
ProfessionalSkills skill =
    new ProfessionalSkills(false, "Personal Skills", 2 , 1);
System.out.println(skill);
skill.setGehaald(true);
System.out.println(skill);
```

Klasse Project

- Deze klasse is een afgeleide klasse (subklasse) van **`LesEenheid`**.
- Deze klasse heeft –naast de eigenschappen van **`LesEenheid`** – de volgende eigenschappen:
    - productCijfer – het cijfer voor de productbeoordeling.
    - procesCijfer – het cijfer voor de procesbeoordeling.
    - methodenEnTechniekenCijfer – het cijfer voor de M&T beoordeling.
- Deze klasse heeft de volgende methoden:
    - Constructor – heeft zes argumenten en initialiseert hiermee de attributen. Let goed op de volgorde van de attributen (zie Class Diagram). Maak hierbij handig gebruik van de constructor van **`LesEenheid`**. Schrijf zo min mogelijk dubbele code!
    - Drie setters voor de attributen productCijfer, procesCijfer en methodenEnTechniekenCijfer.
    - Methode **`String toString()`** – geeft een String representatie terug van een vak. Daarin worden dus alle attributen meegenomen, ook die van **`LesEenheid`**.
      Maak hierbij handig gebruik van de **`toString`** methode van **`LesEenheid`**. Schrijf zo min mogelijk dubbele code! De String die de methode teruggeeft moet er als volgt uitzien (voorbeeld):
      `Fasten Your Seatbelts, 12 ects, studiejaar 1 (7.4, 6.8, 8.0)`
- Test deze klasse door in de Main een nieuw vak aan te maken en deze vervolgens te printen. Je kunt hierbij gebruik maken van de volgende code:

```
Project project = new Project(7.4, 6.8, 8.0,
    "Fasten Your Seatbelts", 12 , 1);
System.out.println(project);
```

Opdracht B

In deze opdracht gaan we de klassen van opdracht A aanpassen met extra functionaliteit.

- Voeg in iedere subklasse (Vak, ProfessionalSkills en Project) een extra constructor toe. Deze heeft drie argumenten: naam, ects en studiejaar. Omdat bij deze constructor het resultaat niet wordt meegeven, moet je dat zelf op een default waarde initialiseren. Voor een wel of niet gehaald is dat **`false`** en voor een cijfer is dat een **`-1`**. Zorg dat je geen magic numbers in je applicatie gebruikt, maar maak er een constante van (zie code conventions). Bedenk goed in welke klasse je deze constante declareert.
  TIP: maak in de extra constructor handig gebruik van de bestaande constructor in dezelfde subklasse en roep deze aan. Schrijf zo min mogelijk dubbele code!

- Voeg in iedere subklasse een methode **`isAfgerond()`** toe. Deze methode controleert of het afgerond is en retourneert dit. Cijfers moeten daarvoor minstens 5.5 zijn (en bij een Project moeten alle cijfers minstens 5.5 zijn).
  Ook hier: geen magic numbers: maak een constante ONDERGRENS_VOLDOENDE en bedenk in welke klasse je deze declareert.
- Test deze klasse door in de klasse **`Main`** de onderstaande code te plakken. Controleer goed of de methode **`isGehaald()`** elke keer de juiste waarde weergeeft!

```
Vak nogEenVak = new Vak("Databases", 3, 1);
System.out.println(nogEenVak.toString() + ", afgerond: " +
     nogEenVak.isAfgerond());
nogEenVak.setCijfer(5.499999);
System.out.println(nogEenVak.toString() + ", afgerond: " +
     nogEenVak.isAfgerond());
nogEenVak.setCijfer(5.5);
System.out.println(nogEenVak.toString() + ", afgerond: " +
     nogEenVak.isAfgerond());

ProfessionalSkills nogEenSkill = new ProfessionalSkills("ICT Ethics", 2, 2 );
System.out.println(nogEenSkill.toString() + ", afgerond: " +
     nogEenSkill.isAfgerond());
nogEenSkill.setGehaald(true);
System.out.println(nogEenSkill.toString() + ", afgerond: " +
     nogEenSkill.isAfgerond());

Project nogEenProject = new Project("Agile Development", 12, 1);
System.out.println(nogEenProject.toString() + ", afgerond: " +
     nogEenProject.isAfgerond());

nogEenProject.setMethodenEnTechniekenCijfer(5.499999);
nogEenProject.setProcescijfer(5.5);
nogEenProject.setProductcijfer(5.5);
System.out.println(nogEenProject.toString() + ", afgerond: " +
     nogEenProject.isAfgerond());

nogEenProject.setMethodenEnTechniekenCijfer(5.5);
nogEenProject.setProcescijfer(5.4999999);
System.out.println(nogEenProject.toString() + ", afgerond: " +
     nogEenProject.isAfgerond());

nogEenProject.setProcescijfer(10);
nogEenProject.setProductcijfer(5.499999);
System.out.println(nogEenProject.toString() + ", afgerond: " +
     nogEenProject.isAfgerond());

nogEenProject.setProductcijfer(10);
System.out.println(nogEenProject.toString() + ", afgerond: " +
     nogEenProject.isAfgerond());

nogEenProject.setMethodenEnTechniekenCijfer(10);
System.out.println(nogEenProject.toString() + ", afgerond: " +
     nogEenProject.isAfgerond());
```