OOP1 – Practicumopdracht 5

Bedrijf

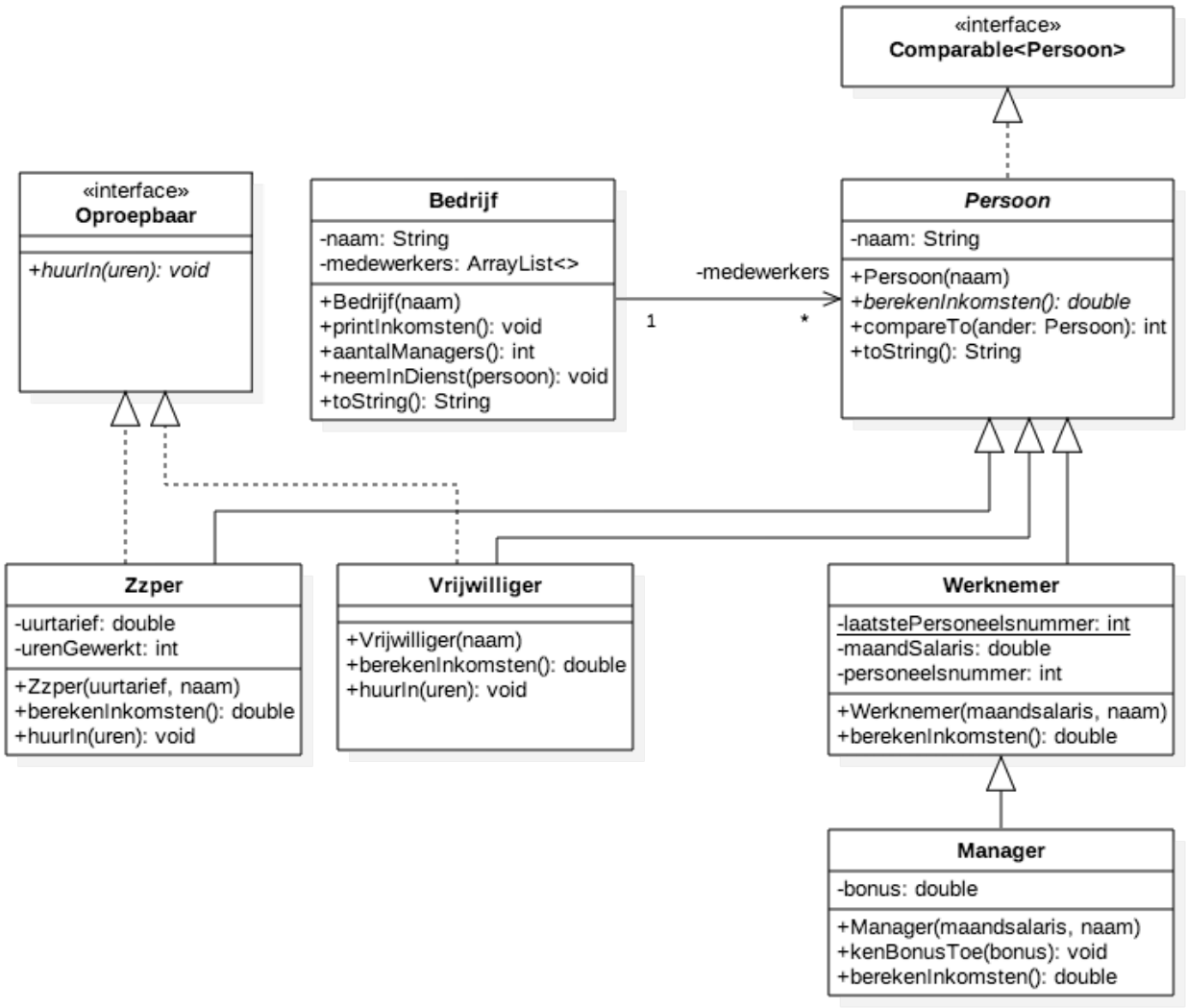
Inleiding

In deze opdracht ga je een bedrijf met werknemers, vrijwilligers en zzp'ers simuleren. Een bedrijf kan personen als medewerker in dienst nemen. Aan het bedrijf kan worden gevraagd hoeveel managers het heeft. Ook kan het bedrijf de inkomsten van alle medewerkers printen. De inkomsten van een medewerker is afhankelijk van het soort medewerker.

Opdracht

In deze uitdagende opdracht wordt er minder voorgekauwd, en moet je meer zelf bedenken. Daarom heb je er ook twee weken tijd voor.

Bouw de structuur, gegeven door het volgende klassendiagram:



De klasse **`Main`** staat niet in dit diagram, maar de code daarvan is wel aangeleverd in het bijgeleverde NetBeans project. Download dit van de VLO.

Hieronder tien verduidelijkingen van het klassendiagram.[1]

1. Alle methoden en constructors moeten de argumenten hebben zoals omschreven.
2. Denk bij constructors goed na over de initialisatie van de variabelen.
3. Het type van de argumenten is overal bewust weggelaten. Denk steeds goed na wat het type zou moeten zijn.
4. Zorg ervoor dat de klasse **`Persoon`** en de klasse **`Bedrijf`** hun eigen **`toString()`** methode hebben. Bij een **`Persoon`** is dat alleen maar diens naam. Voor de gewenste output van **`Bedrijf`** zie het voorbeeld aan het eind. Tip: Maak hierbij slim gebruik van de **`toString()`** van **`Persoon`**.
5. De interface **`Comparable`** hoe je niet zelf te maken, die bestaat al in Java. Wel moet je een eigen **`compareTo()`** methode maken in de klasse **`Persoon`**. Deze moet ervoor zorgen dat personen alfabetisch op naam kunnen worden gesorteerd.
   Zie Liang, §13.6 voor het gebruik van **`Comparable`**.
6. De interface **`Oproepbaar`** wordt geïmplementeerd door personen die op oproepbasis werken, dat wil zeggen de **`Zzper`** en de **`Vrijwilliger`**.
7. De klasse **`Persoon`** is abstract.
8. De klasse **`Werknemer`** heeft een personeelsnummer. Dit wordt automatisch gegenereerd bij het aanmaken van een nieuwe werknemer. Personeelsnummers starten bij 1000, en zijn opeenvolgend.
9. De methode **`berekenInkomsten()`** van iedere klasse moet de inkomsten berekenen. Dit is voor iedere klasse verschillend:
   - Bij een **`Persoon`** is de methode abstract.
   - De inkomsten van een **`Vrijwilliger`** zijn er niet (=0).
   - De inkomsten van een **`Zzper`** worden berekend aan de hand van het uurtarief en het aantal uren dat hij heeft gewerkt.
     Het aantal uren dat hij heeft gewerkt wordt opgehoogd als hij wordt ingehuurd.
     Inhuren gaat via de methode **`huurIn()`**, afkomstig van de interface **`Oproepbaar`**.
   - De inkomsten van een **`Werknemer`** zijn gelijk aan het maandsalaris.
   - De inkomsten van een **`Manager`** worden berekend door het maandsalaris te verhogen met de hoogte van de bonus.
     De bonus kan worden toegekend met de methode **`kenBonusToe()`**.
10. De volgende methoden van de klasse **`Bedrijf`** hebben extra uitleg nodig:
    - **`neemInDienst()`**. Via deze methode kan een medewerker in dienst worden genomen. Deze wordt dan aan de lijst van medewerkers toegevoegd.
    - **`printInkomsten()`**. Deze methode print een lijst van medewerkers met daarbij hun inkomsten. Maak hierbij slim gebruik van de methode **`toString()`** en **`berekenInkomsten()`** van andere klassen. LET OP: de lijst moet gesorteerd zijn, alfabetisch op naam. Gebruik hiervoor je **`compareTo()`** methode. Zie Liang, §13.6.
    - **`aantalManagers()`**. Deze methode geeft het aantal managers van het bedrijf terug. Tip: loop hiervoor door de medewerkers van het bedrijf en gebruik **`instanceof`**.

[1] De theorie voor punt 5, 6 en 7 worden in week 6 behandeld. Sla deze eventueel eerst over.

Je kunt alles testen door de aangeleverde klasse **`Main`** te gebruiken. Haal het commentaar in de methode **`main()`** weg en run de applicatie. De code in deze methode doet het volgende:

1. Maak een bedrijf aan.
2. Voeg verschillende medewerkers toe, huur sommige in, ken bonussen toe.
3. Print het bedrijf.
4. Print de inkomsten van alle medewerkers.
5. Print hoeveel managers het bedrijf heeft.

Als je het leuk vindt mag je de code natuurlijk uitbreiden.

Voorbeeldoutput

Je mag zelf kiezen hoe alles eruitziet, als het maar duidelijk is. Een voorbeeld van output zou kunnen zijn:

```
Bedrijf HvA heeft 6 medewerkers:
       Jantine Jansen
       Piet Pietersen
       Guuse Goedhart
       Brigitte Baas
       Dirk Teur
       Beun Haas

Inkomsten van alle personen:
       Beun Haas, inkomsten: 4800.0
       Brigitte Baas, inkomsten: 10750.0
       Dirk Teur, inkomsten: 11200.0
       Guus Goedhart, bedankt voor uw inzet!
       Jantine Jansen, inkomsten: 1200.0
       Piet Pietersen, inkomsten: 1300.0

Aantal in management: 2
```

Richtlijnen bij coderen (zie ook HBO-ICT code conventions [ICC])[2]

- Zorg dat je naam en het doel van het programma bovenin staan (ICC #1).
- Gebruik de juiste inspringing (*indentation*) bij de lay-out (ICC #2).
- Let op juist gebruik hoofdletters en kleine letters (ICC #3).
- Gebruik goede namen (ICC #4).
- Vermijd *magic number*s (ICC#5).
- Gebruik javadoc tags: **`@author`**, **`@param`** en **`@return`** (ICC #6).
- Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).
- Vermijd dode code (ICC #8).
- Denk aan *encapsulation* (ICC #9).

[2] Zie ook http://www.cs.armstrong.edu/liang/intro9e/supplement/Supplement1dcodingguidelines.html