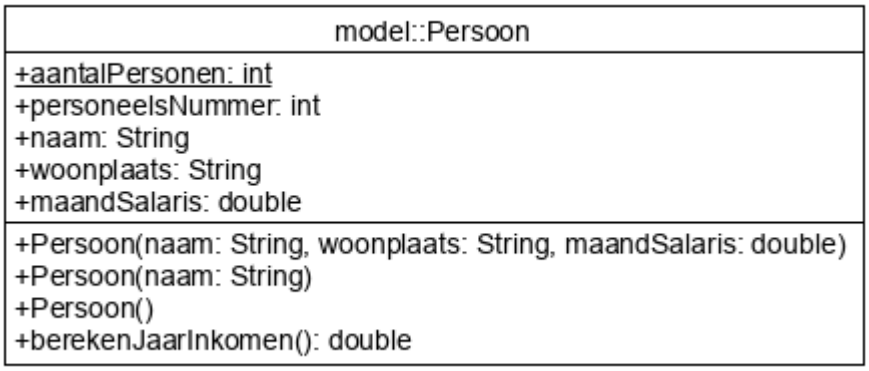
Inhoudsopgave

1 Opdracht Klassen-Objecten-1 Bedrijf

1) Start een nieuw project in IntelliJ en noem dit `Bedrijf`.
2) Maak een package `model` aan.
3) Maak daarbinnen een klasse `Persoon` aan.

Het klassendiagram voor de klasse `Persoon` ziet er als volgt uit:



Merk op dat het attribuut `aantalPersonen` statisch is. Dit attribuut wordt in deze opdracht gebruikt om bij te houden hoeveel personen er toegevoegd zijn (de uitleg volgt hieronder).

4) Voeg de regel
   `public static int aantalPersonen = 0;`
   toe aan de klasse.
   Bedenk dat een statisch attribuut bestaat in de klasse en buiten de instanties beschikbaar is. Het statische attribuut `aantalPersonen` wordt door deze code bij het importeren van de klasse `Persoon` op 0 gezet.

5) Voeg nu de overige attributen toe conform het klassendiagram. Let wel: het attribuut personeelsNummer moet je final maken. Als een Persoon eenmaal een nummer heeft, mag dit niet meer veranderen. In het klassendiagram kunnen we dat helaas niet aangeven.

6) Voeg de all-args constructor als volgt toe:

```
public Persoon(String naam, String woonplaats, double maandSalaris) {
    this.naam = naam;
    this.woonplaats = woonplaats;
    this.maandSalaris = maandSalaris;
    this.personeelsNummer = ++aantalPersonen;
}
```

*Het attribuut `personeelsNummer` wordt dynamisch toegewezen. **Eerst** wordt het statische attribuut `aantalPersonen` met 1 verhoogd (de plussen staan voor het attribuut) en de nieuwe waarde wordt dan toegewezen aan het attribuut `personeelsNummer`. In de volgende stappen wordt dit getoond.*

7) Maak een nieuw package `controller` aan.
8) Maak een nieuwe klasse `BedrijfLauncher` aan.
9) Voeg een `main` methode toe.

10) Voeg de volgende code toe aan je `main` methode:

```
System.out.println(Persoon.aantalPersonen);
Persoon baas = new Persoon("Mark", "Den Haag", 10000);
System.out.println(Persoon.aantalPersonen);
System.out.println(baas.personeelsNummer);
Persoon medewerker = new Persoon("Caroline", "Delft", 4000);
System.out.println(Persoon.aantalPersonen);
System.out.println(medewerker.personeelsNummer);
```

- *In het begin is `aantalPersonen` gelijk aan 0.*
- *Tijdens het instantiëren van `baas` wordt `aantalPersonen` verhoogd met 1 (en is dus gelijk aan 1).*
- *Deze waarde wordt toegewezen aan het attribuut `personeelsNummer` van `baas`.*
- *Tijdens het instantiëren van `medewerker` wordt `aantalPersonen` verhoogd met 1 (en is dus gelijk aan 2).*
- *Deze waarde wordt toegewezen aan het attribuut `personeelsNummer` van `medewerker`.*

11) Run je programma en controleer je uitvoer.
12) Maak nu de tweede constructor `Persoon(naam: String)` aan.
    a) Gebruik de defaultwaarden “Onbekend” voor `woonplaats` en 0 voor `maandSalaris`.
    b) Ken het attribuut `personeelsNummer` toe zoals hierboven is gedaan.
13) Maak nu de default constructor `Persoon()` aan.
    a) Gebruik de defaultwaarde “Onbekend” voor `naam` en de hierboven beschreven defaultwaarden voor `woonplaats` en `maandSalaris`.
    b) Ken het attribuut `personeelsNummer` toe zoals hierboven is gedaan.
14) Voeg de methode `berekenJaarInkomen` toe.
    a) Het jaarinkomen is 12 keer het maandsalaris.
15) Voeg de volgende code toe aan je `main` methode (onder de al bestaande code).

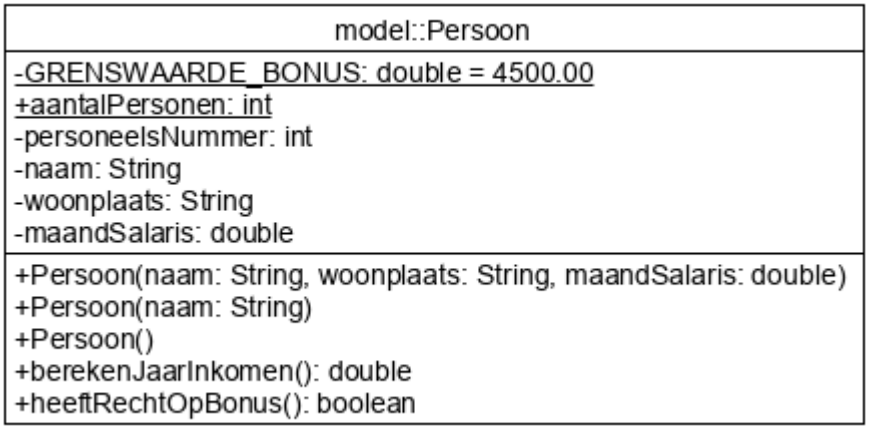
```
Persoon assistent = new Persoon ("Klaas");
Persoon manager = new Persoon();
System.out.println(Persoon.aantalPersonen);
System.out.printf("%s verdient %.2f per jaar\n", baas.naam,
    baas.berekenJaarInkomen());
System.out.printf("%s woont in %s\n", assistent.naam,
    assistent.woonplaats);
```

16) Run je programma, je uitvoer ziet er zo uit:

```
0
1
1
2
2
4
Mark verdient 120000,00 per jaar
Klaas woont in Onbekend
```

2 Opdracht Encapsulation-1 Bedrijf

1) Indien nodig kun je het project `OOP Bedrijf Klassen-Objecten` van de DLO halen. Dit bevat de klassen `BedrijfLauncher` en `Persoon` zoals je die in de opdracht Klassen-Objecten-1 Bedrijf hebt afgesloten.
2) Verander de klasse `Persoon` conform het klassendiagram hieronder:
   a) Voeg de constante `GRENSWAARDE_BONUS` toe.
   b) Maak de vier niet-statische attributen `private`.
   c) Voeg getters en setters toe voor deze vier attributen (de getters en setters zijn niet opgenomen in het klassendiagram). Let wel: voor het attribuut personeelsNummer mag je geen setter opnemen. Dit attribuut is final en kun je dus niet meer aanpassen.
   d) Pas de methode setMaandSalaris() aan, zodat deze een foutmelding geeft als het meegegeven bedrag negatief is. Het maandsalaris wordt dan op 0 gezet.
   e) Pas constructor chaining toe op de drie constructors.
   f) Zorg dat ook in de constructors geen negatief maandsalaris geaccepteerd wordt.
   g) Voeg de methode `heeftRechtOpBonus()` toe. Deze geeft `true` als het maandsalaris groter is dan of gelijk is aan de `GRENSWAARDE_BONUS`.

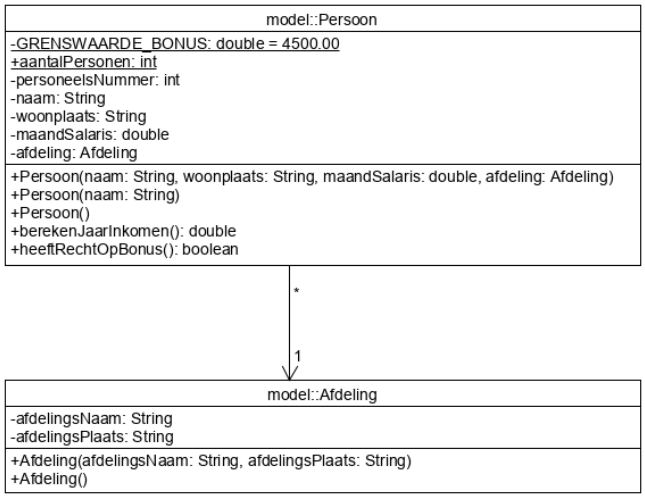


3) Verbeter de fouten, die inmiddels in de code in je main methode zijn ontstaan (benader de attributen via hun getters).
4) Verwijder de laatste twee regels code (beide beginnend met `System.out.printf`) en voeg code toe zodat je uitvoer er als volgt uit komt te zien:

```
0
1
1
2
2
4
Mark verdient 10000,00 en heeft wel recht op een bonus.
Caroline verdient 4000,00 en heeft geen recht op een bonus.
```

3 Opdracht Klassen als attribuut-1 Bedrijf

1) Indien nodig kun je het project `OOP Bedrijf Encapsulation` van de DLO halen. Dit bevat de klassen `BedrijfLauncher` en `Persoon` zoals je die in de vorige opdracht gemaakt zou moeten hebben.
2) In deze opdracht breid je het bedrijf uit conform het volgende klassendiagram.



Een afdeling kan 0 of meer personen bevatten. Een persoon behoort tot 1 afdeling. Een persoon weet op welke afdeling deze zit, een afdeling weet niets van de personen af.

3) Voeg een klasse `Afdeling` toe aan de package `model`.
   a) De default waarden voor `afdelingsNaam` en `afdelingsPlaats` zijn beide “Onbekend”;
   b) Pas constructor chaining toe.
   c) Maak de getters en setters voor de twee attributen.

4) Pas de klasse `Persoon` aan:
   a) Voeg het attribuut `afdeling` van de klasse `Afdeling` toe.
   b) Pas de relevante constructors aan. Gebruik hierbij de default waarden van de klasse `Afdeling`.
   c) Maak een getter en setter voor het attribuut `afdeling`.

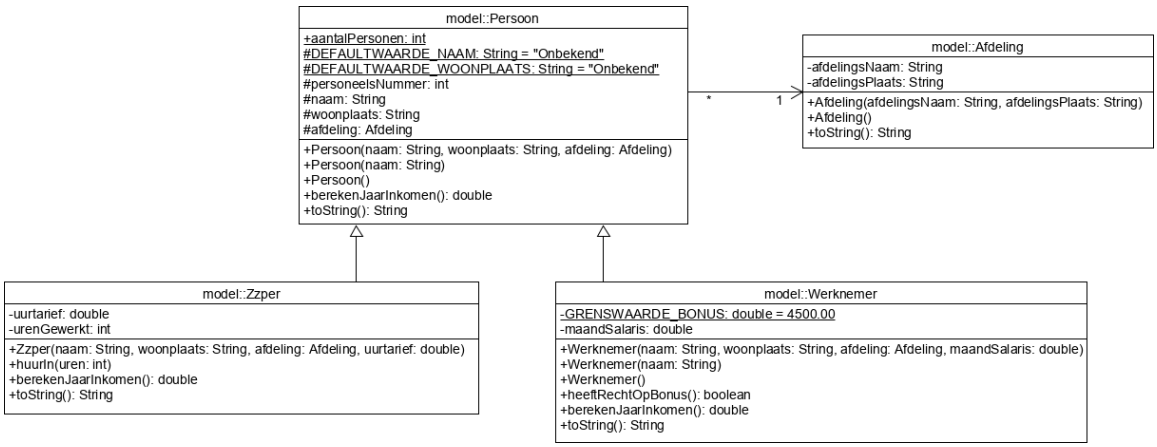
5) In je `main` methode:
   a) Verwijder eventueel de bestaande code of comment deze uit.
   b) Maak een array met de naam `afdelingen` aan met 4 afdelingen:
      i) De afdeling Uitvoering in Hilversum;
      ii) De afdeling Support in Amsterdam;
      iii) De afdeling Management in Almere;
      iv) De afdeling Documentatie in Gouda.
   c) Maak een persoon met de naam `baas` aan (naam: Mark, woonplaats: Den Haag, maandsalaris: 10000, afdeling: Management). Let op: gebruik voor het attribuut `afdeling` de zojuist aangemaakte array.
   d) Maak een persoon met de naam `medewerker` (naam: Caroline, woonplaats: Delft, maandsalaris: 4000, afdeling: Support);
   e) Maak een persoon met de naam `assistent` aan (naam: Klaas, de andere waarden zijn de defaultwaarden);

f) Voeg code toe zodat je uitvoer er als volgt uitziet (de *schuingedrukte* waarden hoeven in je uitvoer niet schuin gedrukt te worden, maar moet je uit de relevante objecten halen):

```
Het aantal personen in het bedrijf is 3
Mark werkt in Almere en woont in Den Haag
Caroline werkt op de afdeling Support en verdient 4000,00
Klaas werkt op de afdeling Onbekend en woont in Onbekend
```

4 Opdracht Overerving-1 Bedrijf

1) Indien nodig kun je het project `OOP Bedrijf Klasse als attribuut` van de DLO halen. Dit bevat de klassen `BedrijfLauncher`, `Persoon` en `Afdeling` zoals je die in de vorige opdracht gemaakt zou moeten hebben.
2) Het bedrijf kent twee soorten personen, werknemers en zzp-ers. In deze oefening worden dit subklassen van de klasse `Persoon`. Hiervoor moet ook de klasse `Persoon` aangepast worden. Hieronder is het klassendiagram van de nieuwe situatie.



3) Voeg de methode `toString()` toe aan de klasse `Afdeling`.
   a) Het gewenste resultaat van de methode is “afdeling *afdelingsNaam* te *afdelingsPlaats*” met de waarde van de attributen op de schuingedrukte plekken.
4) Pas de klasse `Persoon` aan conform het klassendiagram.
   a) Let goed op de visibility modifiers.
   b) Gebruik constanten om de default waarde voor de attributen `naam` en `woonplaats` te bepalen.
   c) Pas de constructors aan en blijf constructor chaining toepassen.
   d) De methode `berekenJaarInkomen()` geeft altijd 0 als resultaat.
   e) Vergeet niet om getters en setters van de verwijderde attributen ook te verwijderen.
   f) Voeg de methode `toString()` toe.
      i) Het gewenste resultaat van de methode is “*naam* woont in *woonplaats* en werkt op afdeling *afdelingsNaam* te *afdelingsPlaats*”.
      ii) Gebruik de `toString()` methode van de klasse `Afdeling`!
5) Maak de subklasse `Werknemer` aan conform het klassendiagram.
   a) Maak de constructors en pas constructor chaining toe. Maak gebruik van de al bekende default waarden.
   b) Maak gebruik van de constructors van de klasse `Persoon`.
   c) Maak de getters en de setters. Zorg weer dat er geen negatief maandsalaris kan worden gegeven en geef de bijbehorende melding.
   d) Het jaarinkomen is 12 maal het maandsalaris plus de eventuele bonus. De bonus is gelijk aan het maandsalaris.

e) Voeg de methode `toString()` toe.
   i) Het gewenste resultaat van de methode is "*naam* woont in *woonplaats* en werkt op afdeling *afdelingsNaam* te *afdelingsPlaats* en is een werknemer met/zonder recht op een bonus".
   ii) Gebruik de `toString()` methode van de klasse `Persoon`!
   iii) De tekst "met/zonder" wordt uiteraard bepaald door de methode `heeftRechtOpBonus()`.

6) Maak de subklasse `Zzper` aan conform het klassendiagram.
   a) Maak gebruik van de constructors van de klasse `Persoon`. De default waarde voor `urenGewerkt` is 0.
   b) Maak de getters en de setters.
   c) Het jaarinkomen is het uurtarief maal het aantal gewerkte uren.
   d) Het aantal uren dat een zzp-er heeft gewerkt wordt opgehoogd als deze wordt ingehuurd. Inhuren gaat via de methode `huurIn()`.
   e) Voeg de methode `toString()` toe.
      i) Het gewenste resultaat van de methode is "*naam* woont in *woonplaats* en werkt op afdeling *afdelingsNaam* te *afdelingsPlaats* en is een zzp-er met een uurtarief van *uurtarief*".
      ii) Gebruik de `toString()` methode van de klasse `Persoon`!
7) Pas je `main` methode als volgt aan:
   a) Maak een array met de naam `afdelingen` aan met 4 afdelingen:
      i) De afdeling Uitvoering in Hilversum;
      ii) De afdeling Support in Amsterdam;
      iii) De afdeling Management in Almere;
      iv) De afdeling Documentatie in Gouda.
   b) Maak een *werknemer* met de naam `baas` aan (naam: Mark, woonplaats: Den Haag, maandsalaris: 10000, afdeling: Management). Let op: gebruik voor het attribuut `afdeling` de zojuist aangemaakte array.
   c) Maak een *werknemer* met de naam `medewerker` (naam: Caroline, woonplaats: Delft, maandsalaris: 4000, afdeling: Support);
   d) Maak een *zzp-er* met de naam `assistent` aan (naam: Klaas, woonplaats: Diemen, uurtarief: 50,00, afdeling: Documentatie);
   e) Huur Klaas in voor 160 uur.
   f) Voeg code toe zodat je uitvoer er als volgt uitziet. Gebruik (impliciet) de methode `toString()`:

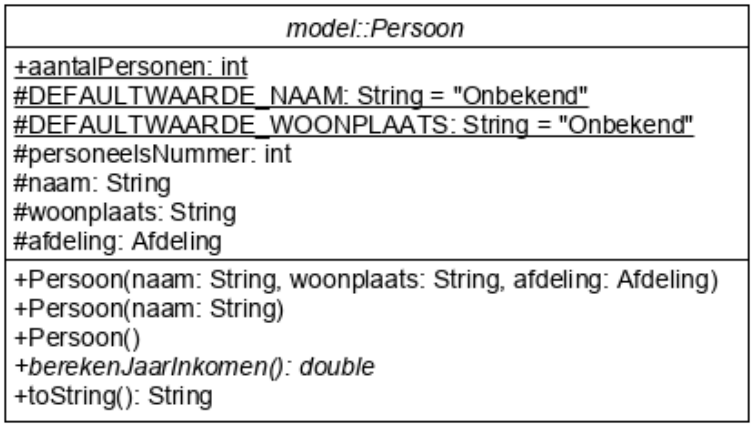
```
Het aantal personen in het bedrijf is 3
Mark woont in Den Haag en werkt op afdeling Management te Almere en is een
werknemer met recht op een bonus
Caroline woont in Delft en werkt op afdeling Support te Amsterdam en is
een werknemer zonder recht op een bonus
Klaas woont in Diemen en werkt op afdeling Documentatie te Gouda en is een
zzp-er met een uurtarief van 50.0
```

   g) Voeg code toe zodat het onderstaande onder de uitvoer hierboven komt:

```
Mark verdient 130000,00 per jaar
Caroline verdient 48000,00 per jaar
Klaas verdient 8000,00 per jaar
```

5 Opdracht Polymorfisme-Abstracte Klasse-1 Bedrijf

1) Indien nodig kun je het project `OOP Bedrijf Inheritance` van de DLO halen. Dit bevat de klassen zoals je die in de vorige opdracht gemaakt zou moeten hebben.
2) Maak klasse `Persoon` en de methode `berekenJaarInkomen()` abstract, conform het onderstaande klassendiagram.



3) Pas je `main` methode als volgt aan:
   a) Maak een array met de naam `afdelingen` aan met vier afdelingen:
      i) De afdeling Uitvoering in Hilversum;
      ii) De afdeling Support in Amsterdam;
      iii) De afdeling Management in Almere;
      iv) De afdeling Documentatie in Gouda.
   b) Maak vier personen aan, gebruik daarbij polymorfisme:
      i) De *werknemer* met de naam `baas` (naam: Mark, woonplaats: Den Haag, maandsalaris: 10000, afdeling: Management). Let op: gebruik hierbij de zojuist aangemaakte array van Afdelingen.
      ii) Een *werknemer* met de naam `medewerker` (naam: Caroline, woonplaats: Delft, maandsalaris: 4000, afdeling: Support);
      iii) Een *zzp-er* met de naam `assistent` aan (naam: Klaas, woonplaats: Diemen, uurtarief: 50,00, afdeling: Documentatie);
      iv) Een *zzp-er* met de naam `projectleider` aan (naam: Ronald, woonplaats: Zaandam, uurtarief: 80,00, afdeling: Uitvoering);
   c) Huur Klaas in voor 160 uur.
   d) Huur Ronald in voor 320 uur.
   e) Maak een array aan van de klasse `Persoon` met vier objecten.
   f) Voeg de werknemers baas en medewerker en de zzp-ers assistent en projectleider toe aan de array.
   g) Voeg onder je `main` methode een methode toe met de signatuur `toonJaarInkomen(Persoon persoon)`, dat de tekst "*Naam* `verdient` *jaarinkomen* `per jaar`" toont op het scherm.
   h) Roep – met behulp van een for-loop – de methode voor elk van de vier items in je array aan, zodat de uitvoer er als volgt uitziet:

```
Mark verdient 130000,00 per jaar
Caroline verdient 48000,00 per jaar
Klaas verdient 8000,00 per jaar
Ronald verdient 25600,00 per jaar
```

6 Opdracht ArrayList-1 Bedrijf

1) Indien nodig kun je het project `OOP Bedrijf Polymorfisme-Abstracte Klasse` van de DLO halen. Dit bevat de klassen zoals je die in de vorige opdracht gemaakt zou moeten hebben.
2) Behoud de array met vier afdelingen, zoals ze al in de `main` methode staan.
3) Maak een arraylist met objecten van de klasse `Persoon` met de naam `personen`.
4) Vul de arraylist als volgt:

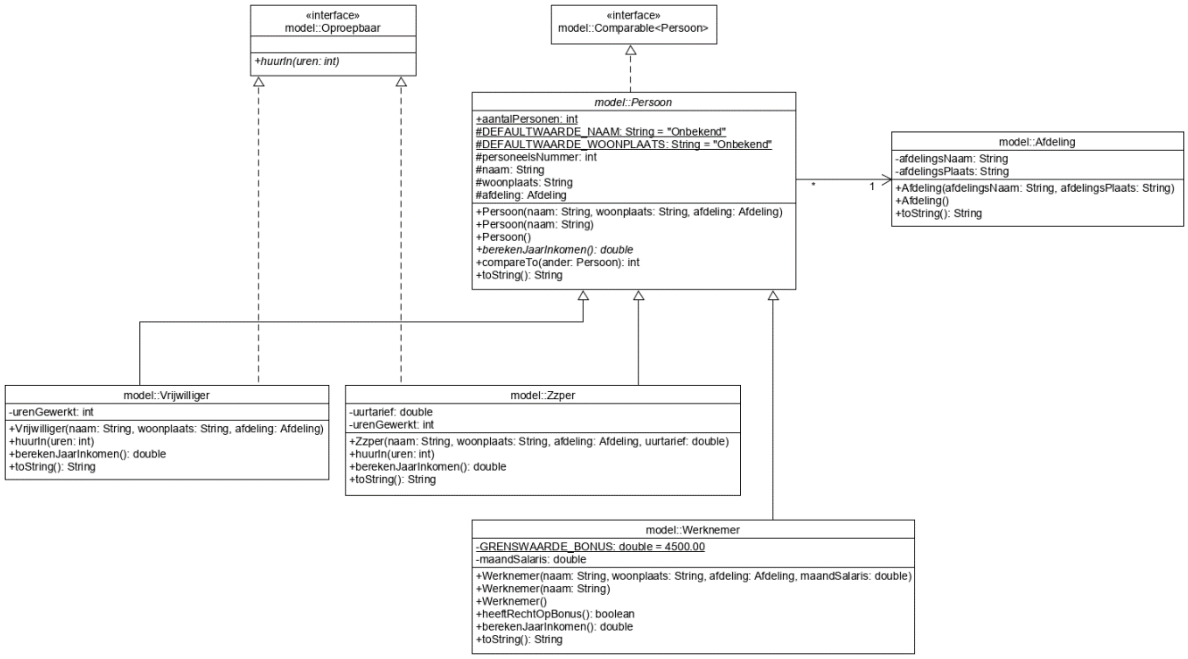
```
personen.add(new Werknemer("Mark", "Den Haag", afdelingen[2], 10000));
personen.add(new Werknemer("Angelique", "Rotterdam", afdelingen[2],
5000));
personen.add(new Werknemer("Caroline", "Delft", afdelingen[1], 4000));
personen.add(new Zzper("Klaas", "Diemen", afdelingen[3], 50.00));
personen.add(new Zzper("Ronald", "Zaandam", afdelingen[0], 80.00));
personen.add(new Zzper("Jannie", "Utrecht", afdelingen[0], 60.00));
personen.add(new Zzper("Anne", "Zwolle", afdelingen[0], 40.00));
```

5) Gebruik een for-each loop met `instanceof` en typecasting om alle zzp-ers in de arraylist voor 320 uur in te huren.
6) Gebruik een for-each loop en de al bestaande methode `toonJaarInkomen()` om de volgende uitvoer te krijgen:

```
Mark verdient 130000,00 per jaar
Angelique verdient 65000,00 per jaar
Caroline verdient 48000,00 per jaar
Klaas verdient 16000,00 per jaar
Ronald verdient 25600,00 per jaar
Jannie verdient 19200,00 per jaar
Anne verdient 12800,00 per jaar
```

7 Opdracht Interface-1 Bedrijf

In deze opdracht pas je het klassendiagram als volgt aan:



- *Een klasse `Vrijwilliger` wordt toegevoegd. Een vrijwilliger kan ook ingehuurd worden, zij het tegen een nultarief.*
- *De methode `huurIn(int uren)` is alleen noodzakelijk voor `Vrijwilliger` en `Zzper` en wordt in een interface geplaatst.*
- *Personen worden gesorteerd met behulp van de interface `Comparable`.*

1) Indien nodig kun je het project `OOP Bedrijf ArrayList` van de DLO halen. Dit bevat de klassen zoals je die in de vorige opdracht gemaakt zou moeten hebben.
2) Voor de klasse `Persoon`:
   a) Zorg dat de klasse de interface `Comparable<Persoon>` implementeert.
   b) Voeg de methode `compareTo()` toe, zodat gesorteerd kan worden op de naam van een persoon.
3) Maak een interface met de naam `Oproepbaar` aan binnen het package `model` met de methode `huurIn(int uren)`. De code van de interface ziet er als volgt uit:

```
package model;

public interface Oproepbaar {
    void huurIn(int uren);
}
```

4) Voeg de interface toe aan de klasse `Zzper` en override de methode `huurIn()`.
5) Voeg de klasse `Vrijwilliger` toe aan het package `model` conform het klassendiagram
   a) De methode `huurIn()` doet hetzelfde als bij de klasse `Zzper`.
   b) De methode `berekenJaarInkomen()` geeft altijd 0.
   c) De methode `toString()` geeft als resultaat "*naam* woont in *woonplaats* en werkt op afdeling *afdelingsNaam* te *afdelingsPlaats* en is een vrijwilliger".

6) In de `main` methode van de launcher:
   a) Gebruik de bestaande code, maar verander het datatype van de lijst personen in List<Persoon>.
   b) Voeg de volgende vier vrijwilligers toe aan de arraylist.

```
personen.add(new Vrijwilliger("Ambi", "Amsterdam", afdelingen[0]));
personen.add(new Vrijwilliger("Naledi", "Gaborone", afdelingen[1]));
personen.add(new Vrijwilliger("Ceren", "Istanboel", afdelingen[2]));
personen.add(new Vrijwilliger("Haining", "Shaoxing", afdelingen[3]));
```

   c) Huur alle vrijwilligers in voor 160 uur.
   d) Sorteer de arraylist `personen` op alfabetische volgorde.
   e) Voeg code toe om de volgende uitvoer te krijgen:

```
Ambi woont in Amsterdam en werkt op afdeling Uitvoering te Hilversum en is
een vrijwilliger
Ambi verdient 0,00 per jaar
Angelique woont in Rotterdam en werkt op afdeling Management te Almere en
is een werknemer met recht op een bonus
Angelique verdient 65000,00 per jaar
Anne woont in Zwolle en werkt op afdeling Uitvoering te Hilversum en is
een zzp-er met een uurtarief van 40.0
Anne verdient 12800,00 per jaar
Caroline woont in Delft en werkt op afdeling Support te Amsterdam en is
een werknemer zonder recht op een bonus
Caroline verdient 48000,00 per jaar
Ceren woont in Istanboel en werkt op afdeling Management te Almere en is
een vrijwilliger
Ceren verdient 0,00 per jaar
Haining woont in Shaoxing en werkt op afdeling Documentatie te Gouda en is
een vrijwilliger
Haining verdient 0,00 per jaar
Jannie woont in Utrecht en werkt op afdeling Uitvoering te Hilversum en is
een zzp-er met een uurtarief van 60.0
Jannie verdient 19200,00 per jaar
Klaas woont in Diemen en werkt op afdeling Documentatie te Gouda en is een
zzp-er met een uurtarief van 50.0
Klaas verdient 16000,00 per jaar
Mark woont in Den Haag en werkt op afdeling Management te Almere en is een
werknemer met recht op een bonus
Mark verdient 130000,00 per jaar
Naledi woont in Gaborone en werkt op afdeling Support te Amsterdam en is
een vrijwilliger
Naledi verdient 0,00 per jaar
Ronald woont in Zaandam en werkt op afdeling Uitvoering te Hilversum en is
een zzp-er met een uurtarief van 80.0
Ronald verdient 25600,00 per jaar
```