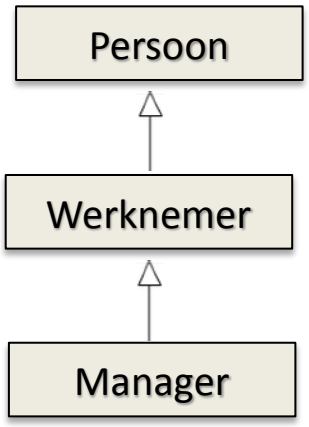
Oefeningen Hoofdstuk 10

Maak een Java Project aan met naam “H10”.

Per oefening maak je een aparte package. Voor oefening 1 geef je deze als naam “be.pxl.h10.oef1”.

Oefening1

Voor deze oefening maken we gebruik van de klasse Persoon uit het voorbeeld.



- Alle eigenschappen zijn enkel toegankelijk binnen de klasse zelf.
- Alle methoden zijn van overal toegankelijk.
- Vermijd om 2 keer dezelfde code te programmeren en hergebruik zoveel mogelijk code die je al hebt!

klasse Werknemer

1. Maak een klasse Werknemer die afgeleid is van Persoon.
   Om deze klasse in de huidige package te kunnen gebruiken plaats je bovenaan in je code (onder de naam van de package) volgende regel:
   ```
   import be.pxl.h10.voorbeeld.Persoon;
   ```
   Je kan deze code automatisch genereren via “Ctrl-Shft-O”.
2. Een Werknemer heeft buiten een naam en een voornaam een functie en een salaris.
3. Een Werknemer kan op 2 manier aangemaakt worden:
   a. door enkel naam en voornaam mee te geven: het salaris wordt ingesteld op 1800 en de functie op “algemeen bediende”.

b. door naam, voornaam, salaris en functie mee te geven.

Roep op een zinvolle manier vanuit de ene constuctor de andere op.

4. Voorzie getters en setters voor alle eigenschappen.
5. Herdefinieer de methode print(). Zorg dat alle gegevens afgedrukt worden.
6. Het minimum-salaris is 1000. Maak hiervoor een klasse-variabele aan en zorg ervoor dat een salaris nooit lager kan zijn dan het toegestane minimum. Zorg ook dat je dit minimum salaris kan opvragen.
7. Doe het nodige om te weten hoeveel Werknemer-objecten gecreëerd werden.

klasse Manager

1. Maak een klasse Manager die afgeleid is van de klasse Werknemer.
2. Buiten een naam, voornaam, functie en salaris heeft een manager een bonus.
3. Voorzie 2 constructors:
   a. door een waarde mee te geven voor naam, voornaam, salaris en functie. De bonus wordt ingesteld op 50.
   b. door een waarde mee te geven voor alle eigenschappen.
4. Voorzie een get- en set-methode om de bonus te kunnen opvragen en wijzigen.
5. Herdefinieer de methode getSalaris. Het salaris van de manager is het salaris van de werknemer + de bonus.
6. Doe het nodige om te weten hoeveel Manager-objecten gecreëerd werden.
7. Voorzie een methode getProcAandeelManagers() die het procentueel aandeel van de managers in het totaal aantal werknemers teruggeeft (met 2 decimalen).

Maak een klasse WerknemerApp met een main-methode waarin je 2 Werknemer-objecten en minstens 1 Manager-object aanmaakt en alle methoden uittest.

Let zeker op het volgende:

- worden de juiste default-waarden ingesteld wanneer gebruik gemaakt wordt van de constructors waarbij niet alle waarden voor de eigenschappen meegegeven worden?
- is het loon bij niemand lager dan het minimumloon?
- wordt bij de managers het juiste loon afgedrukt via de methode print()? (je mag de methode print() niet herdefiniëren in de klasse Manager)
- klopt het procentueel aandeel van de managers tov het totaal aantal werknemers?

Oefening2

klasse Reis

1. Van deze klasse mogen geen objecten kunnen gemaakt worden.
2. Eigenschappen: bestemming en prijs.
3. Een reis kost minstens 5 euro. Voorzie hiervoor een variabele en zorg ervoor dat deze minimumprijs steeds gewaarborgd is.
   De minimumprijs moet opgevraagd kunnen worden.
4. De eerste positie van een bestemming mag nooit een cijfer zijn. Als dit toch zo is, wordt dit verwijderd.
5. Voorzie een constructor waarbij enkel de bestemming meegegeven wordt als parameter (de prijs wordt ingesteld op de minimumprijs) en een constructor met 2 parameters. Roep op een zinvolle manier in de ene constructor de andere op.
6. Voorzie getters en setters voor alle eigenschappen.
7. Voorzie een methode print() die de gegevens van reis drukt als volgt:
   `Reis met bestemming Brussel kost 12.50 euro.`
   De prijs wordt steeds afgedrukt met 2 decimalen.

klasse TreinReis

1. Afgeleid van Reis.
2. Extra eigenschappen: nationaal (true voor binnenlandse ritten, false voor internationale ritten) en specificatie. Voor nationale ritten kan de specificatie enkel “IC”, “IR”, “L” of “P” zijn. Maak hiervoor een array aan met de juiste waarden en zorg voor de nodige controle. Indien een foute waarde opgegeven wordt, wordt de specificatie “IC”.
   Voor internationale ritten wordt de specificatie niet gecontroleerd (vb Thalys).
3. Voorzie volgende constructors:
   a. met 1 parameter (bestemming): het wordt een nationale rit van het type “IC”.
   b. met parameters voor bestemming, prijs, nationaal en specificatie.
4. Voorzie een getter en setter voor de specificatie.
5. Voorzie een methode print() die de gegevens van een treinreis drukt als volgt:

```
Reis met bestemming Brussel kost 12.50 euro.
Nationale treinreis (IC)
```

klasse VliegtuigReis

1. Afgeleid van Reis.
2. Extra eigenschappen: vluchtnummer (String). Het vluchtnummer kan een combinatie zijn van cijfers en letters, maar het eerste teken moet steeds de eerste letter van de bestemming zijn. Doe hierop een controle en voeg zo nodig vooraan de eerste letter van de bestemming toe.
3. De minimumprijs van een VliegtuigReis wordt vastgelegd op 25. Doe het nodige.
   Zorg er eveneens voor dat de minimumprijs opgevraagd kan worden.
4. Voorzie volgende constructors:
   a. met 1 parameter (bestemming): het vluchtnr wordt samengesteld als volgt: eerste letter van de bestemming, gevolgd door "999".
   b. met parameters voor bestemming, prijs en vluchtnummer.
5. Voorzie een getter en setter voor het vluchtnummer.
6. Voorzie een methode print() die de gegevens van een vliegtuigreis drukt als volgt:

```
Reis met bestemming Dublin kost 32.50 euro.
Vliegtuigreis (vluchtnr D526)
```

klasse GeboekteReis

Deze klasse dient om alle informatie betreffende 1 grotere reis (waarbij verschillende keren overgestapt moet worden) bij te houden.

1. Eigenschappen: naam (van de passagier), een array van reizen.
2. Voorzie een constructor met waarden voor de naam van de passagier en het aantal reizen dat in zijn boeking zit.
3. Voorzie een methode voegReisToe() om reizen toe te voegen. Zorg ervoor dat je niet meer reizen kan toevoegen dan dat er voorzien zijn voor de betreffende boeking.
4. Voorzie een methode print() die een afdruk maakt als volgt:

```
Reis van Pedro Salopetti
==> reis 1 Reis met bestemming Zaventem kost 5,00 euro
Nationale treinreis (IC)
==> reis 2 Reis met bestemming Madrid kost 65,00 euro
Vliegtuigreis (vluchtnr MX478)
==> reis 3 Reis met bestemming Barcelona kost 25,00 euro
Vliegtuigreis (vluchtnr B999)
==> reis 4 Reis met bestemming Rosas kost 10,75 euro
Internationale treinreis (local train)
deze passagier moet 3 keer overstappen
```

Maak een klasse ReisApp waarin je alles uittest.
Ga na in de gemaakte klassen of volgende begrippen aan bod zijn gekomen : data hiding, inheritance, constructor overloading, method overriding en polymorfisme.
Schrijf in commentaar in je programma waar je deze begrippen gebruikt hebt.

Oefening3

Voor deze oefening maken we gebruik van de klasse Persoon uit het voorbeeld.

klasse Sporter

1. Maak een klasse Sporter die afgeleid is van Persoon.
   Om deze klasse in de huidige package te kunnen gebruiken plaats je bovenaan in je code (onder de naam van de package) volgende regel:
   ```
   import be.pxl.h10.voorbeeld.Persoon;
   ```
   Je kan deze code automatisch genereren via "Ctrl-Shft-O".
2. Buiten een naam heeft een sporter een omschrijving van de beoefende sport.
3. Zorg ervoor dat een Sporter-object op 2 manieren kan gecreëerd worden:
   - met opgave van de naam, voornaam van de sporter. De sport is dan "onbepaald'.
   - met opgave van de naam, voornaam samen met de beoefende sport.
4. Voorzie een teller, die het aantal aangemaakte sporters bijhoudt. De waarde van de teller moet opgevraagd kunnen worden.
5. Voorzie een methode om de sport te wijzigen.
6. Voorzie een methode print() die alle gegevens van een sporter op het scherm afdrukt.
   Voorbeeld van een afdruk:
   ```
   An Peeters
   zwemmen
   ```

klasse Voetballer

1. Maak een klasse Voetballer. Een Voetballer is een Sporter met als bijkomende eigenschappen club (vb. KSKHasselt) en opstelling(bvb. aanvaller).
   Van deze klasse mogen geen subklassen gemaakt kunnen worden.
2. De opstelling kan slechts 1 van de volgende waarden hebben: middenvelder, aanvaller, verdediger, onbepaald. Maak hiervoor een array aan die je gebruikt om te controleren. De opstelling wordt steeds in kleine letters bijgehouden. Als een foute waarde wordt ingesteld, wordt ze aangepast naar "onbepaald".
3. De creatie van een Sporter-object kan op 2 manieren:

   a. door opgave van de naam en voornaam (de club en opstelling krijgen dan de waarde "onbepaald", de omschrijving van de sport is uiteraard "voetbal")
   b. door opgave van naam, voornaam, club en opstelling
4. Voorzie een getter en een setter voor de eigenschappen opstelling, club.
5. Voorzie de mogelijkheid om de array met juiste waarden voor de opstelling te retourneren.
6. Voorzie een methode print() die alle gegevens van een voetballer op het scherm afdrukt. Voorbeeld van een afdruk:

```
Dirk Horstens
voetbal
club: KSKHasselt opstelling: verdediger
```

Maak een klasse SportApp aan waarin je het volgende doet:

1. Maak een array waarin je minstens 10 sporters (waaronder voetballers) opneemt.
2. Voorzie een afdruk als volgt (eerst de sporters die geen voetballer zijn, vervolgens de voetballers gegroepeerd per opstelling):

```
Overzicht sporters (behalve voetbal
naam: An Bex
zwemmen
naam: Els Nilis
volleybal
naam: Sofie Baerts
zwemmen
naam: Leen Adams
atletiek
naam: Miet Loos
zwemmen

overzicht voetballers volgens opstelling
*** aanvaller ***
naam: Niels Vos
voetbal
club :KSKHasselt opstelling: aanvaller
*** verdediger ***
naam: Joren Vos
voetbal
club :KSKHasselt opstelling: verdediger
naam: Joni Dirix
voetbal
club :KSKHasselt opstelling: verdediger
*** middenvelder ***
naam: Jelle Maes
voetbal
club :KSKHasselt opstelling: middenvelder
*** onbepaald ***
naam: Martijn Hox
voetbal
club :KSKHasselt opstelling: onbepaald
```