Oefententamen OOP Make IT Work

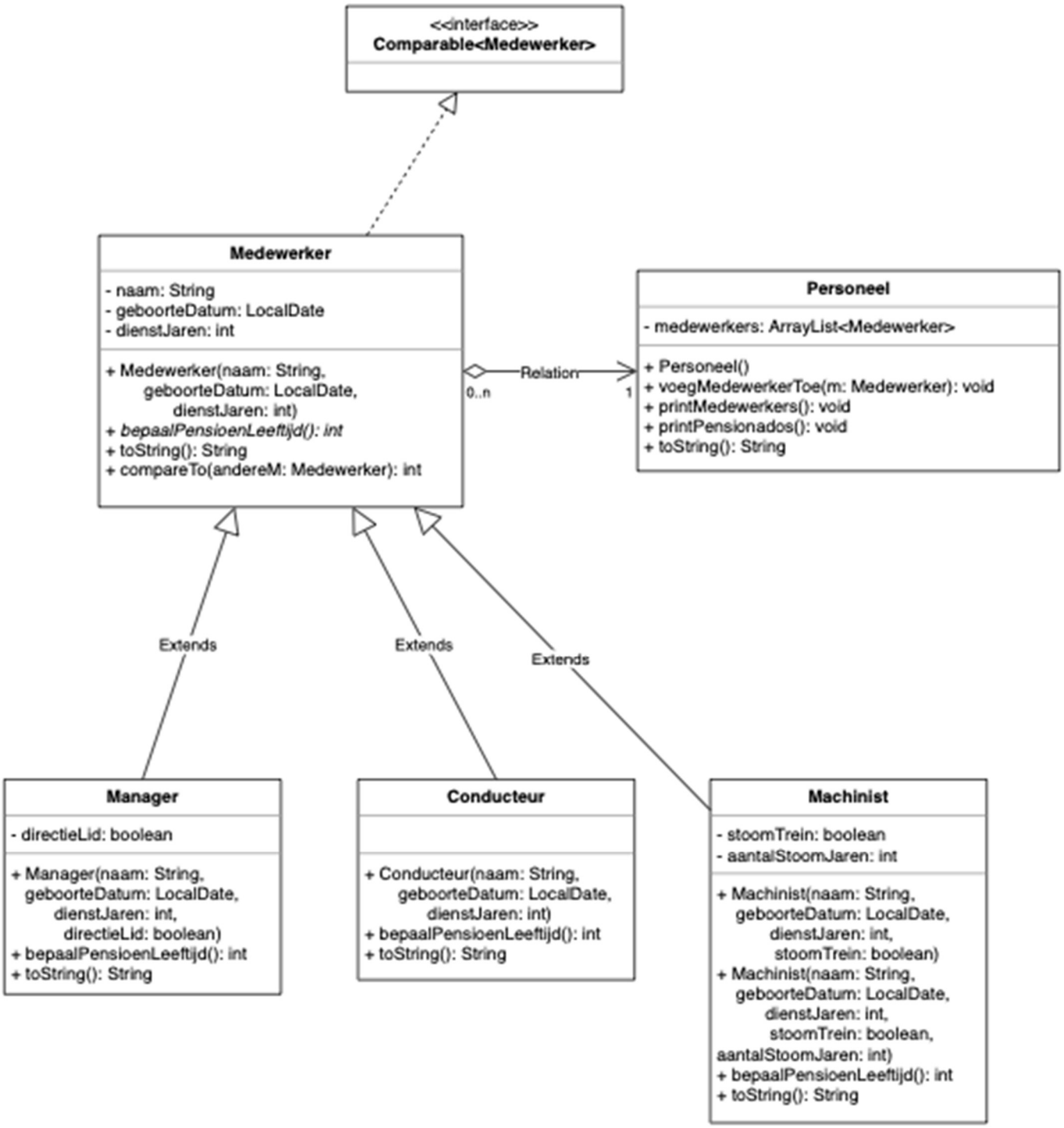
NS Pensioen

Opmerkingen

1. Je krijgt voor deze opgave een startproject genaamd NSPensioen.
2. Je krijgt een sql-script om een database te maken met een user.
3. Upload na afloop een zip van je hele project naar Blackboard.
4. Deze toets bestaat uit 8 pagina's en is dubbelzijdig afgedrukt.
5. Het gebruik van het internet is toegestaan.
6. Daarnaast is toegestaan het gebruik van het boek, je eigen aantekeningen en al het materiaal op je eigen laptop (inclusief uitwerkingen van practicumopgaven).
7. Als je het tentamen inlevert checkt de docent of de upload in Blackboard is gelukt. Verlaat het lokaal niet voordat die check is uitgevoerd en je toestemming hebt gekregen!

Klassendiagram NS Pensioen

De pensioenleeftijd van de medewerkers van de Nederlandse Spoorwegen hangt af van hun functie. Er moet software ontwikkeld worden om bij te houden wie wanneer met pensioen gaat. Hiervoor is het volgende class diagram opgezet voor de klassen in de package model.



Algemene aanwijzingen

- De `PensioenLauncher`–class is al voor je gemaakt en bevat veel testcode. Maak hier gebruik van! De voorbeeldoutput is ook gebaseerd op deze testcode.
- In het class diagram zijn geen getters en setters opgenomen. Voeg zelf de getter(s) en setter(s) toe die je nodig denkt te hebben. Je mag niet standaard alle getters en setters opnemen, maar alleen degene die je nodig hebt.
- In het class diagram staan geen klassen die nodig zijn voor het maken van een connectie met een database en ook geen DAO-klassen. Een DBaccess klasse en een AbstractDAO zijn wel al aanwezig in je project.
- Zorg dat je `toString()` methodes dezelfde output genereren als in de outputvoorbeelden.
- De interface `Comparable` bestaat natuurlijk al in Java, die hoef je dus niet zelf te maken.
- Als je zelf nog zaken wil toevoegen (omdat je denkt dat ze nodig zijn) die niet in het class diagram staan, geef dat dan duidelijk met commentaar aan in de code

Deel I: Bouw klassenstructuur

Stap 0: Verander de welkomstboodschap

Verander de boodschap zodat je eigen naam, je studentnummer en je klas verschijnt.

Run de applicatie. De output moet er als volgt uit zien.

```
Welkom bij de Pensioen app van de NS, geschreven door: <Naam>
```

Stap 1: Abstracte klasse Medewerker (15 punten)

Maak de abstracte klasse `Medewerker` volgens het class diagram. Houd hierbij rekening met de onderstaande eisen:

- LET OP: Het implementeren van de interface `Comparable` en de methode `compareTo()` hoef je pas in stap 6 te doen.
- De methode `bepaalPensioenLeeftijd()` is abstract.
- Override de methode `toString()`, zodat deze de naam, de leeftijd en het aantal dienstjaren, en de pensioenleeftijd teruggeeft in één string. Dit kun je nog niet testen!
    - Tip: gebruik `Period.between(geboorteDatum, LocalDate.now()).getYears()` om de leeftijd te bepalen.
    - Gebruik voor de pensioenleeftijd de methode `bepaalPensioenLeeftijd()`. De string moet er als volgt uitzien (voorbeeld):

```
Piet Paaltjens
Leeftijd: ? jaar met ? dienstjaren.
Pensioenleeftijd: ? jaar.
```

Stap 2: Subklasse Machinist (15 punten)

Maak de subklasse `Machinist` volgens het class diagram. Houd hierbij rekening met de volgende eisen:

- Implementeer de constructors op zo'n manier dat je dubbele code zoveel mogelijk voorkomt.
- Als voor het attribuut `aantalStoomJaren` geen waarde aan de constructor wordt meegegeven, dan moet dit attribuut de waarde `0` krijgen.
- Override de methode `bepaalPensioenLeeftijd()` en implementeer deze als volgt:
    - Als `stoomTrein` op `false` staat, dan is de pensioenleeftijd `67` jaar.
    - Als `stoomTrein` op `true` staat, dan is de pensioenleeftijd afhankelijk van het aantal dienstjaren dat is doorgebracht op een stoomtrein:
        - De basis pensioenleeftijd is `65` jaar.
        - Per `5` jaar op een stoomtrein gaat er `1` jaar extra af van de pensioenleeftijd.
- Override de methode `toString()` en implementeer deze zoals het voorbeeld hieronder. Voorkom dubbele code en hergebruik zoveel mogelijk.
    - Tip: maak gebruik van een `String.replace()`.

Gebruik `testStap2()` om je implementatie te testen. De output moet er als volgt uitzien.

```
--- Test stap 2 ---

Piet Paaltjens, machinist
Leeftijd: 70 jaar met 34 dienstjaren waarvan 7 jaar op een stoomtrein.
Pensioenleeftijd: 64 jaar.

Fred Teeven, machinist
Leeftijd: 60 jaar met 2 dienstjaren waarvan 0 jaar op een stoomtrein.
Pensioenleeftijd: 67 jaar.

Rob de Nijs, machinist
Leeftijd: 76 jaar met 27 dienstjaren waarvan 12 jaar op een stoomtrein.
Pensioenleeftijd: 63 jaar.
```

Stap 3: Subklasse Conducteur (5 punten)

Maak de subklasse `Conducteur` volgens het class diagram. Houd hierbij rekening met de volgende eisen:

- Override de methode `bepaalPensioenLeeftijd()` en implementeer deze als volgt:
    - Als het aantal `dienstJaren` korter is dan `25`:
        - dan is de pensioenleeftijd `68` jaar.
    - Als het aantal `dienstJaren` langer is dan `25`:
        - dan is de pensioenleeftijd `67` jaar.
- Override de methode `toString()` en implementeer deze zoals het voorbeeld hieronder.

Gebruik `testStap3()` om je implementatie te testen. De output moet er als volgt uitzien.

```
--- Test stap 3 —

Kees van Deuren, conducteur
Leeftijd: 45 jaar met 12 dienstjaren
Pensioenleeftijd: 68 jaar.

Jan van Splinteren, conducteur
Leeftijd: 60 jaar met 30 dienstjaren.
Pensioenleeftijd: 67 jaar.

Willem Vermeend, conducteur
Leeftijd: 70 jaar met 10 dienstjaren.
Pensioenleeftijd: 68 jaar.
```

Stap 4: Klasse Manager (10 punten)

Maak de subklasse `Manager` volgens het class diagram. Houd hierbij rekening met de volgende eisen:

- Override de methode `bepaalPensioenLeeftijd()` en implementeer deze als volgt:
  - Als `directielid` op `true` staat:
    - dan is de pensioenleeftijd `70` jaar.
  - Als `directielid` op `false` staat:
    - dan is de pensioenleeftijd `65` jaar.
- Override de methode `toString()` en implementeer deze zoals het voorbeeld hieronder.

Gebruik `testStap4()` om je implementatie te testen. De output moet er als volgt uitzien.

```
--- Test stap 4 ---

Jan-Willem van Deuren, manager (directielid)
Leeftijd: 45 jaar met 12 dienstjaren
Pensioenleeftijd: 70 jaar.

Berend Hoogervorst, manager
Leeftijd: 60 jaar met 30 dienstjaren.
Pensioenleeftijd: 65 jaar.

Willemijn van Driesen, manager (directielid)
Leeftijd: 57 jaar met 10 dienstjaren.
Pensioenleeftijd: 70 jaar.
```

Stap 5: Klasse Personeel (10 punten)

Maak de klasse `Personeel` volgens het class diagram.

- LET OP: De methodes printMedewerkers(), printPensionados() en toString() zijn pas relevant in Deel II en hoeven nu nog niet geïmplementeerd te worden.

Gebruik `testStap5()` om je implementatie te testen. De output moet er als volgt uitzien.

```
--- Test stap 5 ---
Alle medewerkers zijn aan het personeel toegevoegd!
```

Deel II: Gebruik klassen, bestand en database

Stap 6: Sorteer en print alle medewerkers (10 punten)

De medewerkers van de NS gaan vanwege de nieuwe CAO na de staking op verschillende leeftijden met pensioen.
Voer de volgende stappen uit:

- Zorg dat de klasse `Medewerker` de interface `Comparable` implementeert (zie class diagram) en override vervolgens de methode `compareTo()`.
- Override de methode `toString()` in de klasse `Personeel` en implementeer deze zoals de output hieronder.
    - Sorteer de medewerkers op pensioenleeftijd.

Gebruik `testStap6()` om je implementatie te testen. De output moet er ongeveer als volgt uitzien (hieronder staan 6 van de 9 medewerkers, je output moet ze alle 9 tonen).

```
--- Test stap 6 —

Rob de Nijs, machinist
Leeftijd: 76 jaar met 27 dienstjaren waarvan 12 jaar op een stoomtrein.
Pensioenleeftijd: 63 jaar.

Berend Hoogervorst, manager
Leeftijd: 60 jaar met 30 dienstjaren.
Pensioenleeftijd: 65 jaar.

Jan van Splinteren, conducteur
Leeftijd: 60 jaar met 30 dienstjaren.
Pensioenleeftijd: 67 jaar.

Fred Teeven, machinist
Leeftijd: 60 jaar met 2 dienstjaren waarvan 0 jaar op een stoomtrein.
Pensioenleeftijd: 67 jaar.

Willem Vermeend, conducteur
Leeftijd: 70 jaar met 10 dienstjaren.
Pensioenleeftijd: 68 jaar.

Willemijn van Driesen, manager (directielid)
Leeftijd: 57 jaar met 10 dienstjaren.
Pensioenleeftijd: 70 jaar.
```

Stap 7: Print alle medewerkers die met pensioen zijn (5 punten)

Soms wil de NS een kaartje sturen naar alle medewerkers die met pensioen zijn.

- Implementeer de methode `printPensionados()` in de klasse `Personeel`.

Gebruik `testStap7()` om je implementatie te testen. De output moet er als volgt uitzien (de volgorde van de medewerkers in de output kan verschillen):

```
--- Test stap 7 —

Rob de Nijs, machinist
Leeftijd: 76 jaar met 27 dienstjaren waarvan 12 jaar op een stoomtrein.
Pensioenleeftijd: 63 jaar.

Piet Paaltjens, machinist
Leeftijd: 70 jaar met 34 dienstjaren waarvan 7 jaar op een stoomtrein
Pensioenleeftijd: 64 jaar

Willem Vermeend, conducteur
Leeftijd: 70 jaar met 10 dienstjaren.
Pensioenleeftijd: 68 jaar.
```

Stap 8: Print alle medewerkers naar een bestand (10 punten)

De lijst met alle medewerkers die nog in dienst zijn moet in een bestand gezet worden, zodat de lijst gemaild en uitgeprint kan worden.

- Implementeer de methode `printMedewerkers()` in klasse `Personeel`.
- Zorg dat in de methode een tekst bestand gemaakt wordt met de naam `medewerkers.txt` waarin de medewerkers worden weggeschreven. Sla dit bestand op in de folder ‘resources’ van je project (met pad ‘src/main/resources/’).

Gebruik `testStap8()` om je implementatie te testen. De output in het tekstbestand moet er als volgt uitzien (de volgorde van de medewerkers in de output kan verschillen):

```
Berend Hoogervorst, manager
Leeftijd: 60 jaar met 30 dienstjaren
Pensioenleeftijd: 65 jaar

Fred Teeven, machinist
Leeftijd: 60 jaar met 2 dienstjaren
Pensioenleeftijd: 67 jaar

Jan van Splinteren, conducteur
Leeftijd: 60 jaar met 30 dienstjaren
Pensioenleeftijd: 67 jaar

Kees van Deuren, conducteur
Leeftijd: 45 jaar met 12 dienstjaren
Pensioenleeftijd: 68 jaar

Jan-Willem van Deuren, manager (directielid)
Leeftijd: 45 jaar met 12 dienstjaren
Pensioenleeftijd: 70 jaar

Willemijn van Driesen, manager (directielid)
Leeftijd: 56 jaar met 10 dienstjaren
Pensioenleeftijd: 70 jaar
```

Stap 9: Haal een lijst met medewerkers uit de database (10 punten)

Je hebt een sql-script gekregen waarin een database genaamd 'Pensioen' wordt gemaakt met de tabel 'Machinist'. Let wel: er is geen subtypering toegepast en alle attributen van de Java klasse 'Medewerker' zijn er ook in opgenomen. In het script worden vijf records aangemaakt en ook een user met password.

In je project staat al een DBaccess klasse met de juiste properties om een connectie te maken met de database Pensioen.

In de main methode staat code voor stap 9 die je moet uncommenten. Je kunt dan testen of je connectie werkt.

Maak een MachinistDAO klasse die het mogelijk maakt om alle machinisten die meer ervaring op de stoomtrein hebben dan een bepaald aantal jaren uit de database te halen. De tabel Machinist heeft een primary key medewerkernummer. De klasse Machinist heeft geen (overgeërfde) eigenschap medewerkernummer en die hoef je niet toe te voegen in de klasse. Je hoeft de medewerkernummer daarom niet op te halen uit de tabel, want je hebt de medewerkernummer niet nodig voor het instantiëren van Machinist objecten.

- Maak de klasse MachinistDAO met methode `getMachinistMetMeerStoomtreinErvaring(int jaren)`. In deze methode moeten de juiste records uit de tabel worden gehaald en als `ArrayList<Machinist>` worden gereturned.
- Let op: om een datumveld uit Workbench om te zetten in een LocalDate kun je de methode `LocalDate.parse(resultSet.getString("geboortedatum"))` gebruiken.
- Voeg in de main methode code toe om de array af te drukken. De output moet er als volgt uitzien (de volgorde van de medewerkers in de output kan verschillen):

```
--- Test stap 9 ---

Rob de Nijs, machinist
Leeftijd: 76 jaar met 27 dienstjaren waarvan 12 jaar op een stoomtrein
Pensioenleeftijd: 63 jaar

Lee Towers, machinist
Leeftijd: 73 jaar met 32 dienstjaren waarvan 25 jaar op een stoomtrein
Pensioenleeftijd: 60 jaar
```

Let op de code conventions (10 punten)

- Zorg dat je naam en het doel bij elke class 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 numbers (ICC#5).
- Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).
- Vermijd dode code (ICC #8).
- Denk aan encapsulation (ICC #9).