MAKE IT WORK

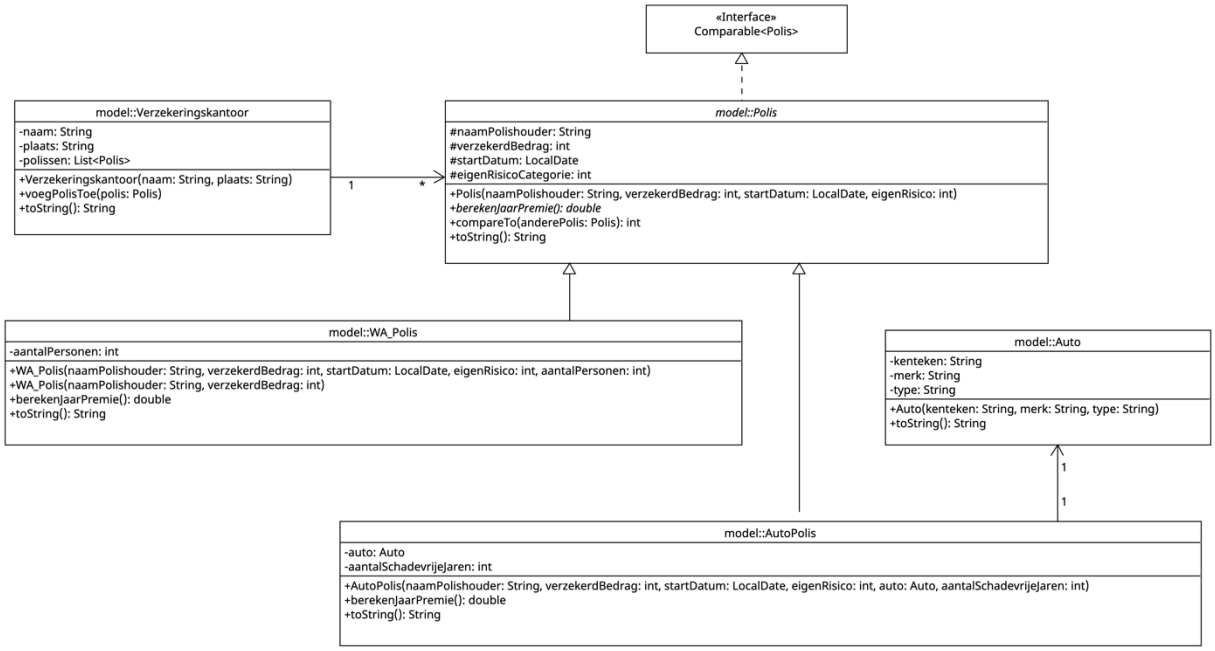
Verzekeringsmaatschappij OOP

Inleiding

Verzekeringsmaatschappij Snits gaat digitaliseren. Er moet software ontwikkeld worden om gegevens betreffende de polissen en schadeclaims vast te leggen. Het doel van deze opdracht is om de klassen te bouwen, die nodig zijn voor de applicatie.

Beschrijving

Het onderstaande klassendiagram moet worden gebouwd. Het is een deel van het database ontwerp in opdracht 2. Uiteraard moet dubbele code voorkomen worden.



Algemene aanwijzingen

- Haal het startproject AssessmentVerzekeringskantoor op van de DLO. In de package `controller` staat een Launcher klasse een `main` methode en in de package `model` staat al de klasse `Auto`.
- *De Launcher kun je gebruiken om je eigen code te testen. Zie het eind van dit document voor de code die je **moet** laten zien in de Launcher. De overige code hoef je niet te laten zien, maar laat goed werkende code in de Launcher staan. We raden je aan om regelmatig je code te testen!*
- In de *Launcher* staat code om een lijst van auto's te maken. Je kunt auto's uit de lijst gebruiken om te testen en is er ook een opdracht die van de lijst gebruikt maakt.
- In het klassendiagram 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 degenen die je nodig hebt.
- Zorg dat je `toString()` methodes dezelfde output genereren als in de outputvoorbeelden.
- Merk op dat datums als datatype de `LocalDate` klasse gebruiken. Je kunt dan handig gebruik maken van de methodes van een `LocalDate` object zoals `now()` voor de huidige datum.

- 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 klassendiagram staan, geef dat dan duidelijk met commentaar aan in de code.

Abstracte klasse Polis

1. De datum wordt default in Amerikaanse opmaak (yyyy-mm-dd) weergegeven. Je kunt hiervoor de `toString()` methode van de `LocalDate` klasse gebruiken.
2. Er kan gekozen worden voor een eigen risico categorie 1 (Laag), 2 (Midden) of 3 (Hoog). De hoogte van het eigen risico wordt per polis type bepaald en speelt in deze opdracht geen rol. Geef een foutmelding als een eigen risico categorie wordt ingevoerd dat anders is dan een van deze waarden en zet de eigen risico categorie van deze polis op 1.
3. Zorg dat de methode `compareTo()` de polissen op datum sorteert. Maak hierbij gebruik van de `compareTo()` methode van de `LocalDate` klasse.
4. Zorg dat de `toString()` methode bijvoorbeeld de volgende uitvoer geeft:

```
Polis op naam van O.B. Bommel
    Verzekerd bedrag: 10000 euro
    Startdatum: 2020-07-31
    Eigen risico categorie: 2
```

Subklasse WA_Polis

1. Implementeer de constructors op zo'n manier dat je dubbele code zoveel mogelijk voorkomt.
2. Als alleen de naam van de polishouder en het verzekerde bedrag worden meegegeven dan moet de startdatum op de huidige datum gezet worden, het eigen risico op (categorie) 1 en het aantal personen op 1. Voor de huidige datum kun je handig gebruik maken van de methode `now()` van de `LocalDate` klasse.
3. Override de methode `berekenJaarPremie()` en implementeer deze als volgt:
   a. De jaarpremie van een WA polis is een percentage van het verzekerde bedrag. Dat is het basisbedrag. Dit percentage is afhankelijk van het gekozen eigen risico conform de onderstaande tabel:

Categorie	Percentage
1	0,06%
2	0,05%
3	0,04%

   b. De jaarpremie is het basisbedrag vermenigvuldigd met het aantal personen.

4. Override de methode toString() en implementeer deze zoals het voorbeeld hieronder. Voorkom dubbele code en hergebruik zoveel mogelijk.
5. Zorg dat de `toString()` methode bijvoorbeeld de volgende uitvoer geeft (het verzekerd bedrag is 1 miljoen euro):

```
Polis op naam van Tom Poes
    Verzekerd bedrag: 1000000 euro
    Startdatum: 2021-09-01
    Eigen risico categorie: 1
    Aantal personen: 4
    Jaarpremie: 2400,00 euro
```

MAKE IT WORK

Subklasse Autopolis

1. Het aantal schadevrije jaren is minimaal 0 en maximaal 40. Geef de relevante foutmelding als het aantal schadevrije jaren hier niet aan voldoet en zet dan het aantal schadevrije jaren op 0. (Zie opdracht Programming: `Het aantal schadevrije jaren moet minimaal 0 zijn!` respectievelijk `Het aantal schadevrije jaren mag maximaal 40 zijn!`)
2. Override de methode `berekenJaarPremie()` en implementeer deze als volgt:
   a. De jaarpremie van een autopolis is een percentage van het verzekerde bedrag. Dat is het basisbedrag. Dit percentage is afhankelijk van het gekozen eigen risico conform de onderstaande tabel:

Categorie	Percentage
1	10%
2	9%
3	8%

   b. Op basis van het aantal schadevrije jaren wordt een korting verleend. Deze korting is 5% op de jaarpremie per schadevrij jaar, met een maximum van 70%. Je betaalt dus altijd minimaal 30% van het basisbedrag.
3. Zorg dat de `toString()` methode bijvoorbeeld de volgende uitvoer geeft:

```
Polis op naam van Wammes Waggel
    Verzekerd bedrag: 10000 euro
    Startdatum: 2021-07-31
    Eigen risico categorie: 2
    Auto: Volkswagen Kever met kenteken 74-OBB-3
    Aantal schadevrije jaren: 8
    Jaarpremie: 540,00 euro
```

Klasse Verzekeringskantoor

1. Een verzekeringskantoor beheert polissen.
2. Met de methode voegPolisToe() kun je polissen aan een verzekeringskantoor toevoegen.
3. Zorg dat de `toString()` methode bijvoorbeeld de volgende uitvoer geeft:

```
Polissen op Verzekeringskantoor Bommelstein te Rommeldam:
Polis op naam van Tom Poes
    Verzekerd bedrag: 1000000 euro
    Startdatum: 2021-09-01
    Eigen risico categorie: 1
    Aantal personen: 4
    Jaarpremie: 2400,00 euro

Polis op naam van Wammes Waggel
    Verzekerd bedrag: 10000 euro
    Startdatum: 2021-07-31
    Eigen risico categorie: 2
    Auto: Volkswagen Kever met kenteken 74-OBB-3
    Aantal schadevrije jaren: 8
    Jaarpremie: 540,00 euro
```

4. Pas de `toString()` methode aan, zodat deze de polissen op volgorde van startdatum toont.

MAKE IT WORK

Launcher

Zorg dat deze code in de Launcher aanwezig is:

1. Geef een welkomstboodschap met je naam en studentnummer
2. Laat zien dat een polis een foutmelding geeft als de ingevoerde eigen risico categorie onjuist is.
3. Laat zien dat een autopolis een foutmelding geeft als het aantal schadevrije jaren te laag is.
4. Laat zien dat een autopolis een foutmelding geeft als het aantal schadevrije jaren te hoog is.
5. Laat zien dat je polissen kunt toevoegen aan een verzekeringskantoor, voeg tenminste drie polissen toe.
6. Laat zien dat alle `toString()` methodes goed geïmplementeerd zijn.
7. Voeg een methode toe aan de Launcher die de auto's van een bepaald type print. Geef de methode de signature `public static void toonTypeAutos(List<Auto> autoLijst, String type).` Roep de methode aan om van de autoLijst alle 'electrische' auto's te tonen.

Let op de code conventions

1. Zorg dat je naam en het doel bij elke klasse bovenin staan (ICC #1).
2. Gebruik de juiste inspringing (indentation) bij de lay-out (ICC #2).
3. Let op juist gebruik hoofdletters en kleine letters (ICC #3).
4. Gebruik goede namen (ICC #4).
5. Vermijd magic numbers, gebruik dus constanten (ICC#5).
6. Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).
7. Vermijd dode code (ICC #8).
8. Denk aan encapsulation (ICC #9).