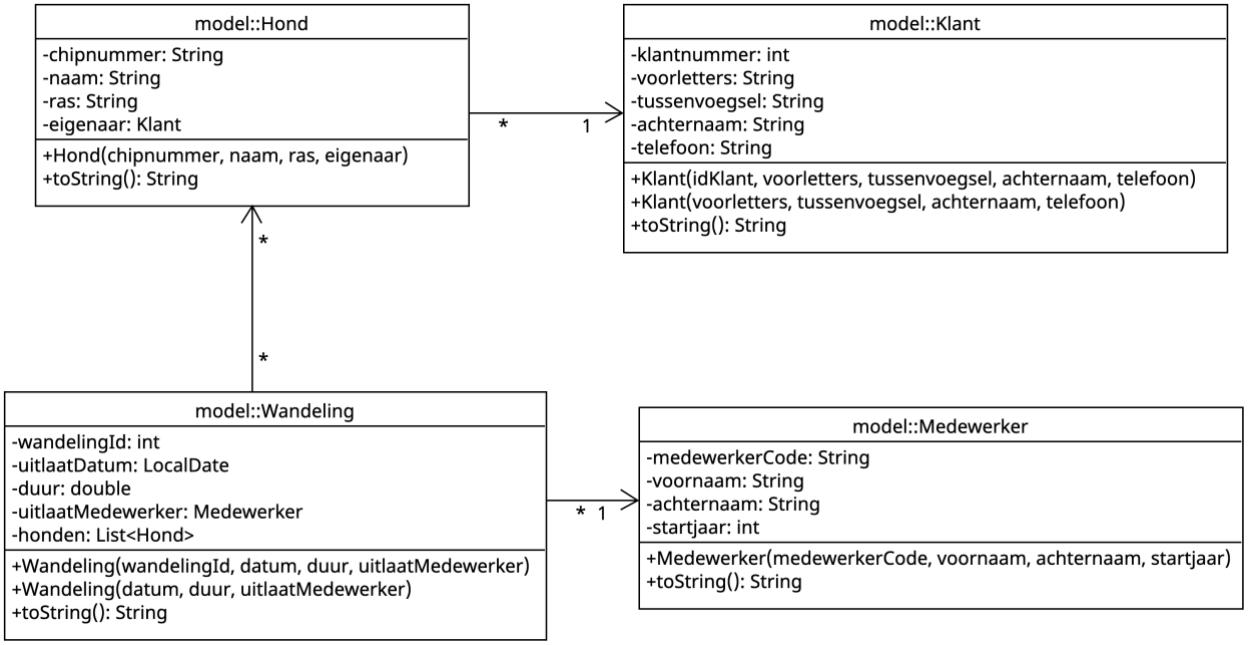
MAKE IT WORK

Opdracht JDBC Hondendienst

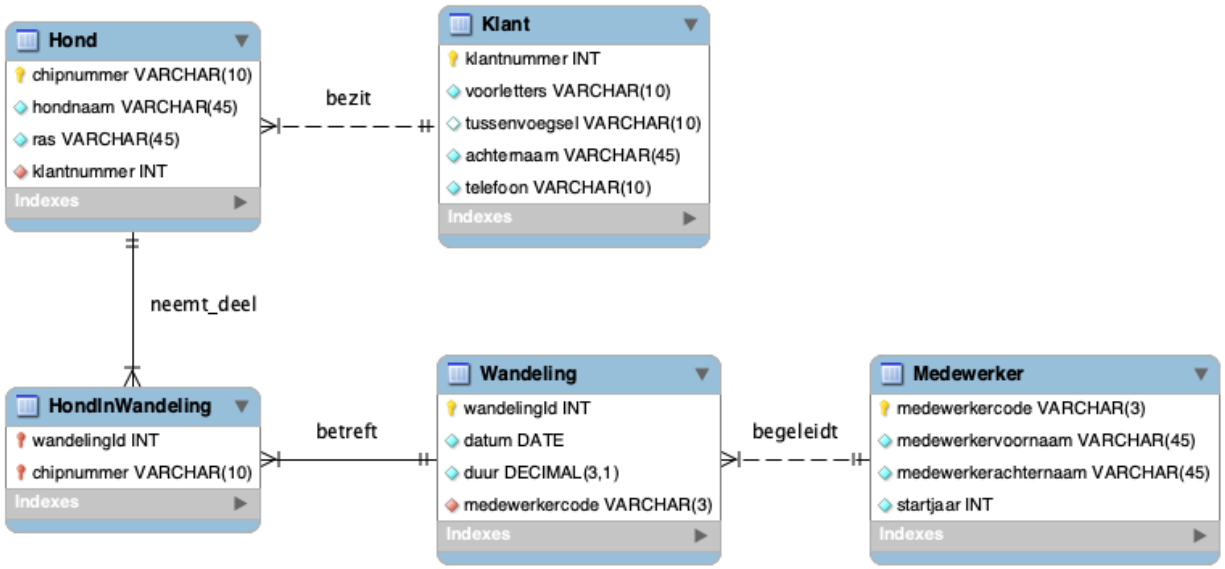
Inleiding

Hieronder zie je het klassendiagram ten behoeve van een applicatie met gegevens van een honden uitlaatdienst.



Merk op dat een medewerker bij een wandeling meerdere honden tegelijk meeneemt. Een hond kan ook in meerdere wandelingen (op verschillende dagen) meegenomen worden.
Een honden attribuut in de Wandeling klasse houdt bij welke honden er mee gaan. De honden staan in een lijst.
Omdat de relatie tussen wandeling en hond een veel-op-veel relatie is, heeft de bijbehorende database een koppeltabel. Let wel: de koppeltabel is niet zichtbaar als object in de Java-applicatie.
Ter vereenvoudiging van de opdracht houden we bij een medewerker alleen de voor- en achternaam bij. Mochten er medewerkers met een tussenvoegsel zijn, dan wordt dat toegevoegd aan de achternaam.

Hieronder zie je het fysieke datamodel van de database met gegevens van de uitlaatdienst.



Maak DAO klassen

Voorbereiding

a. Op GitLab staat een startproject met de naam "`HondenDienststartproject`". Je komt daar via de link [hondendienststartproject].
b. Clone het project met de SSH url (zie eventueel het Git deel in de studiewijzer Voorbereiding).
c. Maak een nieuw project in IntelliJ en kies Project from Version Control.
d. Zet het project in een folder naar keuze.
e. Het project heeft misschien geen verwijzing naar een SDK. Open de klasse HondenDienstLauncher. Als je foutmeldingen ziet, ga dan naar File/Project Structure en controleer of daar een SDK gekozen is bij Project (zie ook het Git deel in de studiewijzer Voorbereiding).

Stap 1: Database aanmaken, connectie testen

a. In de map "resources" van je project staan het bestand "`HondenDienstCreateInsert.sql`". Voer het script uit in Workbench.
b. Open het startproject in IntelliJ en test de connectie in de Launcher. Je krijgt als het goed is een lijst met klanten te zien.
c. Als je de melding krijgt "Driver niet gevonden", ga dan weer naar File/Project Structure. De verwijzing naar de mysql-library is wellicht fout. Kijk bij Problems, je ziet een melding dat de library niet gebruikt is. Klik op Fix en vervolgens op "Add to Depencies. Test opnieuw de connectie.

Oefenen met git.

a. Ga naar de terminal in IntelliJ en typ git status. Doordat je een repository gecloned hebt, heb je gelijk een lokale repository.
b. Maak voor elke stap in de deze opdracht een nieuwe branch.
c. Checkout de nieuwe branch.
d. Als je code goed is, commit dan de code op de branch.
e. Checkout de main branch
f. Merge vervolgens de branch in de main branch.
g. Delete de branch.

MAKE IT WORK

Stap 2: KlantDAO aanvullen

a. Schrijf een methode `slaKlantOp(Klant klant)` die een `Klant` object toevoegt aan de tabel `Klant` in de database. Let wel: de primary key van de tabel klant staat op auto increment.
b. Schrijf een methode `public Klant getKlantPerId(int klantId)` die een `Klant` uit de tabel `Klant` haalt op basis van het klantnummer. Let wel: deze methode geeft dus precies 1 klant terug en niet een lijst van klanten.
c. Haal de comments weg in de Launcher om de methoden in de `KlantDAO` te testen.

Stap 3: HondDAO maken

a. Voeg de klasse `HondDAO` toe aan de package `database`. Deze klasse moet een subklasse van de `AbstractDAO` klasse zijn.
b. Schrijf een methode `slaHondOp(Hond hond)` die een object `Hond` als record toevoegt aan de tabel `Hond`. Let wel: het record bevat een foreign key `klantnummer`, terwijl het object `Hond` een referentie heeft naar een object `Klant`. Je moet dus van het object `Klant` het klantnummer nemen om in het record `Hond` te zetten. Uiteraard moet de klant al bestaan in je database.
c. Schrijf een methode `public Hond getHondPerId(String chipnummer)` die een `Hond` uit de database haalt op basis van een chipnummer. In dit geval krijg je van de klant alleen het klantnummer terug van de database. Je moet nu een klantDAO object gebruiken om de bijbehorende Klant uit de database te halen. Je hebt daarvoor de methode `getKlantPerId()` ter beschikking. Zodoende kun je toch aan het `Hond` object een `Klant` object meegeven.
d. Schrijf een methode `public List<Hond> getHondenPerKlant(Klant klant)` die alle honden van een bepaalde klant uit de database haalt.
e. Schrijf een methode `getHonden()` die alle honden uit de database haalt en die in de vorm van een `List<Hond>` teruggeeft.
f. Haal de comments weg in de Launcher om de methoden in de `HondDAO` te testen.

Stap 4: WandelingDAO maken

a. Voeg de klasse `WandelingDAO` toe aan de package `database`. Deze klasse moet een subklasse van de `AbstractDAO` klasse zijn.
b. Schrijf een methode `slaWandelingOp(Wandeling wandeling)` die een `Wandeling` object toevoegt aan de database. De methode slaat in eerste instantie alleen informatie op in de tabel `Wandeling`. De informatie over de honden die meegaan met de wandeling komt in de volgende stap. Let wel: de tabel `Wandeling` in de database heeft een primary key die op auto increment staat ingesteld.

Stap 5: HondInWandelingDAO maken

Een `Wandeling` object bevat een lijst met honden. Welke honden meegaan met een wandeling moet worden opgeslagen in de koppeltabel `HondInWandeling`. Ook bij het ophalen van een wandeling uit de database moeten de bijbehorende honden opgehaald worden. Ook nu is die informatie te halen uit de koppeltabel `HondInWandeling`. We hebben een Database Access Object nodig voor de koppeltabel.

a. Voeg de klasse `HondInWandelingDAO` toe aan de package `database`. Deze klasse moet een subklasse van de `AbstractDAO` klasse zijn.
b. Schrijf een methode `slaHondenInWandelingOp(Wandeling wandeling)` die de lijst honden van een wandeling langsgaat en de juiste informatie in de tabel `HondInWandeling` opslaat.
c. Roep de methode op de juiste plek aan in de methode `slaWandelingOp()`, die je in stap 4 hebt geschreven. Direct na het opslaan van een `wandeling` record kunnen nu ook de juiste records in `HondInWandeling` worden opgeslagen.
d. Schrijf een methode `List<Hond> getHondenPerWandeling(Wandeling wandeling)` om de honden van een wandeling op basis van de records in de koppeltabel op te halen.

e. Haal de comments weg in de Launcher om de methoden in de `WandelingDAO` en de `HondInWandelingDAO` te testen.

Stap 6 WandelingDAO uitbreiden

a. Schrijf nu in de klasse WandelingDAO een methode `getWandelingenPerMedewerker(Medewerker medewerker)` die een lijst wandelingen van een bepaalde medewerker uit de database haalt en daarvan een lijst met `Wandeling` objecten maakt. Je moet aan de wandelingen ook de honden toevoegen. Gebruik hierbij de methode `List<Hond> getHondenPerWandeling(Wandeling wandeling)` van de `HondInWandelingDAO` klasse.

b. Haal de comments weg in de Launcher om de methoden in de uitgebreide `WandelingDAO` te testen.