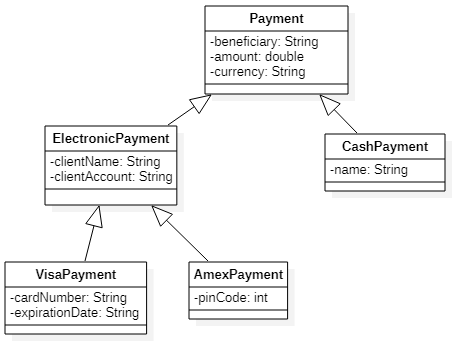
Inheritance

Download startproject.zip, unzip this project, rename it and put it in your corresponding GitHub-folder and add the following classes (in a package called model) according to the UML-diagram below. Use as a strategy: SINGLE\_TABLE



1. Make sure you create the no-arg constructor and all getters and setters for each entity
2. If you programmed exactly what is in the class diagram and you run your application, the corresponding table should exist in your database (check this in your Database tab).
3. Now run the SQL script, *lesson\_payment.sql,* to fill the corresponding table. Check this in your database tab. The table should be filled with the following records:

* 4 Cash payments
* 2 Visa payments
* 1 Amex payments

Afbeelding met tekst

Automatisch gegenereerde beschrijving

1. The statement "spring.jpa.hibernate.ddl-auto=create-drop" in your properties file will cause the database to be deleted after closing your program. So you will have to refill the database with the script “*script trainings.sql”* every time you run your program.

Then add a first link to index.html which takes you to results via the controller to result.html where a list of all payments is shown. Of these payments, you show the beneficiary, the amount and the currency:

Afbeelding met tekst, persoon, schermopname

Automatisch gegenereerde beschrijving

Add in the index.html 4 more links. Each link takes you to result.html, each time with a different list:

1. A list of all payments in ascending order of amount

Afbeelding met tekst

Automatisch gegenereerde beschrijving

1. A list of all cash payments

Afbeelding met tekst, brief

Automatisch gegenereerde beschrijving

1. A list of all electronic payments sorted by currency

Afbeelding met tekst, brief

Automatisch gegenereerde beschrijving

1. A list of all payments with an amount greater than a value passed via parameter on the index page:

Afbeelding met Rechthoek

Automatisch gegenereerde beschrijving

Afbeelding met tekst

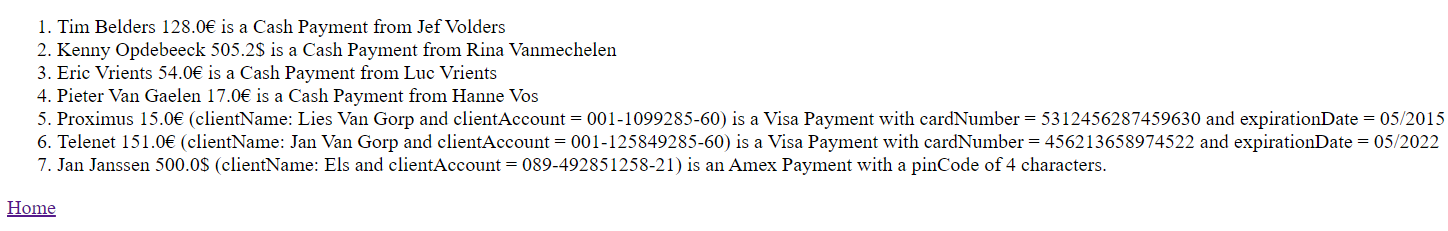
Automatisch gegenereerde beschrijving

Use keyword queries and/or JPQL

Tip: If you want to get back all objects of the electronic payment subclass, you can do so with this query: **SELECT e FROM ElectronicPayment e**

Another solution is to create a separate repository for this subclass.

Now change the output of alle payments by adding the information with regard to the type of payment and the additional data of this type of payment:



Tip: add an appropriate toString()-method in every class and use polymorphy to program this efficiently.

Finally add one more link on the homepage. This link should take you to a page on which the user can add a cash payment. On this page the user must fill out all attributes related to a CashPayment.

After submitting this new payment, this payment should be saved in the database and the result.html page should show all payments including the newly saved payment.