CA2: Martin og Dennis

For description of API services and JSON format, see link "API documentation" on the main page.

Description of chosen design:

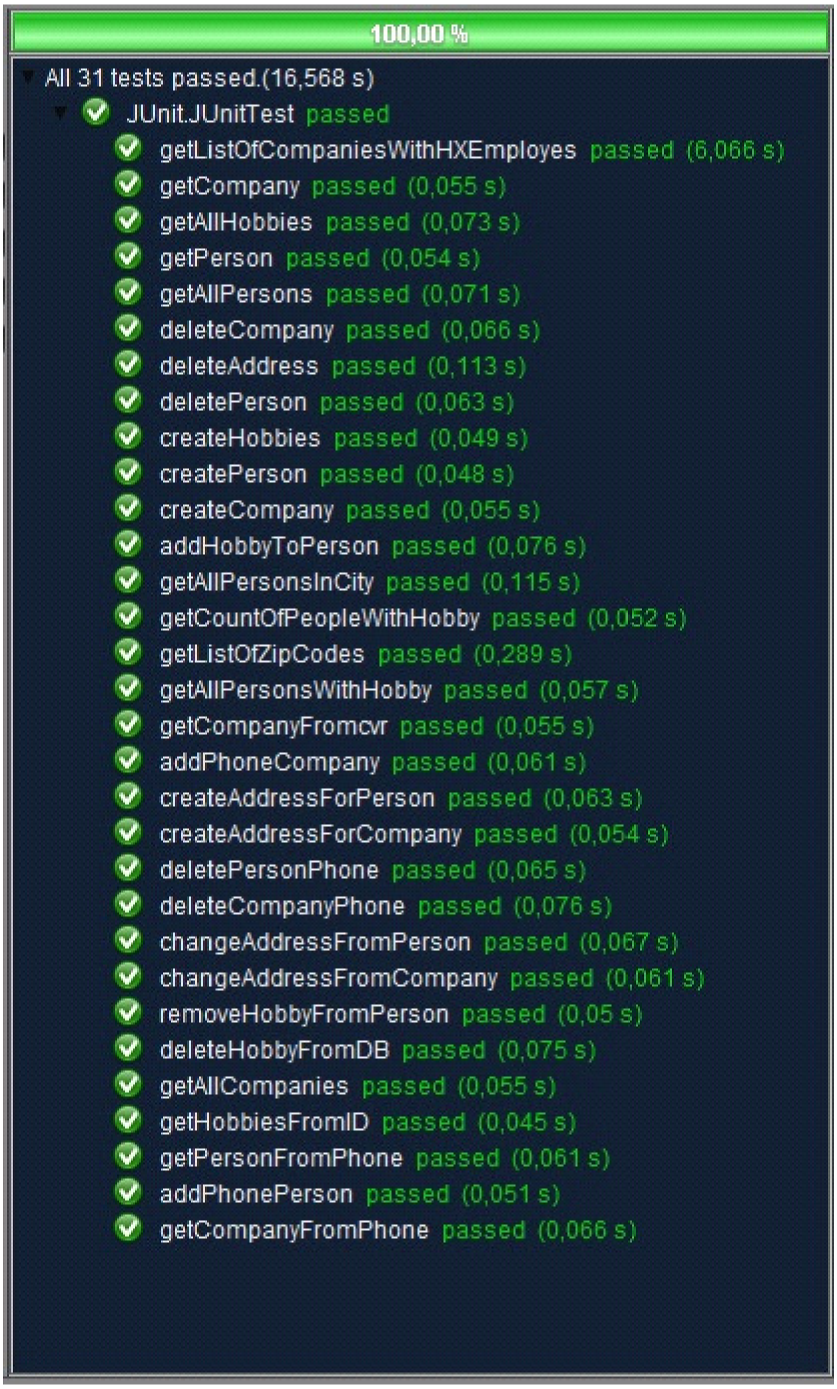
We are utilizing the facade design pattern. This means we have a facade class which acts as an intermediary between our REST api and our entities/database, instead of our REST api making direct calls to the database.

Error handling:

So far we have implemented two types of Exceptions: EntityNotFoundException and NotNumericException. The first one gets thrown if an em.find does not return an object (which means that no entry in the database corresponds to the provided id) and the second is thrown if a json value is expected to be an integer, but is not.

Test strategy and JUnit Tests:

We have tested the facade using JUnit tests. In order to check that a method behaves correctly, we have made a seperate call to the database besides the method being tested, and compared the two results. We have also tested most methods in the REST api with Postman.



```
100,00 %
All 31 tests passed.(16,568 s)
  ✓ JUnit.JUnitTest passed
    ✓ getListOfCompaniesWithHXEmployes passed (6,066 s)
    ✓ getCompany passed (0,055 s)
    ✓ getAllHobbies passed (0,073 s)
    ✓ getPerson passed (0,054 s)
    ✓ getAllPersons passed (0,071 s)
    ✓ deleteCompany passed (0,066 s)
    ✓ deleteAddress passed (0,113 s)
    ✓ deletePerson passed (0,063 s)
    ✓ createHobbies passed (0,049 s)
    ✓ createPerson passed (0,048 s)
    ✓ createCompany passed (0,055 s)
    ✓ addHobbyToPerson passed (0,076 s)
    ✓ getAllPersonsInCity passed (0,115 s)
    ✓ getCountOfPeopleWithHobby passed (0,052 s)
    ✓ getListOfZipCodes passed (0,289 s)
    ✓ getAllPersonsWithHobby passed (0,057 s)
    ✓ getCompanyFromcvr passed (0,055 s)
    ✓ addPhoneCompany passed (0,061 s)
    ✓ createAddressForPerson passed (0,063 s)
    ✓ createAddressForCompany passed (0,054 s)
    ✓ deletePersonPhone passed (0,065 s)
    ✓ deleteCompanyPhone passed (0,076 s)
    ✓ changeAddressFromPerson passed (0,067 s)
    ✓ changeAddressFromCompany passed (0,061 s)
    ✓ removeHobbyFromPerson passed (0,05 s)
    ✓ deleteHobbyFromDB passed (0,075 s)
    ✓ getAllCompanies passed (0,055 s)
    ✓ getHobbiesFromID passed (0,045 s)
    ✓ getPersonFromPhone passed (0,061 s)
    ✓ addPhonePerson passed (0,051 s)
    ✓ getCompanyFromPhone passed (0,066 s)
```

Strategy used to implement inheritance and why:

We made a class called InfoEntity, which contains a string called email, and an Address object, since these are attributes are shared by both the Person entity and Company entity. Both of these child classes extends InfoEntity. They also both contain a list of phones, though in retrospect, this phonelist could also have been placed in InfoEntity, as it is also a shared attribute. Apart from that, the rest of the attributes in Person and Company are different.

Who did what:

We have worked closely together on most parts of the project, though one person usually had the general responsibility for certain parts. The facade and entities were made together, as well with the JUnit tests. In the REST api, Martin worked mostly on the Person api, and Dennis mostly did Company and Address api. Martin wrote this document. We both worked on error handling. As for the html pages, Martin worked mainly on person.html and apidocumentation.html, and Dennis worked mainly on company.html, createCompany.html, hobby.html and zipcode.html.