Lab 7

Part A:

Given is the project **Lab7PartA**.

First run the code and notice that both the Account and the Customer is saved in the database.
Now change the method saveCustomer() in the CustomerRepository like this:

```
public interface CustomerRepository extends
JpaRepository<Customer, Long> {
    default void saveCustomer(Customer customer) {
        throw new RuntimeException("could not save customer");
//            save(customer);
    }
}
```

Run the application and notice that we have an account in the database without customer. Modify the application such that you never can have an account without a customer in the database.

Part B:

Copy and paste **Lab7PartA to Lab7PartB.**
First change the CustomerRepository back like this:

```
public interface CustomerRepository extends
JpaRepository<Customer, Long> {
    default void saveCustomer(Customer customer) {
//      throw new RuntimeException("could not save customer");
        save(customer);
    }
}
```

Now modify the application such that
When both the Account and Customer are saved correctly we send an email to the customer with the text “Welcome <customername>”
When we cannot save the Account and Customer correctly in one transaction, we send an email to the customer with the text “We could not create your account <accountnumber>”

Part C:

Now we want to add tracing to this application.

We need a separate table with the name **tracerecord** which stored the result of the transactional method createCustomerAndAccount(). This table saves the current date and time, and the result of the transaction.

When both the Account and Customer are saved correctly we save a record to the database with the current date and time and the text "**Customer <customername> created with account <accountnumber>"**
When we cannot save the Account and Customer correctly in one transaction, we save a record to the database with the current date and time and the text "**Could not create customer <customername> with account <accountnumber>"**

Part D:

Copy and paste the Bank application from Lab 5 part C.
Modify the new Bank application as such that all methods of the AccountService class are transactional. You can do this by placing the **@Transactional** annotation on the class instead of the individual methods.
Now modify the domain class Account and make all relationships LAZY.
If you run the application everything should still work.
Then remove the @Transactional annotation and notice that you get errors.
Then add the @Transactional annotation again.

Explain in a document why it is that we don't need eager loading anymore and the Application still works with lazy loading.

What to hand in:

1. A separate zip file with the solution of part A
2. A separate zip file with the solution of part B
3. A separate zip file with the solution of part C
4. A PDF of part D