Lab 7 - Classes

Preparations

If you couldn't finish the previous exercise, you can copy and paste the pervious solution from the *labsSolutions* folder.

In this lab, we'll convert some interfaces to classes.

Exercise 1 - Converting interfaces to classes

1. Inside the main.ts file, change the `interface` keyword in front of `BankAccount` to `class`.
2. Change the `createBankAccount` function to be the `constructor` of `BankAccount`
3. Do the same for the `Customer` and `Iban` interfaces. Make sure that all attributes are assigned in their constructors. Be sure to let the `Iban` class be entirely `readonly`.
4. Convert `generateIban` to a static method in the `Iban` class. Feel free to rename it to `generate`.
5. Make sure that both `Customer` and `Iban` have a `format` method. Refactor `formatName` and `formatIban` for this purpose.

Exercise 2 - Creating a Bank class.

We want to add a `Bank` class. We expect it to have a lot of configuration, so we decide to create a interface called `BankConfig`.

1. Inside the main.ts file, create an `interface` called `BankConfig`.
2. Add the following fields (type: `string`)
    - `name`, `countryCode` and `bankCode`.
3. Create a class called `Bank`, with:
    - A `private` field `config` of type `BankConfig`
    - A `private` field `accounts` of type `BankAccount[]`
    - A `constructor` which accepts a `BankConfig` and assigns it to its own `config`.
4. Create a (public) method called `createAccount` which creates a bank account for a given customer and adds it to the private `accounts` array. It should also print `'[${bankName}] welcomes ${account}'` to the console.
5. Make sure the `bankCode` from the `config` attribute is used to instantiate the `Iban` instances inside the `createAccount` method.
    - Feel free to remove the old `DEFAULT_BANK_CODE` and `DEFAULT_COUNTRY_CODE` fields.
6. Both fields `config` and `accounts` should never be reassigned. Make sure this is the case.
7. Test it out! Remove the old `bankAccounts` array. Instead, create a new bank and create some accounts. Make sure everything works and you don't have compile errors.

```
const bank = new Bank({ bankCode: 'TYPE', countryCode: 'NL', name:
'Typed bank'});
bank.createAccount(new Customer('Alfred', 'Kwak', 'Jodocus'));
bank.createAccount(new Customer('Brad', 'Pit'));
bank.createAccount(new Customer('Jack', 'Sparrow'));
```