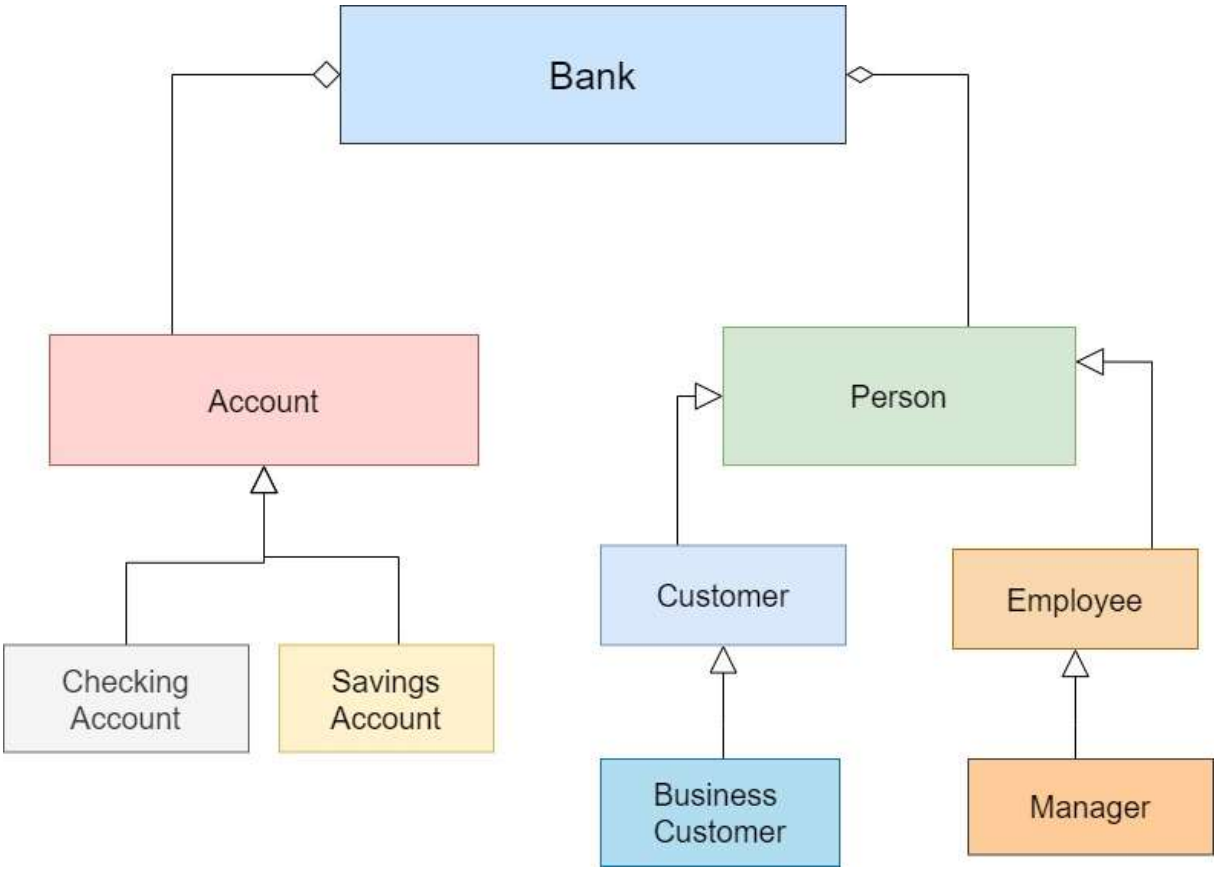
BankSystem

A repository containing the Bank System Console Application for the Clean Code Course at FMI.



Commands:

Class Menu contains all functionalities of the BankSystem:

- **void listCustomers(Bank&);**
- **void addNewCustomer(Bank&);**
- **void addNewBusinessCustomer(Bank&);**
- **void deleteCustomer(Bank&);**
- **void listAllAccounts(Bank&);**
- **void listCustomerAccount(Bank&);**
- **void addNewCheckingAccount(Bank&);**
- **void addNewSavingAccount(Bank&);**
- **void deleteAccount(Bank&);**
- **void withdrawFromAccount(Bank&);**
- **void depositToAccount(Bank&);**
- **void transfer(Bank&);**
- **void getLoan(Bank&);**
- **void listEmployees(Bank&);**

Classes and public methods:

Account

- **Abstract class**
- **Account(std::string IBAN, std::string ownerID, double balance); -** constructor of class Account;
- **void deposit(double amount);** - deposit to an account. It accepts a double amount as an argument and returns void.
- **void withdraw(double amount); -** withdraw from an account. It accepts a double amount as an argument and returns void.

- **virtual void printInformation() = 0;** – pure virtual method that will be overwritten in the derivatives. It doesn't accept any arguments and returns void.

CheckingAccount

- **CheckingAccount(std::string IBAN, std::string customerID, double balance); -** constructor of class CheckingAccount;
- **void printInformation() override;** – override printInformation() function from abstract class Account. It doesn't accept any arguments and returns void. It prints information about CheckingAccount – IBAN, customer ID, account balance;

SavingsAccount

- **SavingsAccount(std::string IBAN, std::string customerID, double balance, double interestRate); -** constructor of class SavingsAccount;
- **void printInformation() override;** – override printInformation() function from abstract class Account. It doesn't accept any arguments and returns void. It prints information about SavingsAccount– IBAN, customer ID, account balance, interest rate;

Loan

- **Loan(double amount, double loanRate, std::string ownerID); -** constructor of class Loan;

Person

- **Abstract class**
- **void validateAge(int age);** - check if age is between 18 and 100;
- **virtual void printInformation() = 0;** – pure virtual method that will be overwritten in the derivatives. It doesn't accept any arguments and returns void.

Customer

- **Customer(std::string name, int age, std::string ID); -** constructor of class Customer;

- **void printInformation() override;** – override printInformation() function from abstract class Person. It doesn't accept any arguments and returns void. It prints information about Customer – name, ID, age.

BusinessCustomer

- **BusinessCustomer(std::string name, int age, std::string ID, std::string businessType); -** constructor of class BusinessCustomer;
- **void printInformation() override;** – override printInformation() function from abstract class Person. It doesn't accept any arguments and returns void. It prints information about BusinessCustomer – name, id, age, businessType;

Employee

- **Employee(std::string name, int age, std::string position, double salary);-** constructor of the class;
- **void printInformation() override;** – override printInformation() function from abstract class Person. It doesn't accept any arguments and returns void. It prints information about Employee – name, position, salary;

Manager

- **Manager(std::string name, int age, std::vector<Employee*> employees, double salary); -** constructor of the class;
- **void addEmployee(std::string name, int age, std::string position, double salary); -** add an employee. It accepts string name, int age, string position and double salary as arguments and returns void.
- **void removeEmployee(int employeeID);** - delete an employee. It accepts an int employeeID and returns void.
- **int getNumberOfEmployees() const;** - returns the number of employees as an integer;
- **void listEmployees();** - prints information about employees and returns void;

Bank

- **Bank(std::string name, Manager* manager, std::map<std::string, Customer*> customers, std::map<std::string, Account*> accounts); -** constructor of the class;
- **Customer* getCurrentCustomer(std::**string **customerID) const; -** returns a Customer with given customerID as argument;
- **Account* getCurrentAccount(std::string IBAN) const; -** returns an Account with given IBAN number as argument;
- **void addCustomer(std::string name, int age, std::string customerID);** - add new customer. It accepts a string name, int age and string customer ID as arguments and returns void;
- **void addBusinessCustomer(std::string name, int age, std::string customerID, std::string businessType); -** add new business customer. It accepts a string name, int age and string customer ID as arguments and returns void;
- **void deleteCustomer(std::string customerID); -** delete a customer. It accepts a string customer ID as argument and returns void;
- **void listCustomers(); -** print information about all customers in the bank. It doesn’t accept any arguments and returns void.
- **void addCheckingAccount(std::string IBAN, std::string customerID, double balance); -** add a new checking account. It accepts a string IBAN, string customer ID and double balance as arguments and returns void.
- **void addSavingsAccount(std::string IBAN, std::string customerID, double balance, double interestRate); -** add a new savings account. It accepts a string IBAN, string customer ID, double balance and double interest rate as arguments and returns void.
- **void deleteAccount(std::string IBAN); -** delete an account. It accepts a string IBAN as argument and returns void.

- **void listCustomerAccount(std::string IBAN); -** print information about an account in the bank. It accepts a string IBAN and returns void.
- **void listAccounts(); -** print information about all accounts in the bank. It doesn't accept any arguments and returns void.
- **void listEmployees(); -** print information about all employees in the bank. It doesn't accept any arguments and returns void.
- **void getLoan(std::string customerID, double loanRate, double amount); -** add a new loan. It accepts a string customer ID, double loan rate, double amount as arguments and returns void.
- **void transfer(std::string fromIBAN, std::string toIBAN, double amount); -** transfer money from an account to an account. It accepts a string IBAN and double amount as arguments and returns void.
- **void deposit(std::string IBAN, double amount); -** deposit money to an account. It accepts a string IBAN and double amount as arguments and returns void.
- **void withdraw(std::string IBAN, double amount); -** withdraw money from an account. It accepts a string IBAN and double amount as arguments and returns void.
- **bool isValidTransaction(double amount, std::string IBAN);** - check if a transaction is valid. It accepts a string IBAN and double amount as arguments and returns bool.
- **bool IbanExists(std::string IBAN); -** check if IBAN exists. It accepts a string IBAN as argument and returns bool.
- **bool isCustomerExist(std::string customerID); -** check if customer ID exists. It accepts a string customer ID as argument and returns bool.

Menu

It includes all functionalities that can be chosen by the client of the BankSystem:

- **void listCustomers(Bank&);**
- **void addNewCustomer(Bank&);**
- **void addNewBusinessCustomer(Bank&);**
- **void deleteCustomer(Bank&);**
- **void listAllAccounts(Bank&);**
- **void listCustomerAccount(Bank&);**
- **void addNewCheckingAccount(Bank&);**
- **void addNewSavingAccount(Bank&);**
- **void deleteAccount(Bank&);**
- **void withdrawFromAccount(Bank&);**
- **void depositToAccount(Bank&);**
- **void transfer(Bank&);**
- **void getLoan(Bank&);**
- **void listEmployees(Bank&);**