1. Objeto número 1 é a de conta, um cadastro normal com alguns atributos privados, em seguida alguns métodos Get e Set, para buscar e alterar os dados e as ações que fazemos com a conta. Em seguida duas classes abstratas que herdam da conta.

import { TaxasJuros } from "./taxasJuros.js";

import { TIPOTRANSACAO } from "./tipoTransacao.js";

import { Transacao } from "./transacao.js";

export class Conta {

#cliente;

#numero;

#agencia;

#dataCriacao;

#saldo;

#status;

#transacoes;

//atributo (static) da classe Conta

static contador = 0;

constructor(cliente, numero, agencia, saldo) {

if (this.constructor == Conta) {

throw new Error('Não é possível instanciar a classe Conta');

} else {

this.#cliente = cliente;

this.#numero = numero;

this.#agencia = agencia;

this.#dataCriacao = new Date().toLocaleDateString();

this.#saldo = saldo;

this.#status = true;

this.#transacoes = [];

Conta.contador++;

}

}

//método (static) da classe conta

static mostrarContador() {

return `O número de contas já criadas é: ${Conta.contador}`;

}

// depositar da conta

depositar(valor) {

this.#saldo += valor;

let trans = new Transacao(TIPOTRANSACAO.creditar, new Date().toLocaleDateString(), valor, null, '+');

this.#transacoes.push(trans);

}

// sacar da conta

sacar(valor) {

if (this.#saldo >= TaxasJuros.calcularIOF(valor)) {

this.#saldo -= TaxasJuros.calcularIOF(valor);

let trans = new Transacao(TIPOTRANSACAO.debitar, new Date().toLocaleDateString(), valor, null, '-');

this.#transacoes.push(trans);

} else {

// lançar um erro

console.error('Erro: Saldo insuficiente ' + valor + ' maior do que o saldo ' + this.#saldo + '.');

}

}

// transferir da conta

transferir(valor, contaFav) {

if (this.#saldo >= valor) {

this.#saldo -= valor;

let trans = new Transacao(TIPOTRANSACAO.transferir, new Date().toLocaleDateString(), valor, contaFav.cliente, '-');

this.#transacoes.push(trans);

contaFav.#saldo += valor;

let transFav = new Transacao(TIPOTRANSACAO.transferir, new Date().toLocaleDateString(), valor, this.#cliente, '+');

contaFav.#transacoes.push(transFav);

} else {

console.error('Erro: Saldo insuficiente ' + valor + ' maior do que o saldo ' + this.#saldo + '.');

}

}

// realizar pagamento da conta

pagar(valor) {

if (this.#saldo >= valor) {

this.#saldo -= valor;

let trans = new Transacao(TIPOTRANSACAO.pagar, new Date().toLocaleDateString(), valor, null, '-');

this.#transacoes.push(trans);

} else {

// lançar um erro

console.error('Erro: Saldo insuficiente ' + valor + ' maior do que o saldo ' + this.#saldo + '.');

}

}

// necessário pois não tem em JS o protected

render(rendimento){

this.#saldo += rendimento;

let trans = new Transacao(TIPOTRANSACAO.render, new Date().toLocaleDateString(), rendimento, null, '+');

this.#transacoes.push(trans);

}

// necessário pois não tem em JS o protected

cobrarTaxa(taxa){

this.#saldo -= taxa;

let trans = new Transacao(TIPOTRANSACAO.cobrarTaxa, new Date().toLocaleDateString(), taxa, null, '-');

this.#transacoes.push(trans);

}

// mostrar saldo da conta

mostrarSaldo() {

return "Saldo: R$ " + this.#saldo;

}

getSaldo(){

return this.#saldo;

}

// toString da conta

toString() {

return `\tAgência: ${this.#agencia.numero} | Nº Conta: ${this.#numero}\n`;

}

// mostrar extrato da conta

mostrarExtrato() {

let extrato = "\t\t\tEXTRATO DE CONTA\n";

extrato += "\t---------------------------------------------------\n";

extrato += this.toString();

extrato += this.#cliente.toString();

extrato += "\t---------------------------------------------------\n";

extrato += "\tDATA\t\tHISTÓRICO\t\tVALOR (R$)\n";

extrato += "\t---------------------------------------------------\n";

for (const trans of this.#transacoes) {

extrato += trans.toString();

}

extrato += "\t---------------------------------------------------\n";

extrato += `\tSaldo:\t\t R$ ${this.#saldo.toFixed(2)}`;

return extrato;

}

/\* // Outras linguagens

getCliente() {

return this.#cliente;

}

setCliente(cliente) {

this.#cliente = cliente;

}\*/

get cliente() {

return this.#cliente;

}

get saldo() {

return this.#saldo;

}

get numero() {

return this.#numero;

}

set numero(numero) {

this.#numero = numero;

}

/\*setNumero(numero) {

this.#numero = numero;

}\*/

get agencia() {

return this.#agencia;

}

set agencia(agencia) {

this.#agencia = agencia;

}

get dataCriacao() {

return this.#dataCriacao;

}

get status() {

return this.#status;

}

set status(status) {

this.#status = status;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import { Conta } from './conta.js'

export class ContaCorrente extends Conta {

#taxaManutencao

constructor(cliente, numero, agencia, saldo, taxaManutencao) {

super(cliente, numero, agencia, saldo);

this.#taxaManutencao = taxaManutencao;

}

cobrarTaxaManutencao(){

super.cobrarTaxa(this.taxaManutencao);

}

get taxaManutencao() {

return this.#taxaManutencao;

}

set taxaManutencao(taxaManutencao) {

this.#taxaManutencao = taxaManutencao;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import { Conta } from "./conta.js";

export class Poupanca extends Conta{

#taxaRendimento;

constructor(cliente, numero, agencia, saldo, taxaRendimento){

super(cliente, numero, agencia, saldo);

this.#taxaRendimento = taxaRendimento;

}

/\*

// Polimorfísmo de sobrecarga de construtor, não existe em JS.

constructor(cliente, numero, agencia){

super(cliente, numero, agencia, 0.0);

this.#taxaRendimento = 0.02;

}\*/

// Polimorfísmo de sobrescrita de método da mãe, não existe em JS.

depositar(valor) {

valor += valor\*this.#taxaRendimento;

super.depositar(valor);

}

render(qtdMeses){

let saldoAtual = super.getSaldo();

for (let i = 0; i < qtdMeses ; i++) {

saldoAtual += saldoAtual\*this.#taxaRendimento;

}

let rendimento = saldoAtual - super.getSaldo();

super.render(rendimento);

}

/\*

// Polimorfísmo de sobrecarga de método, não existe em JS.

render(){

console.log("render passou aqui");

let saldoAtual = super.getSaldo();

saldoAtual += saldoAtual\*this.#taxaRendimento;

let rendimento = saldoAtual - super.getSaldo();

super.render(rendimento);

}\*/

}