Semana 15.2

Javascript - programando a orientação a objetos

variaveis

const cliente1Nome = 'Ricardo'

const cliente1CPF = 11122233309

const cliente1Agencia = 1001

const cliente1Saldo = 0

const cliente2Nome = 'Alice'

const cliente2CPF = 11144455589

const cliente2Agencia = 1001

const cliente2Saldo = 0

const cliente3Nome = 'Joao'

const cliente3CPF = 45678955589

const cliente3Agencia = 1001

const cliente3Saldo = 0

agora iremos ver como ficaria sendo uma classe

class Cliente {

    nome;

    cpf;

    agencia;

    saldo

}

const cliente1 = new Cliente();

cliente1.nome = 'Ricardo'

cliente1.cpf = 11122233309

cliente1.agencia = 1001

cliente1.saldo = 0

class ContaCorrente {

    agencia;

    saldo

    sacar(valor) {

        if(this.saldo >= valor) {

            this.saldo -= valor

        }

    }

}

contaCorrenteRicardo.sacar(50)

this -> quem está chamando

class ContaCorrente {

    agencia;

    #saldo = 0;



    mostraSaldo() {

        console.log(`Seu saldo é: ${this.#saldo}`);

    }

    sacar(valor) {

        if(this.#saldo >= valor) {

            this.#saldo -= valor

        }

    }

    depositar(valor) {

        if(valor > 0) {

            this.#saldo += valor

        }

    }

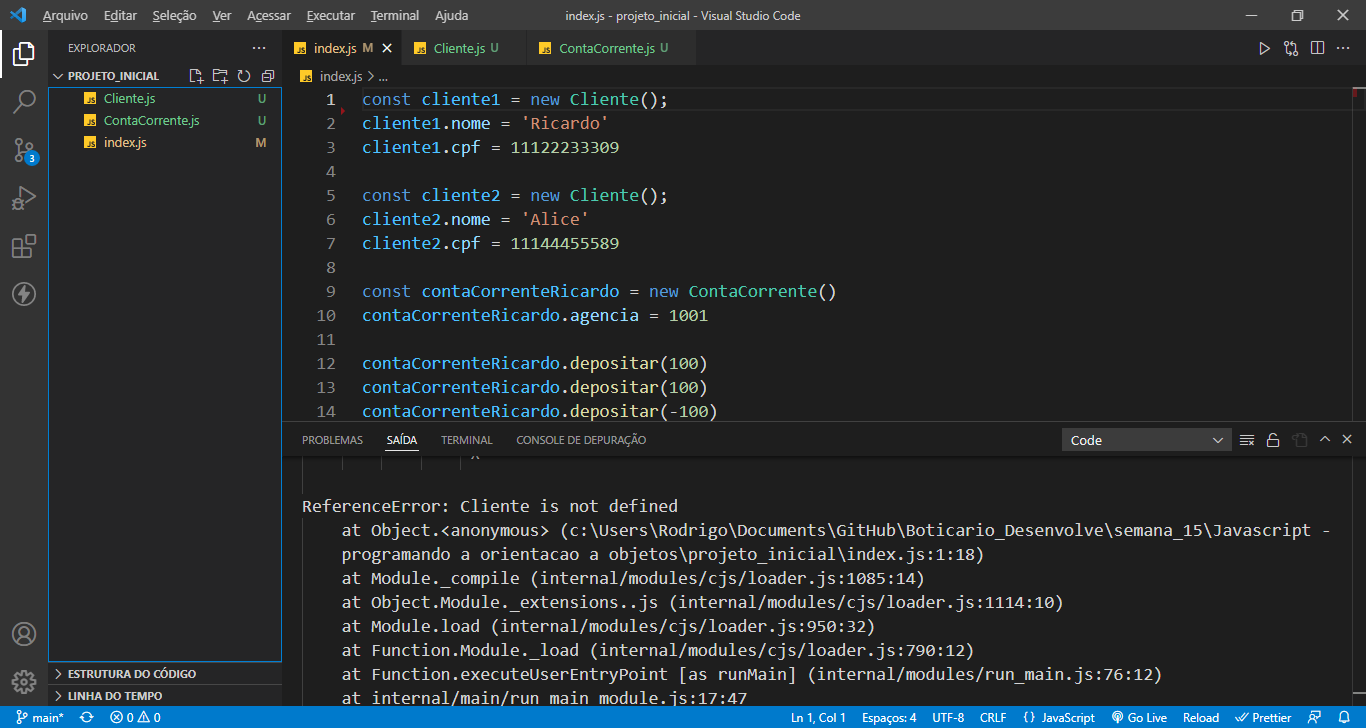
}

Atributo privado, quando for chamado não irá aparecer, so pode ser acessado dentro da classe por ser privado seu uso

console.log(contaCorrenteRicardo);

ContaCorrente { agencia: 1001 }

Cliente.js -> toda a vez que começar um arquivo com a letra MAIUSCULA, significa que está criando uma classe



export class Cliente {

    nome;

    cpf;

}

Exportando uma classe

import { Cliente } from "./Cliente";

importando a classe que está em outro arquivo para pode usa-la

npm init -> para fazer funcionar os módulos em seus projeto ira criar um arquivo package.json, que terá as informações do seu projeto

insira

"type": "module"

No package.json depois de licence coloque o type

transerir(valor, conta) {

        const valorSacado = this.sacar(valor)

        conta.depositar(valorSacado)

    }

Usando multiparametros

contaCorrenteRicardo.transerir(200, conta2)

por dentro dessa função de transferir, ele pega o valor passado e faz o saque usando a função sacar, e em seguida pega a conta passada como parâmetro e usa a função depositar passando o valor

import { Cliente } from "./Cliente";

export class ContaCorrente {

    agencia;

    \_cliente;

    set cliente(novoValor) {

        if(novoValor instanceof Cliente) {

            this.\_cliente = novoValor

        }

    }

Está setando um valor, em uma classe privada caso o valor passado seja uma INSTANCIA(criação) de Cliente

 get cliente() {

        return this.\_cliente

    }

O get funciona para pegar o valor

export class Cliente {

    nome;

    \_cpf;

    get cpf() {

        return this.\_cpf

    }

    constructor(nome, cpf) {

        this.nome = nome;

        this.\_cpf = cpf

    }

}

Como o método de construtor e possível criar valor para a classe

const cliente1 = new Cliente('Ricardo', 11122233309);

const cliente2 = new Cliente('Alice', 11144455589);

cliente2.cpf = 333332232323

agora não será possível atribuir um valor para a variável dessa maneira, so poderá ser atribuída um valor a ela em tempo de construção

export class ContaCorrente {

    static numeroDeContas = 0

    agencia;

    \_cliente;

    \_saldo = 0;

Static numeroDeContas -> sera a variável que ira se referenciar a somente a essa classe

 constructor(cliente, agencia) {

        this.cliente = cliente;

        this.agencia = agencia;

        ContaCorrente.numeroDeContas += 1;

    }

Logo com o construtor cada vez que essa classe ContaCorrente for iniciada a variável numeroDeContas irá acresentar mais 1, cada vez que chama um new ContaCorrente ele ira fazer o acressimo no contador