```
//COMPOSICAO DE CLASSES:

//Um conceito muito importante que deve ser entendido no javascript é o conceito de "Composição de Classes".

//O que é Composição de Classes?

//De maneira sucinta, composição é um estilo de relacionamento entre classes numa POO onde existirá uma classe principal - onde os objetos serão gerados - e essa classe permite que os objetos sejam operados pelas outras classes relacionadas com ela.
//Porém, se a classe principal morrer, as classes relacionadas ficam inuteis e os objetos são eliminados.

//Explicando de forma mais simples, imagine que temos uma classe cliente para um banco que gera um objeto com os dados pessoais de uma pessoa. Mas como cliente do banco, o banco não quer apenas ter os dados da pessoa, ele quer que o cliente tenha uma conta, que possa fazer operações bancárias. Para isso então são criadas as classes "Conta" e "Operações", que se tornarão dependentes do cliente. Ou seja, se a classe cliente morrer as classes "Conta" e "Operações" ficarão inúteis.

//Por isso a Composição é a conhecidíssima "Relação da Morte" do POO.

//Vejamos como utilizá-la na prática:

//Nesse arquivo temos a Classe Principal:
import { ContaCorrente } from './183-Composicao_Ex_Conta.js' //Importamos as classes dependentes...
import { Operacoes } from './183-Composicao_Ex_Operacoes.js'

export class Cliente {
    constructor(nome, cpf, rg){
        this.nome = nome
        this.cpf = cpf
        this.rg = rg
    }

    dadosPessoais(){
        console.log(`
        Nome: ${this.nome}
        CPF: ${this.cpf}
        RG: ${this.rg}
        `)
```

```
    }
}

//Gerando instância para o cliente Ricardo...
let cliente_01 = new Cliente('Ricardo', '123.456.789-1', '235.456.785-7')
let contaCliente_01 = new ContaCorrente(cliente_01, '45654', '25415-4', 1500) //composição com classe ContaCorrente...
let operacoesCliente_01 = new Operacoes(cliente_01, contaCliente_01) //composição com classe Operadores...

//Gerando outra instância para a cliente Ana...
let cliente_02 = new Cliente('Ana', '123.456.789-1', '235.456.785-7')
let contaCliente_02 = new ContaCorrente(cliente_02, '45654', '25415-4', 1000)
let operacoesCliente_02 = new Operacoes(cliente_02, contaCliente_02)

console.log(contaCliente_01)
/*RESULTADO NO CONSOLE:
ContaCorrente {
  cliente: Cliente { //Note que a classe importada "ContaCorrente" recebeu um objeto da classe principal...
    nome: 'Ricardo',
    cpf: '123.456.789-1',
    rg: '235.456.785-7'
  },
  conta: '45654',
  agencia: '25415-4',
  _saldo: 1500
}
*/

operacoesCliente_01.verSaldo()
/*RESULTADO NO CONSOLE:
Saldo: R$ 1500.00
*/

console.log(contaCliente_02)
/*RESULTADO NO CONSOLE:
ContaCorrente {
  cliente: Cliente { nome: 'Ana', cpf: '123.456.789-1', rg: '235.456.785-7' },
```

```
  conta: '45654',
  agencia: '25415-4',
  _saldo: 1000
}
*/

operacoesCliente_02.verSaldo()
/*RESULTADO NO CONSOLE:
Saldo: R$ 1000.00
*/

//Aqui vemos a Composição na prática, vamos tentar fazer uma transferência entre o cliente 1 e o cliente 2, veja o arquivo Opeações para mais detalhes...
operacoesCliente_01.transferencia(500, contaCliente_02)
/*RESULTADO NO CONSOLE:
Fazendo transferência de Ricardo para Ana

Saldo restante de Ricardo R$ 1000.00
Transferência Bem Sucedida!
            Ana recebeu R$ 500.00
*/

//Ricardo teve o seu dinheiro retirado, por que seu relacionamento além de ser uma composição também é uma agregação...
operacoesCliente_01.verSaldo()
/*RESULTADO NO CONSOLE:
Saldo: R$ 1000.00
*/

//Mas Ana tinha um relacionamento apenas de associação na classe Operacoes na função transferência, por isso seu dinheiro não foi depositado, para mudar isso teríamos de ligar o objeto original de Ana com a função "transferencia()" da classe operações...
operacoesCliente_02.verSaldo()
/*RESULTADO NO CONSOLE:
Saldo: R$ 1000.00
*/
```

ARQUIVO DA CLASSE OPERACOES:

```
//CLASSE CRIADA COMO EXEMPLO PARA GERAR UMA COMPOSIÇÃO

//Perceba que por ser uma composição não temos nenhuma chamada para os métodos da classe "Operadores", invés disso todas as chamadas dos métodos estão a cargo da classe principal...
export class Operacoes {
    constructor(cliente, conta){ //a classe Operadores recebe o objeto gerado na classe cliente e na classe conta
        this._cliente = cliente
        this._conta = conta
        this._saldo = conta._saldo //Extraído só o saldo para trabalharmos com maior facilidade...
    }

    deposito(valor){
        this._saldo += valor
        console.log(`\nVocê depositou R$ ${valor.toFixed(2)}`)
        console.log(`Saldo atual R$ ${this._saldo.toFixed(2)}`)
    }

    saque(valor){
        if(this._saldo >= valor){
            this._saldo -= valor
            console.log(`\nSaldo restante de ${this._cliente.nome} R$ ${this._saldo.toFixed(2)}`)
        } else {
            console.log(`\nSeu saldo é insuficiente para sacar ao valor de R$ ${valor.toFixed(2)}`)
            console.log(`Saldo atual R$ ${this._saldo.toFixed(2)}`)
            console.log(`Se deseja sacar mesmo assim use o comando saqueCredito()`)
        }
    }

    saqueCredito(valor){
        this._saldo -= valor
        console.log(`\nSaldo restante de ${this._cliente.nome} R$ ${this._saldo.toFixed(2)}`)
    }

    verSaldo(){
        console.log(`\nSaldo: R$ ${this._saldo.toFixed(2)}`)
```

```
    }

    //Aqui está o pulo do gato da composição...
    transferencia(valor, recebedor){ //a variável recebedor recebe o cliente 2, porém note que ele não foi chamado por através de um relacionamento de composição - como acontece com o cliente 1 - e sim por associação, embora tenhamos acesso aos dados do cliente 2 não podemos alterar nenhum atributo dele, somente do cliente 1.
        console.log(`Fazendo transferência de ${this._cliente.nome} para ${recebedor.cliente.nome}`)
        if(this._saldo >= valor){
            this.saque(valor) //O cliente 1 realmente vai ter os 500 extraídos da sua conta, por ele tem um relacionamento de composição e agregação...
            recebedor._saldo += valor
            console.log( //Parece que Ana vai receber 500 e ficar com 1500, mas isso não vai acontecer com o objeto original.
            `Transferência Bem Sucedida!
            Ana recebeu R$ ${valor.toFixed(2)}
            `
            )
        } else {
            console.log('Saldo insuficente para fazer a transferência.')
        }
    }
}

//Retorne no arquivo da classe principal para ver que foram tirados 500 de Ricardo, mas não foram creditados 500 em Ana.
```

ARQUIVO DA CLASSE CONTA:

```
//CLASSE CRIADA COMO EXEMPLO PARA GERAR UMA COMPOSIÇÃO

export class ContaCorrente {
    constructor(cliente, conta, agencia, saldo){
        this.cliente = cliente
        this.conta = conta
        this.agencia = agencia

        this._saldo = saldo
    }
```

```
}
```