caso 1

/\* no caso 1 eu usei o chat gpt para ter uma base ja que estava com dificuldade, o polimorfismo eu fiz sozinho sem ajuda da IA \*/

// Cliente.ts

class Cliente {

nome: string;

public cnpj: string;

endereco: string;

constructor(nome: string, cnpj: string, endereco: string) {

this.nome = nome;

this.cnpj = cnpj;

this.endereco = endereco;

}

informacaDoCliente(): string{

return `${this.nome} dono do cnpj "${this.cnpj}" comprou... `

}

}

class ClienteContratante extends Cliente {

descontoFrequencia: number;

constructor(nome: string, cnpj: string, endereco: string, descontoFrequencia: number) {

super(nome, cnpj, endereco);

this.descontoFrequencia = descontoFrequencia;

}

informacaDoCliente(): string{

return `O clinte ${this.nome}, que compra frequentimente ganhou um desconto de ${this.descontoFrequencia}`

}

}

class ClienteEsporadico extends Cliente {

taxaAdicional: number;

constructor(nome: string, cnpj: string, endereco: string, taxaAdicional: number) {

super(nome, cnpj, endereco);

this.taxaAdicional = taxaAdicional;

}

informacaDoCliente():string{

return `O cliente ${this.nome} dono do cnpj "${this.cnpj}" teve uma taxa adicional de ${this.taxaAdicional}`

}

}

// Veiculo.ts

class Veiculo {

placa: string;

modelo: string;

capacidadeCarga: number;

tipoCarga: 'seco' | 'refrigerado' | 'perigoso';

status: 'disponível' | 'em manutenção' | 'em rota';

constructor(

placa: string,

modelo: string,

capacidadeCarga: number,

tipoCarga: 'seco' | 'refrigerado' | 'perigoso',

status: 'disponível' | 'em manutenção' | 'em rota'

) {

this.placa = placa;

this.modelo = modelo;

this.capacidadeCarga = capacidadeCarga;

this.tipoCarga = tipoCarga;

this.status = status;

}

}

// Carga.ts

class Carga {

peso: number;

volume: number;

tipo: string;

origem: string;

destino: string;

dataEmbarquePrevista: Date;

dataEntregaPrevista: Date;

constructor(

peso: number,

volume: number,

tipo: string,

origem: string,

destino: string,

dataEmbarquePrevista: Date,

dataEntregaPrevista: Date

) {

this.peso = peso;

this.volume = volume;

this.tipo = tipo;

this.origem = origem;

this.destino = destino;

this.dataEmbarquePrevista = dataEmbarquePrevista;

this.dataEntregaPrevista = dataEntregaPrevista;

}

}

// Motorista.ts

class Motorista {

nome: string;

cnh: string;

categoria: string;

experiencia: number;

veiculoAtribuido: Veiculo | null;

constructor(nome: string, cnh: string, categoria: string, experiencia: number, veiculoAtribuido: Veiculo | null) {

this.nome = nome;

this.cnh = cnh;

this.categoria = categoria;

this.experiencia = experiencia;

this.veiculoAtribuido = veiculoAtribuido;

}

}

// Rota.ts

class Rota {

origem: string;

destino: string;

distancia: number;

tempoEstimado: number;

veiculosPercorridos: Veiculo[];

constructor(

origem: string,

destino: string,

distancia: number,

tempoEstimado: number,

veiculosPercorridos: Veiculo[]

) {

this.origem = origem;

this.destino = destino;

this.distancia = distancia;

this.tempoEstimado = tempoEstimado;

this.veiculosPercorridos = veiculosPercorridos;

}

}

// Entrega.ts

class Entrega {

data: Date;

hora: string;

cliente: Cliente;

carga: Carga;

veiculo: Veiculo;

motorista: Motorista;

status: 'pendente' | 'em andamento' | 'entregue';

observacoes: string;

constructor(

data: Date,

hora: string,

cliente: Cliente,

carga: Carga,

veiculo: Veiculo,

motorista: Motorista,

status: 'pendente' | 'em andamento' | 'entregue',

observacoes: string

) {

this.data = data;

this.hora = hora;

this.cliente = cliente;

this.carga = carga;

this.veiculo = veiculo;

this.motorista = motorista;

this.status = status;

this.observacoes = observacoes;

}

}

// NotaFiscal.ts

class NotaFiscal {

numero: string;

dataEmissao: Date;

cliente: Cliente;

itens: Carga[];

valorTotal: number;

formaPagamento: string;

constructor(

numero: string,

dataEmissao: Date,

cliente: Cliente,

itens: Carga[],

valorTotal: number,

formaPagamento: string

) {

this.numero = numero;

this.dataEmissao = dataEmissao;

this.cliente = cliente;

this.itens = itens;

this.valorTotal = valorTotal;

this.formaPagamento = formaPagamento;

}

}

// Importando classes (se estiver usando módulos, remova os comentários de importação)

// import { Cliente, ClienteContratante, ClienteEsporádico } from './Cliente';

// import { Veiculo } from './Veiculo';

// import { Carga } from './Carga';

// import { Motorista } from './Motorista';

// import { Rota } from './Rota';

// import { Entrega } from './Entrega';

// import { NotaFiscal } from './NotaFiscal';

// Criando clientes

const cliente1 = new ClienteContratante("Transporte Ltda", "12.345.678/0001-90", "Rua Principal, 123", 10);

const cliente2 = new ClienteEsporadico("Loja de Artigos", "98.765.432/0001-00", "Avenida Secundária, 456", 15);

console.log("Clientes:");

console.log(cliente1.informacaDoCliente());

console.log(cliente2.informacaDoCliente());

/\*

// Criando veículos

const veiculo1 = new Veiculo("ABC-1234", "Caminhão Baú", 5000, "seco", "disponível");

const veiculo2 = new Veiculo("XYZ-9876", "Caminhão Frigorífico", 3000, "refrigerado", "em manutenção");

console.log("\nVeículos:");

console.log(veiculo1);

console.log(veiculo2);

// Criando cargas

const carga1 = new Carga(1000, 15, "alimentos", "São Paulo", "Rio de Janeiro", new Date("2024-12-01"), new Date("2024-12-03"));

const carga2 = new Carga(2000, 25, "eletrônicos", "Curitiba", "Porto Alegre", new Date("2024-12-02"), new Date("2024-12-05"));

console.log("\nCargas:");

console.log(carga1);

console.log(carga2);

// Criando motorista

const motorista1 = new Motorista("Carlos Silva", "123456789", "E", 10, veiculo1);

console.log("\nMotorista:");

console.log(motorista1);

// Criando rota

const rota1 = new Rota("São Paulo", "Rio de Janeiro", 430, 7, [veiculo1]);

console.log("\nRota:");

console.log(rota1);

// Criando entrega

const entrega1 = new Entrega(new Date("2024-12-01"), "08:00", cliente1, carga1, veiculo1, motorista1, "pendente", "Carga refrigerada");

console.log("\nEntrega:");

console.log(entrega1);

// Criando nota fiscal

const notaFiscal1 = new NotaFiscal(

"NF-001",

new Date("2024-12-01"),

cliente1,

[carga1, carga2],

5000.00,

"Cartão de Crédito"

);

console.log("\nNota Fiscal:");

console.log(notaFiscal1);

\*/

CASO 2

// caso 2

class Funcionario {

nome: string

public cpf: string

constructor( nome: string, cpf: string){

this.nome = nome

this.cpf = cpf

}

metodoDePagamento(): string {

return `${this.nome} recebe por: ???`

}

}

class Horista extends Funcionario {

constructor(nome: string, cpf:string){

super(nome, cpf)

}

metodoDePagamento(): string {

return `${this.nome} Recebe por hora trabalhada`

}

}

class Assalariado extends Funcionario {

constructor(nome: string, cpf: string){

super(nome, cpf)

}

metodoDePagamento(): string {

return `${this.nome} Recebe por mensalmente uma quantia `

}

}

class Comissionado extends Funcionario {

constructor(nome: string, cpf:string){

super(nome, cpf)

}

metodoDePagamento(): string {

return `${this.nome} Recebe por venda`

}

}

class Ativos {

nomeEquipamneto: string

status: 'disponivel' | 'em manutenção'

atribuidoA: object

constructor(nomeEquipamento: string, status: 'disponivel' | 'em manutenção', atribuidoA: object){

this.nomeEquipamneto = nomeEquipamento

this.status = status

this.atribuidoA = atribuidoA

}

}

class Projetos {

nome: string

inicio: string

fimPrevisto: string

funcionario: object[] = []

constructor(nome: string, inicio: string, fimPrevisto: string){

this.nome = nome

this.inicio = inicio

this.fimPrevisto = fimPrevisto

}

adicionarFuncionario(funci: object){

this.funcionario.push(funci)

}

}

class BaseDeDados {

funcionario: object[] = []

ativos: Ativos[] = []

projetos: Projetos[] = []

adicionarAtivos(ativo: Ativos){

this.ativos.push(ativo)

}

adicionarFuncionario(funci: object){

this.funcionario.push(funci)

}

adicionarProjetos(projctes: Projetos){

this.projetos.push(projctes)

}

}

const hosri = new Horista( "gustavo" , "000.000.000-00")

//console.log(hosri , hosri.metodoDePagamento())

const assa = new Assalariado("pedro", "000.000.000-01")

//console.log(assa, assa.metodoDePagamento())

const comis = new Comissionado("arthur", "000.000.000-02")

//console.log(comis, comis.metodoDePagamento())

const computador = new Ativos("notebook", 'disponivel', assa )

//console.log(computador)

const projeto1 = new Projetos("rumo a marte", "2024/02/01" , "2025/02/01")

projeto1.adicionarFuncionario(hosri)

projeto1.adicionarFuncionario(assa)

//console.log(projeto1)

const projeto2 = new Projetos("rumo a venus", "2025/02/01" , "2026/02/01")

projeto2.adicionarFuncionario(hosri)

projeto2.adicionarFuncionario(comis)

//console.log(projeto2)

const dados = new BaseDeDados()

dados.adicionarFuncionario(comis)

dados.adicionarFuncionario(hosri)

dados.adicionarFuncionario(assa)

dados.adicionarAtivos(computador)

dados.adicionarProjetos(projeto1)

dados.adicionarProjetos(projeto2)

console.log(dados)

CASO 3

//CASO 3

class Cliente {

formaDePagamento: string

combustivelComprado: string

quantidade: string

valorPago: number

constructor(formaDePagamento: string, combustivelComprado: string, quantidade: string, valorPago: number){

this.formaDePagamento = formaDePagamento

this.combustivelComprado = combustivelComprado

this.quantidade = quantidade

this.valorPago = valorPago

}

informacao(): string{

return `A forma de pagamento foi ${this.formaDePagamento}.`

}

}

class Frota extends Cliente {

quantidadeDeVeiculos: number

constructor(quantidadeDeVeiculo: number, formaDepagamento: string, combustivelComprado: string, quantidade: string, valorPago: number){

super(formaDepagamento, combustivelComprado, quantidade, valorPago)

this.quantidadeDeVeiculos = quantidadeDeVeiculo

}

informacao(): string{

return `A quantidade de carros abastecidos foi de ${this.quantidadeDeVeiculos}, e a forma de pagamento foi de ${this.formaDePagamento}.`

}

}

class NotaFiscal extends Frota {

public cpf: string

constructor(cpf: string, quantidadeDeVeiculo: number, formaDepagamento: string, combustivelComprado: string, quantidade: string, valorPago: number){

super(quantidadeDeVeiculo, formaDepagamento, combustivelComprado, quantidade, valorPago)

this.cpf = cpf

}

}

class Informacoes {

notasFiscais: NotaFiscal[] = []

adicionarInformacao(nota: NotaFiscal){

this.notasFiscais.push(nota)

}

}

const fiat = new Cliente("pix" , "etanol" , "5 litros", 100)

//console.log(fiat,fiat.informacao())

const skai = new Frota(55, "check", 'normal', "100 litro", 610)

//console.log(skai,skai.informacao())

const fiscalNota = new NotaFiscal("000.000.000-00", 1, "pix", "aditivada" , "5litros", 31.45)

console.log(fiscalNota)

const informacoesGerais = new Informacoes()

informacoesGerais.adicionarInformacao(fiscalNota)

console.log(informacoesGerais)

CASO 4

class Professor {

nome: string

idioma: 'ingles' | 'espanhol' | 'grego' | 'japones'

idade: number

sala: '101' | '102' | '103' | 'Sala EAD'

constructor(nome: string, idioma: 'ingles' | 'espanhol' | 'grego' | 'japones', idade: number, sala: '101' | '102' | '103' | 'Sala EAD') {

this.nome = nome

this.idioma = idioma

this.idade = idade

this.sala = sala

}

informacoes(): string {

return `O professor ${this.nome} da aula de ${this.idioma}`

}

}

class Aluno extends Professor {

public cpf: string

aulaPersonalizada: string

constructor(nome: string, idade: number, cpf: string, idioma: 'ingles' | 'espanhol' | 'grego' | 'japones', aulaPersonalizada: string, sala: '101' | '102' | '103' | 'Sala EAD') {

super(nome, idioma, idade, sala)

this.cpf = cpf

this.aulaPersonalizada = aulaPersonalizada

}

informacoes(): string{

return `O aluno ${this.nome} tem aula de ${this.idioma}`

}

}

class Turma {

professor: Professor

alunos: Aluno[] = []

sala: '101' | '102' | '103' | 'Sala EAD'

inicioDaTurma: string

fimDaTurma: string

constructor(sala: '101' | '102' | '103' | 'Sala EAD', inicioDaTurma: string, fimDaTurma: string, professor: Professor) {

this.sala = sala

this.inicioDaTurma = inicioDaTurma

this.fimDaTurma = fimDaTurma

this.professor = professor

}

adicionarAluno(aluno: Aluno) {

this.alunos.push(aluno)

}

}

class Escola {

nome: string

cidade: string

turmas: Turma[] = []

constructor(nome: string, cidade: string) {

this.nome = nome

this.cidade = cidade

}

adicionarTurma(turma: Turma) {

this.turmas.push(turma)

}

}

const professorDeIngles = new Professor("mariua claudia", "ingles", 29, "102")

const professorDeGrego = new Professor("paulo", "grego", 31, "101")

const professorDeEspanhol = new Professor("jay joul", "espanhol", 28, "103")

const professorDeJapones = new Professor("shintaro", "japones", 19, "Sala EAD")

const aluno1 = new Aluno("Guilherme", 17, "000.000.000-03", "ingles", "focar mais na fala", "102")

const aluno2 = new Aluno("Pedro", 18, "000.000.000-02", "ingles", "ja tem nivel avançado, focar em...", "102")

const aluno3 = new Aluno("Jão", 16, "000.000.000-04", "ingles", "focar mais na escuta", "102")

const aluno4 = new Aluno("Gustavo", 16, "000.000.000-05", "ingles", "focar mais na escuta", "102")

const aluno5 = new Aluno("Ingrid", 16, "000.000.000-07", "ingles", "focar mais na escuta", "102")

const aluno6 = new Aluno("Roberto", 16, "000.000.000-09", "ingles", "focar mais na escuta", "102")

const aluno7 = new Aluno("Roberta", 16, "033.000.000-03", "ingles", "focar mais na escuta", "102")

const aluno8 = new Aluno("Mike", 16, "000.044.000-03", "ingles", "focar mais na escuta", "102")

const aluno9 = new Aluno("Jack", 16, "011.090.000-03", "ingles", "focar mais na escuta", "102")

const aluno10 = new Aluno("Vasco", 16, "000.000.000-00", "ingles", "focar mais na escuta", "102")

const aluno11 = new Aluno("Shelly", 16, "085.050.000-03", "ingles", "focar mais na escuta", "102")

const aluno12 = new Aluno("Dom", 16, "111.990.040-03", "ingles", "focar mais na escuta", "102")

const turma102 = new Turma("102", "01/08/24", "03/03/26", professorDeIngles)

turma102.adicionarAluno(aluno1)

turma102.adicionarAluno(aluno2)

turma102.adicionarAluno(aluno3)

const turma101 = new Turma("101", "01/02/24", "05/05/25", professorDeGrego)

turma101.adicionarAluno(aluno4)

turma101.adicionarAluno(aluno5)

turma101.adicionarAluno(aluno6)

const turma103 = new Turma("103", "23/03/23", "21/03/24", professorDeEspanhol)

turma103.adicionarAluno(aluno7)

turma103.adicionarAluno(aluno8)

turma103.adicionarAluno(aluno9)

const turmaEAD = new Turma("Sala EAD", "11/11/23", "30/04/25", professorDeJapones)

turmaEAD.adicionarAluno(aluno10)

turmaEAD.adicionarAluno(aluno11)

turmaEAD.adicionarAluno(aluno12)

const senac = new Escola("Senac Centro", "Centro de São leopoldo")

senac.adicionarTurma(turma102)

senac.adicionarTurma(turma101)

senac.adicionarTurma(turma103)

senac.adicionarTurma(turmaEAD)

//console.log(senac)

console.log(professorDeEspanhol.informacoes())

console.log(aluno1.informacoes())