**Raoni Martins Raposo. 🡪** Aluno do Fluency sem Limites.

**Exercícios Unidade 5.**

**Funções e Classes em um Código.**

**Obs: Estou livre para escolher realizar os exercícios com JavaScript ou Python.**

1. **Explique com suas palavras os conceitos de herança e polimorfismo.**

Herança.

A herança permite criar classes de objetos “filhas” que herdam recursos (métodos e propriedades) de suas classes “pai”.

No JavaScript, a herança é prototipada. Isso significa que os objetos herdam de outros objetos por meio de uma referência interna chamada prototype.

Por exemplo, podemos criar uma classe Teacher que herda todos os membros da classe Person, adicionando propriedades específicas para professores, como o assunto que ensinam1.

Polimorfismo.

O polimorfismo é a capacidade de uma referência de classe se associar a instâncias de diferentes classes em tempo de execução.

Isso permite que operações se adaptem automaticamente aos objetos aos quais estão sendo aplicadas.

Em JavaScript, podemos alcançar o polimorfismo ao usar métodos comuns em diferentes classes, mesmo que essas classes tenham implementações específicas.

1. **O diagrama UML abaixo possui um dos princípios de orientação a objetos, análise e descreva o seu entendimento sobre a necessidade da utilização nesse cenário.**

A diagram of a computer code

Description automatically generated with medium confidence

Necessidade da Utilização:

A herança é essencial aqui para evitar duplicação de código. Os atributos e métodos comuns a todos os clientes estão definidos na classe “Cliente”, e as subclasses especializam-se com detalhes específicos.

Isso torna o sistema mais organizado, facilita a manutenção e promove a reutilização de código.

Em resumo, o diagrama demonstra como a herança simplifica a modelagem de classes, permitindo que diferentes tipos de clientes compartilhem características comuns enquanto mantêm suas particularidades.

1. **Você foi contratado para desenvolver uma lista de contatos, suas tarefa inclui:**
2. **Definir quais classes vão atender o requisito.**
3. **Identificar quais atributos devem existir em cada classe.**
4. **Identificar quais métodos são necessários desenvolver.**
5. **Produzir um diagrama UML representando sua solução proposta.**
6. **Implementar a solução proposta.**
7. Definido as Classes:

Para uma lista de contatos, podemos considerar as seguintes classes:

Contato: *Representa um contato genérico com atributos como nome, telefone, e-mail etc.*

PessoaFisica: *Subclasse de Contato, com atributos específicos como CPF, RG, data de nascimento etc.*

PessoaJuridica: *Subclasse de Contato, com atributos como CNPJ, razão social etc.*

1. Atributos das Classes:

Contato:

Nome (String)

Telefone (String)

E-mail (String)

PessoaFisica (além dos atributos de Contato):

CPF (String)

RG (String)

Data de Nascimento (Date)

PessoaJuridica (além dos atributos de Contato):

CNPJ (String)

Razão Social (String)

1. Métodos Necessários:

Contato:

*Construtor para inicializar os atributos.*

*Getters e setters para acessar e modificar os atributos.*

PessoaFisica:

*Construtor que chama o construtor da classe base (Contato).*

*Método para validar o CPF.*

PessoaJuridica:

*Construtor que chama o construtor da classe base (Contato).*

*Método para validar o CNPJ.*

1. Diagrama UML:

O diagrama de classes deve representar as relações entre as classes, herança e associações.

(Não Fiz, mais sei Fazer).

1. Implementação:

Implemente as classes em uma linguagem de programação (JavaScrip, Python) seguindo o diagrama UML.

Crie métodos para adicionar, buscar e remover contatos na lista.

1. **Seu restaurante favorito solicitou o seu serviço como desenvolvedor para criar um menu eletrônico, esse menu tem como objetivo facilitar a apresentação dos pratos servidos no local. Para isso, você deve analisar e construir:**
2. **A representação de um prato contendo os atributos para armazenar o nome, descrição, imagem e o valor.**
3. **O restaurante possui um número indefinido de pratos que precisam ser registrados.**
4. **O menu deve apresentar todos os pratos do restaurante e apresentar as informações do prato quando ele for selecionado.**

**Você deve seguir o conceito de orientação a objetos e pensar em uma implementação que separe as responsabilidades de cada necessidade.**

1. Representação de um prato: *Para representar um prato, podemos criar uma classe chamada “Prato” com os atributos necessários*: //{ em JavaScript}

class Prato {

constructor(nome, descricao, imagem, valor) {

this.nome = nome;

this.descricao = descricao;

this.imagem = imagem;

this.valor = valor;

}

}

1. Registro de pratos: *Vamos criar uma classe “****Restaurante****” que armazena uma lista de objetos “Prato”. A classe “****Restaurante****” terá métodos para adicionar pratos, listar todos os pratos e buscar pratos por nome*:

class Restaurante {

constructor() {

this.pratos = [];

}

adicionarPrato(prato) {

this.pratos.push(prato);

}

listarPratos() {

this.pratos.forEach((prato) => {

console.log(`${prato.nome} - R$ ${prato.valor.toFixed(2)}`);

});

}

buscarPratoPorNome(nome) {

return this.pratos.find((prato) => prato.nome.toLowerCase() === nome.toLowerCase());

}

}

1. Menu eletrônico: *O menu eletrônico pode ser implementado em uma página web ou aplicativo. Você pode usar a classe “****Restaurante****” para gerenciar os pratos e exibir as informações na interface do usuário.*

const meuRestaurante = new Restaurante();

// Criando alguns pratos

const lasanha = new Prato("Lasanha", "Massa com molho de tomate e queijo", "lasanha.jpg", 25,99);

const salmaoGrelhado = new Prato("Salmão grelhado", "Salmão fresco com legumes", "salmão.jpg", 32,50);

// Adicionando os pratos ao restaurante

meuRestaurante.adicionarPrato(lasanha);

meuRestaurante.adicionarPrato(salmaoGrelhado);

// Listando todos os pratos

meuRestaurante.listarPratos();

// Buscando um prato por nome

const nomeBusca = "Lasanha";

const pratoEncontrado = meuRestaurante.buscarPratoPorNome(nomeBusca);

if (pratoEncontrado) {

console.log(`Informações do prato '${nomeBusca}':`);

console.log(`Descrição: ${pratoEncontrado.descricao}`);

console.log(`Valor: R$ ${pratoEncontrado.valor.toFixed(2)}`);

} else {

console.log(`Prato '${nomeBusca}' não encontrado.`);

}