### **Exercício: Consumo de API com Programação Orientada a Objetos (POO)**

### **Contexto**

Neste exercício, você deve consumir uma API pública de informações e aplicar conceitos de **POO** para organizar e estruturar o código. O objetivo é criar um programa que permita buscar dados de usuários fictícios e apresentá-los em um formato organizado.

### **API a ser utilizada**

**API pública**: JSONPlaceholder **HTTP Verb (ou Method): GET**  
Endpoint: <https://jsonplaceholder.typicode.com/users>

Este endpoint retorna uma lista de usuários com dados como nome, endereço, empresa, etc.

### **Objetivo do Programa**

1. Consumir a API de usuários.
2. Mapear os dados em classes.
3. Exibir as informações dos usuários de forma organizada.
4. Usar conceitos de **encapsulamento**, **herança**, **interfaces** e **polimorfismo** para estruturar o código.

### **Requisitos**

1. **Classes e Encapsulamento**
   1. Crie uma classe chamada Usuario com os seguintes atributos:
      1. Id (int)
      2. Nome (string)
      3. Email (string)
      4. Telefone (string)
      5. Endereco (uma outra classe chamada Endereco)
   2. Crie a classe Endereco com os atributos:
      1. Rua (string)
      2. Cidade (string)
      3. CEP (string)
   3. Use **propriedades** para encapsular os atributos.
2. **Interface**
   1. Crie uma interface chamada IConsumidorApi com o método:
      1. List<Usuario> BuscarUsuarios() -> Pesquise sobre Task  
         Esse método será responsável por realizar a chamada à API e retornar uma lista de objetos do tipo Usuario.
3. **Implementação**
   1. Crie uma classe chamada ConsumidorApi que implementa a interface IConsumidorApi:
      1. Realize a chamada à API usando a biblioteca HttpClient.
      2. Faça o mapeamento dos dados recebidos para as classes Usuario e Endereco.
4. **Polimorfismo e Herança**
   1. Adicione uma classe derivada chamada UsuarioDetalhado, que herda de Usuario e possui um novo atributo:
      1. Empresa (string)
      2. Implemente um método ExibirDetalhes() sobrescrito que mostra os detalhes adicionais do usuário.
5. **Menu e Interatividade**
   1. O programa deve permitir ao usuário:
      1. Buscar e exibir a lista de usuários.
      2. Selecionar um usuário para ver detalhes mais aprofundados (usando a classe UsuarioDetalhado).

### **Exemplo de Execução**

==== Sistema de Usuários ====

1. Listar usuários

2. Exibir detalhes de um usuário

3. Sair

Escolha uma opção: 1

Lista de Usuários:

1. Leanne Graham

2. Ervin Howell

3. Clementine Bauch

==== Sistema de Usuários ====

1. Listar usuários

2. Exibir detalhes de um usuário

3. Sair

Escolha uma opção: 2

Digite o ID do usuário que deseja visualizar: 1

Detalhes do Usuário:

Nome: Leanne Graham

Email: Sincere@april.biz

Telefone: 1-770-736-8031 x56442

Endereço: Kulas Light, Gwenborough, 92998-3874

Empresa: Romaguera-Crona

### **O que você vai aprender:**

1. **Consumo de API**:
   1. Uso da biblioteca HttpClient.
   2. Deserialização de JSON usando System.Text.Json.
2. **POO**:
   1. Encapsulamento com classes e propriedades.
   2. Uso de interfaces para abstração de lógica de consumo de API.
   3. Herança com classes derivadas (UsuarioDetalhado).
   4. Polimorfismo ao sobrescrever métodos.
3. **Prática de integração**:
   1. Relacionar APIs externas com classes e objetos no programa.
   2. Manipular listas e dados dinâmicos.