# Sistema de Gerenciamento de Restaurante

**Objetivo: Criar um sistema de gerenciamento de restaurante que permita gerenciar mesas, clientes, pedidos e cardápio.**

**Questões:**

- **Diagrama de Classes UML:**

>Você deve criar um diagrama de classes UML que inclua as seguintes classes:

- **Mesa (Table)**
- **Cliente (Customer)**
- **Pedido (Order)**
- **Item do Cardápio (MenuItem)**
- **Restaurante (Restaurant)**

> Cada classe deve incluir atributos e métodos relevantes.

> O diagrama de classes deve ser feito em mermaid no arquivo *resolucao/diagramas.md*

**Implementação em Python:**

- Você traduzir o diagrama de classes UML para código Python.
- Implementar as classes com atributos, métodos e relacionamentos adequados.
- Usar propriedades na declaração dos atributos e type hints na declaração dos métodos não é obrigatório, mas agrega nota.
- Siga as regras e requisitos do sistema abaixo:

## Requisitos do Sistema:

**Classe Mesa (Table):**

**Atributos:**

- **número da mesa (int)**
- **número de assentos (int)**
- **status (str) [livre/ocupada]**

**Métodos:**

- **disponivel(): Retornar True se a mesa estiver livre, False caso contrário.**
- **reservar(): Alterar o status da mesa para 'ocupada'.**
- **liberar(): Alterar o status da mesa para 'livre'.**

**Classe Cliente (Customer):**

**Atributos:**

- **nome (str)**
- **id_cliente (int)**

**Métodos:**

- **fazer_pedido(restaurante, mesa, itens): Realizar um pedido em uma mesa específica.**

**Classe Pedido (Order):**

**Atributos:**

- **cliente (Customer)**
- **mesa (Table)**
- **itens do pedido (list)**
- **status (str) [pendente/servido]**

**Métodos:**

- **adicionar_item(item): Adicionar um item ao pedido.**
- **remover_item(item): Remover um item do pedido.**
- **fechar_pedido(): Alterar o status do pedido para 'servido'.**

**Classe Item do Cardápio (MenuItem):**

**Atributos:**

- **nome (str)**
- **descrição (str)**
- **preço (float)**

**Métodos:**

- **exibir_informacoes(): Exibir informações do item do cardápio.**

**Classe Restaurante (Restaurant):**

**Atributos:**

- **lista de mesas (list)**
- **lista de clientes (list)**
- **lista de pedidos (list)**
- **cardápio (list)**

**Métodos:**

- **adicionar_mesa(numero, assentos): Adicionar uma nova mesa no sistema.**
- **registrar_cliente(nome): Registrar um novo cliente no sistema.**
- **fazer_pedido(cliente, mesa, itens): Realizar um novo pedido.**
- **fechar_pedido(pedido): Fechar um pedido existente.**

## Regras para Implementação:

1. **Verificação de Disponibilidade: Ao fazer um pedido, o método fazer_pedido da classe Restaurant deve verificar se a mesa está disponível.**
2. **Manutenção de Estado: Ao confirmar um pedido, o status da mesa deve ser alterado para 'ocupada'. Ao fechar um pedido, o status deve ser alterado para 'servido'.**
3. **Informações Consistentes: Métodos que exibem informações (como exibir_informacoes na classe MenuItem) devem retornar dados formatados de maneira clara e concisa.**
4. **Gestão de Listas: A classe Restaurant deve manter listas consistentes de mesas, clientes, pedidos e itens do cardápio, garantindo que cada operação de adição ou remoção seja refletida corretamente nas listas correspondentes.**