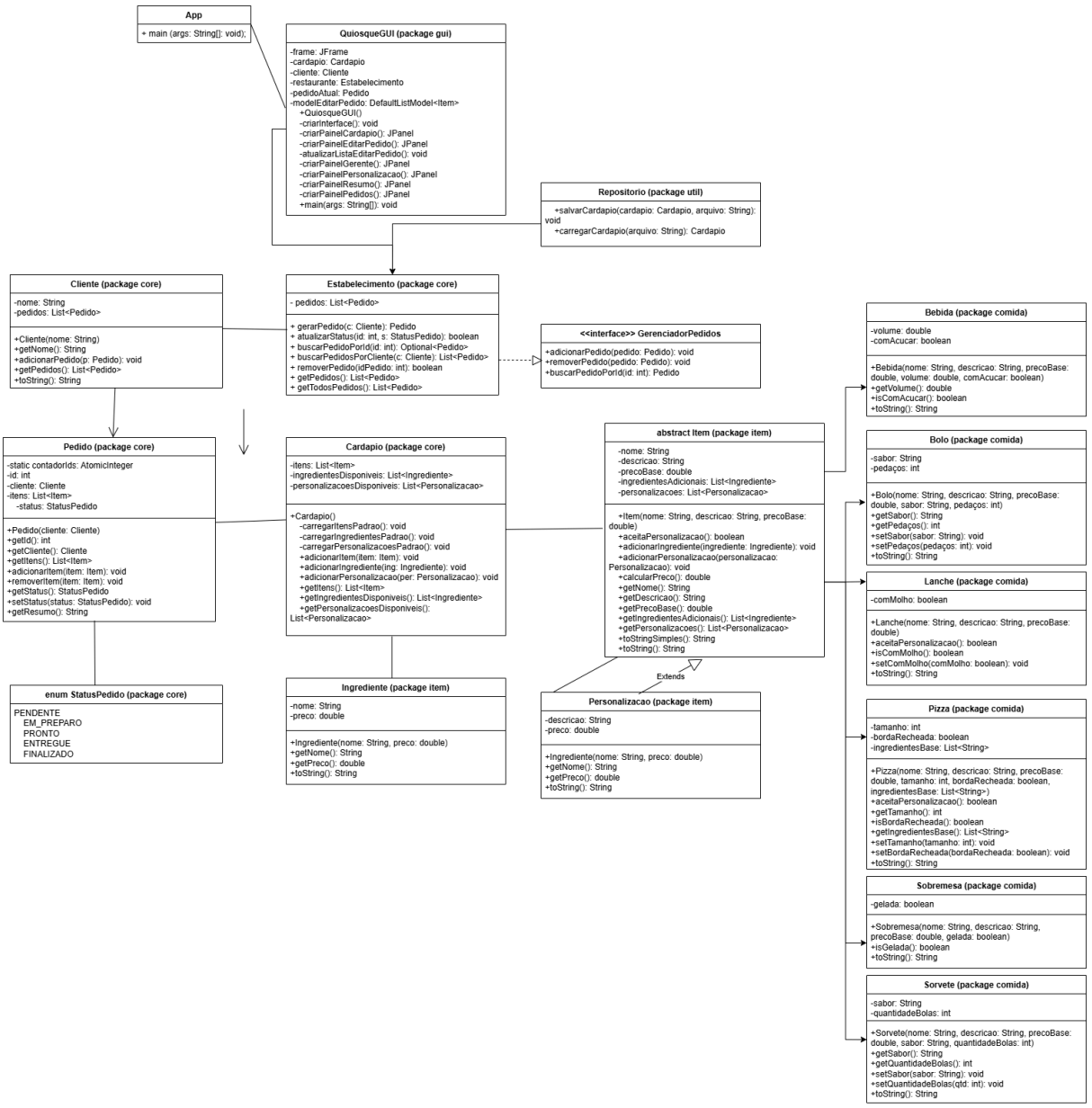
Introdução e Objetivos do Projeto

A crescente demanda por automação e eficiência nos serviços de alimentação tem impulsionado o desenvolvimento de sistemas que otimizem o processo de atendimento em restaurantes e lancherias. Neste contexto, o presente projeto propõe a criação de um sistema orientado a objetos, implementado em Java, que simula um quiosque eletrônico para atendimento ao cliente.

O objetivo principal é proporcionar uma solução modular e reutilizável que permita aos clientes visualizar o cardápio, selecionar e personalizar seus pedidos, além de acompanhar o status do pedido de forma intuitiva. Por outro lado, o sistema também possibilita ao estabelecimento o gerenciamento eficiente dos pedidos em tempo real, desde sua criação até a entrega, garantindo organização operacional e melhoria na experiência do usuário final.

Além disso, o projeto tem a finalidade de aplicar e consolidar conceitos fundamentais da programação orientada a objetos, tais como encapsulamento, herança, polimorfismo, uso de classes abstratas e interfaces, e manipulação de coleções. A proposta contempla ainda uma arquitetura flexível, capaz de ser adaptada para diferentes tipos de estabelecimentos, desde um restaurante universitário até uma lanchonete típica, garantindo escalabilidade e manutenção facilitada.

UML:



Descrição dos Principais Componentes e Decisões de Modelagem

Hierarquia e Classes Abstratas

A modelagem do sistema foi estruturada em torno da abstração dos itens que compõem o cardápio, por meio da classe abstrata **Item**. Esta classe define os atributos essenciais, como nome, descrição e preço base e também um método abstrato para o cálculo do preço final, que pode variar conforme características específicas do item. Essa abstração permite que subclasses concretas implementem suas regras próprias, promovendo polimorfismo e flexibilidade.

Subclasses como **Bebida**, **Lanche** e **Sobremesa** especializam o conceito de item, adicionando atributos e comportamentos específicos. Por exemplo, bebidas possuem tamanhos que influenciam no preço final, lanches podem ter ingredientes adicionais que afetam seu valor, e sobremesas podem possuir características que definem se são servidas geladas ou não. Essa hierarquia facilita a adição futura de novos tipos de produtos, mantendo a coesão e a reutilização do código.

Composição e Relações

A composição foi adotada para representar a relação entre lanches e ingredientes adicionais, permitindo que um lanche seja construído a partir de uma lista flexível de ingredientes, cada um com seu preço específico. Isso possibilita a personalização detalhada dos pedidos, refletindo a realidade dos estabelecimentos que oferecem opções customizáveis.

O **Pedido** é composto por uma lista de itens selecionados pelo cliente, e está associado a um único cliente, refletindo a transação completa de um atendimento. Essa estrutura permite o cálculo dinâmico do valor total do pedido e o acompanhamento individualizado de cada um.

O **Estabelecimento** centraliza o controle operacional, mantendo uma lista de pedidos ativos, gerando números únicos para identificação e permitindo a atualização do status de cada pedido. Essa abordagem facilita o gerenciamento dos fluxos de atendimento e o monitoramento em tempo real dos pedidos em produção.

Encapsulamento e Integridade

Para garantir a integridade dos dados e o controle das operações internas, todas as classes adotam o encapsulamento rigoroso, com atributos privados e métodos públicos para acesso e modificação controlada. Essa prática assegura que os objetos mantenham seu estado consistente e que a lógica de negócio seja preservada, evitando alterações indevidas.

Polimorfismo e Extensibilidade

O método abstrato `calcularPreco()` na classe **Item** é um exemplo clássico de polimorfismo, permitindo que cada subtipo defina como seu preço final é calculado, considerando suas particularidades. Isso torna o sistema altamente extensível, pois novos tipos de itens podem ser introduzidos sem impactar a lógica geral do pedido ou do estabelecimento.

Uso de Enumerações para Controle de Estados

A enumeração **StatusPedido** define claramente os estados possíveis para um pedido — "Em preparo", "Pronto" e "Entregue". Essa definição explícita ajuda a manter a coerência do sistema, facilitando a validação e o controle do fluxo de atendimento, além de possibilitar a extensão para novos status caso seja necessário no futuro.

Coleções e Estrutura de Dados

A escolha pelo uso de coleções do tipo `ArrayList` para armazenar ingredientes, itens de pedido e pedidos ativos reflete a necessidade de uma estrutura dinâmica, que suporte inserções e remoções frequentes durante a operação do sistema. Essa decisão equilibra eficiência e simplicidade, adequada ao contexto de um sistema de atendimento rápido e interativo.