**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA  
UNIDADE EDUCACIONAL PRAÇA DA LIBERDADE**

**Bacharelado em Engenharia de Software**

**Henrique Andrade Cruz**

**Leonardo Viggiano Souza do Carmo**

**TRABALHO PRÁTICO SOBRE PRINCÍPIOS SOLID E PADRÕES DE PROJETO**

Belo Horizonte

2020

**Henrique Andrade Cruz**

**Leonardo Viggiano Souza do Carmo**

**TRABALHO PRÁTICO SOBRE PRINCÍPIOS SOLID E PADRÕES DE PROJETO**

Trabalho de Software apresentado como requisito parcial à aprovação na disciplina Programação Modular

Professor:

Hugo de Paula

Belo Horizonte

2020**SUMÁRIO**

1. Estrutura 4

1.1. Aplicação 4

1.2. Exceções 8

1.3. Testes 8

2. Requisitos 9

2.1. UML 9

# 1. Estrutura

## 1.1. Application

* **Adicional.java**
  + Esse arquivo representa os adicionais que podem ser inseridos na lista de adicionais da classe Pedido. O arquivo é constituído por um Enum que têm como valor o preço de determinado adicional.
  + Opções e seus respectivos valores:
    - VEGANO (3.9)
    - PORCO (5.9)
    - Boi (7.9)
    - CARNE\_EXTRA (4)
    - CREME\_ALHO (1.5)
    - CHILLI (2.5)
    - CROUTONS (2)
    - SHITAKE (6.9)
    - REFRIGERANTE (5.9)
    - O\_CHA\_VERDE (3.9)
    - KO\_CHA\_PRETO (0)
  + Além disso possui o método “preco()” que retorna o preço do adicional passado como parâmetro no construtor.
* **IchiranRamenShop.java**
  + Trata-se de uma classe que herda da interface Restaurante e é constituída pelos atributos:
    - INSTANCE
      * Representa a instância da classe própria classe. É construída no momento que é chamada pela primeira vez. Dessa forma assegura-se que é criada apenas 1 instância da classe, segundo o padrão Singleton.
    - \_pedidosRealizados
      * Uma lista de pedidos do tipo Array.
  + E pelos seguintes métodos:
    - IchiranRamenShop()
      * Construtor da classe, que chama o método “atribuiPedidosRealizados()” para criar inicializar a lista de pedidos que será utilizada posteriormente.
    - getInstance()
      * Retorna o atributo INSTANCE, que representa a própria classe.
    - retornaPedidosRealizados()
      * Retorna a ArrayList inicializada no construtor.
    - atribuiPedidosRealizados()
      * Instancia uma nova ArrayList no atributo “\_pedidosRealizados”.
    - adicionaPedido(Pedido pedido)
      * Verifica se o pedido já existe dentro da lista de pedidos, caso já exista, lançará uma exceção do tipo “ExcecaoAdicionaPedidoDuplicado” e, caso não exista, adicionará o pedido normalmente.
    - removePedido(Pedido pedido)
      * Remove o peido da lista de pedidos realizados.
    - retornaBalancoFinal()
      * Percorre pela lista de pedidos realizados, soma o valor total de cada um e o retorna.
* **Main.java**
  + É o arquivo de reprodução do sistema, que manipula os dados criados e imprime no console o resultado das operações, como balanço final e listagem dos pedidos realizados.
* **Pedido.java**
  + Classe que representa o objeto contendo as informações do pedido a qual possui os atributos:
    - count
      * Contador estático e auto incrementável responsável por garantir que a senha do cliente seja única no sistema.
    - \_senhaCliente
      * Representa a senha (única) do cliente.
    - \_tamanhoPrato
      * Variável do tipo Enum de TamanhoPrato que armazena o tamanho no prato desejado pelo cliente prato.
    - \_adicionais
      * Lista do tipo ArrayList que guarda os possíveis de cada pedido.
    - \_retirado
      * Chave que controla se o pedido foi retirado ou não.
  + E os métodos:
    - Pedido(TamanhoPrato tamanhoPrato, ArrayList<Adicional> adicionais)
      * Construtor da classe, que recebe o tamanho do prato escolhido e uma lista de adicionais já elaborada.
    - retornaSenhaCliente()
      * Retorna a senha (única) do cliente.
    - retornaTamanhoPrato()
      * Retorna o tamanho do prato escolhido pelo cliente.
    - retornaListaAdicionais()
      * Retorna a lista de adicionais escolhidos pelo cliente.
    - foiRetirado()
      * Retorna um valor booleano que define se o pedido foi retirado ou não.
    - atribuiSenhaCliente()
      * Método privado que define a senha (única) do cliente e incrementa o contador em 1.
    - atribuiTamanhoPrato(TamanhoPrato tamanhoPrato)
      * Verifica se o prato já foi retirado, se sim, lançada uma exceção do tipo “ExcecaoProdutoRetirado”, senão, atribuirá o novo tamanho do prato.
    - atribuiAdicionais(ArrayList<Adicional> adicionais)
      * Recebe uma lista de adicionais e então sobrescreve a lista de adicionais do objeto em questão.
    - atribuiFlagRetirado()
      * Configura o atributo privado “\_retirado” para “false”.
    - acrescentaAdicional(Adicional adicional)
      * Verifica se o pedido já foi retirado, se sim, lançará uma exceção do tipo “ExcecaoProdutoRetirado”, senão, é adicionado o novo adicional indicado na lista de adicionais do pedido.
    - removeAdicional(Adicional adicional)
      * Verifica se o pedido já foi retirado, se sim, lançará uma exceção do tipo “ExcecaoProdutoRetirado”, senão, é removido o adicional indicado da lista de adicionais do pedido.
    - retornaPrecoTotal()
      * Obtém o preço total do pedido a partir da soma do preço do tamanho do prato com a soma do preço de cada adicional inserido no pedido.
* **Restaurante.java**
  + Interface que cria um contrato de implementação para os métodos:
    - ArrayList<Pedido> retornaPedidosRealizados()
    - void adicionaPedido(Pedido pedido) throws ExcecaoAdicionaPedidoDuplicado
    - void removePedido(Pedido pedido)
    - double retornaBalancoFinal();
* **TamanhoPrato.java**
  + O arquivo é constituído por um Enum que têm como valor o preço de determinado tamanho de prato.
    - Opções e seus respectivos valores:
      * PEQUENO (3.9)
      * MEDIO (12.9)
      * GRANDE (15.9)

## 

## 1.2. Exceptions

* **ExcecaoAdicionaPedidoDuplicado.java**
  + Classe de exceção customizada que herda de “Exception”, passando a “String” abaixo para o construtor base:
    - “Não é possível inserir o mesmo pedido mais de 1 vez.”
* **ExcecaoProdutoRetirado.java**
  + Classe de exceção customizada que herda de “Exception”, passando a “String” abaixo para o construtor base:
  + “Não é mais possível fazer qualquer tipo de alteração neste pedido, pois o produto já foi retirado.”

## 1.3. Tests

Foram desenvolvidos testes unitários em JUnit para as classes “IchiranRamenShop.java” e “Pedido.java”, respectivamente:

* **IchiranRamenShopTest.java**
* **PedidoTest.java**

# 2. Organização

## 2.1. UML

