Beautiful Manual of Fast Food Restaurant Manager

Introdução

Este documento é um manual que explica o funcionamento do programa Fast Food Restaurant Manager.
Seu Objetivo é possibilitar o uso e facilitar a criação de uma interface amigável ao usuário deste software.

Esboço Estrutural

O programa, orientado a objetos, possui os sequintes elementos:

Cardápio: Possui os pratos oferecidos pelo restaurante, e é único, ou seja, o Restauranto só possui uma única forma de cardápio. A este cardápio pode-se inserir novos produtos, ou retirar produtos que não são mais vendidos.

Prato: É o prato que o restaurante serve. Cada prato possui um nome, por exemplo misto quente, seus ingredientes(pão, queijo, presunto, conforme a receita), e seu preço.

Pedido: Trata-se do pedido que é anotado pelo garçom conforme as escolhas do cliente. Cada pedido é formado por itens.

Item: Cada item possui o prato escolhido e a quantidade, desse prato, desejada. Um item pode conter apenas um único prato.

Caixa: Concluído o pedido, este é levado ao caixa onde o cliente efetua o pagamento do valor total do pedido, para que a preparação do prato seja liberada, e recebe o troco devido. Portanto, o caixa é responsável por abrir um pedido, receber o valor do pedido, devolver o troco devido, e ordenar a preparação dos itens do pedido. O caixa ordena a preparação do pedido ao Cozinheiro.

Cozinheiro: É o encarregado de receber o pedido e programar o robô, que realmente prepara os itens do prato, para devolver o pratos solicitados.

Robô: É uma máquina que prepara pratos conforme solicitado.

Novas Classes:
Do pacote FastFoodRestaurant:

Ingredientes: É uma especialização da classe prato. Foi feito desta forma para que podessemos usar o padrão Decorator.

Filtro: É uma classe criada para o uso de filtros, que foi baseado no padrão Chain of Responsabilities. Cada filho seu é responsável por filtrar uma lista de pratos baseado em um critério.

FiltroTempoPreparo: Filho da classe Filtro. Critério: Tempo de preparo.

FiltroPreco:Filho da classe Filtro. Critério: Preço.

FiltroCaloria: Filho da classe Filtro. Critério: Caloria.

Pacote FastFoodRestaurantWeb-Gerente:

Este pacote foi criado para possibilitar o gerenciamento do restaurante.

GerenteMostraItem: Mostra os itens que serão editados ou excluídos.

GerenteMostraCardapio: Mostra o cardápio, ou seja uma lista dos pratos do restaurante.

GerenteNovoPrato: Cria um novo prato.

GerentePrincipal: Acesso à área de gerenciamento do site.

GerenteRemovePrato: Remove um prato do cardápio.

GerenteRestaurante: Ele é a base da área de gerenciamento do site.

GerenteTask: Responsável pelas ações realizadas na área de gerenciamento.

Funcionamento e Funções para a construção de uma Interface

Antes de mais nada, um restaurante precisa de pratos, refeições, a serem servidos.

Portanto para definir um prato qualquer, exemplo x-salada, deve se escrever assim:

```
objeto := Prato novo: umNome comPreço: umPreço feitoDe: umaListaDeIngredientes.
```

Legenda:

Prato é a classe que possui o método;

novo: receberá o nome do novo prato;

comPreço: recebe o valor que o prato custará ao cliente;

feitoDe: recebe a lista dos ingredientes que fazem parte do prato;

Exemplo:

```
xsalada := Prato novo: 'X-Salada' comPreço: 5.40 feitoDe: #('pão' 'hamburger' 'queijo' 'presunto' tomate' 'alface').
```

Para saber o nome de um prato qualquer, usa-se o método nome, de prato:

```
objeto := objetoPrato  nome.
```

Exemplo:

```
nomeprato := xsalada nome.
```

Para saber o preço de um prato qualquer, usa-se o método preco, de prato:

```
objeto := objetoPrato  preco.
```

Exemplo:

```
precoprato := xsalada preco.
```

Para saber os ingredientes de um prato qualquer, usa-se o método ingredientes, de prato:

```
objeto := objetoPrato  ingredientes.
```

Exemplo:

```
listadeingredientes := xsalada ingredientes.
```

Legenda:

Este método retornará uma lista com todos os ingredientes do prato.

Caso, algum dia, se queira modificar algum desses atributos de um prato, usa-se os seguintes métodos:

nome - para modificar o nome do prato:

```
objetoPrato nome: novonome.
```

Exemplo:

```
xsalada nome: 'McSalada'.
```

ingredientes - modifica a lista de ingredientes necessária para preparar o prato:

```
objetoPrato ingredientes: novalista.
```

Exemplo:

```
xsalada ingredientes: #('pão' 'hamburguer' 'presunto' 'queijo' 'tomate' 'alvace' 'picles').
```

preco - modifica o preço do prato.

```
objetoPrato preco: novopreco.
```

Exemplo:

```
xsalada preco: 8.40.
```

Criados os Pratos, é preciso colocá-los em um cardápio para que o cliente os possa escolher.

Mas antes de mais nada, também é preciso criar um Cardápio, o que é feito a partir de seu respectivo método unico:

```
objeto := Cardapio unico.
```

Exemplo:

```
cardapio := Cardapio unico.
```

Depois de criado os cardápio, pode-se adicionar um novo prato a ele, ou retirar, caso esse prato não seja mais servido.
Essas ações são executadas pelos métodos adiciona e remove:

```
objetoCardapio adiciona: objetoPrato.

objetoCardapio remove: objetoPrato.
```

Legenda:

objetoCardapio: é uma instância da classe Cardapio, como criada logo acima (cardapio).

objetoPrato: é uma instância da classe Prato, como criada logo acima (xsalada).

Exemplo:

```
cardapio adiciona: xsalada.

cardapio remove: xsalada.
```

Um cardápio é capaz de listar os pratos que contém através do método listarPratos que retorna uma lista:

```
objeto := objetoCardapio listarPratos.
```

Exemplo:

```
listadepratos := cardapio listarPratos.
```

Há ainda a possibilidade de esvaziar o cardápio totalmente através do método esvaziar da classe:

```
objetoCardápio esvaziar.
```

Exemplo:

```
cardapio esvaziar.
```

Com o cardápio pronto, o cliente é capaz de fazer o pedido.
Portanto, é preciso solicitar um novo Pedido ao Caixa, pois é ele quem controla o início, pagamento e fim de um pedido. Mas, antes disso tudo, é preciso que haja um Caixa, que é criado pelo método unico da classe Caixa. Pedido e Caixa começam com letras maiúsculas pois se referem a Classes.

```
objeto := Caixa unico.
```

Exemplo:

```
caixa := Caixa unico.
```

Após criado o Caixa, pode-se iniciar um pedido, através do método iniciaPedido:

```
objeto := objetoCaixa iniciaPedido.
```

Exemplo:

```
pedido := caixa iniciaPedido.
```

E fechar um pedido através do método fechaPedido, o que só é feito após o pagamento total do mesmo pelo cliente:

```
objeto := objetoCaixa fechaPedido: valor.
```

Exemplo:

```
troco := cardapio fechaPedido: 50.00.
```

Legenda:

Em Smalltalk a vírgula dos decimais de um número é representada por um ponto.

Depois de criado o Pedido é preciso acrescentar a ele os itens. Mas antes de colocá-los no pedido é preciso criá-los.

Para criar um item usa-se o comando novoComPrato:

```
objeto := Item novoComPrato: objetoprato.
```

Exemplo:

```
item1 := Item novoComPrato: xsalada.
```

Também é possível adicionar uma quantidade, que determina quantas unidades daquele prato o cliente deseja. Para isso usa-se o método adiciona:

```
objetoItem adiciona: valor.
```

Exemplo:

```
item1 adiciona: 3.
```

Com isso o usuário define que deseja 3 unidades de xsalada ,e não 4, como se pode imaginar.

O item também possui outros métodos para facilitar seu manuseio.

Método quantidade retorna a quantidade do prato do item pedido até o momento.

```
objeto := objetoItem quantidade.
```

Exemplo:

```
numeroPedido := item1 quantidade.
```

Já o método preco retorna o preço total do item.

```
objeto := objetoItem preco.
```

Exemplo:

```
precoitem := item1 preco.
```

Caso o cliente tenha feito o pedido de um item com determindo prato, mas depois tenha mudado de idéia, ainda é possível trocar o prato do item através do método prato, que define um prato para um item:

```
objetoItem prato: objetoPrato.
```

Exemplo:

```
item1 prato: xtudo.
```

Ou ainda, se quiser conferir a quantidade de pratos do item pedido, basta usar o método quantidade:

```
objetoItem quantidade.
```

Exemplo:

```
item1 quantidade.
```

Se precisar de alguma informação do prato relacionado ao item, basta usar o método prato, que retorna um objeto Prato:

```
objeto := objetoItem prato.
```

Exemplo:

```
pratoitem1 := item1 prato.
```

Depois de criado o item é hora de colocá-lo em um Pedido. Isso é feito pelo método adiciona da classe Pedido:

```
objetoPedido adiciona: objetoItem.
```

Exemplo:

```
pedido adiciona: item1.
```

Também é possível perguntar a um pedido qual o seu valor total através do método precoTotal:

```
objeto := objetoPedido precoTotal.
```

Exemplo:

```
conta := pedido precoTotal.
```

Ou então, verificar quais são os itens do pedido até o momento, através do método listaItens:

```
objeto := objetoPedido listaItens.
```

Exemplo:

```
listadeitens := pedido listaItens.
```

Após fechado o pedido pelo Caixa, este cria uma instância de cozinheiro pelo método new, se ainda não tiver uma:

```
objeto := Cozinheiro new.
```

Exemplo:

```
cozinheiro = Cozinheiro new.
```

e usa o método prepara, da classe Cozinheiro, para ordenar o preparo dos itens do pedido fechado:

```
objetoCozinheiro prepara: objetoPedido.
```

Exemplo:

```
cozinheiro prepara: pedido.
```

Por fim, temos a classe Robô que na verdade é a classe responsável pelo controle do robô que prepara os pratos. Apesar de já vir pronta com o robô, vamos apenas comentar seus métodos rápidamente. Quem a comanda é a classe cozinheiro.
Ela possui três métodos para seu uso:

novoComEstilo: Cria uma instância(objeto) para o robô e determina um certo estilo de cozinha(Ex.: Japonesa).

prepara: Prepara um ingrediente(Ex.: Arroz, Feijão,etc.).

juntaPrato: Junta os ingredientes preparados pelo método prepara.

Restaram ainda dois métodos que foram colocados para um possível futuro uso na Classe Caixa

cozinheiro - sem passar parâmetros, é o método que retorna o objeto cozinheiro que a classe possui.

Uso:

objetoCaixa cozinheiro.

Exemplo:

caixa cozinheiro.

cozinheiro - com parâmetro do tipo Cozinheiro, é o método que determina um cozinheiro para o caixa.

Uso:

objetoCaixa cozinheiro: objetoCozinheiro.

Exemplo:

caixa cozinheiro: cozinheiro2.

Agora, o sistema do Restaurante possui uma interface WEB, para isso surgiram novas classes e métodos.
Seguem abaixo os novos métodos criados e respectivos exemplos e sintaxes.

Do pacote RestaurantFastFoodManager:

Da classe Filtro:

adicionaFiltro: umFiltro
- Adiciona um filtro ao objeto.

Uso:
objetoFiltro adicionaFiltro: objetoFiltro.

Exemplo:
filtroTempoPreparo adicionaFiltro: filtroCaloria.

filtrar: umaLista
- Executa a função de filtrar respectiva do objeto sobre a lista passada, chama um próximo filtro, se existir, e retorna o array filtrado por essa chamada de método.

Uso:
objetoFiltro filtrar: OrderedCollection.

Exemplo:
filtroCaloria filtrar: listaDePratos.

Das classes FiltroPreço, FiltroCaloria e FiltroTempoDePreparo:

filtrar: umaLista
- Executa o seu filtro específico.Por exemplo, o filtroCaloria seleciona se o prato tem mais ou menos calorias do que a estabelecida.

Uso:
objetoFiltro filtrar: OrderedCollection.

Exemplo:
filtroPreco filtrar: listaPratos.

maiorQue: umValorNumérico
- Estabelece um valor para ser comparado como chão das opções.

Uso:
objetoFiltro maiorQue: umValor.

Exemplo:
filtroTempoPreparo maiorQue: 20.

menorQue: umValorNumérico
- Estabelece um valor para ser comparado como teto das opções.

Uso:

objetoFiltro menorQue: umValor.

Exemplo:

filtroCaloria menorQue: 50.

Da classe Ingrediente:

ingrediente: umPrato

- Recebe um prato.

Uso:

objetoIngrediente ingrediente: objetoPrato.

Exemplo:

    cebola := xsalada.

nome
nome: umNome
preco
preco: umValorNumérico
- São acessores para as variáveis de classe nome e preco, respectivamente.

Uso:
    objetoIngrediente nome "Retorna o nome do ingrediente"
    objetoIngrediente preco: umValor "Atribui uma valor oa ingrediente"

Exemplo:
    tomate preco.
    alface nome: tomate.

Do Pacote FastFoodRestaurantWeb-Gerente

Da classe GerenteMostraItem:

prato
prato: umPrato
- Acessores da variável prato.

Uso:
    objetoGerenteMostraItem prato: objetoPrato.
    objetoPrato := objetoGerenteMostraItem prato.

Exemplo:
    gerenteMostrarCardapio prato: xsalada.
    prato := gerenteMostrarCardapio prato.

Da classe GerenteMostrarCardapio

adicionarPrato
- Inicia o processo de criar um novo prato e agrega-lo..

Uso:

    objetoGerenteMostrarCardapio adicionarPrato.

Exemplo:

apagaPrato: umPrato
- apaga um prato do cardápio.

Uso:

```
objetoGerenteMostrarCardapio apagaPrato: umPrato.
```

Exemplo:
```
gerenteMostrarCardapio apagaPrato: xbacon.
```

editarPrato: umPrato
- Chama um objeto GerenteMostrarItem.

Uso:
```
objetoGerenteMostrarCardapio editarPrato: umPrato.
```

Exemplo:
```
gerenteMostrarCardapio editarPrato: xburguer.
```

mostraPrato: umPrato on: umHtml
- Mostra a informação de um prato no browser.

Uso:
```
objetoGerenteMostrarCardapio mostraPrato: umPrato on: umHtml.
```

Exemplo:
```
gerenteMostrarCardapio mostraPrato: fritas on: html.
```

Da classe GerenteNovoPrato

adicionarPrato: umPrato
- Adicionar um prato ao cardápio. Só agrega se o prato ainda não está no cárdapio.

Uso:
```
objetoGerenteMostrarCardapio adicionarPrato: umPrato.
```

Exemplo:
```
gerenteMostrarCardapio adicionarPrato: beirute.
```

prato
prato: umPrato
- Acessores da variável prato.

Uso:
objetoGerenteNovoPrato prato: objetoPrato.
objetoPrato := objetoGerenteNovoPrato prato.

Exemplo:
gerenteNovoPrato prato.
gerenteNovoPrato prato: xbacon.

Da classe GerentePrincipal

editar
editar: umPrato
- Edita um prato. O primeiro atribui a um prato o retorno de uma chamada da classe GerenteMostraCardapio. O segundo chama a classe mostra item.

Uso:
objetoGerentePrincipal editar.
objetoGerentePrincipal editar: umPrato.

Exemplo:
gerentePrincipal editar.
gerentePrincipal editar: xtudo.

mostraCardapio:
- Mostra o conteúdo do cardápio. Chama a classe GerenteMostrarCardapio.

Uso:
objetoGerentePrincipal mostraCardapio.

Exemplo:
gerentePrincipal mostraCardapio.

Da classe GerenteRemovePrato:

apagaPrato: umPrato

- Apaga um prato do cardápio.

Uso:

objetoGerenteRemovePrato apagaPrato objetoPrato.

Exemplo:

gerenteRemovePrato apagaPrato: xbagunca.

Da classe GerenteRestaurant

canBeRoot

- Indica que está é a classe inicial.

initialize

- Chama um objeto GerenteTask.

Da classe GerenteTask

editarPrato
- Chama um objeto da classe GerenteMostraItem.

mostraCardápio
- Chama um objeto da classe GerentePrincipal.

novoPrato
- Chama um objeto da classe GerenteNovoPrato.

go
- Executa a página WEB.

removePrato
- Chama um objeto da classe GerenteRemovePrato.

Do pacote FastFoodRestaurantWeb

Da classe LojaInfo

info

-Retorna uma mensagem inicial para a página WEB.

Da classe LojaPrincipal

barraDeNavegacao: umHtml

- Cria uma barra de navegação no site com HTML.

Uso:

objetoLojaPrincipal barraDeNavegacao

Exemplo:

lojaPrincipal barraDeNavegacao

caixa: umCaixa

- Atribui um Caix(único) à classe.

Uso:

objetoLojaPrincipal caixa: umCaixa.

Exemplo:
lojaPrincipal caixa: caixa.

cardapio: umCardapio
- Atribui um cardápio à classe

Uso:
objetoLojaPrincipal cardapio: umCardapio.

Exemplo:
lojaPrincipal cardapio: cardapio.

main
- Hospeda as principais funcionalidades do site.

Uso:

objeto := objetoLojaPrincipal main.

Exemplo:

```
mainpage := lojaPrincipal main.
```

mostrarCardapio: umCardapio on: umHtml

- Constrói uma página html contendo o cardápio

Da classe OnlineRestaurant (É a página inicial do sistema. Nela se hospedam as outras.)

description
- Retorna uma descrição, uma String.

Uso:

```
object := objectOnlineRestaurant description.
```

Exemplo:

```
description := onlineRestaurant description.
```

Da classe RestaurantTask.

caixa: umCaixa
- Atribui um caixa a uma variável de classe da classe.

Uso:

```
objetoRestaurantTask caixa: umCaixa.
```

Exemplo:

```
restaurantTask caixa: caixa.
```

cardapio: umCardapio
- Atribui um cardapio a uma variavel de classe da classe.

Uso:

```
objetoRestaurantTask cardapio: umCardapio.
```

Exemplo:

restaurantTask cardapio: cardapio.

montarPrato
- Chama a classe MontaPrato que monta um prato personalizado.

preencheCardapio
- Chama a classe LojaPrincipal e passa pra ela o cardápio.

Da classe RestauranteUtils

geraIdPrato
- Gera um id pro prato transformando seu nome em uma string sem espaços e em letras minúsculas

Da classe VisualizarCardapio

cardapio
cardapio: umCardapio
- São acessores da vairável de classe cardapio, da classe.

main: umaLojaInfo
- Em princípio recebe a mensagem de Boas-Vindas.

mostraCardapio
- Chama a função mostrPratos da classe passando pra ela a lista de pratos do cardapio.

mostraPratos:

- Mostra uma lista de pratos

mostraPrato

- Mostra um prato

pedido: umPedido

- Atribui umPedido à variável de calsse pedido

renderOnPrato

mostra os detalhes de um prato

Da classe VisualizarPedido

pedido: umPedido

- Atribui umPedido à variável de calsse pedido

renderItem: umItem on: umHtml

- Mostra um item na tela

style

- Define o estilo da tabela

troco

- Calcula e retorna o troco

valor

- Recebe um valor e o atribui à variável valor da classe.

Da classe VisualizarPrato

adicionaPedido: umPrato

- Adiciona um prato a um pedido.

pedido: umPedido

- Atribui um pedido à variável pedido da classe.

prato: umItem

- Adiciona um item à variável item da classe.