Universidade Federal do Ceará - Campus Russas
Programação Orientada a Objetos – 2018.2
Professor: Marcio Costa Santos
Trabalho 1 - The warm up!

Capítulo 1

Introdução

Programação orientada à objetos é um dos paradigmas de programação mais utilizados para o desenvolvimento de aplicações, especialmente aplicações web.
Dois momentos são críticos no desenvolvimento de sistemas que utilizam programação orientada à objetos são a concepção e a tradução do modelo conceitual para código fonte.
Neste projeto, será dado um diagrama de classes UML, assim como a especificação conceitual do sistema e os alunos em equipes de até 2 participantes devem traduzir esse diagrama UML para código fonte com o auxílio das especificações providas.
A entrega do trabalho se dará na data definida no SIGAA e pelo SIGAA através de um arquivo zipado nomeado com o nome dos alunos em letras minúsculas, sem caracteres especiais, juntamente com suas matrículas separadas por underline, por exemplo: mariadorosario_00001_joaodasilva_00002.
O arquivo zipado deve conter todo o código fonte, devidamente organizado e comentado; e um arquivo executável (.jar) do projeto. Para geração do .jar os alunos podem encontrar um manual passo a passo aqui. Para os alunos que usam eclipse, uma outra opção pode ser encontrada aqui e aqui
Todas as duplas deverão apresentar o código e responder a uma serie de possíveis questionamentos do professor a respeito da implementação em data divulgada no SIGAA após a entrega do trabalho em ordem definida por sorteio.
Os trabalhos serão avaliados segundo os seguintes critérios

Corretude - 4,0 Neste quesito será verificado a adequação do código fonte com o diagrama UML provido. Os nomes de classes, atributos e métodos devem coincidir assim como sua visibilidade.

Funcionabilidade - 3,0 Neste quesito será considerado se os métodos dos objetos estão implementados de uma maneira correta e se realizam as operações desejadas

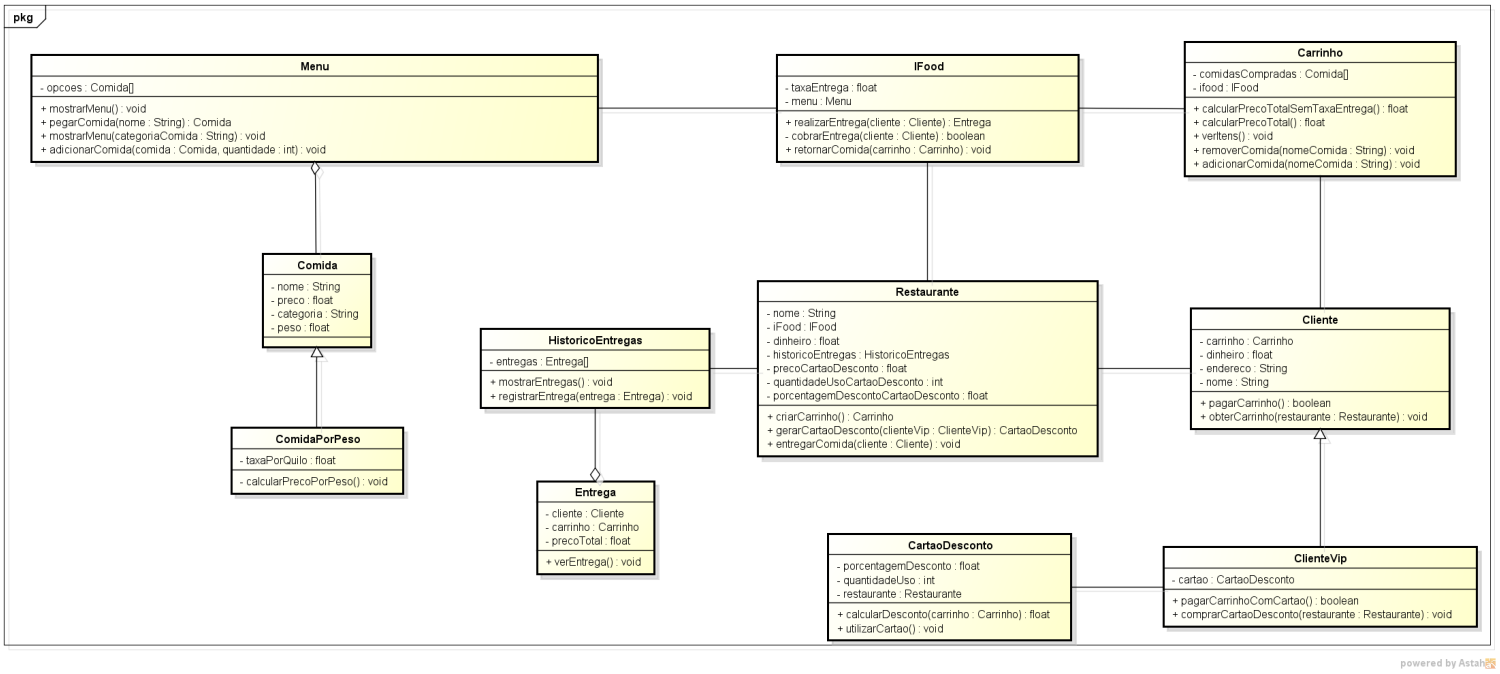
Organização -1,0 Neste quesito será analisado a organização do código.

Tratamento de Execeções - 1,0 Neste quesito será considerado a resistência do código a falhas e má utilização por parte do usuário.

Documentação - 1,0 Neste quesito será considerado os comentários gerais no código.

Capítulo 2

Diagrama de Classes UML



Classe Comida

Esta classe representará uma opção de comida no restaurante, ela será utilizada para compor o menu do restaurante e posteriormente poder ser vendida.

Atributos

Os atributos desta classe representarão características simples das comidas do restaurante.

Exemplo

nome: Feijão de corda

preço: R$ 2,00

categoria: Grão

peso: 200g

Classe ComidaPorPeso

Esta classe herdará de Comida, portanto também será uma Comida, porém se difere no quesito de que seu preço irá variar de acordo com o peso da mesma, ou seja, é necessário realizar um cálculo para isso.

Atributos

O atributo taxaPorQuilo irá ser utilizado para calcular o preço da comida de acordo com seu peso

Exemplo

nome: Feijão de corda (herdado)

preço: R$ 1,66 (herdado) (taxaPorQuilo * peso)

categoria: Grão (herdado)

peso: 200g (herdado)

taxaPorQuilo: R$ 8,30

Métodos

- calcularPrecoPorPeso() : void

Este método será responsável por efetuar o cálculo do preço através da fórmula taxaPorQuilo * peso, ou seja, ele deve ser executado logo quando o objeto for criado. O resultado desta operação será o peso da comida

Classe Menu

Esta classe representará o menu do restaurante, ela será responsável por gerenciar todas as opções de comida no restaurante, como também a entrada e saída de comida no mesmo.

Atributos

O único atributo desta classe será responsável por guardar vários objetos do tipo Comida, que serão todas as porções de comida disponíveis no restaurante no momento, lembrando que poderão haver duas ou mais porções da mesma comida. As comidas podem ser tanto objetos da classe Comida como da ComidaPorPeso.

Exemplo

opcoes: {Feijão de corda, Frango assado, Arroz integral, Feijão de corda}

OBS: O exemplo acima é meramente um exemplo de como ficaria o vetor com elementos adicionados.

Métodos

+ mostrarMenu() : void

Este método deve mostrar na tela todas as opções de comida disponíveis no restaurante.

OBS: Não é para apresentar uma mesma comida mais de uma vez no menu

Um exemplo de saída deste método seria:

---Menu---

Feijão de corda – R$ 2,00 – Grão – 200g

Frango assado – R$ 8,00 – Aves – 300g

+ mostrarMenu(categoriaComida : String) : void

Este método funciona do mesmo modo que o método acima, porém só apresentará as comidas da categoria que for especificada pelo usuário na variávei categoriaComida.

+ adicionarComida(comida : Comida, quantidade : int) : void

Este método adicionará o objeto comida de acordo com o valor da variável quantidade. Por exemplo, se quantidade = 2, então teremos duas daquela comida no menu.

OBS: Não se pode adicionar 0 comidas

OBS: Não se pode adicionar quando o vetor de opções do menu já estiver cheio

+ pegarComida(nome : String) : Comida

Este método irá buscar por uma unidade da comida que estiver o seu nome igual a variável nome especificada no parâmetro.Caso você ache uma unidade desta comida você deve removê-la do menu e após isso retorne a comida achada.

Classe IFood

Esta classe representará um sistema de Ifood de um restaurante, ela será responsável por todas as ações relacionadas entrega e cobrança das entregas de comida.

Atributos

Esta classe possuirá uma referência ao menu do restaurante, podendo assim ter acesso as opções de comida disponíveis. Há também uma variável referente a taxaEntrega, que é o preço cobrado ao realizar uma entrega de comida.

Métodos

+ realizarEntrega(cliente : Cliente) : Entrega

Este método irá montar uma Entrega, contendo seu preço, cliente e o carrinho com todas as comidas que foi pedido pelo dado cliente. Após montar a entrega você irá cobrar o cliente, caso o cliente tenha dinheiro suficiente você deverá retornar a entrega feita, caso contrário você deverá retornar todas as comidas do carrinho do cliente de volta no menu de opções e esvaziar o carrinho do cliente, pois ele não tinha dinheiro para pagar.

+ retornarComida(carrinho : Carrinho) : void

Este método será irá realocar todos os itens do carrinho de volta para o menu do restaurante, ou seja, irá remover itens do carrinho e adicionar de volta ao menu.

- cobrarEntrega(cliente : Cliente) : boolean

Este método tratará da questão de cobrar o cliente o devido dinheiro que deverá ser pago. Ele dirá ser o cliente tem dinheiro suficiente ou não para pagar a conta.

Caso o cliente tenha dinheiro para pagar, você irá debitar o valor a ser cobrado do cliente e retornará true, devido ao sucesso da operação.

Caso contrário, você deverá retornar o valor false, devido ao insucesso da operação.

Classe Carrinho

Esta classe irá representar um carrinho onde o cliente usará para adicionar suas comidas no sistema de ifood. Há a possibilidade de adicionar,remover e ver itens do carrinho, além de ver o preco que você irá pagar pelos itens. Você pode ver o preço de todos os itens que você adicionou sem contar a taxa de entrega, como também poderá ver com a taxa de entrega.

Atributos

iFood: Este atributo será uma referência a um sistema de ifood de um restaurante, ou seja, você poderá, somente, realizar ações com os itens presentes neste ifood.

comidasCompradas: Este atributos é um vetor de Comidas/ComidasPorPeso. É aqui que você irá adicionar todos os seus itens.

Métodos

+ verItens() : void

Você deverá mostrar todos os itens atualmente presentes no seu carrinho.

+ calcularPrecoTotalSemTaxaEntrega() : float

Este método irá calcular o total dos preços de todos os itens no seu carrinho, não incluindo a taxa de entrega.

+ calcularPrecoTotal() : float

Este método irá calcular o total dos preços de todos os itens no seu carrinho contando com o preço de entrega.

+ adicionarComida(nomeComida : String) : void

Este método será responsável por adicionar itens ao seu carrinho. Você deve procurar no menu do ifood se há um item com o nome especificado no parâmetro nomeComida, se houver ao menos um item deste você deve pegar o item do menu e adicionar ao seu carrinho, alocando-o em seu vetor de comidas.

+ removerComida(nomeComida : String) : void

Este método deverá procurar por um item na sua mochila cujo o nome é igual ao parâmetro nomeComida, se houver uma comida em sua mochila com o nome especificado você deve removê-lo de sua mochila e devolvê-lo ao menu do ifood.

Classe Cliente

Esta classe representará um cliente real de um restaurante com ifood, este cliente quando em um ifood poderá pegar um carrinho e adicionar quantos itens quiser, porém deverá pagar pelos itens que pegar com seu dinheiro, ou seja, cada cliente possuirá um carrinho de um restaurante e se relacionará com o sistema de ifood deste restaurante.

Atributos

nome: Representará o nome do cliente

endereço: Representará o endereço do cliente

dinheiro: Representará o dinheiro do cliente

carrinho: Representará a instância de um carrinho de um restaurante

OBS: Você só pode possuir um carrinho se este carrinho estiver relacionado a um restaurante.

Métodos

+ obterCarrinho(restaurante : Restaurante) : void

Este método irá pedir um carrinho ao restaurante especificado no parâmetro do método, se você conseguir o carrinho você irá agora usá-lo para gerenciar seus itens. Este carrinho inicialmente não possui itens.

OBS: Todos os itens em seu carrinho agora serão relacionados ao restaurante especificado no parâmetro.

+ pagarCarrinho() : boolean

Este método irá se relacionar com o ifood do carrinho do cliente e irá pagar o preço total do carrinho ao restaurante. Você deverá retornar true se a operação de pagamento foi bem sucedidade, ou seja, se o cliente teve dinheiro suficiente, caso contrário você deve retornar false.

OBS: A operação de pagamento só poderá ser realizada, se e somente se, o cliente já possuir um carrinho.

Classe ClienteVip

Esta classe irá herdar de Cliente, ou seja, todos os atributos e métodos de Cliente também irão fazer parte desta classe. A diferença aqui é que este ClienteVip possui a possibilidade de comprar um cartão de desconto, que irá prover desconto nas compras deste cliente em um Restaurante específico.

OBS: Um cartão de desconto obtido em um restaurante A não pode ser utilizado em um restaurante B.

Métodos

+ pagarCarrinhoComCartao() : boolean

Este método irá pagar todo o carrinho, porém com um decréscimo de acordo com o desconto que o cartão provê.

Exemplo:

precoCarrinho = 50; (itens + taxa de entrega)

porcentagemDesconto = 5;

Logo você irá pagar R$ 47.50 devido aos 5% de desconto.

OBS: Você só poderá pagar com seu cartão se este ainda estiver habilitado a ser utilizado, ou seja, o atributo quantidadeUso em CartaoDesconto deve ser maior que 0. Logo após efetuar o pagamento você deve decrementar este atributo quantidadeUso em uma(1) unidade.

+ comprarCartaoDesconto(restaurante : Restaurante) : void

Este método será responsável pela compra do Cartão de desconto no restaurante localizado no parâmetro da função. Este cartão será vinculado com este restaurante, e somente com este. Se o cliente tiver dinheiro suficiente, você deverá decrementá-lo de acordo com o preco do cartão estabelecido no restaurante e no final da operação você deve dar o cartão do cliente. Caso o dinheiro do cliente não seja suficiente, ele não irá comprar o cartão.

Classe CartaoDesconto

Esta classe representará um cartão que provê desconto nas compras de comida em um, e somente um, restaurante. Este cartão possui uma porcentagem de desconto, que é utilizado para descontar sobre o preço total de uma compra, possui um tempo de vida fixo que é diminuido a cada uso e possui uma referência para o restaurante que o aceita.

Métodos

+ utilizarCartao() : void

Este método irá decrementar a vida útil do cartão(quantidadeUso) em uma(1) unidade.

+ calcularDesconto(carrinho : Carrinho) : float

Este método receberá um carrinho e irá devolver o valor do desconto que o cartão oferece baseado no preço total(itens + taxaEntrega) do carrinho recebido como parâmetro.

Exemplo:

precoCarrinho = 50; (itens + taxa de entrega)

porcentagemDesconto = 5;

Logo, o valor de retorno deste método será 2.50, que é o valor a ser descontado do total.

Classe Entrega

Esta classe representará uma entrega feita pelo ifood de um restaurante quando um cliente pede a entrega e a paga com sucesso. Esta classe contém o Cliente que fez o pedido da entrega, o Carrinho do cliente que fez o pedido e o valor total ganho naquela entrega.

OBS: valor total é o total recebido dos itens do carrinho + taxa de entrega

Métodos

+ verEntrega() : void

Este método irá simplesmente mostrar todos os dados referentes a um objeto da classe Entrega.

Classe HistoricoEntregas

Esta classe representa um histórico de todas as entregas já realizadas pelo sistema ifood de um restaurante. Esta classe possui um vetor que irá armazenar todas as entregas de um restaurante.

Métodos

+ mostrarEntregas() : void

Este método irá mostrar todas as informações referentes a cada entrega já feita.

+ registrarEntrega(entrega : Entrega) : void

Após uma entrega ser feita com sucesso pelo restaurante ela deve ser armazenada no registro de entregas, este método recebe uma entrega que foi realizada e o aloca no vetor de entregas já realizadas pelo restaurante.

Classe Restaurante

Esta classe é uma das mais importantes deste projeto, pois tudo irá ser relacionado a ela. O restaurante será responsável por gerenciar o seu sistema de ifood, seu históricо de entregas, gerações de cartões de desconto, criações de carrinhos para os clientes e entregar os pedidos de um cliente.

Atributos

nome: Representa o nome do restaurante

dinheiro: Representa todo o dinheiro que o restaurante ganha com as entregas de comida

precoCartaoDesconto: Representa o preço a ser pago quando um ClienteVip tenta comprar um cartão de desconto

quantidadeUsoCartaoDesconto: Representa o número de vezes que um cartão daquele restaurante poderá ser utilizado.

porcentagemDescontoCartaoDesconto: Representa a quantidade, em %, de desconto que o cartão do restaurante irá oferecer para o ClienteVip

iFood: Representa o sistema de ifood do restaurante

historicoEntregas: Representa o histório de entregas do restaurante

Métodos

+ criarCarrinho() : Carrinho

Este método irá gerar um carrinho com o sistema de ifood do restaurante e irá retorná-lo. Este carrinho será então utilizado por um cliente do restaurante.

+ gerarCartaoDesconto(clienteVip : ClienteVip) : CartaoDesconto

Este método irá gerar um cartão de desconto para o clienteVip recebido como parâmetro. Se o cliente possuir dinheiro suficiente para pagar pelo cartão, então você deve debitar este valor do cliente e acresentá-lo no dinheiro do restaurante.

+ entregarComida(cliente : Cliente) : void

Este método irá se comunicar com o ifood do restaurante solicitando-o que realize a entrega do pedido. Caso a entrega seja bem sucedida, você deverá acrescer o dinheiro do restaurante com o dinheiro ganho na entrega e no final você deverá salvar esta entrega no seu histórico de entrega, para fins de consulta futura. Caso a entrega não seja bem sucedidade, você não precisa realizar nenhuma ação.