**Relatório Técnicas de Construção de Programas**

**Trabalho Prático Etapa II – Prof: Erika Fernandes Cota**

**Grupo: Alexandre Fabian, Thiago Nitschke, Rodrigo Okido**

**INTRODUÇÃO:**

Desenvolver um projeto orientado a objetos a partir de uma especificação recebida. Neste caso, a possível solução que será apresentado neste relatório trata-se de um Restaurante. Para mostrar a solução pensada, será demonstrado toda a criação dos diagramas de classe e sequencia construídos para que seja possível a visualização de uma possível implementação do objetivo desejado. Para entendermos melhor o que cada objeto no diagrama representa, detalharemos eles de uma forma mais esclarecedora abaixo.

**\*\* CLASSES:**

Uma breve introdução das classes do projeto. Para melhor entendimento dos principais métodos do projeto, eles serão detalhados em outro setor mais abaixo.

**>> Funcionario – Abstract Class**

Funcionário é uma das classes fundamentais do projeto. Sendo uma classe abstrata, é nesta classe que será associado tudo que um funcionário que trabalha em um restaurante possui em comum. Neste caso, temos o nome, um salário e um código. Este código possui o objetivo de dar a cada funcionário, um código único que será usado para login ao sistema. Ressaltando que, funcionário é uma classe abstrata, e por isso, não é possível fazer uma instancia desta classe.

**>> Atendente – Herda a classe Funcionario**

Esta classe possuirá todos os métodos relativos ao Atendente. Todo Atendente poderá verificar mesas livres, ocupar mesa, reservar ou cancelar Mesas.

**>> Gerente – Herda a classe Funcionario**

Esta classe possuirá todos os métodos relativos ao Gerente. Todo gerente poderá iniciar turno, finalizar turno, verificar estoque, checar folha de pagamento e checar relatório de gastos.

**>> Garçom** **- Herda a classe Funcionario**

Esta classe possuirá todos os métodos relativos ao Garçom. Todo garçom pode abrir um pedido, atualizar pedido e fechar uma mesa.

**>> Cozinheiro** – **Herda a classe Funcionario**

Esta classe possuirá todos os métodos relativos ao cozinheiro. Todo cozinheiro neste caso, poderá iniciar uma preparação, ou finalizar ela.

**>> AuxiliarCozinha** **– Herda a classe Funcionario**

Esta classe possuirá todos os métodos necessários associados ao Auxiliar de Cozinha. Todo Auxiliar libera e checa uma mesa.

**>> Mesa** – Todas as mesas do restaurante possuem um setor e um número de lugares. Para melhor identificarmos cada objeto do tipo Mesa criado, associamos cada objeto com um código especifico. Assim, é facilitado a busca e o controle de como e onde está cada objeto a partir apenas deste código.

**>> Pedido** – O pedido está associado diretamente ao garçom. Cada pedido criado está associado a uma mesa e uma lista de pedidos. Esta lista será preenchida com os códigos associados a cada item do cardápio (disponível no arquivo disponibilizado no moodle). Cada item no cardápio, possui um determinado código, e assim, apenas o código será usado para montar a lista de pedidos do cliente. Nesta classe, podemos adicionar um pedido (a partir de uma mesa), excluir (a partir de uma mesa) e pegar a lista de códigos (ou seja, os pedidos do cardápio).

**>> Prato –** A classe prato serve para criamos cada item que fará a composição do cardápio do restaurante. Todo prato possui um código, nome, custo, valor de venda, tempo de preparo, uma lista de ingredientes e um tipo. Lembrando que cada objeto prato criado, pode ser uma entrada, prato principal, saladas, bebida, sobremesa ou sopas (depende do construtor).

**>> Estoque** – Estoque é uma classe para guardarmos cada ingrediente do restaurante.

**>> Ingredientes** – Esta classe é para criarmos cada ingrediente que conterá o restaurante. Todo ingrediente possui um nome, quantidade e um determinador.

**>> Salário** – A classe salário possui o propósito de facilitar o manuseio dos salários de cada funcionário do restaurante. Tendo como atributos o cargo composto pelo funcionário, o valor fixo de salário dele, e a porcentagem de lucro de extras.

**>> BancoDeDados** – Esta é a classe responsável pela inicialização do restaurante. Ela guardará todas as informações necessárias para o funcionamento do sistema. Fará a inicialização do cardápio, das mesas e os salários.

**>> AdministracaoDeContaRestauranteImpl –** Ponto de entrada do sistema. Gera o menu para utilização do mesmo. A partir do login de algum dos atores do sistema, é possível utilizar todas funcionalidades do sistema.

**>> Turno -**

**\*\* PRINCIPAIS FUNCIONALIDADES DO SISTEMA:**

***1 - Atendente***

**Cancelar Reserva:** Para cancelar uma mesa é chamada a função “cancelarReservaMesa (mesa: Mesa, Date : date). A função chama a função “checarReserva (date:Date)” da classe Mesa, e verifica se o determinado objeto mesa desejado com aquela determinada data possui alguma reserva. Caso positivo, o método retorna ***true***, ***false*** caso contrário.

**Reservar Mesa:** Para reservar uma mesa é utilizada um método chamado “reservarMesa (mesa: Mesa, Date : date)”. O método verifica se naquela determinada mesa desejada possuirá alguma reserva antes. Para fazer esta verificação, é chamada um método “checarReserva (Date: date)” da classe Mesa. Caso a mesa desejada já possua uma reserva realizada, o método “checarReserva” deve retornar ***true*** e, consequentemente, não pode ser autorizada a reserva naquela mesa. Caso contrário, o método retornará ***false***, e consequentemente, a reserva poderá ser realizada tranquilamente alterando o estado booleano do atributo “reservado” para ***true*** através do método “setReservaStatus” da classe Mesa.

***2 – Auxiliar de Cozinha***

**Liberar Mesa**: Para liberar uma mesa, isto está diretamente relacionado a ação de “Limpar Mesa” (abaixo). Para verificar se uma mesa pode ser liberada novamente, a mesa obrigatoriamente deve estar antes no status que está liberado para limpeza. Estando liberado para limpeza, o auxiliar executa a limpeza, e ao final dela, libera seu uso novamente alterando seu status.

**Limpar Mesa**: Para limpar mesa, o auxiliar acessa o método “checarMesasParaLimpeza (List<Mesa> : ArrayList)”. Recebendo uma lista de mesa por parâmetro, o método irá varrer esta lista e verificar quais mesas estão com seu status disponível para limpeza. Para pesquisar o status, será buscado pelo método “getStatus()” da classe Mesa. Depois que o método varrer toda a lista, será exibida quais mesas estão disponíveis para limpar para o Auxiliar. Podendo assim, executar após a ação de **Liberar Mesa** (Explicado acima).

***3 – Cozinheiro –***

**Iniciar preparação:**

**Finalização de pedido:**

***4 – Garçom***

**Atender uma Mesa:**

**Atualizar Pedido:**

**Fechar Mesa:**

***5 – Gerente***

**Finalizar Turno:**

**Folha de Pagamentos:**

**Iniciar Turno:**

**Relatório de Gastos:**

**Relatório Estoque:**