UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DA COMPUTAÇÃO

Relatório Myfood 1

Aluno: Ezequiel Pereira Alves
Matrícula: 22210893
Curso: CIÊNCIA DA COMPUTAÇÃO

Relatório Myfood

1. Visão Geral

O sistema de gerenciamento abrange o gerenciamento de usuários, produtos, pedidos, empresas e entregas utilizando um design orientado a objetos em Java. O sistema é dividido em várias classes, cada uma com responsabilidades específicas.

2. Descrição das Classes e Métodos

2.1. Classe Sistema

Responsabilidades: Gerenciar usuários, restaurantes e produtos, além de carregar e salvar dados em um arquivo.

Métodos:
- public void adicionarUsuario(Usuario usuario):
Descrição: Adiciona um novo usuário ao sistema.
Detalhes: Verifica se o CPF e o e-mail são únicos. Se forem válidos, o usuário é adicionado ao Map.

- public Usuario getUsuario(int id):
Descrição: Retorna um usuário pelo seu ID.
Detalhes: Utiliza o Map para acessar o usuário correspondente.

- public void adicionarRestaurante(Restaurante restaurante):
Descrição: Adiciona um novo restaurante ao sistema.
Detalhes: Adiciona o restaurante ao Map com um ID gerado automaticamente.

- public Restaurante getRestaurante(int id):
Descrição: Retorna um restaurante pelo seu ID.
Detalhes: Utiliza o Map para acessar o restaurante correspondente.

- public void criarProduto(int restauranteId, Produto produto):
Descrição: Cria um novo produto para um restaurante específico.
Detalhes: Verifica a existência do restaurante. Adiciona o produto à lista de produtos do restaurante e atualiza o Map com um ID gerado automaticamente.

- public void editarProduto(int restauranteId, Produto produto):
Descrição: Edita um produto existente em um restaurante específico.

Detalhes: Verifica a existência do restaurante e do produto. Atualiza o produto na lista de produtos do restaurante e no Map.

- public List listarProdutos(int restauranteId):
Descrição: Lista todos os produtos de um restaurante específico.
Detalhes: Obtém o restaurante pelo ID e retorna a lista de produtos associada.
Aqui estão as descrições para o código apresentado:

-criarPedido recebe os IDs de um cliente e uma empresa. Ele valida se ambos existem, garantindo que um dono de empresa não faça um pedido em sua própria empresa. Também verifica se o cliente já possui um pedido em aberto na mesma empresa. Se todas as validações forem bem-sucedidas, cria um novo pedido e o adiciona às listas relevantes, retornando o número do pedido criado.

-adicionarProduto permite adicionar um produto a um pedido existente. Ele verifica se o pedido está em aberto, se o produto existe e se pertence à empresa associada ao pedido. Se todas as condições forem atendidas, o produto é adicionado ao pedido.

-fecharPedido altera o estado de um pedido para "preparando", encerrando-o. Ele primeiro verifica se o pedido existe e, caso positivo, atualiza seu estado.

-getPedidos permite acessar informações sobre um pedido específico, como cliente, empresa, estado, valor e produtos. Ele garante que o pedido exista e que o atributo solicitado seja válido, lançando exceções apropriadas para condições inválidas.

-removerProduto é responsável por remover um produto de um pedido. Ele verifica se o nome do produto é válido e se o pedido está em aberto. Caso contrário, lança exceções. Se o produto for encontrado no pedido, ele é removido.

-liberarPedido muda o estado de um pedido para "pronto", permitindo que seja enviado para entrega. Antes disso, verifica se o pedido existe e se já foi preparado.

-getNumeroPedido obtém o número de um pedido associado a um cliente e uma empresa, com base em um índice fornecido. Ele valida se o cliente e os pedidos existem, garantindo que o índice fornecido seja válido.

-obterPedido retorna o ID do pedido que um entregador pode retirar para entrega. Ele valida se o entregador existe, se está associado a alguma empresa e se está em uma entrega ativa. Retorna o pedido mais antigo que está pronto para entrega ou lança uma exceção se não houver pedidos disponíveis.

-criarEntrega inicia uma entrega para um pedido específico. Ele verifica se o pedido está pronto, se o entregador não está em outra entrega e se é um entregador válido associado à

empresa do pedido. Se todas as condições forem atendidas, um novo objeto de entrega é criado e registrado.

-getEntrega recupera informações sobre uma entrega específica com base em seu ID e um atributo solicitado, validando se a entrega existe e se o atributo é válido.

-entregar marca uma entrega como concluída, alterando o estado do pedido associado para "entregue". Ele valida se a entrega existe e se o pedido correspondente também está disponível.

-getIdEntrega busca e retorna o ID da entrega associada a um pedido específico, lançando uma exceção se nenhuma entrega for encontrada.

2.2. Classe Restaurante
Responsabilidades: Armazenar informações sobre o restaurante e gerenciar produtos.

Métodos:
- public void adicionarProduto(Produto produto):
Descrição: Adiciona um produto à lista de produtos do restaurante.
Detalhes: Adiciona o produto à lista interna de produtos (List).

- public void editarProduto(Produto produto):
Descrição: Edita um produto existente no restaurante.
Detalhes: Substitui o produto existente na lista de produtos com base no ID do produto.

- public Produto getProduto(int idProduto):
Descrição: Retorna um produto pelo ID.
Detalhes: Pesquisa o produto na lista de produtos pelo ID.

- public List listarProdutos():
Descrição: Retorna a lista de todos os produtos do restaurante.
Detalhes: Retorna a lista interna de produtos (List).

2.3. Classe Produto
Responsabilidades: Representar um produto com atributos como ID, nome, valor e categoria.

Métodos:
- public int getId():
Descrição: Retorna o ID do produto.
Detalhes: Acessa o campo privado id.

- public void setId(int id):
Descrição: Define o ID do produto.

Detalhes: Atualiza o campo privado id.

- public String getNome():
Descrição: Retorna o nome do produto.
Detalhes: Acessa o campo privado nome.

- public void setNome(String nome):
Descrição: Define o nome do produto.
Detalhes: Atualiza o campo privado nome.

- public double getValor():
Descrição: Retorna o valor do produto.
Detalhes: Acessa o campo privado valor.

- public void setValor(double valor):
Descrição: Define o valor do produto.
Detalhes: Atualiza o campo privado valor.

- public String getCategoria():
Descrição: Retorna a categoria do produto.
Detalhes: Acessa o campo privado categoria.

- public void setCategoria(String categoria):
Descrição: Define a categoria do produto.
Detalhes: Atualiza o campo privado categoria.

2.4. Classe Usuario
Responsabilidades: Representar usuários com atributos comuns e especializações específicas para clientes e donos de restaurante.

Métodos em Usuario:
- public int getId():
Descrição: Retorna o ID do usuário.
Detalhes: Acessa o campo privado id.

- public void setId(int id):
Descrição: Define o ID do usuário.
Detalhes: Atualiza o campo privado id.

- public String getNome():
Descrição: Retorna o nome do usuário.
Detalhes: Acessa o campo privado nome.

- public void setNome(String nome):

Descrição: Define o nome do usuário.
Detalhes: Atualiza o campo privado nome.

- public String getEmail():
Descrição: Retorna o e-mail do usuário.
Detalhes: Acessa o campo privado email.

- public void setEmail(String email):
Descrição: Define o e-mail do usuário.
Detalhes: Atualiza o campo privado email.

Métodos em DonoDeRestaurante:
- public String getCpf():
Descrição: Retorna o CPF do dono do restaurante.
Detalhes: Acessa o campo privado cpf.

- public void setCpf(String cpf):
Descrição: Define o CPF do dono do restaurante.
Detalhes: Atualiza o campo privado cpf.

- public List getRestaurantes():
Descrição: Retorna a lista de restaurantes gerenciados pelo dono.
Detalhes: Acessa o campo privado restaurantes.

Métodos em Cliente:
- public List getPedidos():
Descrição: Retorna a lista de pedidos feitos pelo cliente.
Detalhes: Acessa o campo privado pedidos.

2.5 Classe Empresa

Responsabilidades: Representar uma empresa com atributos como ID, tipo, nome, endereço e lista de entregadores.

Métodos:
- Empresa(String tipoEmpresa, String nome, String endereco): Construtor que inicializa uma nova empresa com tipo, nome e endereço fornecidos. O ID é gerado automaticamente a partir de um contador estático.

- List<Entregador> getEntregadores(): Retorna a lista de entregadores associados à empresa. Se a lista de entregadores ainda não tiver sido criada, ela é inicializada.

- void setEntregadores(List<Entregador> entregadores): Define a lista de entregadores da empresa.

- int getId(): Retorna o ID único da empresa.

- String getNome(): Retorna o nome da empresa.

- String getEndereco(): Retorna o endereço da empresa.

- String getTipoEmpresa(): Retorna o tipo de empresa (ex: mercado ou farmácia).

- boolean isMercado(): Método abstrato que indica se a empresa é do tipo "mercado". Precisa ser implementado pelas subclasses.

- boolean isFarmacia(): Método abstrato que indica se a empresa é do tipo "farmácia". Precisa ser implementado pelas subclasses.

- void setAtributo(String atributo, String valor): Método abstrato que define um valor para um atributo específico. Precisa ser implementado pelas subclasses.

- String getAtributo(String atributo): Retorna o valor de um atributo específico, como nome, endereço ou tipo de empresa. Caso o atributo fornecido seja inválido, lança uma exceção IllegalArgumentException.

Observações:
- A classe é abstrata, o que significa que ela serve como base para subclasses específicas de empresas, como "Mercado" ou "Farmacia".
- A lista de entregadores é criada sob demanda, ou seja, apenas quando o método getEntregadores é chamado.

2.5.1 Classe Farmacia

Responsabilidades: Representar uma farmácia como um tipo específico de empresa, com atributos adicionais como se está aberta 24 horas e o número de funcionários.

Métodos:
- Farmacia(String tipoEmpresa, String nome, String endereco, boolean aberto24Horas, int numeroFuncionarios): Construtor que inicializa uma farmácia com tipo de empresa, nome, endereço, se está aberta 24 horas e o número de funcionários.

- boolean isMercado(): Implementação do método abstrato da classe Empresa, retornando false, indicando que não é um mercado.

- boolean isFarmacia(): Implementação do método abstrato da classe Empresa, retornando true, indicando que é uma farmácia.

- void setAtributo(String atributo, String valor): Método que define um valor para um atributo específico. Atualmente não possui implementação.

- boolean getAberto24Horas(): Retorna se a farmácia está aberta 24 horas.

- int getNumeroFuncionarios(): Retorna o número de funcionários da farmácia.

- String getAtributo(String atributo): Retorna o valor de um atributo específico, como se está aberta 24 horas ou o número de funcionários. Se o atributo não for reconhecido, delega a chamada para o método getAtributo da classe Empresa.

Observações:
- A classe herda de Empresa e adiciona características específicas de uma farmácia, como o atributo aberto24Horas.
- O método getAtributo lida com atributos específicos de farmácias, além de chamar a implementação da classe base para outros atributos.

2.5.6 Classe Mercado

Responsabilidades: Representar um mercado como um tipo específico de empresa, com atributos adicionais como horário de abertura, horário de fechamento e o tipo de mercado.

Métodos:
- Mercado(String tipoEmpresa, String nome, String endereco, String abre, String fecha, String tipoMercado): Construtor que inicializa um mercado com tipo de empresa, nome, endereço, horário de abertura, horário de fechamento e o tipo de mercado (supermercado, minimercado ou atacadista).

- boolean isMercado(): Implementação do método abstrato da classe Empresa, retornando true, indicando que é um mercado.

- boolean isFarmacia(): Implementação do método abstrato da classe Empresa, retornando false, indicando que não é uma farmácia.

- String getAbre(): Retorna o horário de abertura do mercado no formato HH:MM.

- String getFecha(): Retorna o horário de fechamento do mercado no formato HH:MM.

- String getTipoMercado(): Retorna o tipo de mercado (supermercado, minimercado ou atacadista).

- void setAbre(String abre): Define o horário de abertura do mercado.

- void setFecha(String fecha): Define o horário de fechamento do mercado.

- void setAtributo(String atributo, String valor): Define um valor para um atributo específico. Se o atributo for "abre" ou "fecha", define o horário correspondente. Outros atributos são tratados pela classe base.

- String getAtributo(String atributo): Retorna o valor de um atributo específico, como horário de abertura, fechamento ou tipo de mercado. Se o atributo não for reconhecido, delega a chamada para o método getAtributo da classe Empresa.

Observações:
- A classe herda de Empresa e adiciona características específicas de mercados, como os horários de funcionamento e o tipo de mercado.
- O método setAtributo permite modificar atributos específicos, enquanto o método getAtributo trata tanto atributos próprios quanto os da classe base.

2.7 Classe Pedido

Responsabilidades: Representar um pedido, contendo informações sobre o cliente, a empresa, os produtos, o estado do pedido e seu valor total. A classe também permite a adição, remoção de produtos e a mudança de estado do pedido.

Atributos:
- `int numero`: Número único do pedido gerado automaticamente com um contador.
- `String cliente`: Nome do cliente que fez o pedido.
- `String empresa`: Nome da empresa de onde o pedido foi feito.
- `String estado`: Estado atual do pedido, que começa como "aberto" e pode ser alterado durante o processo.
- `List<Produto> produtos`: Lista de produtos que compõem o pedido.
- `float valor`: Valor total do pedido, atualizado conforme produtos são adicionados ou removidos.

Métodos:
- `Pedido(String cliente, String empresa)`: Construtor que inicializa um pedido com o nome do cliente, nome da empresa, estado "aberto", uma lista vazia de produtos e valor inicial zero.

- `int getNumero()`: Retorna o número do pedido.

- `String getCliente()`: Retorna o nome do cliente que fez o pedido.

- `String getEmpresa()`: Retorna o nome da empresa relacionada ao pedido.

- `String getEstado()`: Retorna o estado atual do pedido.

- `List<Produto> getProdutos()`: Retorna a lista de produtos do pedido.

- `float getValor()`: Retorna o valor total do pedido.

- `void adicionarProduto(Produto produto)`: Adiciona um produto à lista de produtos do pedido e atualiza o valor total com o preço do produto.

- `void finalizarPedido()`: Altera o estado do pedido para "preparando".

- `boolean removerProdutoPorNome(String nomeProduto)`: Remove um produto da lista de produtos pelo nome. Se o produto for encontrado e removido, o valor do pedido é atualizado, e o método retorna `true`. Caso contrário, retorna `false`.

- `void setEstado(String pronto)`: Altera o estado do pedido para um valor fornecido.

Observações:
- O número do pedido é gerado automaticamente usando um contador estático que incrementa a cada novo pedido.
- O valor do pedido é mantido atualizado conforme produtos são adicionados ou removidos.
- O estado do pedido pode ser "aberto", "preparando", ou outro estado conforme o método `setEstado`.

2.8 Classe Entrega

Responsabilidades: Representar uma entrega, associando-a a um pedido, entregador e destino, além de permitir a obtenção de atributos relacionados ao pedido e ao entregador.

Métodos:
- Entrega(int id, int idPedido, int idEntregador, String destino): Construtor que inicializa a entrega com um ID, ID do pedido, ID do entregador e o destino da entrega.

- int getId(): Retorna o ID da entrega.

- int getIdPedido(): Retorna o ID do pedido associado à entrega.

- int getIdEntregador(): Retorna o ID do entregador associado à entrega.

- String getDestino(): Retorna o destino da entrega.

- String getAtributo(String atributo): Retorna o valor de um atributo específico relacionado à entrega, como o ID do pedido, o nome do entregador, o cliente, a empresa associada ao pedido ou o destino da entrega.

  - Caso o atributo seja "pedido", retorna o ID do pedido.

  - Caso o atributo seja "entregador", carrega o usuário correspondente ao ID do entregador e retorna o nome, caso o entregador exista.

  - Caso o atributo seja "cliente", carrega o pedido correspondente ao ID do pedido e retorna o cliente associado, caso o pedido exista.

  - Caso o atributo seja "empresa", retorna a empresa associada ao pedido, caso o pedido exista.

  - Caso o atributo seja "destino", retorna o destino da entrega.

  - Se o atributo não for reconhecido, lança uma exceção IllegalArgumentException.

Observações:
- A classe utiliza dois métodos auxiliares, `GuardaPedidos.carregarPedidos()` e `SalarUsuarios.carregarUsuarios()`, para carregar os pedidos e usuários salvos.

- A limpeza dos mapas de pedidos e usuários (`pedidos.clear()` e `usuarios.clear()`) é feita após o uso dos dados.

- A verificação de validade dos atributos "entregador", "cliente" e "empresa" é feita através da verificação da existência do pedido ou do entregador, lançando exceções em casos inválidos.

2.9 Classe Entregador

Responsabilidades: Representar um entregador, herdar atributos e comportamentos da classe Usuário e incluir informações específicas relacionadas ao veículo e à placa.

Atributos:
- `String veiculo`: O tipo de veículo utilizado pelo entregador.
- `String placa`: A placa do veículo do entregador.

Métodos:

- `Entregador(String nome, String email, String senha, String endereco, String veiculo, String placa)`: Construtor que inicializa um entregador com nome, e-mail, senha, endereço, tipo de veículo e placa.

- `String getVeiculo()`: Retorna o tipo de veículo do entregador.

- `void setVeiculo(String veiculo)`: Define o tipo de veículo do entregador.

- `String getPlaca()`: Retorna a placa do veículo do entregador.

- `void setPlaca(String placa)`: Define a placa do veículo do entregador.

- `String getEmail()`: Retorna o e-mail do entregador, utilizando o método da superclasse Usuário.

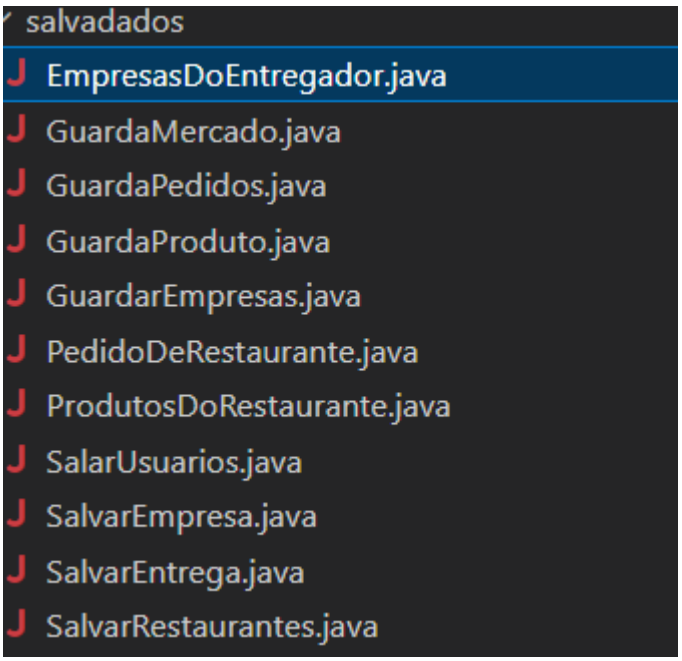
- `String getAtributo(String atributo)`: Retorna o valor de um atributo específico relacionado ao entregador.
  - Caso o atributo seja "placa", retorna a placa do veículo.
  - Caso o atributo seja "veiculo", retorna o tipo de veículo.
  - Para outros atributos, chama o método `getAtributo` da superclasse Usuário.

- `boolean podeCriarEmpresa()`: Retorna `false`, indicando que o entregador não tem permissão para criar uma empresa.

- `boolean ehEntregador()`: Retorna `true`, indicando que a instância é um entregador.

Observações:
- A classe Extensora, `Usuario`, fornece atributos e métodos comuns, como nome, e-mail e senha.
- A classe pode ser usada para gerenciar entregadores em um sistema de entrega, onde cada entregador tem um veículo associado e pode ser identificado pela sua placa.

2.10 SalvarDados



As entidades na classe `salvadados` são responsáveis por gerenciar a persistência dos dados do sistema. Elas desempenham um papel crucial, pois são encarregadas de salvar e carregar informações essenciais, garantindo que os dados sejam armazenados de forma segura e possam ser recuperados de maneira eficiente. Essas entidades permitem que o sistema mantenha seu estado entre as execuções, facilitando a continuidade das operações e melhorando a experiência do usuário.

3. Design e Arquitetura
Encapsulamento: Cada classe e método é projetado para proteger e gerenciar dados internos, expondo apenas o necessário para interações externas.

Modularidade: A divisão em classes separadas (Sistema, Restaurante, Produto, Usuario e suas subclasses) promove uma arquitetura modular e facilita a manutenção e a extensibilidade.

Persistência: Utiliza arquivos de texto para salvar e carregar dados. Embora simples, isso permite que o sistema preserve o estado entre execuções. Considerações futuras podem incluir a migração para um banco de dados mais robusto.

Validação: Implementa verificações para garantir a integridade dos dados, como CPF e e-mail únicos. Validações adicionais garantem que os produtos e usuários sejam gerenciados corretamente.