**Projeto Maid Coffee – Plataforma para Registro de Pedidos.**

Bárbara Cervígni/12522225785, Danilo dos Santos Freitas/12522219074, Matheus Martins Garcia/125221101218, Miguel Rios da Silva/12522217469 e Rodrigo Cantele Cuencas/1252222044

Ciência da Computação / Ciência de Dados, Campus Paulista, Universidade Anhembi Morumbi

e-mail: 1252222044@ulife.com.br

**Resumo:** O artigo a seguir demonstra as etapas envolvidas na criação de uma aplicação em Java que tem como objetivo gerenciar e registrar os pedidos de uma cafeteria, solucionando problemas comuns como erros e cobranças incorretas de forma precisa e interativa. A aplicação é composta por três telas principais: “Menu Inicial”, “Fazer Pedidos” e “Verificar Pedidos”, e duas telas secundarias: “Tela de Erro” e “Tela de Conclusão”. Durante o desenvolvimento do projeto, foram definidos os requisitos básicos para a criação da aplicação, foram utilizados diversos softwares de design gráfico, como *Canvas, VistaCreate, IloveIMG, removebg e PNGWing*, para desenvolver a interface, e a codificação foi realizada utilizando a *IDE NetBeans*, com o auxílio do *JDK 8*. Ao longo do processo de desenvolvimento, todos os requisitos básicos estabelecidos foram explorados e atendidos, resultando em melhorias significativas, como a precisão no registro e cálculo dos pedidos. Essas melhorias trazem benefícios como a redução de erros, o aumento da eficiência operacional e a melhoria na satisfação do cliente para a cafeteria.

**Palavras-chave:** aplicação em Java, Registro, Pedidos, Interface, Método.

***Abstract:*** *The following article demonstrates the steps involved in creating a Java application that aims to manage and register the orders of a coffee shop, solving common problems like errors and incorrect charges in an accurate and interactive way. The application is composed of three main screens: “*Menu Inicial*”, “*Fazer Pedidos*” and “*Verificar Pedidos*”, and two secondary screens: “*Tela de Erro*” and “*Tela de Conclusão*”. During the development of the project, the basic requirements for the creation of the application were defined, several graphic design software were used, such as Canvas, VistaCreate, IloveIMG, removebg and PNGWing, to develop the interface, and the coding was done using the NetBeans IDE, with the help of JDK 8. Throughout the development process, all the basic requirements established were explored and met, resulting in significant improvements, such as the accuracy in the registration and calculation of orders.*

*These improvements bring benefits such as reduced errors, increased operational efficiency, and improved customer satisfaction for the coffee shop.*

***Keywords:*** *Java application, Registration, Orders, Interface, Methods.*

**Introdução**

O seguinte trabalho demonstra a criação de um programa em Java e os processos realizados para a obtenção dos resultados. O programa consiste na criação de uma plataforma para gerenciar e efetuar os registros dos pedidos feitos em uma cafeteria, dividido em uma estrutura de três telas principais e duas secundárias, oferecendo funcionalidades essenciais para o registro e acompanhamento dos pedidos.

A primeira tela é o "Menu Inicial", onde o usuário pode escolher entre as opções "Fazer Pedidos" e "Verificar Pedidos". Ao escolher a opção "Fazer Pedidos", o usuário é direcionado para a tela correspondente, onde poderá adicionar itens ao pedido de forma intuitiva com o auxílio de uma lista de opções do cardápio. O usuário poderá selecionar os itens desejados, preencher informações adicionais como o nome do funcionário e nome do cliente, além de adicionar observações, se necessário. Esses detalhes são armazenados em uma lista de pedidos, que ficará disponível para consulta na tela de "Verificar Pedidos". Caso nenhum dos campos disponíveis seja preenchido e o botão "Confirmar?" seja acionado, aparecerá a "Tela de Erro", que consiste em uma tela de aviso que comunica ao usuário que o pedido não foi efetuado. Por outro lado, se as informações forem devidamente preenchidas, aparecerá a "Tela de Conclusão", que também é uma tela de aviso confirmando ao usuário que seu pedido foi armazenado com sucesso. Na tela de "Verificar Pedidos", o usuário terá acesso aos pedidos registrados até o momento. Nessa visualização, é possível ver algumas informações de cada pedido, incluindo o nome do funcionário, nome do cliente e valor total do pedido.

O programa possui uma interface amigável que incentiva a interação e oferece o benefício adicional de salvar e revisar os pedidos. Esse recurso permite que a equipe de funcionários seja mais ágil em sua abordagem com o cliente e permite que sejam mais eficientes no processamento de pedidos.

No decorrer do projeto, foi explorada a implementação dessas funcionalidades por meio da utilização de diferentes estruturas de controle, classes e métodos, oferecendo a todo o grupo um aumento significativo no conhecimento e desenvolvimento de aplicações realizadas com a linguagem Java. O objetivo final é garantir que o programa seja executado de forma eficiente e eficaz, buscando simplificar o processo de recebimento e gerenciamento de pedidos, proporcionando uma experiência limpa e estruturada para os funcionários da cafeteria e uma melhor experiência para os clientes.

**Materiais e métodos**

Nesta seção, será apresentado o processo para desenvolver a aplicação. Primeiramente, fizemos uma análise dos principais requisitos da aplicação, entre eles as exigências do professor da UC, a identificação das telas necessárias e suas funcionalidades. Esses requisitos foram essenciais para orientar o desenvolvimento do sistema e garantir que ele atendesse ao seu propósito.

Em seguida, iniciamos o processo de design da interface do sistema, utilizando softwares como *Canvas* e *VistaCreate* para criar as telas e os conteúdos visuais da aplicação, e os softwares *IloveIMG*, *removebg* e *PNGWing* para aperfeiçoar o design. Com essas ferramentas foi possível desenhar e implementar as diferentes telas e elementos de interação do sistema de acordo com as necessidades e requisitos definidos anteriormente.

Por fim, partimos para a etapa de codificação do sistema da aplicação. Essa etapa consistiu no desenvolvimento das classes, métodos e objetos necessários para a implementação das funcionalidades do sistema e da interface gráfica. Para isso, utilizamos a *IDE NetBeans*, bastante reconhecida e empregada no desenvolvimento de aplicações Java juntamente com o *JDK 8 (Java Development Kit)*.

Durante as etapas do processo, foram realizados testes para assegurar o funcionamento da aplicação e prevenir possíveis problemas ou erros, além de comunicação constante entre os membros do grupo. Foram organizadas reuniões semanais usando a plataforma *Discord* para compartilhar informações, esclarecer dúvidas e definir os próximos passos.

**Resultados**

Nesta etapa daremos continuidade com a documentação de nosso projeto, aprofundando-se nas classes que o compões e esclarecendo seu funcionamento s citando de qual forma atendemos os requisitos pedidos.

**Estrutura geral do código** – A forma como estruturamos nosso código ficou dividida entre os pacotes *View.backend* (que abriga a lógica de programação, na qual daremos foco nesta documentação) e *View* (responsável pelo *frontend*, ou seja, as telas de nossa aplicação).

A primeira classe que gostaríamos de apresentar é a “Pedido”, sendo ela a responsável por registrar o nome do funcionário responsável pelo pedido, nome do cliente, salvar eventuais observações, código do pedido, o pedido em sí, número de itens e o valor total. Todos estes atributos são definidos na figura abaixo:

Texto

Descrição gerada automaticamente

Figura 1: Classe Pedido

Um padrão interessante a ser ressaltado, e que se repetirá ao longo de todo nosso código, é que os atributos estão todos privados, sendo assim, eles somente serão acessíveis por meio dos métodos, *getters* e *setters*. A aplicação do encapsulamento é um dos pontos chaves da programação orientada a objeto, tendo em mente que diminui a dependência direta nos atributos e oferece um controle mais seguro sobre o acesso e manipulação de dados.

**Métodos construtores** – Os métodos construtores utilizados na classe Pedido estão especificados na figura 2 e desempenham um papel fundamental na inicialização da classe.

O primeiro construtor serve para inicializarmos uma classe com os atributos “zerados” e que poderão ser inicializados por meio dos métodos *setters* futuramente. Por outro lado, o segundo método construtor, receberá como entrada os dados necessários para inicializar todos os atributos definidos da classe (*idPedido, NomeFuncionario, Nome cliente* etc.). Dessa forma garantimos que todos os atributos serão definidos e inicializados corretamente.

Texto, Aplicativo

Descrição gerada automaticamenteFigura 2: Métodos construtores da classe Pedido

**Métodos, *getters* e *setters*** – Uma vez explicados os atributos da classe e seus métodos construtores, podemos abordar os métodos, *getters* e *setters* que serão utilizados para se comunicar com a classe e os objetos que serão instanciados futuramente.

Boa parte do processamento de dados que poderia ser realizado nesta classe foi dividido entre as classes *“*GerenciadorPedidos*”* e *“ArrayObjetos”*, as quais abordaremos futuramente, e por conta disso o funcionamento desta classe ficou relativamente simples.

Os métodos que foram criados são um conjunto padrão de *getters* e *setters*, utilizados somente para acessar e alterar os valores dos atributos, por conta de sua simplicidade, não estremos nos aprofundando neles.

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

Figura 3: Métodos construtores da classe Pedido

A próxima classe é chamada de GerenciadorPedidos, responsável por gerenciar os pedidos do sistema, por meio dos objetos criados a partir dela.

Esta classe começa declarando os atributos IdPedidos e *arrObj* novamente como privados, a fim de manter o encapsulamento enquanto o método construtor inicializa o IdPedido com o valor 0.

Em seguida temos o principal método da classe, sendo ele o novoPedido*()*, sendo responsável por criar um novo pedido criando uma instância da classe pedido, estabelecida anteriormente, e atribuindo os valores do método aos atributos por meio de seus *getters e setters*.

A lógica deste método, é que caso o retorno que o método recebe para o atributo id será -1, o programa entende que se trata de um novo pedido, logo é preciso incrementar o id em um (uma das solicitações do projeto) e sendo criado um pedido.

Caso seja diferente de -1, o programa entende que se trata da alteração de um pedido já existente e através do método *editObjeto* altera o pedido.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Figura 4: Classe GerenciadorPedidos

Juntamente com isso temos o método *selecionarPedido()* que interage com a classe *ArrayObjetos*, onde guardamos as informações para que possam ler lidas e utilizadas ao longo de todo o programa.

A classe *ArrayObjetos* tem como função gerenciar o “*Array*” de objetos do tipo Pedido, que estão sendo gerados pela classe GerenciaPedido. Primeiramente ela possui dois atributos privados: “*TamanhoArray*”, que como o nome sugere, representa o tamanho do *Array* e “*Array*” que efetivamente armazena os objetos do tipo “Pedido”.

Texto

Descrição gerada automaticamente

Figura 6: Classe *ArrayObjetos*

O construtor “*ArrayObjeto*()” inicializa o “tamanhoArray” com o valor 1 e cria um novo *array* de pedido com este tamanho. Em seguida, temos o método “*addObjeto*()” que recebe um objeto do tipo pedido como parâmetro e o adiciona ao *array* criado anteriormente. Se o último elemento do *array* for nulo, significa que ainda há espaço disponível no *array*. Nesse caso, o método percorre o *array* de trás para frente até encontrar a primeira posição nula ou a posição inicial.

Em seguida, ele adiciona o objeto nessa posição. Caso o último elemento não seja nulo, será necessário incrementar o tamanho do *array* por meio do método *“setTamanhoArray()”* e adiciona-lo na última posição criada.

Texto

Descrição gerada automaticamente

Figura 7: Métodos da classe *ArrayObjetos*

O método *“setTamanhoArray()”* por sua vez recebe um número inteiro e define o tamanho do *array* para este valor e cria um *array* temporário, copiando os elementos presentes no *array* original para o *array* temporário. O tamanho do *array* temporário é o mesmo do *array* original, pois ele precisa comportar temporariamente os dados que estavam no primeiro *array* enquanto é criado um novo com maior capacidade.

Dependendo do atributo "*tamanhoArray*" em relação ao *array* temporário, podem acontecer três situações. A primeira é quando o valor de "*tamanhoArray*" for maior que o tamanho do *array* temporário. Nesse caso, os dados do *array* temporário são copiados de volta para o array atual, sendo que os valores são copiados até a nova posição "*tamanhoArray* - 1". A segunda situação ocorre quando o valor de "*tamanhoArray*" for menor ou igual ao tamanho do *array* temporário. Nesse caso, os dados do *array* temporário são copiados de volta para o *array* atual, sendo que os valores são copiados até a posição "*tamanhoArray* - 1". A terceira situação é quando o valor de "*tamanhoArray*" for igual ao tamanho do *array* temporário. Nesse caso, não é necessário copiar os dados de volta, pois o *array* atual já possui a capacidade adequada.

Os demais métodos possuem o funcionamento mais simples, porém continuam tendo extrema importância para o funcionamento do código, tendo em mente que permitem a manipulação do *array*. O método “*selecionarObj*()” retorna o objeto de um *array* por meio do id, “*getTamanhoArray*()” simplesmente retorna o tamanho do *array*, “*getArray*()” retorna o *array* de objetos do tipo pedido e por fim “*editObjeto*()” substitui o objeto atual de um *array*, usando o id para se localizar, por outro fornecido como parâmetro.

**Discussão**

Por fim, todo o desenvolvimento da aplicação se deu pensando nas adversidades que podem ocorrer ao efetuar um pedido em algum estabelecimento, como por exemplo o pedido chegar errado ou, mesmo que esteja correto, o valor ser cobrado de forma incorreta na hora do pagamento. A aplicação ajuda a resolver isso fornecendo aos funcionários de uma cafeteria uma forma melhor e mais confiável de anotar e calcular com maior precisão os itens e valores do pedido. Com a implementação desse projeto, a cafeteria poderá adquirir diversos benefícios como por exemplo a redução de erros nos pedidos, o aumento da eficiência operacional e a melhoria da satisfação do cliente, e ao mesmo tempo garantir que os funcionários executem o registro de pedidos de forma mais eficiente, interativa e intuitiva.

Concluímos que foram explorados e realizados todas os desafios e requisitos definidos pelo grupo ao longo do desenvolvimento deste programa, destacando-se a importância na melhoria da experiência do cliente e na otimização das operações do MaidCoffee.

**Referências**

MARKENZON. Estruturas de dados e seus algoritmos. Rio de Janeiro: LTC, 2010. E-book.Disponível em:https://integrada.minhabiblioteca.com.br/#/books/978- 85-216-2995-5/. Acesso em: 10 jun. 2023.

MANZANO, J.A.N.G., OLIVEIRA, J.F. Algoritmos: lógica para desenvolvimento de programação de computadores. 25ª ed. São Paulo: Érica, 320 p. ISBN 9788536502212. Acesso em: 17 jun. 2023

RIBEIRO, J.A. Introdução à programação e aos algoritmos. Rio de Janeiro: LTC, E-book. Disponível em:https://integrada.minhabiblioteca.com.br/#/books/97885 21636410/. Acesso em: 18 jun. 2023.

DEITEL, Harvey M.; DEITEL, Paul J. Java: como programar. 6. ed. São Paulo: Pearson Prentice Hall, 2005. xl, 1110 p. + 1 CD-ROM ISBN 8576050196.