Relatório do Projeto POO Financial Services (POOFS)

03/12/2024

Beatriz Rodrigues, nº 2022146346

Rita Ramos, nº 2022257681

Índice

Visão geral

O projeto desenvolve uma aplicação em Java para gestão financeira, chamada POO Financial Services (POOFS), que facilita o gerenciamento de clientes, faturas e produtos em uma empresa. A aplicação oferece funcionalidades como registro de clientes e faturas, cálculo de valores com e sem IVA, importação/exportação de dados e geração de estatísticas, com foco em eficiência e organização.

Objetivos

A aplicação implementa as seguintes funcionalidades:

1. Criação, edição e listagem de clientes.
2. Criação, edição e listagem de faturas.
3. Visualização de detalhes completos de faturas.
4. Registro de produtos com diferentes taxas de IVA.
5. Importação e exportação de faturas em ficheiros ".txt".
6. Apresentação de estatísticas sobre as faturas.

Estrutura do Projeto

Para a realização deste projeto, foram definidas as seguintes classes:

- **Cliente**: Armazena informações sobre os clientes (nome, NIF e localização). Inclui validações para evitar duplicação de NIFs e para garantir que a localização seja uma das opções válidas (Continente, Madeira ou Açores). O método **localizacaoToIndex** transforma a localização em um índice numérico, usado para cálculos regionais de IVA.
- **Fatura**: Responsável por registrar detalhes das faturas, como cliente associado, data de emissão, lista de produtos e valores. A geração automática de números de fatura utilizando um atributo estático (*contadorNumeroFatura*) elimina a necessidade de verificação manual, melhorando a eficiência. Além disso, a classe inclui a importação da classe **"java.util.Date"** para registrar a data de emissão da fatura, o que permite maior controle e organização. Através dos métodos **calcularTotalComIVA** e **calcularTotalSemIVA**, a classe calcula os valores totais da fatura considerando os produtos associados, taxas regionais e possíveis descontos.
- **Main:** Executa o programa, permitindo ao utilizador escolher ações através de um menu. Inicializa o ficheiro de objetos, carrega os dados (se existirem) e oferece opções para gerir clientes, faturas e exibir estatísticas. No final, os dados são guardados no ficheiro de objetos e o programa é encerrado.
- **Produto**: Classe abstrata que representa um produto genérico. Define atributos essenciais, como *nome, descricao, quantidade, valorUnitSemIVA e codigo*, além de métodos comuns, como **calcularValorSemIVA**. Os métodos abstratos, como **calcularValorComIVA** e **toTexto**, devem ser implementados pelas subclasses, garantindo que comportamentos específicos sejam definidos

conforme o tipo de produto (polimorfismo). Esta classe é abstrata para garantir que algumas funcionalidades sejam definidas apenas em subclasses concretas.

- **ProdutoAlimentar**: Classe abstrata que é subclasse de Produto, representando os produtos alimentares. Adiciona o atributo *bio* para indicar se o produto é biológico, e implementa o método ***descontoBio*** para aplicar um desconto na taxa de IVA caso o produto seja biológico. Mantém o método abstrato **calcularValorComIVA**, que deve ser implementado pelas subclasses, permitindo fazer implementações diferenciadas do método que efetua o cálculo do valor com IVA, conforme as características do produto alimentar. Esta classe não pode ser instanciada diretamente, servindo como superclasse para produtos alimentares das três diferentes taxas de IVA (reduzida, intermédia e normal).
- **ProdutoAlimentarTaxaIntermedia**: Subclasse de *ProdutoAlimentar* que representa produtos alimentares sujeitos a taxas intermédias de IVA. Introduz os atributos *categoria* e *listaTaxas*, permitindo associar diferentes taxas de IVA conforme a localização do cliente (Continente, Madeira ou Açores). A classe implementa o método **calcularValorComIVA** para calcular o valor com IVA, aplicando um aumento de 1% na taxa para a categoria "vinho" e aplicando um desconto a produtos biológicos. Também implementa o método **toTexto**, que gera uma descrição detalhada do produto (usada para escrever no ficheiro de texto posteriormente). Esta classe permite personalizar a aplicação de taxas de IVA e descontos, dependendo das especificidades do produto.
- **ProdutoAlimentarTaxaNormal**: Subclasse de *ProdutoAlimentar* que representa produtos alimentares sujeitos a taxas normais de IVA. Introduz o atributo *listaTaxas*, que contém as taxas de IVA associadas à localização do cliente (Continente, Madeira ou Açores). A classe implementa o método **calcularValorComIVA**, aplicando a taxa correspondente com base na localização do cliente e aplicando um desconto a produtos biológicos. Também implementa o método **toTexto**, que gera uma descrição detalhada do produto(usada para escrever no ficheiro de texto posteriormente). Como já foi dito para a classe *ProdutoAlimentarTaxaIntermedia*, esta classe fornece uma implementação específica para o cálculo do valor com IVA, adaptando a lógica às taxas normais.
- **ProdutoAlimentarTaxaReduzida:** Subclasse de *ProdutoAlimentar* que representa produtos alimentares sujeitos a taxas reduzidas de IVA. Esta classe introduz os atributos *certificacoes* (lista as certificações associadas ao produto) e *listaTaxas* (contém as taxas de IVA com base na localização do cliente). O método ***calcularValorComIVA*** aplica a taxa correspondente à localização do cliente e, se o produto tiver 4 certificações, reduz a taxa em 1%. Além disso, a classe efetua desconto para produtos biológicos. O método **toTexto** gera uma descrição detalhada do produto, incluindo as certificações (usado para escrever no ficheiro de texto posteriormente). Como já foi dito para as classes anteriores, esta classe também implementa o cálculo específico do IVA, ajustando-se às taxas reduzidas.
- **ProdutoFarmacia**: É uma subclasse de *Produto* e representa produtos farmacêuticos. Esta é uma classe abstrata, o que significa que não pode ser instanciada diretamente, mas define os atributos e métodos necessários para as suas subclasses. As subclasses devem implementar os métodos **calcularValorComIVA**, que define como o IVA irá ser calculado, e **toTexto**, que descreve o produto(usado para escrever no ficheiro de texto posteriormente). Além disso, a classe herda o método ***calcularValorSemIVA*** de *Produto*. A classe serve como uma base para os vários produtos farmacêuticos, delegando as implementações específicas para as duas subclasses (ProdutoFarmaciaComPrescricao e ProdutoFarmaciaSemPrescricao).

- **ProdutoFarmaciaComPrescricao**: É uma subclasse de *ProdutoFarmacia*, especializada em produtos farmacêuticos com prescrição médica. Ela adiciona atributos como *nomeMedico* (médico responsável pela prescrição do produto) e *listaTaxas* (taxas de IVA associadas ao produto). A classe implementa o método **calcularValorComIVA**, que calcula o valor total com base nas taxas de IVA específicas para cada localização do cliente(Continente, Madeira ou Açores). O método **toTexto** fornece uma descrição detalhada do produto, incluindo o nome do médico e as taxas (utilizado para escrever no ficheiro de texto posteriormente). A classe também utiliza o método **calcularValorSemIVA** da classe pai para determinar o preço sem IVA.
- **ProdutoFarmaciaSemPrescricao**: Subclasse de *ProdutoFarmacia*, destinada a produtos farmacêuticos sem prescrição médica. Estão incluídos os atributos *categoria* (classificação do produto) e *listaTaxas* (taxas associadas ao produto). O método **calcularValorComIVA** calcula o preço com base nas taxas e aplica um desconto de 1% para a categoria "animais". O método **toTexto** fornece a descrição do produto, incluindo a sua categoria (utilizado para escrever no ficheiro de texto posteriormente). A classe também herda o método **calcularValorSemIVA** da classe pai para determinar o valor sem IVA.
- **SistemaPOOFS**: A classe tem como objetivo principal fazer a gestão de clientes, produtos e faturas, permitindo a criação, edição, visualização e exportação/importação de dados. Ela utiliza atributos como listas para organizar os clientes, faturas e produtos disponíveis. Os principais métodos incluem:
    - Gestão de Clientes:
        - Criação, edição e listagem de clientes com validação para nome, NIF e localização.
    - Gestão de Faturas:
        - Criação de faturas associadas a clientes, incluindo adição de produto(s);
        - Edição de detalhes da fatura como cliente, data (formatada para DD/MM/YYYY) e produtos;
        - O método **visualizarFatura** mostra as informações da fatura (número da fatura, dados do cliente, detalhes de cada produto da fatura) e para cada produto mostra o valor sem Iva, taxa do IVA, o valor do IVA e o valor total com IVA. De seguida, exibe os totais da fatura, incluindo o valor total sem IVA, o total do IVA e o total com IVA;
        - O método **mostrarEstatisticas** mostra as estatísticas sobre as faturas, incluindo o número total de faturas, número de produtos e os valores totais com e sem IVA, além do total de IVA. Por fim, o método **faturaJaExiste** verifica se uma fatura com um dado número fornecido já existe na lista de faturas
    - Gestão de Ficheiros:
        - O método **salvarDadosObj** salva os dados dos clientes e faturas num ficheiro de objetos através do *ObjectOutputStream*. A lista de clientes e faturas é também gravada no arquivo. Além disso, foi criado o método **carregarDadosObj** que carrega os dados dos clientes e as faturas a partir de um ficheiro de objetos (utilizando *ObjectInputStream*). Caso o arquivo não seja encontrado ou se ocorrer algum erro durante este processo, o programa será inicializado com listas vazias. No fim, atualiza o contador do número de faturas para garantir que o próximo número da fatura seja o correto;

- O método **importarFaturas** importa as faturas a partir de um ficheiro de texto. Para isso processa a informação relacionada a cada fatura (número, entre outros) com validação para formato ".txt". Para cada tipo de produto é criado o respetivo objeto da respetiva classe e é adicionado à fatura. Este método verifica também se a fatura já existe anteriormente antes de a adicionar à lista de faturas (para evitar faturas em duplicado na lista) e verifica se o conteúdo do ficheiro está dentro dos padrões. O método **exportarFaturas** exporta todas as faturas registadas para um ficheiro de texto ".txt". Cada fatura é detalhada no ficheiro (através do método **toTexto** implementado nas classes referido anteriormente) com as informações do cliente e dos produtos, dados do produto, valores calculados, entre outros.

Nota: nas classes **ProdutoAlimentarTaxaIntermedia** (aumento = 1) e **ProdutoAlimentar** (desconto = 0.9) inicialmente foram criadas variáveis para a representação dos valores 1 (variável aumento de taxa de IVA) e 0.9 (variável desconto: caso o produto seja biológico multiplica a taxa de IVA por este valor, aplicando um desconto de 10%), respetivamente. No entanto, esta solução criou vários problemas e, por isso, os valores foram aplicados diretamente no código, não estando associados a nenhuma variável.

Funcionalidades da aplicação

Criar e Editar cliente

O cliente é registado com validação de NIF e localização. Caso o NIF já exista, uma mensagem de aviso é exibida. O sistema permite que sejam adicionadas e editadas as informações do cliente, incluindo nome, NIF e localização.

Listar Clientes

Mostra a lista de todos os clientes registados no sistema. Cada cliente é mostrado com seu nome, NIF e localização.

Criar e Editar Faturas

As faturas podem ser criadas e editadas, associando um cliente a uma fatura e adicionando os produtos desejados. O número da fatura é gerado automaticamente, tornando mais simples a utilização do programa. A edição de uma fatura permite alterar os produtos, quantidades e valores da fatura.

Listar Faturas

Exibe todas as faturas registadas, com informações como o número da fatura, cliente associado, data da fatura e valores totais.

Visualizar Fatura

Ao inserir o número da fatura pretendida,mostra todos os detalhes dessa fatura, incluindo o cliente associado, produtos, valores sem IVA, valores do IVA e valores totais com IVA.

Importar Faturas

Permite importar faturas de um ficheiro de texto. O sistema processa as informações das faturas, verifica se já existem e adiciona os produtos com os respetivos tipos e valores.

Exportar Faturas

Exporta todas as faturas para um ficheiro de texto. Cada fatura é detalhada, incluindo informações sobre o cliente, produtos e valores, com a possibilidade de calcular e exibir o IVA.

Estatísticas

Exibe estatísticas sobre as faturas registadas, incluindo o número total de faturas, número de produtos, e o total de valores com e sem IVA. Também calcula o total de IVA cobrado.

Conclusão

Em conclusão, ao longo deste projeto foi possível aprofundar os conhecimentos na linguagem de programação Java, com especial foco na criação de classes abstratas, implementação de herança, polimorfismo e outros conceitos fundamentais da programação orientada a objetos. Por fim, a aplicação cumpre os objetivos definidos, entregando uma solução robusta para gestão financeira, com integração de funcionalidades úteis para usuários finais.