Contents

1 Relatório das Classes do projeto

O projeto implementa um total de 9 classes responsáveis pelas regras de negócio, sem contar todos os *controllers* implementados para fazer os *models* funcionarem de forma correta.

A seguir serão apresentadas todas as classes, com seus respectivos métodos.

1.1 Descrição das Classes

1.1.1 User

Essa é uma classe abstrata responsável por fornecer um modelo de herança para as classes `Manager` e `Employee`. Ela contém informações gerais de usuário.

Seus métodos implementados são apenas dos getters e setters para os atributos existentes.

1.1.2 Manager

Essa é uma classe responsável por gerenciar a criação de todos os gerentes da empresa, e além dos atributos e métodos importados da classe `User`, também contém os atributos `type` (que pode ser tanto especificando que esse gerente é responsável pela equipe de vendas ou pela equipe de compras) e também um array `subordinates` (que é responsável por guardar todos os funcionários que reportam a esse gerente).

Seus métodos implementados são, além dos devidos getters e setters:

- `addSubordinate`: responsável por adicionar um único subordinado no array `subordinates`.
- `addSubordinates`: responsável por adicionar um array de subordinados no array `subordinates`.
- `removeSubordinate`: responsável por remover um único subordinado do array `subordinates`.
- `removeSubordinates`: responsável por remover um array de subordinados do array `subordinates`.

1.1.3 Employee

Essa é uma classe responsável por gerenciar a criação de todos os funcionários da empresa, e além dos atributos e métodos importados da classe `User`, também contém os atributos `type` (que pode ser ou a especificação de funcionário responsável por *vendas* ou por *compras*), `manager` (que indica qual é o gerente do funcionário) e `seniority` (que indica se o funcionário é *Júnior*, *Pleno*, *Senior* ou *Lead*).

Seus métodos implementados são, além dos devidos getters e setters:

- `getPurchasesTimeframe`: responsável por obter quais foram as compras realizadas por um determinado funcionário em um intervalo de tempo.
- `getSalesTimeframe`: responsável por obter quais foram as vendas realizadas por um determinado funcionário em um intervalo de tempo.

1.1.4 Partner

Essa é uma classe abstrata responsável por fornecer um modelo de herança para as classes `Supplier` e `Customer`. Ela contém informações gerais de uma pessoa física ou jurídica.

Seus métodos implementados são apenas dos getters e setters para os atributos existentes.

1.1.5 Supplier

Essa é uma classe responsável por gerenciar a criação de todos os fornecedores da empresa, e além dos atributos e métodos importados da classe `Partner`, também contém o atributo `products` (que é um array com todos os produtos possíveis de se obter desse fornecedor).

Seus métodos implementados são, além dos devidos getters e setters:

- `addProduct`: responsável por adicionar um único produto no array `products`.
- `addProducts`: responsável por adicionar um array de produtos no array `products`.
- `removeProduct`: responsável por remover um único produto do array `products`.
- `removeProducts`: responsável por remover um array de produtos do array `products`.
- `getSalesTimeframe`: responsável por obter quais foram as vendas realizadas em um determinado intervalo de tempo que possuiam como produto vendido algo obtido por meio desse fornecedor.

1.1.6 Customer

Essa é uma classe responsável por gerenciar a criação de todos os clientes da empresa.

Seus métodos implementados são, além dos devidos getters e setters:

- `getPurchasesTimeframe`: responsável por obter quais foram as compras realizadas por um determinado cliente em um intervalo de tempo.

1.1.7 Product

Essa é uma classe responsável por gerenciar a criação de todos os produtos vendidos pela empresa, possuindo os atributos `description` (breve descrição do produto), `minPrice` (preço mínimo que o produto pode ser vendido), `name` (nome do produto), `productId` (identificador único do produto), `suggestedPrice` (preço sugerido para o produto), `supplier` (qual foi o fornecedor do qual a empresa adquiriu o produto), `stock` (qual é a quantidade total de produtos disponíveis em estoque).

Seus métodos implementados são, além dos devidos getters e setters:

- `isCriticalStockLevel`: responsável por verificar no estoque da empresa se um determinado produto está com nível muito baixo em estoque. (isso é necessário para que a empresa possa dar prioridade a esses produtos na hora de aquisição de novas mercadorias).

- `salesVolumeTimeframe`: responsável por obter qual é o volume de vendas que esse produto teve em um determinado intervalo de tempo.

1.1.8 Operation

Essa é uma classe responsável por gerenciar a criação de todas as operações de venda da empresa, possuindo os atributos `customer` (com a identificação de qual foi o cliente que realizou a compra), `date` (a data da operação de compra), `employee` (qual foi o funcionário responsável por efetivar a operação de compra ou venda), `product` (qual foi o produto vendido), `quantity` (qual foi a quantidade de produto vendido), `supplier` (qual foi o fornecedor do qual a empresa adquiriu o produto), `type` (se foi uma operação de compra ou venda), `unitPrice` (qual foi o preço unitário para cada produto vendido).

Seus métodos implementados os devidos getters e setters.

1.1.9 Report

Essa é uma classe responsável por gerenciar a criação de todos os relatórios de compra e venda da empresa, possuindo os atributos `from` (qual a data de início para se pesquisar as operações), `to` (qual a data de fim para se pesquisar as operações), `issueDate` (em que data o relatório foi criado), `type` (se é um relatório de compra ou de venda).

Seus métodos implementados são, além dos devidos getters e setters:

- `customersActiveTimeframe`: responsável por obter quais foram os clientes que realizaram compras com a empresa no período especificado.
- `customersHighVolume`: responsável por obter quais foram os clientes que realizaram compras em um valor total alto em um período de tempo.
- `customersInactive`: responsável por obter quais foram os clientes que não realizaram nenhuma comprar dentro de um intervalo de tempo.
- `productsNotSoldTimeframe`: responsável por obter quais foram os produtos que não foram vendidos dentro de um intervalo de tempo.
- `productsSoldTimeframe`: responsável por obter quais foram os produtos vendidos em um intervalo de tempo.

- `sellersHighPerformance`: responsável por obter quais foram os vendedores com melhor performance na empresa dentro de um intervalo de tempo.
- `sellersLowPerformance`: responsável por obter quais foram os vendedores com baixa performance na empresa dentro de um intervalo de tempo.
- `suppliersHighVolume`: responsável por obter quais foram os fornecedores cujas mercadorias adiquiridas correspondem a um grande percentual do total vendido pela empresa.
- `suppliersSoldProductsTimeframe`: responsável por obter quais foram os produtos vendidos por um determinado fornecedor dentro de um intervalo de tempo.