|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Curso: | Ciência da Computação | | |  | |  |
| Disciplina: | POO | | |  | **Valor** | 20,0 pts |
| Professor (a): | João Paulo C. Aramuni | | |
| Nome: |  | | |  | **Nota** |  |
| Nº da Atividade/Nome: | AAI |  |  |
| Data: |  |  |  |  |  |  |
| Valor: | 20,0 pts |  |  |  |  |  |

**Observações: O trabalho é individual**: É permitido discutir com outros colegas os problemas e as estratégias para solucioná-los, mas as implementações das soluções, bem como as resoluções dos exercícios propostos devem ser feitas somente pelo aluno. Trabalhos plagiados da Internet ou de outros colegas não serão corrigidos e receberão nota 0 (zero).

**Data Limite para Entrega**: Definida em sala. Não serão tolerados atrasos!

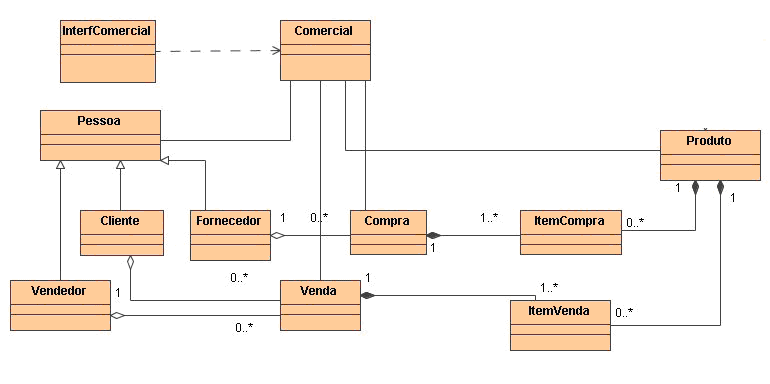
**O que deve ser entregue**: Deverá ser entregue todo o código necessário para a execução do programa. O código deve ser entregue via SINEF, em arquivo ZIP.

**Observações importantes**: Será avaliado: Documentação; nomenclatura de pacotes, classes, atributos e métodos; desenvolvimento dentro da metodologia de orientação a objetos; verificação detalhada do funcionamento; interface gráfica e usabilidade. Trabalhos entregues com erro de compilação não serão avaliados.

**SisCom - Sistema Comercial de Controle de Compras e Vendas**

Implemente um conjunto de classes para um sistema de controle de compras e vendas. A área comercial necessita manter informações sobre seus fornecedores, compras destes fornecedores, clientes e vendas para estes clientes. Para isso, foi feito o seguinte projeto que deve ser implementado usando Java:

**Diagrama Simplificado de Classes**



* Índice: Azul: feito / vermelho: não entendido / preto: não feito/ laranja: sera feito pelo bando de dados/verde: comentarios
* **Classe Pessoa**: (**Classe Abstrata**)
* Atributos: codigo (int), nome (String), telefones (String), email (String), dataCad (Date). Todos com acesso privado, o código é um número sequencial;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método abstrato retornando o tipo de pessoa.
* Método **compareTo** reimplentada da **interface Comparable**; Parâmetro a Pessoa. Este método vai comparar dois nomes de pessoas e retornar zero caso os nomes sejam iguais, um para maior e -1 para menor. (compare to em 1 linha so)
* Método **toString** retornando uma String com os atributos da classe formatados.
* **Classe Cliente**: (**Herança da Classe Pessoa**)
* Atributos: cpf (String), limiteCredito (double). Todos com acesso privado;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método toString retornando uma String com os atributos da classe formatados.
* **Classe Vendedor**: (**Herança da Classe Pessoa**)
* Atributos: cpf (String), metaMensal (double). Todos com acesso privado;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método toString retornando uma String com os atributos da classe formatados.
* **Classe Fornecedor**: **(Herança da Classe Pessoa)**
* Atributos: cnpj (String), nomeContato (String). Todos com acesso privado;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método toString retornando uma String com os atributos da classe formatados.
* **Classe Compra**:
* Atributos: numCompra (int), fornecedor (Fornecedor), compraItens (lista de ItemCompra implementada usando a classe ArrayList), dataCompra (Date). Todos com acesso privado, o número da compra é um número sequencial;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método toString retornando uma String com os atributos da classe formatados.
* **Classe ItemCompra**:
* Atributos: produto (Produto), quantCompra (int), valorCompra (double)(substituído por valor unitário). Todos com acesso privado;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método toString retornando uma String com os atributos da classe formatados.
* **Classe Venda**:
* Atributos: numVenda (int), cliente (Cliente), vendedor (Vendedor), vendaItens (lista de ItemVenda implementada usando a classe ArrayList), formaPagto (int) (1-Pagto á Vista; 2-Pagto a prazo – cheque pré-datado), dataVenda (Date). Todos com acesso privado, o número da venda é um número sequencial;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método toString retornando uma String com os atributos da classe formatados.
* **Classe ItemVenda**:
* Atributos: produto (Produto), quantVenda (int), valorVenda (double)(DESNECESSARIO). Todos com acesso privado;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método toString retornando uma String com os atributos da classe formatados.
* **Classe Produto**:
* Atributos: codigo (int), nome (String), precoUnitario (double), estoque (int), estoqueMinimo (int), dataCad (Date). Todos com acesso privado, o código do produto é um número sequencial;
* Método construtor para inicializar os atributos por meio de parâmetros;
* Métodos *get* e *set* para obter e modificar cada um dos valores dos atributos.
* Método para adicionar quantidade do produto no estoque. Parâmetro: quantidade a ser incrementada;
* Método para decrementar quantidade do produto no estoque. Parâmetro: quantidade a ser decrementada. Observação: caso não consiga decrementar quantidade do produto no estoque lançar uma exceção da classe **SisComException** passando como parâmetro o nomeProduto, o estoque e a mensagem “Estoque Insuficiente. ”;(MEIO INUTIL SE USAR UPDATE NO LUGAR)
* Método compareTo reimplementado da **interface Comparable**; Parâmetro o Produto. Este método vai comparar dois nomes de produtos e retornar zero caso os nomes sejam iguais, um para maior e -1 para menor.
* Método toString retornando uma String com os atributos da classe formatados.
* **Classe** **SisComException**: **(Herança da Classe Exception)**
* Atributos: nomeProduto (String), estoque (int), mensagemErro (String). Todos com acesso privado;
* Método construtor para inicializar todos atributos por meio de parâmetros;
* Método construtor para inicializar o atributo mensagemErro por meio de parâmetro;
* Métodos *get* para obter cada um dos valores dos atributos.
* **Classe Comercial**:
* Atributos: pessoas (lista de Pessoas), produtos (lista de Produtos), compras (lista de Compras) e lista de vendas (lista de Vendas). As listas devem ser implementadas usando a classe ArrayList. Todos com acesso privado;
* Método para inserir uma nova **pessoa**. Parâmetro a pessoa;

**Observações**:

* Não poderá ser cadastrado fornecedor com cnpj já cadastrado;
* Não poderá ser cadastrado cliente com cpf já cadastrado;
* A meta mensal de vendas do vendedor tem que ser maior do que zero e o cpf não pode ser repetido;
* Método para excluir uma pessoa do banco de dados. Parâmetro a pessoa.

**Observações**:

* O fornecedor não pode ser excluído se tiver compra registrada para ele;
* O cliente não pode ser excluído se tiver venda registrada para ele;
* O vendedor não pode ser excluído se tiver venda registrada para ele;
* Método para fazer uma **compra** para um fornecedor. Parâmetro: o fornecedor e a lista de ItemCompra(ArrayList). Incrementar o estoque para os produtos comprados.

**Observação**: A lista de ItemCompra não pode possuir produtos repetidos

* Método para excluir uma compra de um fornecedor. Parâmetro: número da compra. Decrementar o estoque para os produtos comprados.
* Método para fazer uma **venda** para um cliente. Parâmetro: o cliente, o vendedor, a forma de pagamento e a lista de ItemVenda(ArrayList). A venda não pode ser realizada se a forma de pagamento for igual a 2 (Venda a prazo através de Cheque pré-datado) e o total da venda for superior ao limite de crédito que o cliente possui. Decrementar o estoque para os produtos vendidos.

**Observação**: A lista de ItemVenda não pode possuir produtos repetidos

* Método para excluir uma venda de um cliente. Parâmetro: número da venda. Incrementar o estoque para os produtos vendidos.
* Método para inserir um novo **produto**. Parâmetros: o produto.
* Método para excluir um produto. Parâmetro: o código do produto. O produto não pode ser excluído se tiver compra ou venda registrada para ele;
* Método para buscar **cliente** por cpf;
* Método para obter lista de clientes em ordem alfabética por parte do nome do cliente;
* Método para buscar **fornecedor** por cnpj;
* Método para obter lista de fornecedores em ordem alfabética por parte do nome do fornecedor;
* Método para buscar **vendedor** por cpf;
* Método para obter lista de vendedores em ordem alfabética por parte do nome do cliente;
* Método para buscar **produto** por codigo; (ACHO QUE SERIA MELHOR PESQUISAR PELO NOME)
* Método para obter lista de produtos em ordem alfabética por parte do nome do produto;
* Método para obter lista de produtos com estoque abaixo do mínimo em ordem alfabética;
* Método para obter **lista de vendas** em ordem de cliente e data da venda decrescente. Parâmetro: parte do nome do cliente e período das vendas;
* Método para obter lista de **vendas** em ordem de vendedor e data da venda decrescente. Parâmetro: parte do nome do vendedor e período das vendas;
* Método para obter lista de **compras** em ordem de fornecedor e data da compra decrescente. Parâmetro: parte do nome do fornecedor e período das compras;
* Método para buscar **Estatística de Vendas por cliente** (Nome, quantas vezes comprou e valor total). Parâmetro: período das vendas. Retorno: Lista EstatisticaClientes.
* Método para buscar **Estatística de Vendas por vendedor** (Nome, quantas vezes vendeu e valor total). Parâmetro: período das vendas. Retorno: Lista EstatisticaVendedores.
* Método para buscar **Estatística de Compras por fornecedor** (Nome, quantas vezes vendeu e valor total). Parâmetro: período das compras. Retorno: EstatisticaFonecedores.
* **Classe InterfComercial**:
* Método para apresentar um menu (interface de caracteres). Todos os métodos abaixo, com exceção do main, mais a opção para sair, devem ser chamados no menu.
* Método para ler os dados armazenados em arquivo.
* Método para gravar os dados em arquivo;
* Método para inserir um novo **fornecedor** no banco de dados;
* Método para excluir um fornecedor do banco de dados;
* Método para inserir um novo **cliente** no banco de dados;
* Método para excluir um cliente do banco de dados;
* Método para inserir um novo **produto** no banco de dados;
* Método para excluir um produto do banco de dados;
* Método para inserir um novo **vendedor** no banco de dados;
* Método para excluir um vendedor do banco de dados;
* Método para fazer uma **compra** para um fornecedor.
* Método para excluir uma compra de um fornecedor;
* Método para fazer uma **venda** para um cliente;
* Método para excluir uma venda de um cliente;
* -------------------------------------------------------------
* Método para consultar **fornecedor** por cnpj;
* Método para imprimir **lista de fornecedores** em ordem alfabética por parte do nome do fornecedor;
* Método para consultar o **cliente** por cpf;
* Método para imprimir **lista de clientes** em ordem alfabética por parte do nome do cliente;
* Método para consultar o **vendedor** por cpf;
* Método para imprimir **lista de vendedores** por parte do nome em ordem alfabética;
* Método para consultar **produto** por código;
* Método para imprimir **lista de produtos** em ordem alfabética por parte do nome do produto;
* Método para imprimir **lista de produtos** com estoque abaixo do mínimo em ordem alfabética;
* Método para imprimir **lista de vendas por período** em ordem de cliente e data da venda decrescente.
* Método para imprimir lista de **vendas por período** em ordem de vendedor e data da venda decrescente.
* Método para imprimir lista de **compras por período** em ordem de fornecedor e data da compra decrescente.
* Método para imprimir **Estatística de Vendas agrupado por cliente** **por período de vendas** (Nome, quantas vezes comprou e valor total).
* Método para imprimir **Estatística de Vendas agrupado por vendedor** **por período de vendas** (Nome, quantas vezes vendeu e valor total).
* Método para imprimir **Estatística de Compras agrupado por fornecedor** **por período de compras** (Nome, quantas vezes vendeu e valor total).

O usuário deve ser informado de todas as ações tomadas. Por exemplo: Na busca de um cliente ele deve ser informado se o cliente não existir no arquivo de dados. Todos os métodos deverão ser privados com exceção do método main.

Observe a separação entre as classes de negócio (Pessoa, Cliente, Fornecedor, Vendedor, Compra, ItemCompra, Venda, ItemVenda, Produto e SisComException) e a classe InterfComercial.

As classes de negócio não devem possuir nada que crie uma dependência da interface. Desta forma, se posteriormente você decidir mudar a interface, por exemplo, usar uma interface gráfica, as classes de negócio não precisam ser alteradas. A classe Comercial atua como uma classe gerente, fazendo a interligação das classes de negócio com a classe de interface. Em um projeto mais profissional, provavelmente existiriam várias interfaces e classes gerente.

**Observações a serem seguidas:**

* As classes, atributos e métodos descritos constituem o projeto básico do sistema. Pode-se adicionar outros métodos e atributos necessários para completar o sistema. Porém qualquer mudança “radical” no projeto deve ser discutida com o professor;
* **Leitura de dados do usuário e impressão na tela devem ser feitos somente na classe InterfComercial**;
* **Todas as exceções deverão ser tratadas**.
* **Use polimorfismo sempre que possível**.
* Usar as interfaces Serializable, Comparable e Comparator.
* As classes de negócio e a classe de interface deverão ficar em pacotes diferentes;
* A classe **LtpLib** deverá ser utilizada no projeto.
* Usar o método **sort** da classe java.util.Collections para fazer todas as classificações do aplicativo.
* Faça comentários no formato **JavaDoc** para todas as classes e todos os métodos. Gere uma documentação usando este utilitário. Documentação mínima para classe: Descrição, pacote, autor, versão e data de criação. Documentação mínima para método: Descrição, parâmetros, valor retornado, exceções;

**Interface Gráfica com Usuário:**

* A interface Gráfica com Usuário corresponde à 25% do valor do trabalho! (5 pts).
* A apresentação do projeto em laboratório, em data agendada pelo professor, corresponde à 25% do valor do trabalho! (5 pts).
* Faça uma interface moderna, com layout bem trabalhado e boa usabilidade.
* Os exemplos do Material Didático (como o SisLocadora) podem ser usados como referência, mas não devem ser copiados.
* Você deve utilizar os componentes das bibliotecas AWT e SWING vistas durante o semestre.

**Bom trabalho!**

Dúvidas: [joaopauloaramuni@fumec.br](mailto:joaopauloaramuni@fumec.br)