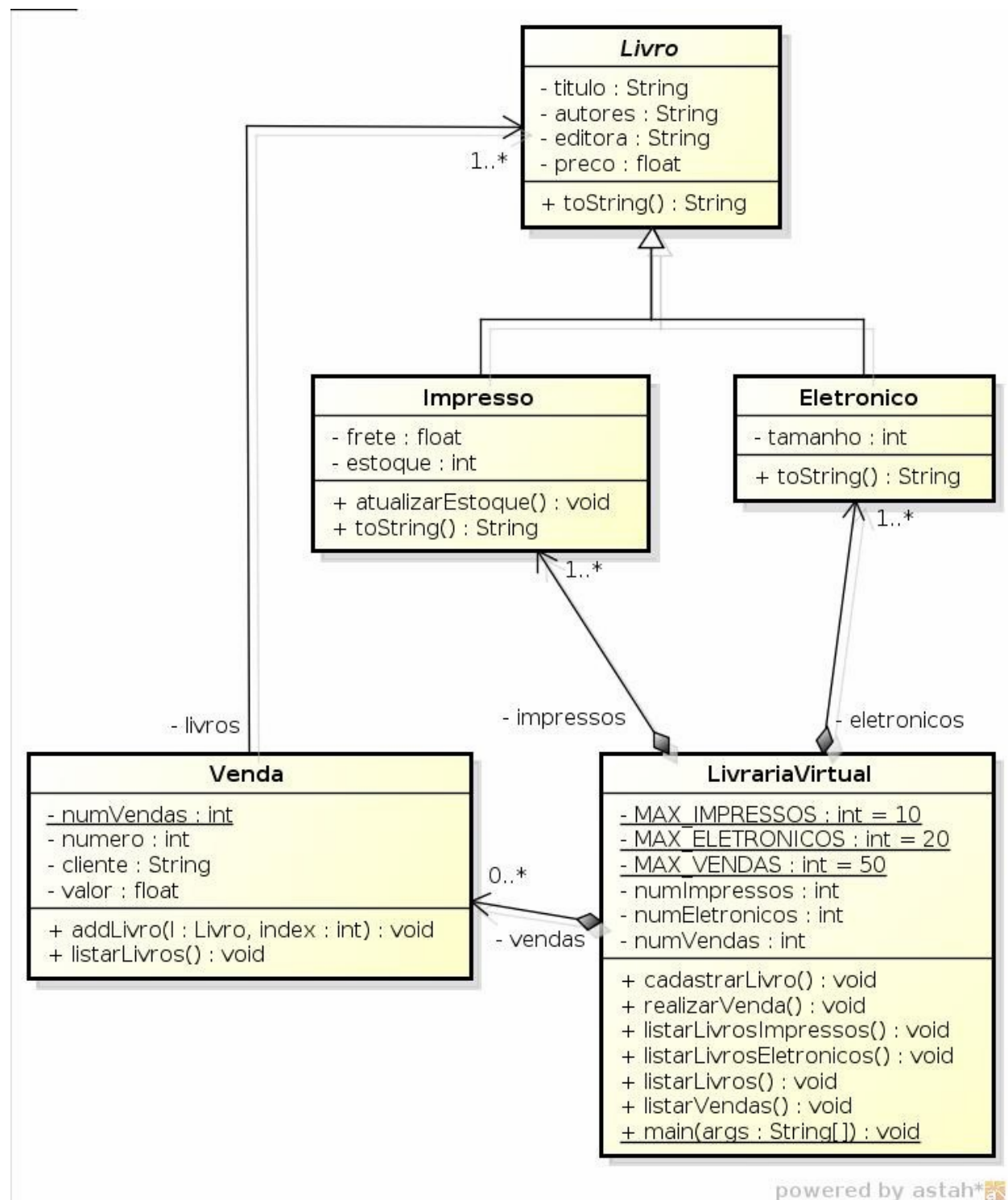


Desafio – Livraria Virtual

1. Introdução

Este projeto consiste em implementar um sistema de gerenciamento de uma livraria virtual, explorando os conceitos de composição, herança e polimorfismo.

O sistema deve seguir o diagrama de classes UML mostrado abaixo, aonde os construtores e os métodos de acesso (*getters* e *setters*) foram omitidos:



2. Descrição do Sistema

O sistema deverá ser baseado em um menu, entrada de dados via console, com as seguintes opções:

- a) **Cadastrar livro**: esta opção permite ao usuário cadastrar um livro;

- . b) **Realizar uma venda:** esta opção permite ao usuário realizar a venda de um ou mais livros;
- . c) **Listar livros:** o sistema deverá listar todos os livros cadastrados, sejam eles eletrônicos ou impressos;
- . d) **Listar vendas:** o sistema deverá listar todas as vendas realizadas;
- . e) **Sair do programa:** encerra a execução do programa.

3. Descrição das Classes

A seguir serão descritas as classes do sistema.

3.1 Livro

A classe abstrata Livro possui 4 atributos:

- a) **título:** título do livro;
- b) **autores:** nome do autor ou dos autores do livro;
- c) **editora:** nome da editora do livro;
- d) **preço:** preço do livro.

Os métodos de **acesso** (**getters** e **setters**) e o(s) **construtor**(es) desta classe e das demais classes foram omitidos e devem ser implementados mesmo que você não os julgue necessário. O outro método obrigatório da classe **Livro** é descrito a seguir:

- a) String **toString()**: devolve uma representação textual dos atributos de um livro.

3.2 Impresso

A classe **Impresso** representa um livro impresso e possui 2 atributos:

- . a) **frete:** frete cobrado para entrega do livro;
- . b) **estoque:** número de exemplares do livro em estoque. A seguir são descritos os métodos da classe Impresso:
- . a) **void atualizarEstoque()**: este método deve subtrair 1 do valor do

atributo estoque;

- . b) **String toString**: este método devolve uma representação textual de todos dos atributos de um livro impresso.

3.3 Eletronico

A classe **Eletronico** representa um livro eletrônico e possui 1 atributo:

- a) **tamanho**: representa o tamanho do arquivo eletrônico do livro em KB.

A seguir é descrito mais um método obrigatório da classe **Eletronico**:

- a) **String toString**: este método devolve uma representação textual de todos dos atributos de um livro eletrônico.

3.4 Venda

A classe **Venda** possui 5 atributos:

- . a) **livros**: um vetor de referências a objetos do tipo Livro. Representa os livros associados a uma venda;
- . b) **numVendas**: atributo estático que representa a quantidade de vendas realizadas. Deve ser incrementado de 1 sempre que uma nova venda for realizada;
- . c) **numero**: representa o número da venda. É um valor sequencial com início em 1 e que é incrementado a cada venda. Utilize o valor do atributo numVendas para definir o valor desse atributo;
- . d) **cliente**: nome do cliente que comprou o(s) livro(s);
- . e) **valor**: valor total da venda. A seguir são descritos os métodos da classe **Venda**:
 - . a) **addLivro(l: Livro, index: int)**: adiciona o livro **l** na posição **index** do array livros;
 - . b) **listarLivros()**: lista todos os livros da venda.

3.5 LivrariaVirtual

A classe **LivrariaVirtual** possui 9 atributos:

- a) **MAX_IMPRESSOS**: constante que representa o número máximo de livros impressos que podem ser cadastrados;
- b) **MAX_ELETRONICOS**: constante que representa o número máximo de livros eletrônicos que podem ser cadastrados;
- c) **MAX_VENDAS**: constante que representa o número máximo de vendas que podem ser cadastradas;
- d) **impressos**: vetor de referências a objetos da classe Impresso, representa os livros impressos cadastrados;
- e) **eletronicos**: vetor de referências a objetos da classe Eletronico, representa os livros eletrônicos cadastrados;
- f) **vendas**: vetor de referências a objetos da classe Venda, representa as vendas realizadas;
- g) **numImpressos**: número de livros impressos cadastrados;
- h) **numEletronicos**: número de livros eletrônicos cadastrados;
- i) **numVendas**: número de vendas realizadas. A seguir são descritos os métodos da classe LivrariaVirtual:
 - a) **cadaststrarLivro()**: este método é invocado quando a primeira opção do menu do sistema (Cadastrar livro) for selecionada. O usuário deve informar o tipo de livro que será cadastrado: impresso, eletrônico ou ambos. Depois o sistema deve solicitar os dados do tipo de livro escolhido (ou de ambos). Se não houver mais espaço no vetor para cadastrar um novo livro, o sistema deve exibir uma mensagem;
 - b) **realizarVenda()**: este método é invocado quando a segunda opção do menu do sistema (**Realizar uma venda**) é selecionada. O sistema deve solicitar o nome do cliente e a quantidade de livros que ele deseja comprar. Depois, para cada livro, o sistema deve solicitar seu tipo (impresso ou eletrônico), exibir a lista de livros do tipo escolhido e permitir que o usuário escolha um dos livros dessa lista. Utilize os métodos **listarLivrosImpressos()** e **listarLivrosEletronicos()** descritos a seguir;

- c) **listarLivrosImpressos()**: exibe no vídeo no formato de tabela os dados de todos os livros impressos cadastrados. Utilize o método `toString()` da classe `Impresso`;
- d) **listarLivrosEletronicos()**: exibe no vídeo no formato de tabela os dados de todos os livros eletrônicos cadastrados. Utilize o método `toString()` da classe `Eletronico`;
- e) **listarLivros()**: este método é invocado quando a terceira opção do menu do sistema (Listar livros) é selecionada. O método exibe no vídeo os dados de todos os livros impressos e eletrônicos cadastrados. Utilize os métodos **listarLivrosImpressos()** e **listarLivrosEletronicos()**;
- f) **listarVendas()**: este método é invocado quando a quarta opção do menu do sistema (Listar vendas) é selecionada. O método exibe no vídeo os dados de todas as vendas realizadas;
- g) **main(args: String[])**: este método deve instanciar um objeto da classe `LivrariaVirtual`, exibir repetidamente o menu de opções e invocar os métodos apropriados a partir da seleção do usuário.

Observações:

- a) Os fontes da aplicação devem enviados para o GitHub até a data de entrega do projeto;
- b) Critérios de avaliação: Utilizar banco de dados para inserir os pedidos sera um diferencial.

Corretude	70%
Interface	20%
Legibilidade	10%