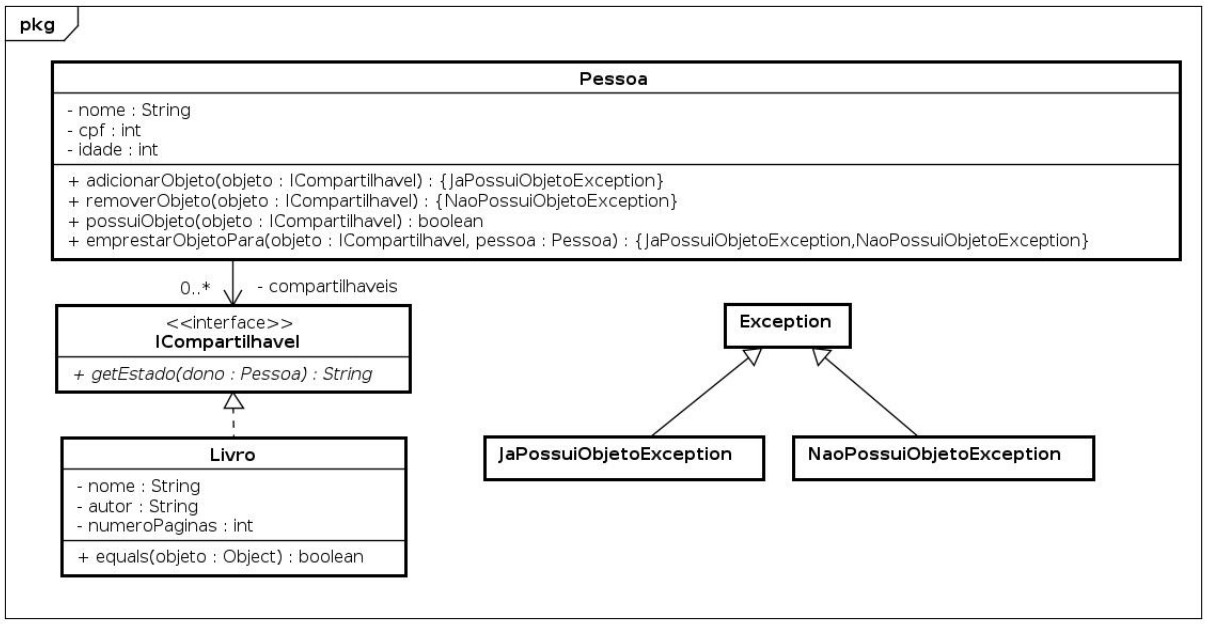
UDESC - Fundação Universidade do Estado de Santa Catarina — Monitor: Vinícius Takeo
Professor: Fabiano Baldo — vtkwki@gmail.com
Disciplina: POO 2020.1 — github.com/takeofriedrich

A lista deve ser entregue até o dia **31/07/2020**, às 23h59, no Moodle, os arquivos devem ser compactados em um arquivo *.zip* ou *.tar*. O arquivo compactado deverá conter o projeto Eclipse ou Netbeans da lista. **Não serão aceitos projetos com os códigos-fonte no formato *.class*!**

Lista 4 - Exceções, Coleções e Interface

1. Observe a figura a seguir e implemente o que se pede nos itens abaixo:



a) (1.0) Implemente as classes *JaPossuiObjetoException* e *NaoPossuiObjetoException* que estendem a classe Exception. Elas devem conter dentro delas as mensagens de "objeto já existe na lista" e "objeto não existe na lista" respectivamente.

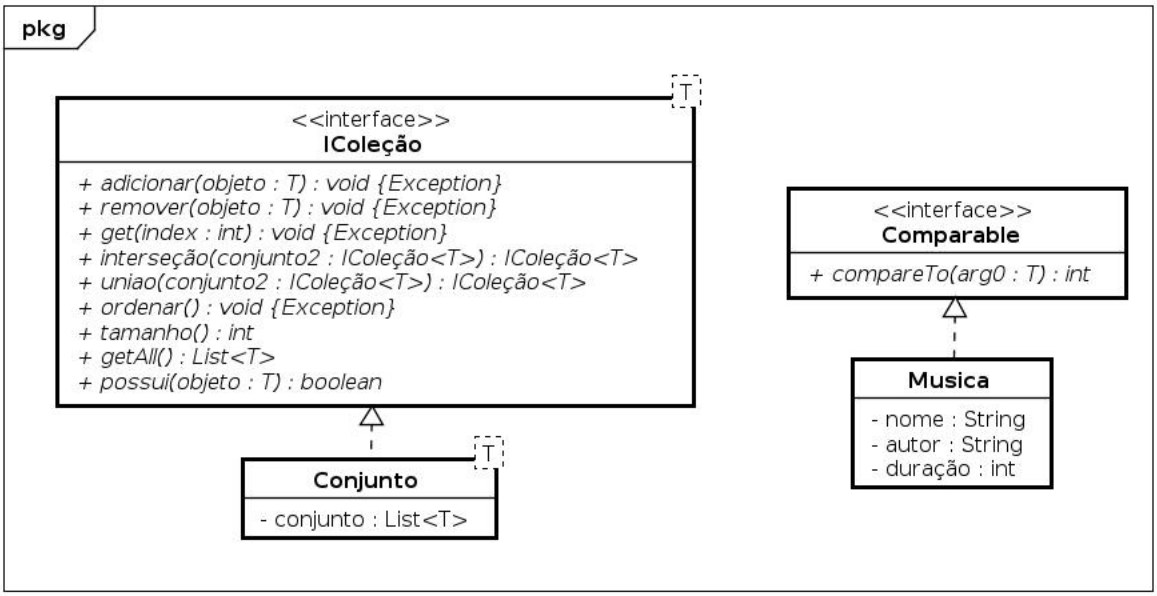
b) (1.5) Implemente a interface *ICompartilhavel*, a classe Livro e crie mais duas classes representando objetos que, assim como um livro, podem ser "compartilhados" por pessoas. Essas classes devem possuir ao menos 3 atributos cada e implementar o seu método **equals()**. Outros exemplos de **compartilhaveis** são: selos, figurinhas, moedas e etc. O método **getEstado()** deve retornar uma String a ser exibida no console, contendo ao menos um atributo do objeto e o nome da Pessoa ao qual o objeto pertence.

c) (1.5) Implemente a classe Pessoa, lançando adequadamente as exceções descritas no diagrama pelos seus métodos. Por exemplo, ao tentar adicionar um objeto que já existe na lista o método **adicionarObjeto()** lança a exceção *JaPossuiObjetoException*. Já o

método **removerObjeto()** lança a exceção *NaoPossuiObjetoException*. O método **emprestarObjetoPara()** apenas repassa as exceções lançadas por **adicionarObjeto()** e **removerObjeto()**.

d) (0.5) Crie um método **main()**, instancie dois objetos da classe Pessoa, para cada pessoa, instancie e adicione dois objetos distintos que implementem a interface **ICompartilhavel**. Após isso, troque os objetos compartilhaveis entre as Pessoas.

2. A partir da figura a seguir, implemente o que se pede:



a) (1.5) Crie a interface **IColeção** e a classe **Conjunto** que a implementa usando Generics (T). A classe **Conjunto** contém um atributo **conjunto** que deve ser do tipo LinkedList ou ArrayList e implementar os seguintes métodos:

- **adicionar(T objeto):** esse método deve adicionar um objeto T ao conjunto e caso esse objeto já esteja no conjunto deve lançar uma exceção informando isso;
- **remover(T objeto):** esse método remove um objeto T caso ele esteja no conjunto, caso contrário deve lançar uma exceção informando que o objeto não existe;
- **get(int index):** esse método retorna o objeto que se encontra na posição passada como parâmetro. Caso essa posição seja inválida, o método deve lançar uma exceção informando que isso ocorreu;
- **interseccao(Colecao<T> colecao):** esse método deve retornar uma coleção contendo apenas os objetos que se encontram tanto nesse conjunto quanto no conjunto passado como parâmetro do método;

- **uniao(Colecao<T> colecao):** esse método retorna uma coleção contendo todos os objetos presentes em ambos os conjunto (como um conjunto será retornado, não podem haver elementos repetidos);
- **ordenar():** esse método deve ordenar os dados do conjunto;
- **tamanho():** esse método retorna a quantidade de itens presentes no conjunto;
- **getAll():** retorna uma List contendo todos os objetos presentes no conjunto;
- **possui(T objeto):** retorna true caso o conjunto possua esse objeto e false caso contrário;

Utilize os métodos providos pela classe LinkedList ou ArrayList do *Framework Collection* utilizada para implementar os métodos acima.

b) (1.0) Crie uma classe Musica para representar as músicas. Essa classe deve possuir os seguintes atributos: nome, autor e duração (em segundos). Para essa classe Musica, implemente o método **equals()** da interface *Comparable*. A interface *Comparable* já é fornecida pela JDK. Para implementar a interface *Comparable*, implemente o método **compareTo()**, esse método retorna -1 se o objeto em questão é menor que o passado como parâmetro do método e 1 caso seja maior. Utilize o atributo duração para determinar se uma música é "maior" ou "menor" que outra.

c) (1.0) Crie um método **main()** e instancie um **Conjunto** para objetos do tipo **Musica**. Instancie também 5 objetos do tipo **Musica**, set seus atributos e utilize o método **ordenar()** desse conjunto. Exiba no console utilizando o método **toString()** da classe Musica a lista como era antes da ordenação e como ficou após a ordenação;

d) (1.0) Instancie um segundo **Conjunto** para objetos do tipo **Musica** e adicione 5 músicas dentro da coleção. Utilize os métodos **intereseccao()** e **uniao()** do Conjunto, exibindo no console por meio do método **toString()** os resultados antes e depois da aplicação dos métodos;

e) Extra (0.5) Altere o método **compareTo()** da classe Musica para utilizar o atributo nome ao invés da duração.