UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO

UM ESTUDO SOBRE A LISTA ORDENADA

João Vítor Demaria Venâncio

Florianópolis
2018

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO

UM ESTUDO SOBRE A LISTA ORDENADA

Relatório técnico apresentado como requisito parcial para obtenção de aprovação na disciplina Estrutura de Dados, no Curso de Sistemas de Informação, na Universidade Federal de Santa Catarina.

Prof. Dr. José Eduardo de Lucca

João Vítor Demaria Venâncio

Florianópolis
2018

RESUMO

O trabalho é sobre a construção e o teste de uma Lista Encadeada Simples Ordenada, com conceitos já abordados em sala de aula e a estrutura de dado implementada na linguagem Java.

Palavras-chave: Estrutura, Dados, Lista Ordenada, Java.

SUMÁRIO

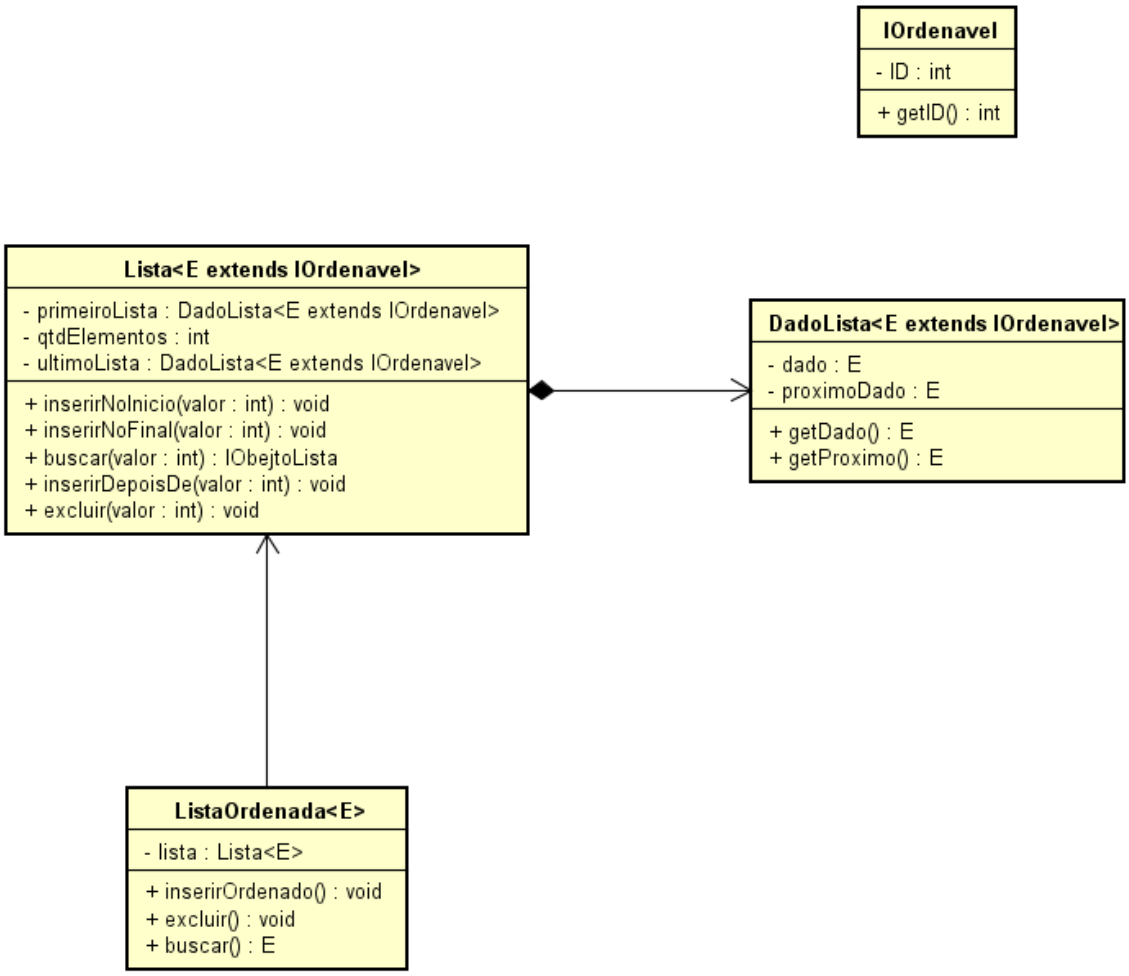
1. INTRODUÇÃO

Para se ter um entendimento melhor sobre como as listas encadeadas ordenadas funcionam, o professor José Eduardo de Lucca solicitou a criação de uma Lista Encadeada e que, a partir dela, criaríamos uma lista ordenada. As exigências para a criação da lista simples (encadeada de uma via) seria ela ter os métodos públicos inserirOrdenado(elemento completo), excluir(ID do elemento a ser excluído) e buscar(ID do elemento a ser procurado).

Além disso, a lista ordenada deve também ser genérica, possibilitando que qualquer um desenvolvedor possa utilizar a lista criada para armazenar seus próprios objetos. Para tanto, o trabalho foi desenvolvido na linguagem Java.

2. CONCEPÇÃO

Para começar a realizar a lista, foi idealizado o projeto inicial de como a lista deveria ser e como se comportaria, para tanto, foi construído o seguinte quadro:

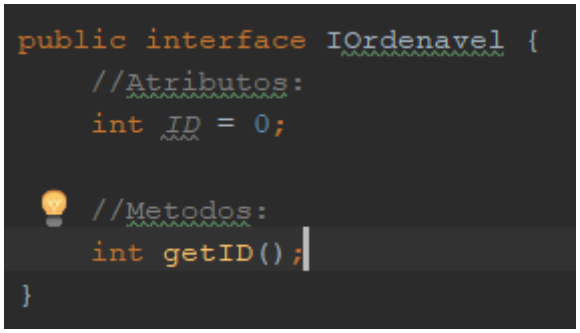


A lista encadeada (Lista) foi idealizada a partir do que vimos em sala da aula, entretanto, ao invés de utilizar um pacote de dados int, criei uma “pacote” de dados genéricos chamado de DadoLista, no qual recebe como argumento um tipo de dado (Objeto) que implementa a interface IOrdenavel. Da lista criada, foi feita a ListaOrdenada.

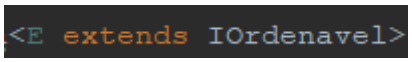
A necessidade de se fazer uma Interface, a IOrdenavel, era de assegurar que todo objeto que eu teria dentro da minha lista possuísse um campo que eu pudesse comparar para realizar a ordenação dos meus elementos. Para tanto, estipulei que todos os dados deveriam ter um dado de identificação (ID) e que, de preferência, fossem únicos.

3. DA CRIAÇÃO DO PROJETO

Comecei implementando pelo IOrdenavel, que seria dali que meu projeto andaria, nada mais simples do que exigir um método que retorne um int o qual seria meu suposto ID, o ID = 0 é só para ele retornar 0 caso a pessoa não faça nada de especial para definir os IDs dos Objetos:



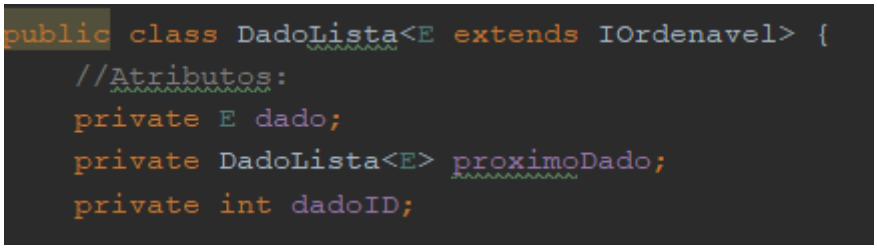
Depois dela, parti a criação do DadoLista e a Lista. A dificuldade foi encontrada na implementação do Generics, visto que era algo que não tinha feito antes. Para tanto, foi utilizado o material disponibilizado pelo moodle do professor de Lucca. É de se frisar que era necessário pedir como parâmetro um objeto que implementasse a interface IOrdenavel, visto que só gostaria de aceitar objetos que possuíssem o campo de ID:



Na Lista, os métodos inserirNoInicio() e inserirNoFinal() foram implementados conforme visto em sala, entretanto o buscar(), inserirDepoisDe() e excluir() foram diferentes.

No inserirDepoisDe(), excluir() e buscar() a dificuldade foi como relizar uma pesquisa na lista para achar o valor para inserir depois, achar o valor para excluir e o valor para buscar. Isso através da ID. Em primeira instância, gostaria de criar uma Objeto que implementasse a IOrdenavel, mas não posso porque desconheço o tipo de objeto que pode estar sendo enviado como parâmetro. Para mim, era necessário colocar esse objeto dentro de um DadoLista(pacote de dados) para poder para meu loop de pesquisa caso encontrasse a ID que eu queria, mas de forma orgânica e sem precisar fazer mais de uma verificação (como vimos na aula que perderíamos muito da performance na pesquisa).

Para tanto, alterei como o meu DadoLista funcionava. Fiz com que desse para através do DadoLista já retirar o ID (puxando do Objeto que implementa IOrdenvael) e colocando como uma atributo:



Próximo passo era permitir que, eu pudesse alterar o dadoID para os casos de quando eu quiser inserir um DadoLista no final da minha lista para efeito de comparação, para quando meu loop passar por toda a lista procurando pela ID que eu quero, ele parar no final caso não encontre antes. Para tanto, adicionei o método setDadoID():

```
public void setDadoID (int novoDadoID) {
    this.dadoID = novoDadoID;
}
```

Nesse momento surgiu outra questão: como controlar a questão do programador folgado que sem querer alterou um dado de ID do meu objeto e não atualizou no DadoLista? E se esse programado alterou o ID do meu DadoLista e ele não condiz com o do meu Objeto com o ID certo? Resolverei isso no método getDadoID:

```
/**
 * Ele ve se existe algum dado de Objeto, se tiver, retorna o ID desse objeto, caso contrario,
 * retorna 0 ou um ID setado pelo metodo setDadoID().
 *
 * @return the ID from the Generic Objeot extended from the IOrdenavel Interface..
 * @return 0 or any settable int (by setDadoID() method) if this DadoLista doesn't have any Generic Objeot.
 */
public int getDadoID () {
    if (this.dado == null) {
        return this.dadoID;
    } else {
        this.dadoID = this.dado.getID();
        return this.dadoID;
    }
}
```

Dessa forma, se eu não colocar nenhum dado no meu DadoLista, eu posso alterar meu ID desse DadoLista para o que eu quiser, assim posso fazer minha pesquisa com uma performance melhor (colocando esse Pacote de dados para o final da minha lista). Caso eu tenho um dado no meu DadoLista, ele sempre vai retornar o ID do meu Objeto que implementa IOrdenavel, não o do meu DadoLista.

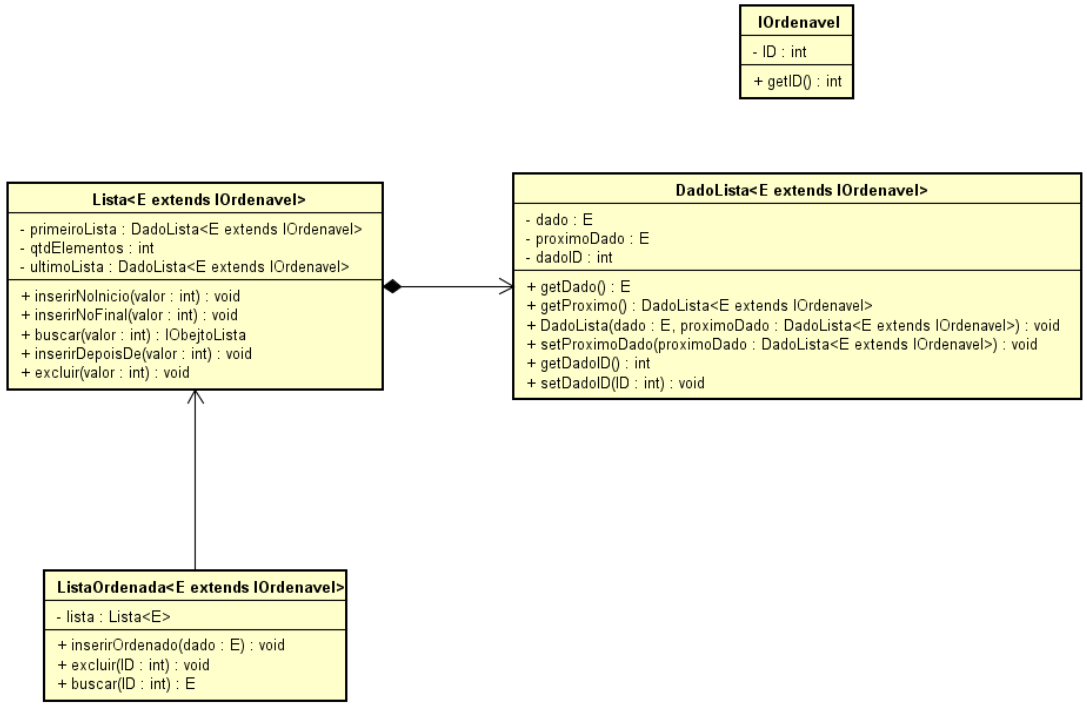
Um exemplo é o do método inserirDepoisDe(), em que coloco um pacote de dados no final da lista com o ID do meu Objeto que quero inserir e passo pela lista procurando por outros objetos com IDs maiores, caso achar, ele é inserido antes do dado de maior ID (inserindo depois da ID), caso não achar, ele é adicionado como ultimo e o meu código não quebra o sistema, visto que não vou dar null pointer caso eu chegar no final e não achar a ID menor, pois coloquei um pacote de dados como segurança:

```
DadoLista<E> auxilairPesquisa = new DadoLista<E>( dado: null,  proximoDado: null);
auxilairPesquisa.setDadoID(ID);
this.ultimoLista.setProximoDado(auxilairPesquisa);

DadoLista<E> auxilairInclusao = this.primeiroLista;

while (auxilairInclusao.getProximo().getDadoID() < ID) { //Procurar pela menor ID antes do que eu quero achar
    auxilairInclusao = auxilairInclusao.getProximo();
}
if (auxilairInclusao.getProximo().equals(this.ultimoLista.getProximo())) { //Caso ele for o ultimo a ser inserido:
    this.ultimoLista.setProximoDado(null);
    this.inserirNoFinal(novoObjeto); //Ja faz qtdElementos++
} else { //Inserir depois da menor aparicao do ID selecionado ou no ID selecionado:
    this.ultimoLista.setProximoDado(null);
    auxilairInclusao.setProximoDado(new DadoLista<E>(novoObjeto, auxilairInclusao.getProximo()));
    this.qtdElementos++;
}
```

Então, no final do projeto, as minhas classes ficaram dessa forma:

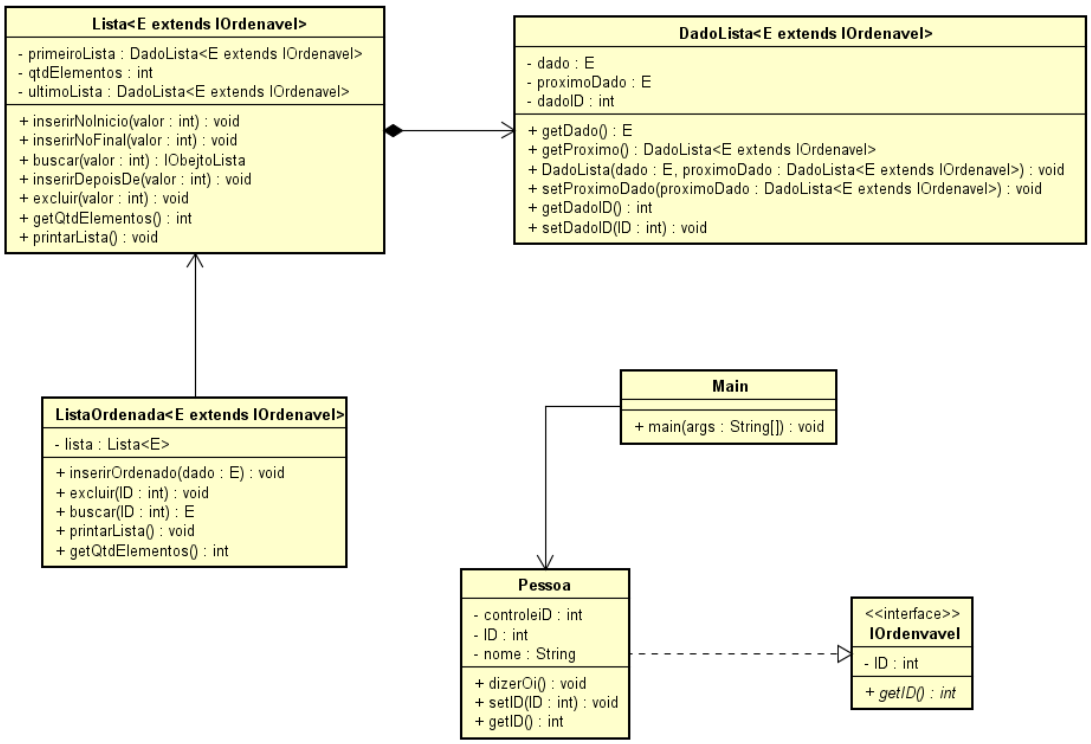


4. TESTES

Para realizar os testes, criei duas classes: o Main e o Pessoa. O Pessoa eu usei como dados para popular a minha ListaOrdenada criada, sendo que Pessoas implementa a interface estipulado (IOrdenavel). Já o Main é onde os testes realmente acontecem. Foram criados duas ListaOrdenadas, e em cada uma deles foram feitos os testes de inserirOrdenado, excluir e buscar.

Para realizar os testes, foi implementado mais dois métodos em ListaOrdenada e Lista, o printarLista(), que imprime todos os IDs da lista no console e o getQtdElementos(), que retorna a quantidade de elementos das

listas. Dessa forma, o projeto final ficou:



Foi adicionado 8 pessoas, alguns de forma desordenada para o algoritmo ordenar, foi pedido o número de inserções na lista e para realizar uma função para dar a ID e retornar o nome da Pessoa dizendo oi:

```
0
1
2
3
4
5000
5010
5020

8
Legolas de ID=2 disse oi!
```

Na outra lista foi adicionado 4 pessoas de IDs desordenadas, pedido o número de inserções e para o ID de int 10 dizer oi. Ddepois foi retirado os Objetos de IDs 3 e 0 e pedido o número de inserções. No final foi adicionado mais duas pessoas de IDs diferentes e pedido o número de inserções. O output foi:

```
----------NOVA LISTA---------

10
11
3
0

0
3
10
11

4
Han Solo de ID=10 disse oi!
10
11

2

5
10
11
20

4
```