Lista 4 - Listas encadeadas e listas encadeadas ordenadas

ESTRUTURA DE DADOS I – Pedro Nuno Moura

Monitores: Victor Hugo Souza Wirz

Para as questões abaixo, considere a implementação de Lista Encadeada e Lista Encadeada Ordenada vistas em sala de aula.

ATENÇÃO: Para todas as questões, deve ser informada e explicada a complexidade computacional alcançada.

1) Qual a complexidade do método `tamanho()` presente na classe `Lista` disponibilizada? É possível melhorá-lo? Caso seja possível, faça as alterações necessárias para tal e diga a complexidade do novo método. Você tem liberdade para criar novos métodos e atributos, assim como modificar métodos e atributos já existentes na classe.

2) Elabore um algoritmo que, dadas duas listas encadeadas ordenadas, intercale as duas listas de forma que a lista resultante seja também ordenada.

3) Implemente uma lista ordenada genérica (usando *Generics* de Java) que armazene nomes em ordem alfabética. Lembre-se de manter a prioridade de ordenação da lista nos métodos implementados. Para cada método, diga a sua complexidade.

4) Suponha que uma frase é representada por uma lista encadeada, sendo que cada elo contém uma string representando um *token*, isto é, um caracter de pontuação ou uma palavra. Escreva um algoritmo para retornar a frase inteira.

5) Suponha agora uma derivação do exercício anterior, em que se deseje reverter uma frase usando listas encadeadas. Além de inverter a ordem da frase em si, também se deseja inverter a ordem das letras de cada palavra individualmente. Implemente então um método para realizar essa tarefa. Note que cada letra de cada palavra deve estar armazenada em um objeto Elo da lista. Tem-se liberdade para pensar na melhor forma de modelar tal problema.

6) Um diretório é uma lista de arquivos e outros diretórios. Assim sendo, crie um programa que receba o nome de um diretório e imprima o nome de todos os arquivos e outros diretórios contidos naquele, de modo que o conteúdo de cada diretório seja recursivamente listado (de forma indentada) sob o nome do diretório pai. Considere o seguinte exemplo:

```
Jogos
    Campo Minado
    Paciência
    World of Warcraft
Faculdade
    EDD1
    EDD2
    AA
    ACEs
        ACE1
        ACE2
        ACE3
    TPD
Filmes
    Super-heróicos
        Marvel
            Vingadores
            Homem-Aranha
        DC
            Superman
    Drama
        Brilho Eterno de Uma Mente Sem Lembranças

    Comédias Românticas
        (Pasta Vazia)
```

Dicas:

- A estrutura de pastas pode ser *hard-coded* no código do seu programa. Pode ser usada a estrutura exemplificada acima ou qualquer outra que cubra os mesmos casos;
- A classe de lista encadeada usada não pode ser de inteiros como a classe básica `Lista` vista em sala de aula. Crie uma nova versão da classe ou a modifique usando *Generics* (ambas versões serão aceitas). As listas devem ser do tipo das classes que vocês criarem para modelar as entidades descritas no enunciado; e
- O bom uso de herança ou de uma interface facilita a construção e impressão da estrutura de pastas.

Seu programa deve ter apenas um comando `System.out.println(diretorio)` para produzir a saída esperada.

7) No desenvolvimento de um jogo, é necessário armazenar os inimigos que seu personagem está enfrentando na fase. Um inimigo é definido como uma instância da classe “Inimigo”, que possui a seguinte estrutura:

```
public class Inimigo
{
      private int vida;
      private double danoPorSegundo;

      public void setVida(int vida);
      public int getVida();
      public void setDanoPorSegundo(double danoPorSegundo);
      public double getDanoPorSegundo();
}
```

Por sua vez, os inimigos são armazenados em uma lista encadeada que possui os seguintes métodos:

```
public void inserir(Inimigo inimigo);
public Inimigo localizar(Inimigo inimigo);
public int efetuarDano(Inimigo inimigo, double dano);
```

Considere que o método `efetuarDano(Inimigo inimigo, double dano)` remove o inimigo da estrutura caso o dano sofrido reduza a vida do objeto `inimigo` a zero. Dessa maneira, implemente essa estrutura de dados para o jogo em questão.