UNIVERSIDADE REGIONAL DE BLUMENAU
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
ALGORITMOS E ESTRUTURAS DE DADOS
PROFESSOR GILVAN JUSTINO

Trabalho – 1

O objetivo deste trabalho é construir um programa capaz de avaliar se a estrutura de um arquivo HTML está correta. Um arquivo HTML é um arquivo utilizado para criar documentos para a web. Trata-se de um arquivo em formato texto constituído de comandos denominados de *tag*, utilizados para formatar o texto a ser exibido no navegador de internet. Veja um exemplo de conteúdo de um arquivo HTML simples:

```
<html>
<body>
<h1>Aqui cabeçalho do arquivo</h1>
<p>Meu parágrafo da página web.</p>
<p>Meu segundo
parágrafo.</p>
</body>
</html>
```

Neste arquivo, as *tags* (comandos) são: `html`, `body`, `h1` e `p`. Observe a terceira linha do arquivo que é constituído da *tag* `h1`. Esta linha possui uma *tag de início*, chamada de `<h1>` e uma *tag final*, chamada de `</h1>`. Normalmente as *tags* do arquivo HTML estão em pares, como exposto neste exemplo, sendo que a *tag final* é igual à *tag de início*, porém possui um caractere "/" após o caractere "<". O mesmo ocorre com as outras *tags* (`html`, `body` e `p`).

É importante observar também que uma *tag final* somente pode ser inserida quando não houver outra nova *tag de início* que ainda não possui sua *tag final.* Isto é, a *tag final* `</body>` não pode ser colocada antes de `</h1>`, já que a *tag de início* `<h1>` deve ser finalizada antes.

Um arquivo HTML pode ser constituído de diversas *tags*, além de `html`, `body`, `h1` e `p` (estes são apenas alguns exemplos de *tags*).

O seu programa deverá permitir receber um arquivo a ser analisado e o programa deverá verificar se a estrutura de *tag inicial* e *tag final* estão corretas. Para resolver este problema, implemente uma pilha de *tags*: quando você extrair do arquivo uma *tag de início,* coloque-o numa pilha. Quando ler uma *tag final*, retire um dado da pilha e verifique se o dado retirado corresponde a *tag de início* da *tag final*, isto é, verifique se forma um par. Caso não formar, há um erro de formatação de *tags*.

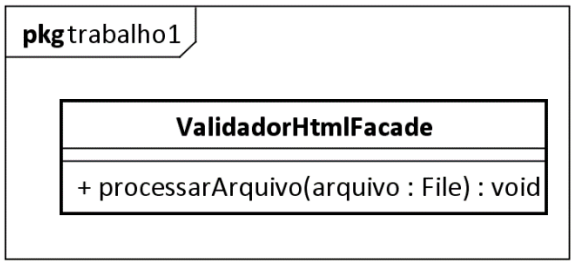
Interpreta-se que o arquivo está bem formatado quando todas as *tags de início* possuem suas respectivas *tags finais*. Caso houver, a estrutura do arquivo está mal formatada e seu programa deverá informar ao usuário que existem *tags de início* sem *tag final*.

Existe algumas exceções para se tratar neste programa:

a) Linhas em branco devem ser desprezadas;
b) Uma *tag* pode ser constituída de atributos, como o exemplo abaixo:
   `<a href="http://www.w3schools.com">Isto é um link</a>`
   Neste exemplo, a *tag de início* é `<a>` e ela possui o atributo `href`. Os atributos não são importantes para nossa análise, mas deve-se reconhecer que a tag é `<a>` e não `<a href...>`.
c) Existe um conjunto de *tags*, denominadas de *singleton tags*. Estas *tags* não têm uma *tag final* e portanto, não deve ser validado se a tag possui uma *tag final*. Considere que as *singleton tags* são: `meta`, `base`, `br`, `col`, `command`, `embed`, `hr`, `img`, `input`, `link`, `param`, `source` e `!DOCTYPE`. Todas estas *tags* podem ou não ser utilizadas com atributos.

Quando um arquivo estiver corretamente formatado você também deverá apresentar a relação de *tags* utilizadas no arquivo, bem como a quantidade de vezes que apareceram no arquivo. No exemplo anterior, o programa deveria informar que a tag `html` foi utilizada uma vez e a *tag* `p` foi utilizada 2 vezes, por exemplo.

Para construir a solução, considere a construção de uma classe denominada `ValidadorHtmlFacade` que deve ter um método chamado `processarArquivo()`, como no diagrama de classes abaixo.



Esta classe deve receber como argumento um arquivo (objeto da classe `File`) e apresentar na tela (console ou tela gráfica) a análise do arquivo. O projeto pode ser organizado em mais classes, porém para testar o programa, será utilizado o método `processarArquivo()` da classe `ValidadorHtmlFacade`. Todas as classes do projeto devem pertencer ao pacote `trabalho1`.

Os requisitos deste trabalho são:

1. Construir diagrama de classes da solução;
2. Criar um programa que implemente a classe `ValidadorHtmlFacade`, conforme descrito acima;
3. O programa deverá avaliar a estrutura do arquivo fornecido pelo usuário e indicar se o arquivo está bem formatado ou não, considerando os pares *tag de início/fim*, *singleton tags* e *tags* com atributos;
4. Se o arquivo estiver bem formatado, deverá ser apresentado na tela (console ou tela gráfica) uma relação das *tags* encontradas bem como a frequência de cada uma. As *singleton tags* também devem ser computadas*;*
5. Se o arquivo não estiver bem formatado, o programa deverá indicar qual a razão, que poderá ser:
   5.1. Foi encontrada uma *tag final* inesperada (aguardava-se determinada *tag final* mas foi encontrada outra). Deve-se indicar qual a *tag final* encontrada e qual a *tag final* esperada;
   5.2. Faltam *tags* finais. Neste caso, apresentar quais as *tags finais* esperadas mas não encontradas.

O trabalho pode ser feito em dupla. Deve ser submetido no AVA na pasta “Trabalho 1” até 07/10/2017, num arquivo compactado com o nome “trabalho1.zip”, contendo todos os fontes e o diagrama de classes em formato .jpg. Após esta data, serão descontados 2 pontos para cada dia de atraso. Caso o trabalho seja feito em dupla, o trabalho deverá ser submetido apenas na pasta de um dos alunos. Neste caso, usar a anotação `@author` na classe `ValidadorHtmlFacade` com os nomes dos membros da equipe. Não é preciso enviar uma classe com o método `main()` pois para validar a implementação será utilizada uma classe construída pelo professor.

A solução que for implementada não pode utilizar nenhuma classe de coleções da API Java (`ArrayList`, `HashMap`, etc). Utilize apenas as estruturas de dados implementadas nos exercícios em laboratório.