///AXUR

Teste técnico: Software Development Intern

Requisitos funcionais

Objetivo: a partir de uma URL, obter o trecho de texto contido no nível mais profundo da estrutura HTML de seu conteúdo. Por exemplo:

`http://hiring.axreng.com/internship/example1.html`

```
<html>
  <head>
    <title>
      Este é o título.
    </title>
  </head>
  <body>
    Este é o corpo.
  </body>
</html>
```

Na estrutura HTML acima, o trecho desejado como retorno é "`Este é o título.`" (sem as aspas), porque está em 3 níveis de profundidade (`html > head > title`), enquanto o trecho "`Este é o corpo.`" está em 2 níveis (`html > body`). Se dois ou mais trechos estiverem no nível máximo de profundidade do documento, o primeiro deles deve ser retornado.

Para simplificar o escopo do problema, a solução deve se basear nas seguintes premissas:

1. O código HTML está dividido em linhas;
2. Cada linha pode ser apenas de um dos seguintes tipos:
   a. Tag de abertura (exemplo: `<div>`)
   b. Tag de fechamento (exemplo: `</div>`)
   c. Trecho de texto (exemplo: "`Este é o corpo.`")
3. Uma mesma linha não pode conter dois tipos de conteúdo;
4. Apenas elementos HTML com pares de tags de abertura e fechamento são utilizados (exemplo: `<div>` e `</div>`, mas não `<br/>`)
5. Tags de abertura não possuem atributos (contra-exemplo: `<a href="link.html">`).

Cada linha pode ou não ter espaços iniciais, utilizados meramente para indentação, que devem ser ignorados. Linhas em branco também devem ser ignoradas.

Opcional: pontos bônus serão concedidos caso a solução seja capaz de identificar estruturas HTML mal-formadas, retornando nesse caso a mensagem "`malformed HTML`" (sem as aspas).

Requisitos técnicos

1. A solução deve ser desenvolvida como um programa Java a ser compilado e executado pela linha de comando, utilizando o JDK 17.
2. Não é permitido o uso de quaisquer bibliotecas e *frameworks* externos ao JDK. Também não é permitido o uso de *packages* e classes nativos do JDK relacionados à manipulação de HTML, XML ou DOM (como `javax.xml` ou quaisquer outros).
3. Para a compilação do programa deve ser suficiente executar o seguinte comando a partir do diretório que contém o(s) arquivo(s) de código fonte, sem quaisquer alterações:
   `javac HtmlAnalyzer.java`
4. Para a execução do programa deve ser suficiente executar o seguinte comando, a partir do diretório onde foi feita a compilação (item anterior), alterando apenas o argumento que contém a URL a ser analisada para uma URL válida:
   `java HtmlAnalyzer inserir-url-aqui`
5. O programa deve gerar apenas os seguintes tipos de *output* no console padrão:
   a. Linha de trecho de texto identificado no HTML; ou
   b. Mensagem "`malformed HTML`" (caso implementada funcionalidade que vale pontos bônus); ou
   c. Mensagem "`URL connection error`" (caso não seja possível obter o conteúdo HTML por falha de conexão).
6. O código fonte da solução (apenas arquivos `.java` e opcionalmente um `README.md`, compatíveis com UTF-8) deve ser entregue em um arquivo *tar* (`.tar` ou `.tar.gz`), cujo nome deve ser igual ao nome do(a) candidato(a), sem acentos ou cedilhas e com espaços substituídos por *underscore* ("_"). Não deve haver diretórios dentro do arquivo *tar*. Para um candidato chamado "Fulano de Tal", o arquivo entregue deve ter o nome `fulano_de_tal.tar` (ou `.tar.gz`).

Avaliação

A solução será avaliada de acordo com os requisitos funcionais e técnicos descritos nas seções anteriores, bem como em relação ao uso de boas práticas de programação e *design* de software orientado a objetos. Testes automatizados serão aplicados pela equipe de avaliação da Axur, de forma que **soluções que não seguirem à risca os requisitos serão desclassificadas automaticamente**.