Web Ex. 1.5.1 (Word and line count)

Faça o Web Exercise 1.5.1 (Word and line count). Seu programa Wc.java deve receber a entrada na entrada padrão e deve enviar para a saída padrão o número de linhas, palavras e caracteres, separados pelo caractere '\t' (TAB).

Por exemplo,

```
$ java-introcs Wc < Wc.java
30 93 715
$ java-introcs Wc < bible_KJ.txt
100223 824150 4351874
```

O arquivo bible_KJ.txt está disponível em: `https://www.ime.usp.br/~yoshi/DATA/Gutenberg/`

Seu programa será uma versão simples do utilitário wc do Unix:

```
$ wc < Wc.java
30 93 715
$ wc < bible_KJ.txt
100223 824150 4351874
```

Alguns programas que podem ajudá-lo estão disponibilizados (Cat.java, CharCount.java, LineCount.java).

Nesse exercício, uma "palavra" é uma sequência contígua (um segmento) maximal de caracteres diferentes de espaço. Aqui um "espaço" é basicamente um branco, uma mudança de linha ('\n'), um TAB ('\t') etc. Para saber se um caractere ch é um espaço, você pode usar *Character.isWhitespace(ch)*.

Ex.

```
if (Character.isWhitespace(ch))
   StdOut.println("Eh espaco");
else
   StdOut.println("Nao eh espaco");
```

Importante: para contar as palavras no texto, você deve implementar um algoritmo baseado no seguinte esquema. Coloque a entrada padrão inteira em um string s; leia os caracteres de s um a um; nesse processo, você deve identificar os segmentos maximais de não-espaços (isto é, as palavras); toda vez que você identificar uma palavra, você deve incrementar seu contador.

Bônus. Um jeito mais fácil de contar palavras é baseado no uso do método split() para strings. Veja

`http://stackoverflow.com/questions/7899525/how-to-split-a-string-by-space`

Escreva uma variante de seu programa chamado Wc2.java baseado nessa ideia (o uso de split() basicamente trivializa esse exercício).