

Semana 1

File Tools



André Souto

Unidade Curricular de
Laboratório de Programação

2017/2018

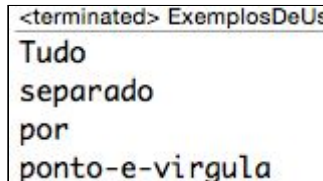
Objectivos

- Leitura e manipulação de textos contidos em ficheiros;
- Manipulação de Strings.
- Conversão do tipo char em int e vice versa.

Antes de Começar

De modo a poder realizar este projecto deverá recordar como utilizar as classes que lhe permitam ler e escrever dados em ficheiros de texto. Sugestão: Em algumas classes, como por exemplo na classe *String*, existem métodos (por exemplo **split**) que permitem que dada uma *String* **padrao** como argumento se possa dividir a *String* em vários pedaços de acordo com o **padrao** dado.

```
public class ExemplosDeUso {  
  
    public static void main(String args[]) {  
        String linha = "Tudo;separado;por;ponto-e-virgula";  
  
        String[] resultados = linha.split(";");  
  
        for (String str: resultados) {  
            System.out.println(str);  
        }  
    }  
}
```



```
<terminated> ExemplosDeUso  
Tudo  
separado  
por  
ponto-e-virgula
```

A *String* que o método `split` recebe representa um padrão que é usado como separador dos elementos da *String*. Na verdade é uma expressão regular que pode ser muito simples ou muito complexa. Mais à frente no semestre irá haver uma aula dedicada ao assunto. No projecto apenas precisa de saber que se quiser separar por dois identificadores distintos deve indicá-los entre parênteses rectos. Por exemplo, se tivesse usado a expressão `linha.split("[-;]")` além de separar por `;` também separaria pelo travessão `-`. O mesmo se pode aplicar ao método **replace**.

Internamente, o tipo de dados `char` são representados por números. O código de representação de caracteres mais usado é o código ASCII (**American Standard Code for Information Interchange**) que representa cada caracter por um número inteiro de forma sequencial. A correspondência entre caracteres e o número que o representa pode ser consultada em <http://www.asciitable.com/>. Por exemplo, o

caractere `a` tem código ASCII 97. Tirando partido desta correspondências os tipos primitivos `char` e `int` têm a particularidade de poderem ser transformados de forma automática um num outro usando a tabela de conversão de código ASCII. Por exemplo a instrução

```
System.out.println((int) 'a' + " " + (char) 98);
```

imprime na consola o resultado `"97 b"`.

Enunciado

Relembre a equipe TUTEDECC ("Tem Um TEXto Demasiado Claro? Contacte-nos!") de Introdução à Programação e o seu estudo sobre várias técnicas de tratamento de textos, quer para obscurecimento, quer para análise.

Com este projecto pretende-se enriquecer o leque de técnicas à disposição da equipe da TUTEDECC. Assim, é necessário criar novas formas de manipular ficheiros de texto e processá-los linha a linha.

As técnicas pretendidas a aplicar em cada *String* correspondente a uma linha do ficheiro de texto, são as seguintes:

- **Capitalização personalizada:** que pode ser feita de três formas distintas
 - **Tudo minúsculas** - todo o texto é transformado em minúsculas.
 - **Tudo maiúsculas** - todo o texto é transformado em maiúsculas.
 - **Apenas as primeiras letras maiúsculas** - as únicas letras em maiúscula são a da primeira de cada palavra do início de linha ou sempre que haja um ponto seguido de um espaço. Assuma que o texto não tem indentação.
- **Omissão de caracteres:** Retira do texto original um conjunto dado de caracteres.
- **Trocar algarismos por letras:** Depois de eliminar do texto todos os espaços, todas as vírgulas e todos os pontos finais, troca os algarismos pela sua escrita em português (troca 0 por zero, 1 por um, 2 por dois, 3 por tres (sem acentos), etc).
- **Rotação das letras:** Que assumindo que o texto apenas tem como pontuação pontos finais e vírgulas, elimina essa pontuação e os os espaços e aplica uma rotação de n posições a cada uma das letras, isto é, se a rotação for de 5, o "a" e o "A" devem ser substituídos por "f", o "b" e o "B" por "g",..., o "z" e o "Z" por "e".

Para medir quão eficazes são estas técnicas, a TUTEDECC quer também implementar algumas métricas de similitude e para isso precisa de métodos para as seguintes tarefas:

- **Contagem e ordenação**
 - **Por ordem alfabética** - que contabiliza a frequência relativa de cada letra (minúscula ou maiúscula) e ordena alfabeticamente essas frequências;
 - **Por ordem de frequência** - que contabiliza a frequência relativa de cada letra (minúscula ou maiúscula) e ordena por ordem decrescente da frequência (se houver duas com a frequência igual, por ordem alfabética).
- **Identificação das linhas** (e o número de ocorrências) que cada um dos padrões dados aparece nessa linha e em todo o texto.

Note que neste trabalho, qualquer das técnicas de tratamento referidas é para ser aplicada linha a linha num ficheiro de texto. Assim sendo e tal como se descreve a seguir, deve criar métodos com assinaturas que recebem os nomes de ficheiros de texto a usar (quer de leitura dos textos, quer para escrita das alterações necessárias) e métodos auxiliares que permitam aplicar as referidas técnicas a uma *String*.

O que fazer

Deve desenvolver a classe TUTEDECC e os métodos com as descrições feitas acima e cujas assinaturas são as apresentadas a seguir. Note que as *Strings* identificam os nomes dos ficheiros.

- **public static void** capitalizar (String fileIn, **int** tipo, String fileOut) **throws** IOException que aplica linha a linha do ficheiro de texto *fileIn* a capitalização pretendida e escreve o resultado no ficheiro com o nome *fileOut*. A capitalização a aplicar depende do parâmetro *tipo*:
 - Se *tipo* tiver o valor -1 aplica a capitalização tudo minúsculas;
 - Se *tipo* tiver o valor 1 aplica a capitalização tudo maiúsculas;
 - Se *tipo* tiver o valor 0 aplica a capitalização à primeira letra da linha ou à primeira letra depois de um ponto final.

capitalizar(in, -1, out)	capitalizar(in, 1, out)	capitalizar(in, 0, out)
se o ficheiro in contiver o texto: uma linha. na Mesma segunda linha	se o ficheiro in contiver o texto: uma linha. na Mesma segunda linha	se o ficheiro in contiver o texto: uma linha. na Mesma segunda linha
o ficheiro out deve conter: uma linha. na mesma segunda linha	o ficheiro out deve conter: UMA LINHA. NA MESMA SEGUNDA LINHA	o ficheiro out deve conter: Uma linha. Na mesma Segunda linha

- **public static void** retiraCaracteres (String fileIn, String fileLetras, String fileOut) **throws** IOException que retira os caracteres (*case sensitive*) escritos na primeira linha do ficheiro de texto de nome *fileLetras* do ficheiro de texto com o nome *fileIn*, linha a linha. O resultado deve ser guardado no ficheiro com o nome *fileOut*.

```
retiraCaracteres(in, out, "um ")

se o ficheiro in contiver o texto:
uma linha. na Mesma
segunda linha

o ficheiro out deve conter:
alinha.naMesa
segndalinha
```

- **public static void** `numerosPorLetras` (`String fileIn`, `String fileOut`) **throws** `IOException` que a cada linha do ficheiro de texto com o nome `fileIn` retira os pontos finais e os espaços e troca todos os algarismos por a sua escrita em extensão.

```
numerosPorLetras(in, out)

se o ficheiro in contiver o texto:
1. 2 3 4 ABC
5 6 ... 7

o ficheiro out deve conter:
umdoistresquatroABC
cincoseisete
```

- **public static void** `rotacao` (`String fileIn`, `String fileOut`, `int quanto`) **throws** `IOException` que assumindo que a pontuação existente no ficheiro de texto `fileIn` são o ponto final (.), a vírgula (,) e só contém letras e espaços aplica linha a linha a remoção da pontuação e dos espaços e uma rotação de `quanto` a cada uma das letras.

```
rotacao(in, out, 6)

se o ficheiro in contiver o texto
abc. z

e o int for 6 (ou seja uma rotação de 6):

o ficheiro out deve conter:
ghif
```

- **public static void** `frequenciasLetras` (`String fileIn`, `String fileOut`, `int tipo`) **throws** `IOException` que, de acordo com o parâmetro `tipo` faz a tarefa de contagem e ordenação descrita acima.
 - Se `tipo` tiver o valor 0 deve apresentar os resultados das frequências relativas por ordem alfabética;
 - Se `tipo` tiver o valor 1 deve apresentar os resultados das frequências relativas por ordem decrescente (e depois alfabética) das frequências.

```
frequenciasLetras(in, out, 0)

se o ficheiro in contiver o texto:
ccc bb a

o ficheiro out deve conter:
a 0.16666...
b 0.33333...
c 0.5
d 0.0
...
```

```
frequenciasLetras(in, out, 1)

se o ficheiro in contiver o texto:
ccc bb a

o ficheiro out deve conter:
c 0.5
b 0.3333...
a .01666...
d 0
...
```

- **public static void** detectaPadrao (String fileIn, String filePadroes, String fileOut) **throws** IOException que lê do ficheiro de texto com o nome filePadroes os padrões que estão separados por espaço e procura linha a linha no ficheiro de texto com o nome fileIn cada um dos padrões, indicando (se aparecerem!) quantas vezes aparecem e no final a contagem total de cada um dos padrões, conforme o exemplo a seguir.

```
detectaPadrao(in, out, padroes)

se o ficheiro in contiver o texto:
abc cd abab
ab

e o ficheiro padroes contiver o texto:
ab cd de

o ficheiro out deve conter:
ab ocorre na linha 1, 3 vez(es)
cd ocorre na linha 1, 1 vez(es)
ab ocorre na linha 2, 1 vez(es)

Numero de ocorrencias dos padroes:
ab 4
cd 1
de 0
```

Para aferir (em parte) a correção do seu projecto pode usar a classe `RunSemana1.java` que lhe é dada conjuntamente com os ficheiros de input e de output esperados bem como as classes de testes individuais.

Atenção

Antes de submeter o trabalho, recorde o protocolo de submissão dos trabalhos, certificando-se que **TODOS** os passos descritos no documento são cumpridos.

O que entregar

Deve criar o ficheiro **semana1.zip**, contendo os ficheiros `.java`, com a instrução

zip semana1.zip *.java

Deve seguir as indicações dadas no documento com os procedimentos de submissão.