



#R4E

Software Developer

Algoritmia e Programação

Ficheiros



Conteúdo

- Ler Ficheiros de Texto
- Escrever Ficheiros de Texto
- Casos de Uso

Ficheiro

“Conjunto de dados gravados no suporte físico de um sistema informático”¹

- É um conjunto de informação guardado num suporte de armazenamento (secundário) de forma durável.
- *“Pode ser considerado como um objeto, possuindo um nome que o identifica, atributos e valores”².*
- Pode conter dados estruturados ou não.
 - Não estruturados - possuem apenas sequências de bytes.
 - Estruturados - podem ser organizados em registos ou em árvore, por ex.
- Um ficheiro de texto é composto por caracteres

Ficheiro

File

- É uma representação abstrata de nomes de caminho de ficheiros e diretórios no disco.

Métodos:

```
public boolean exists()  
public boolean isFile()  
public boolean isDirectory()  
public String getPath()  
public String getName()  
...
```

```
File file = new File("/temp/samplefile1.txt");
```

```
File file = new File("c:\\temp\\samplefile1.txt");
```

Separadores:

/ (UNIX + Windows)
\\ (Windows)

Ficheiros de Texto

PrintWriter (java.io.PrintWriter)

- Permite escrever texto formatado num ficheiro de texto.
- Permite usar métodos de formatação tal como em System.out: *print()*, *println()*, *printf()*

```
public static void usingPrintWriter() throws FileNotFoundException {
```

```
    String fileContent = "Hello. Welcome to APROG.";
```

Exceção verificável

```
    File file = new File("c:/temp/samplefile3.txt");
```

```
    PrintWriter printWriter = new PrintWriter(file);
```

Criar ficheiro

```
    printWriter.print(fileContent);
```

```
    printWriter.println();
```

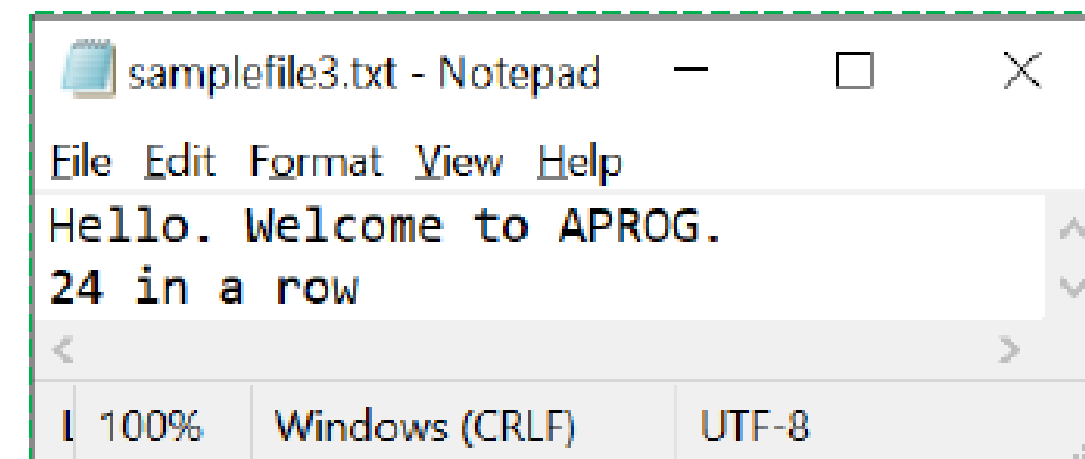
```
    printWriter.printf("%d in a %s", 24, "row");
```

```
    printWriter.close();
```

Fechar ficheiro

```
}
```

```
PrintWriter printWriter = new PrintWriter(new File("c:/temp/samplefile3.txt"));
```



Ficheiros de Texto

Formatter (java.util.Formatter)

- Permite escrever texto formatado num ficheiro de texto.
- Permite usar o método de formatação *format()*

```
public static void usingFormatter() throws FileNotFoundException {
```

```
    String fileContent = "Hello. Welcome to APROG.";
```

Exceção verificável

```
    File file = new File("c:/temp/samplefile3.txt");
```

```
    Formatter formatter = new Formatter(file);
```

Criar ficheiro

```
    Formatter formatter = new Formatter(new File("c:/temp/samplefile3.txt"));
```

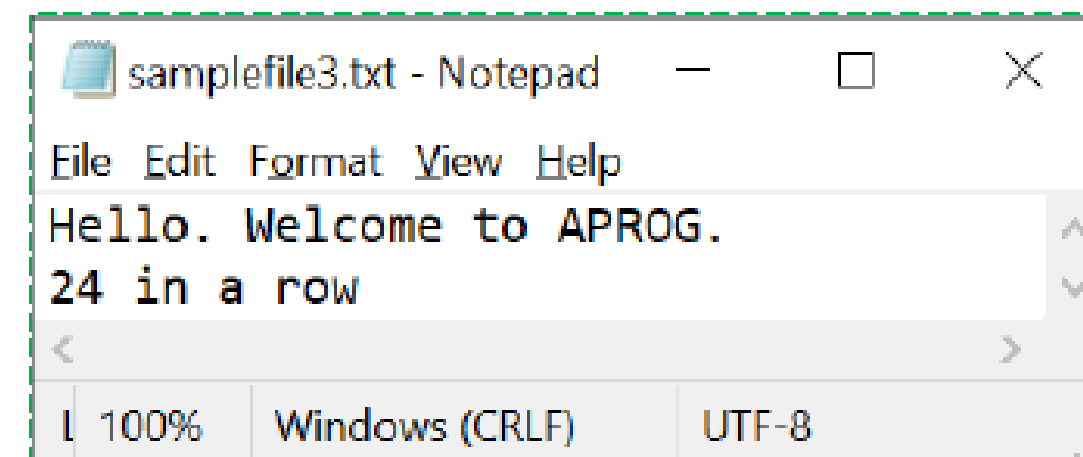
```
    formatter.format(fileContent + "\n");
```

```
    formatter.format("%d in a %s", 24, "row");
```

```
    formatter.close();
```

Fechar ficheiro

```
}
```



Ficheiros de Texto

Scanner (java.util.Scanner)

- Permite converter texto para tipos primitivos e strings (exceto char)
- O texto pode ser obtido de diversas fontes, por exemplo:
 - Teclado (System.in)
 - Strings
 - Ficheiros
- Permite separar o texto em *tokens*, que são sequências de caracteres separados por *delimitadores*
- Por omissão, os *delimitadores* são: espaço, tab e mudança de linha
- Os tokens resultantes podem ser convertidos em valores de diferentes tipos usando os vários métodos “*next...*”: *nextInt()*, *nextDouble()*, *next()*, *nextLine()*...

Ficheiros de Texto

Scanner (java.util.Scanner)

```
public static void usingScanner() throws FileNotFoundException {
```

```
    File file = new File("samplefile3.txt");
```

```
    Scanner sc = new Scanner(file);
```

```
    System.out.println( sc.nextLine() );
```

```
    System.out.println( sc.nextInt() );
```

```
    System.out.println( sc.next() );
```

```
    System.out.println( sc.nextLine() );
```

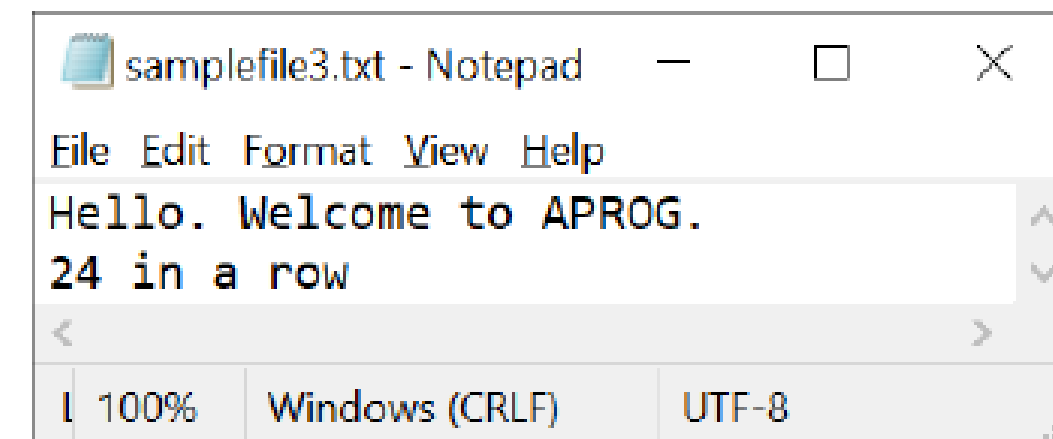
```
    scanner.close();
```

```
}
```

Exceção verificável

Abrir ficheiro

Scanner sc = new Scanner(new File("c:/temp/samplefile3.txt"));



```
Hello. Welcome to APROG.  
24  
in  
a row
```


Ficheiros de Texto

Scanner (java.util.Scanner)

Verificar se existe informação na entrada de dados

`hasNext()`

`hasNextInt()`

`hasNextDouble()`

`hasNextFloat()`

`hasNextLine()`

...

Ficheiros de Texto

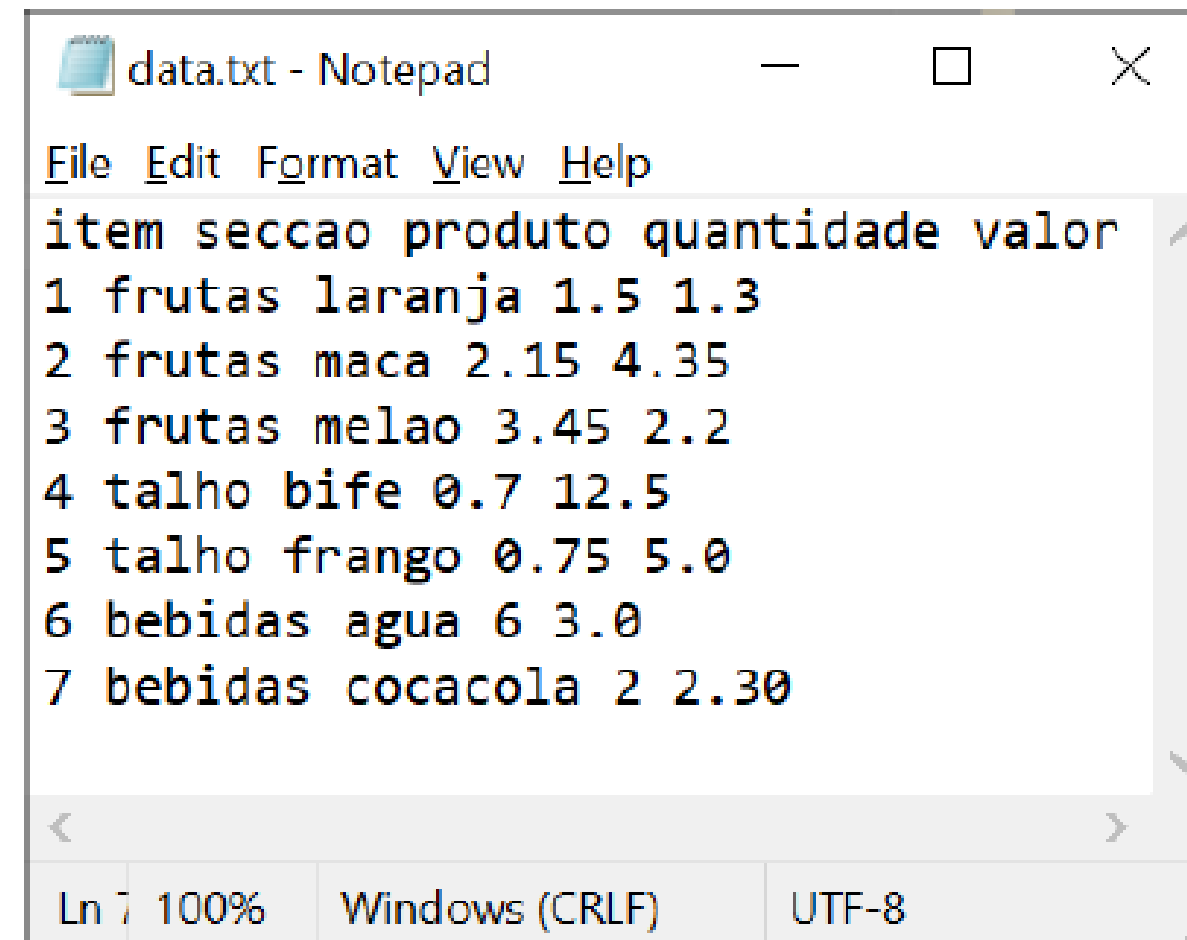
Notas importantes

- Se tentar aceder a um ficheiro de texto que não existe, é lançada a exceção *FileNotFoundException*
- Deve sempre fechar um ficheiro após a utilização (*close()*) para libertar recursos.
- O método *close()* invoca o método *flush()* para garantir que antes de libertar os recursos, sejam primeiro gravados no ficheiro quaisquer bytes em buffer.

Ficheiros de Texto

Considere o seguinte:

- Um ficheiro de texto contém informação sobre as compras efetuadas num espaço comercial.
- A 1ª linha do ficheiro contém o significado do conteúdo de cada uma das linhas seguintes.
- Cada linha do ficheiro (exceto a 1ª) contém informação sobre um item particular.
- O conteúdo do ficheiro é o seguinte:



```
data.txt - Notepad
File Edit Format View Help
item seccao produto quantidade valor
1 frutas laranja 1.5 1.3
2 frutas maca 2.15 4.35
3 frutas melao 3.45 2.2
4 talho bife 0.7 12.5
5 talho frango 0.75 5.0
6 bebidas agua 6 3.0
7 bebidas cocacola 2 2.30
Ln 7 100% Windows (CRLF) UTF-8
```

Ficheiros de Texto

Problema1: Ler e visualizar a informação de todos os items comprados.

Scanner

```
public static void lerLinhasInteirasDoFicheiro() throws FileNotFoundException {
```

```
    Scanner in = new Scanner( new File("data.txt") );
```

```
    String linha = in.nextLine(); //linha do cabecalho
```

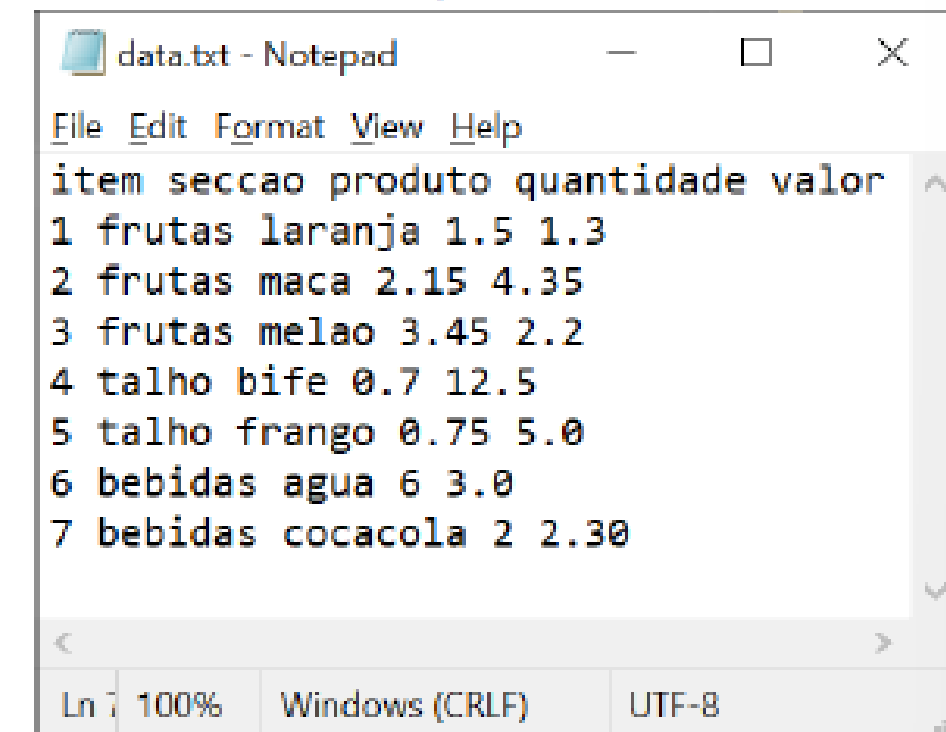
```
    while (in.hasNextLine()) {  
        linha = in.nextLine();
```

```
        System.out.println(linha);
```

```
    }
```

```
    in.close();
```

```
}
```



```
data.txt - Notepad  
File Edit Format View Help  
item seccao produto quantidade valor  
1 frutas laranja 1.5 1.3  
2 frutas maca 2.15 4.35  
3 frutas melao 3.45 2.2  
4 talho bife 0.7 12.5  
5 talho frango 0.75 5.0  
6 bebidas agua 6 3.0  
7 bebidas cocacola 2 2.30  
Ln 1 100% Windows (CRLF) UTF-8
```

```
1 frutas laranja 1.5 1.3  
2 frutas maca 2.15 4.35  
3 frutas melao 3.45 2.2  
4 talho bife 0.7 12.5  
5 talho frango 0.75 5.0  
6 bebidas agua 6 3.0  
7 bebidas cocacola 2 2.30
```

Ficheiros de Texto

Problema2: visualizar o valor total gasto nas compras.

Scanner

```
public static void calcularTotalDasCompras() throws FileNotFoundException {
```

```
    Scanner in = new Scanner(new File("data.txt"));
```

```
    String linha = in.nextLine(); //linha do cabeçalho
```

```
    int item;
```

```
    String seccao, produto;
```

```
    double quantidade, valor, total = 0;
```

```
    while (in.hasNext()) {
```

```
        item = in.nextInt();
```

```
        seccao = in.next();
```

```
        produto = in.next();
```

```
        quantidade = in.nextDouble();
```

```
        valor = in.nextDouble();
```

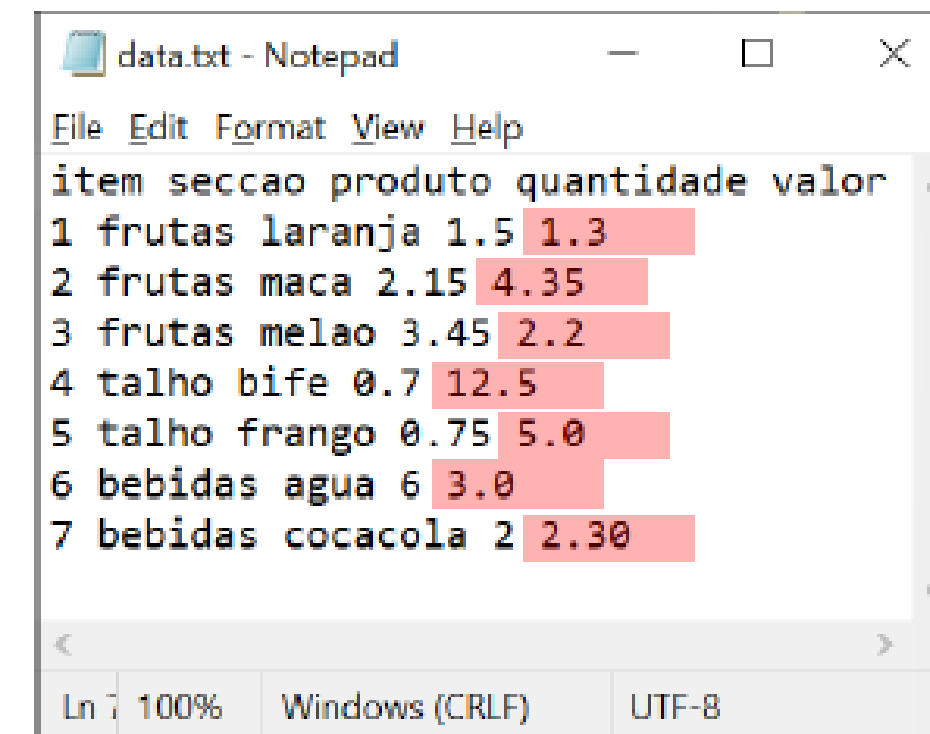
```
        total += valor;
```

```
    }
```

```
    in.close();
```

```
    System.out.println("total=" + total);
```

```
}
```



```
data.txt - Notepad
File Edit Format View Help
item seccao produto quantidade valor
1 frutas laranja 1.5 1.3
2 frutas maca 2.15 4.35
3 frutas melao 3.45 2.2
4 talho bife 0.7 12.5
5 talho frango 0.75 5.0
6 bebidas agua 6 3.0
7 bebidas cocacola 2 2.30
Ln 7 100% Windows (CRLF) UTF-8
```

```
total=30.6500000000000002
```

String.split()

O método `split()` corta uma string sempre que encontra nela uma expressão idêntica à expressão do parâmetro do método `split` e constrói um array resultado com as partes cortadas da string.

String linha = "1|frutas|laranja|1.5|1.3"

Corta a string sempre que encontra um espaço

String[] itensDaLinha = linha.split(" ");

itensDaLinha	"1"	0
	"frutas"	1
	"laranja"	2
	"1.5"	3
	"1.3"	4

Resulta num array de strings

String.split()

Exemplo: Imprimir todas as palavras de uma string (que estão separadas por espaços).

```
String linha = "1 frutas laranja 1.5 1.3"
```

delimitador

```
String[] itensDaLinha = linha.split(" ");
```

```
for(int i=0; i < itensDaLinha.length; i++)  
    System.out.println( itensDaLinha[i] );
```

itensDaLinha	"1"	0
	"frutas"	1
	"laranja"	2
	"1.5"	3
	"1.3"	4

```
1  
frutas  
laranja  
1.5  
1.3
```

Ficheiros de Texto

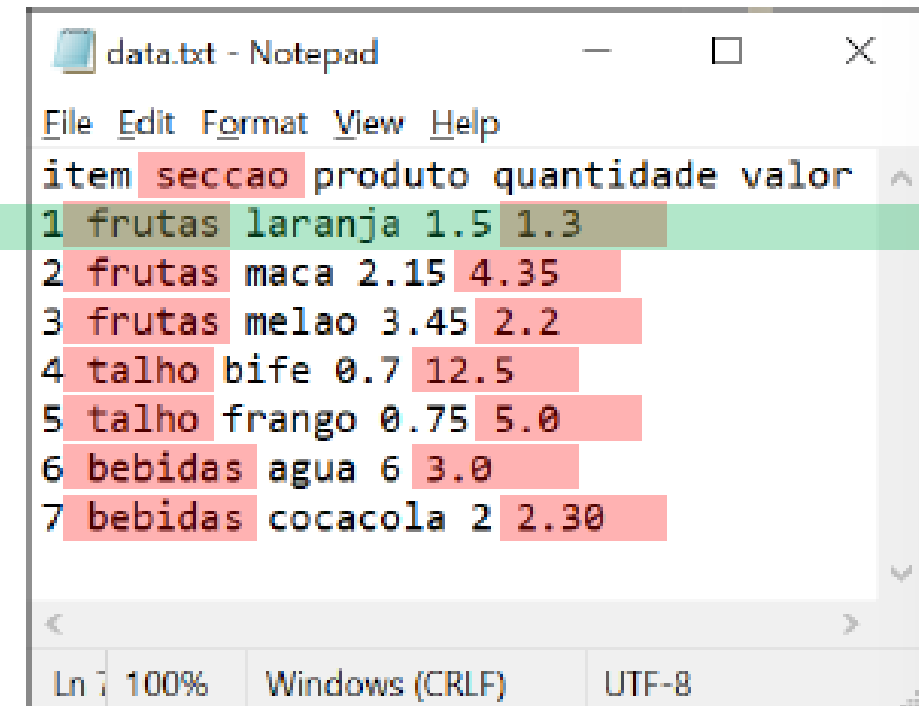
Problema3: visualizar o valor total gasto numa secção específica.

Scanner

```
public static void totalGastoNaSeccao(String seccao) throws FileNotFoundException {  
  
    Scanner in = new Scanner(new File("data.txt"));  
  
    String linha;  
    double total = 0;  
  
    while (in.hasNextLine()) {  
        linha = in.nextLine();  
        String[] itensDaLinha = linha.split(" ");  
  
        if (seccao.equals(itensDaLinha[1])) {  
            total += Double.parseDouble(itensDaLinha[4]);  
        }  
    }  
    in.close();  
  
    System.out.println("seccao [" + seccao + "] total=" + total);  
}
```

itensDaLinha	"1"	0
	"frutas"	1
	"laranja"	2
	"1.5"	3
	"1.3"	4

seccao [frutas] total=7.85



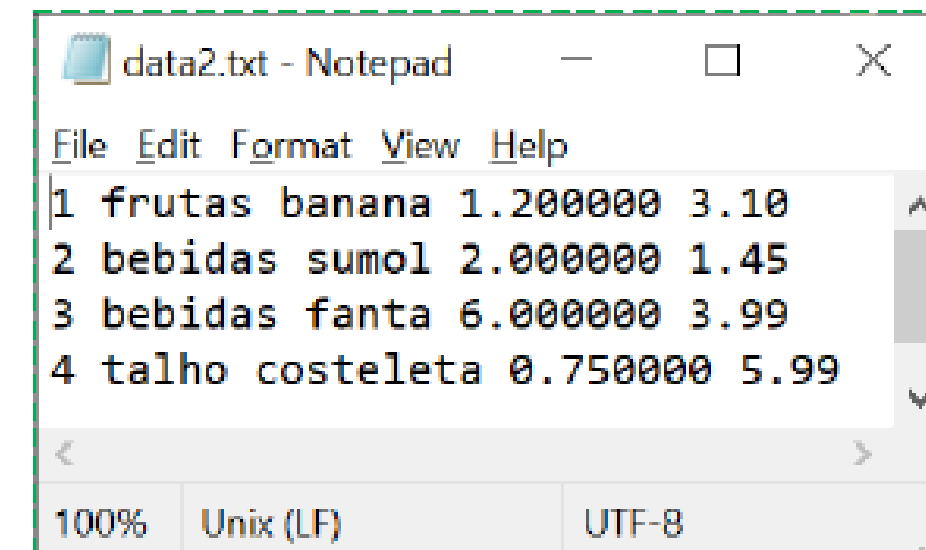
item	seccao	produto	quantidade	valor
1	frutas	laranja	1.5	1.3
2	frutas	maca	2.15	4.35
3	frutas	melao	3.45	2.2
4	talho	bife	0.7	12.5
5	talho	frango	0.75	5.0
6	bebidas	agua	6	3.0
7	bebidas	cocacola	2	2.30

Ficheiros de Texto

Problema4: gravar no ficheiro "data2.txt" alguns items de compras.

PrintWriter

```
public static void escreverParaFicheiro() throws FileNotFoundException {  
    PrintWriter out = new PrintWriter(new File("data2.txt"));  
  
    String[] arr1 = {"frutas", "bebidas", "bebidas", "talho"};  
    String[] arr2 = {"banana", "sumol", "fanta", "costeleta"};  
    double[] arr3 = {1.2, 2, 6, 0.75};  
    double[] arr4 = {3.1, 1.45, 3.99, 5.99};  
  
    for (int item = 0; item < 4; item++) {  
        out.printf("%d %s %s %f %.2f\n", (item + 1), arr1[item], arr2[item], arr3[item], arr4[item]);  
    }  
    out.close();  
}
```

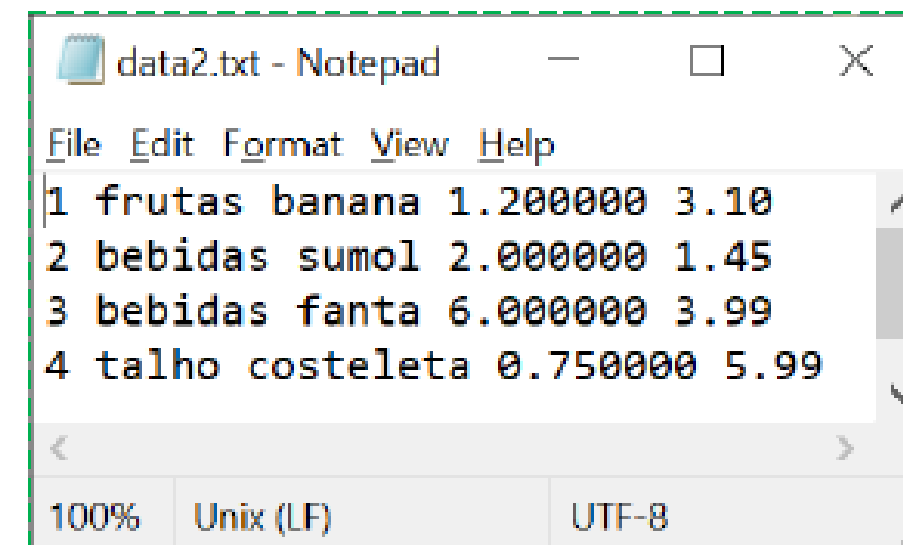


Ficheiros de Texto

Problema4.1: gravar no ficheiro “data2.txt” alguns items de compras.

Formatter


```
public static void escreverParaFicheiro() throws FileNotFoundException {  
    Formatter out = new Formatter(new File("data2.txt"));  
  
    String[] arr1 = {"frutas", "bebidas", "bebidas", "talho"};  
    String[] arr2 = {"banana", "sumol", "fanta", "costeleta"};  
    double[] arr3 = {1.2, 2, 6, 0.75};  
    double[] arr4 = {3.1, 1.45, 3.99, 5.99};  
  
    for (int item = 0; item < 4; item++) {  
        out.format("%d %s %s %f %.2f\n", (item + 1), arr1[item], arr2[item], arr3[item], arr4[item]);  
    }  
    out.close();  
}
```



Ficheiros de Texto

Notas importantes

- Se tentar aceder a um ficheiro de texto que não existe, é lançada a exceção *FileNotFoundException*
- Deve sempre fechar um ficheiro após a utilização (*close()*) para libertar recursos.
- O método *close()* invoca o método *flush()* para garantir que antes de libertar os recursos, sejam primeiro gravados no ficheiro quaisquer bytes em buffer.



Ficheiros de Texto

Casos de Uso

Array - Ordenação (Crescente)

Ideia: Cada posição do array guarda o menor dos elementos existentes à sua frente. Quando encontra um elemento menor à sua frente troca de posição com ele.

- posição 0 : compara-se com os elementos nas posições [1 , n]
- posição 1 : compara-se com os elementos nas posições [2 , n]
- posição 2 : compara-se com os elementos nas posições [3 , n]
- ...
- posição (n-1) : compara-se com o elemento na posição [n]

Array - Ordenação (Crescente)

Considere-se o seguinte array:

64	25	12	22	11
----	----	----	----	----

- posição 0: compara-se com os elementos nas posições [1 , n].
- Quando encontra um elemento menor à sua frente troca de posição com ele.

64	25	12	22	11
25	64	12	22	11
25	64	12	22	11
12	64	25	22	11
12	64	25	22	11
12	64	25	22	11
11	64	25	22	12

Array - Ordenação (Crescente)

- posição 1 : compara-se com os elementos nas posições [2 , n]
- Quando encontra um elemento menor à sua frente troca de posição com ele.

11	64	25	22	12
11	25	64	22	12
11	25	64	22	12
11	22	64	25	12
11	22	64	25	12
11	12	64	25	22

Array - Ordenação (Crescente)

- posição 2 : compara-se com os elementos nas posições [3 , n]
- Quando encontra um elemento menor à sua frente troca de posição com ele.

11	12	64	25	22
11	12	25	64	22
11	12	25	64	22
11	12	22	64	25

Array - Ordenação (Crescente)

- posição (n - 1) : compara-se com o elemento na posição [n]
- Quando encontra um elemento menor à sua frente troca de posição com ele.

11	12	22	64	25
11	12	22	25	64

Array - Ordenação (Crescente)

```
public static void ordenarArray(double[] arr) {
```

```
    for (int idx1 = 0; idx1 < arr.length - 1; idx1++) {
```

Percorre todas as posições,
exceto a última

```
        for (int idx2 = idx1 + 1; idx2 < arr.length; idx2++) {
```

```
            if (arr[idx2] < arr[idx1]) {
```

Percorre todas as posições
à frente da anterior

```
                double aux = arr[idx1];
```

```
                arr[idx1] = arr[idx2];
```

```
                arr[idx2] = aux;
```

Se elemento menor à frente

Troca de valores

Ficheiros de Texto

Problema5: Pretende-se ler dum ficheiro de texto (notasAlunos.csv) as notas de vários alunos a 5 disciplinas e visualizar as suas médias ordenadas de forma decrescente. O ficheiro contém em cada linha a seguinte informação :
Aluno,nota1,nota2,nota3,nota4,nota5.

Que estruturas de dados utilizar?

Nomes: Array unidimensional de Strings

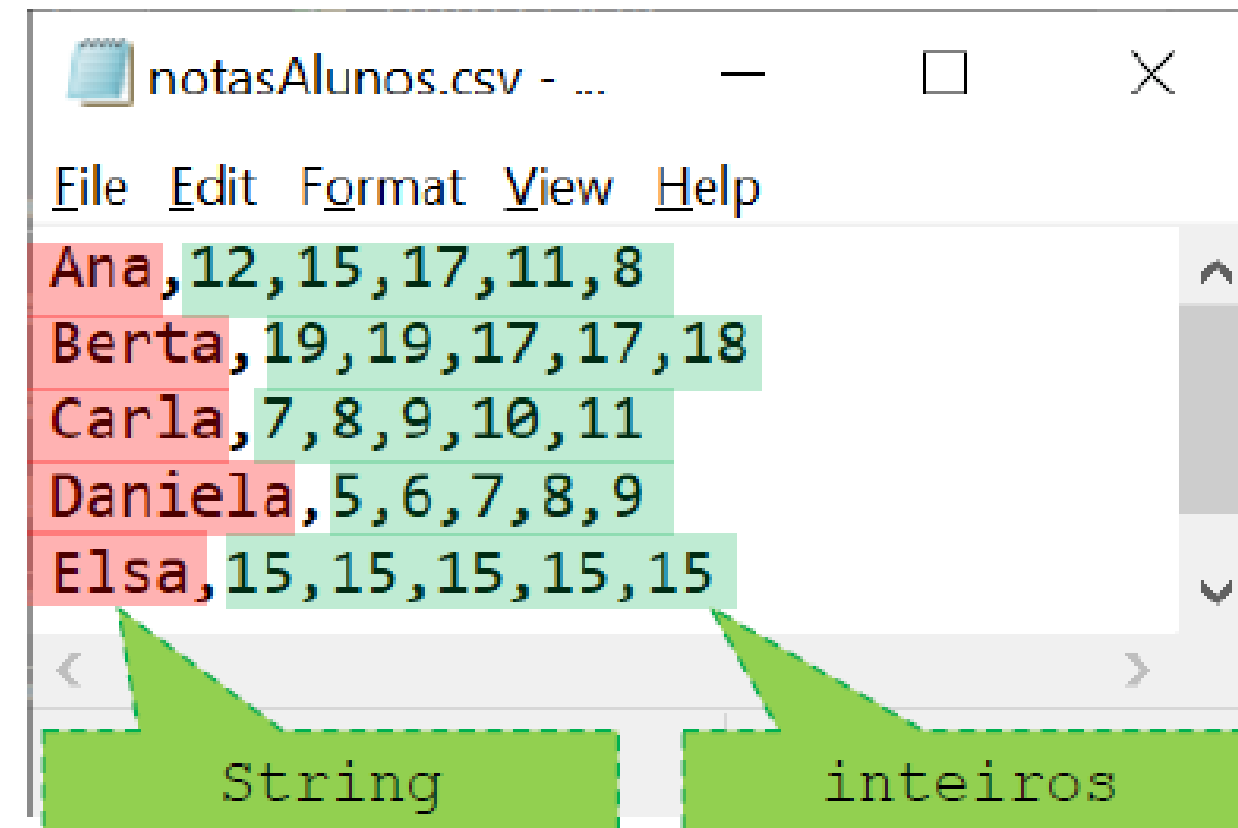
`String[] arrNomes;`

Notas: Array bidimensional de inteiros

`int[][] arrNotas;`

Médias: Array unidimensional de reais

`double[] arrMedias;`



arrNomes	Ana	arrNotas	12	15	17	11	8	arrMedias	
	Berta		19	19	17	17	18		
	Carla		7	8	9	10	11		
	Daniela		5	6	7	8	9		
	Elsa		15	15	15	15	15		

Ficheiros de Texto

Problema5: Pretende-se ler dum ficheiro de texto (notasAlunos.csv) as notas de vários alunos a 5 disciplinas e visualizar as suas médias ordenadas de forma decrescente. O ficheiro contém em cada linha a seguinte informação :
Aluno,nota1,nota2,nota3,nota4,nota5.

Que estrutura modular utilizar? Tentar separar responsabilidades.

- **lerDadosDoFicheiro()**
 - ler informação do ficheiro e preencher as estruturas de dados
 - Nomes dos alunos (arrAlunos)
 - Notas dos alunos (arrNotas)
- **calcularMedias()**
 - Preencher a estrutura de dados relativa à média (arrMédias)
- **ordenarInformacao ()**
 - Organizar as estruturas de dados de acordo com o critério pretendido
- **visualizarMedias()**
 - Visualizar o nome dos alunos (arrNomes) e respectivas médias (arrMedias)

Ficheiros de Texto

Problema5: Pretende-se ler dum ficheiro de texto (notasAlunos.csv) as notas de vários alunos a 5 disciplinas e visualizar as suas médias ordenadas de forma decrescente. O ficheiro contém em cada linha a seguinte informação :
Aluno,nota1,nota2,nota3,nota4,nota5.

```
public class Exercicio {  
    static final int MAX_ALUNOS=20;  
    static final int DISCIPLINAS=5;  
    static final String FILE_IN="notasAlunos.csv";
```

Não se conhece a dimensão!
Assume-se valores máximos

```
public static void main(String[] args) throws FileNotFoundException {
```

```
    String[] arrNomes=new String[MAX_ALUNOS];  
    int[][] arrNotas=new int[MAX_ALUNOS][DISCIPLINAS];  
    double[] arrMedias = new double[MAX_ALUNOS];
```

Podia definir-se a dimensão
mais tarde, após conhecer a
quantidade efetiva de alunos

```
...  
}  
}
```

arrNomes		arrNotas						arrMedias	

Quantos alunos existem?

Ficheiros de Texto

lerDadosDoFicheiro()

```
public class Exercicio {
    static final int MAX_ALUNOS=20;
    static final int DISCIPLINAS=5;
    static final String FILE_IN="notasAlunos.csv";

    public static void main(String[] args) throws FileNotFoundException {

        String[] arrNomes=new String[MAX_ALUNOS];
        int[][] arrNotas=new int[MAX_ALUNOS][DISCIPLINAS];
        double[] arrMedias = new double[MAX_ALUNOS];

        int totalDeAlunos = lerDadosDoFicheiro(arrNomes, arrNotas, FILE_IN);
        ...
    }
}
```

Total de linhas lidas do ficheiro = total de alunos.

arrNomes	Ana	arrNotas	12	15	17	11	8	arrMedias	
	Berta		19	19	17	17	18		
	Carla		7	8	9	10	11		
	Daniela		5	6	7	8	9		
= 5	Elsa		15	15	15	15	15		

Ficheiros de Texto

Exceção verificável

```
public static int lerDadosDoFicheiro(String[] arrNomes, int[][] arrNotas, String nomeFicheiro) throws  
FileNotFoundException {
```

```
    Scanner in = new Scanner(new File(nomeFicheiro));
```

Abrir ficheiro

```
    int qtdAlunos = 0;
```

```
    while (in.hasNextLine()) {
```

Verificar se existe linha no ficheiro

```
        String line = in.nextLine();
```

Extrair linha do ficheiro

```
        String[] itens = line.split(",");
```

Cortar pelas virgulas

```
        arrNomes[qtdAlunos] = itens[0];
```

Guardar nome

```
        for (int disciplina = 0, item = 1; item < itens.length; disciplina++, item++) {
```

```
            arrNotas[qtdAlunos][disciplina] = Integer.parseInt(itens[item]);
```

Guardar notas

```
        }
```

```
        qtdAlunos++;
```

```
    }
```

```
    in.close();
```

Fechar ficheiro

```
    return qtdAlunos;
```

```
}
```

Ficheiros de Texto

```
public static int lerDadosDoFicheiro(String[] arrNomes, int[][] arrNotas, String nomeFicheiro) throws  
FileNotFoundException {
```

```
    Scanner in = new Scanner(new File(nomeFicheiro));
```

```
    int qtdAlunos = 0;
```

```
    while (in.hasNextLine()) {
```

① String line = in.nextLine();

② String[] itens = line.split(",");

③ arrNomes[qtdAlunos] = itens[0];

④ for (int disciplina = 0, item = 1; item < itens.length; disciplina++, item++) {
 arrNotas[qtdAlunos][disciplina] = Integer.parseInt(itens[item]);

```
    }
```

```
    qtdAlunos++;
```

```
}
```

```
in.close();
```

```
return qtdAlunos;
```

```
}
```

line="Ana,12,15,17,11,8" ①

②

itens	"Ana"	0
	"12"	1
	"15"	2
	"17"	3
	"11"	4
	"8"	5

		0	1	2	3	4
arrNomes	Ana					

③

		0	1	2	3	4
arrNotas		12	15	17	11	8

④

Ficheiros de Texto

V1 - Método recebe o array das médias para preencher

```
public static void calcularMedias(int[][] arrNotas, double[] arrMedias, int totalAlunos) {  
  
    for (int aluno = 0; aluno < totalAlunos; aluno++) {  
  
        int soma = 0;  
        for (int disciplina = 0; disciplina < DISCIPLINAS; disciplina++) {  
            soma += arrNotas[aluno][disciplina];  
        }  
  
        arrMedias[aluno] = (double) soma / DISCIPLINAS;  
    }  
}
```

		0	1	2	3	4		
arrNomes	Ana	0	12	15	17	11	8	0
	Berta	1	19	19	17	17	18	1
	Carla	2	7	8	9	10	11	2
	Daniela	3	5	6	7	8	9	3
	Elsa	4	15	15	15	15	15	4
		5						5
		6						6

		0	1	2	3	4		
arrNotas		0	12	15	17	11	8	0
		1	19	19	17	17	18	1
		2	7	8	9	10	11	2
		3	5	6	7	8	9	3
		4	15	15	15	15	15	4
		5						5
		6						6

		0	1	2	3	4		
arrMedias	12.6	0						
	18	1						
	9	2						
	7	3						
	15	4						
		5						
		6						

Ficheiros de Texto

V2 - Método cria, preenche e retorna o array das médias

```
public static double[] calcularMedias(int[][] arrNotas, int totalAlunos) {  
    double[] arrMedias = new double[totalAlunos];  
  
    for (int aluno = 0; aluno < totalAlunos; aluno++) {  
        int soma = 0;  
        for (int disciplina = 0; disciplina < DISCIPLINAS; disciplina++) {  
            soma += arrNotas[aluno][disciplina];  
        }  
  
        arrMedias[aluno] = (double) soma / DISCIPLINAS;  
    }  
    return arrMedias;  
}
```

Cria array com dimensão efetiva de alunos

Retorna o array criado e preenchido

			0	1	2	3	4					
arrNomes	Ana	0	arrNotas	12	15	17	11	8	0	arrMedias	12.6	0
	Berta	1		19	19	17	17	18	1		18	1
	Carla	2		7	8	9	10	11	2		9	2
	Daniela	3		5	6	7	8	9	3		7	3
	Elsa	4		15	15	15	15	15	4		15	4
		5							5			
		6							6			

Ficheiros de Texto

```


public static void ordenarInformacao(String[] arrNomes, int[][] arrNotas, double[] arrMedias, int totalAlunos) {
    for (int idx1 = 0; idx1 < totalAlunos - 1; idx1++) {
        for (int idx2 = idx1 + 1; idx2 < totalAlunos; idx2++) {
            if (arrMedias[idx2] > arrMedias[idx1]) {
                //.....[troca media]
                double auxMedia = arrMedias[idx1];
                arrMedias[idx1] = arrMedias[idx2];
                arrMedias[idx2] = auxMedia;

                //.....[troca nome]
                String auxNome = arrNomes[idx1];
                arrNomes[idx1] = arrNomes[idx2];
                arrNomes[idx2] = auxNome;

                //.....[troca notas]
                for (int disciplina = 0; disciplina < DISCIPLINAS; disciplina++) {
                    int auxNota = arrNotas[idx1][disciplina];
                    arrNotas[idx1][disciplina] = arrNotas[idx2][disciplina];
                    arrNotas[idx2][disciplina] = auxNota;
                }
            }
        }
    }
}

```

troca todas as colunas da linha

		0	1	2	3	4		
	Ana	0	12	15	17	11	8	0
	Berta	1	19	19	17	17	18	1
	Carla	2	7	8	9	10	11	2
	Daniela	3	5	6	7	8	9	3
	Elsa	4	15	15	15	15	15	4
		5						5
		6						6
arrNomes		arrNotas					arrMedias	

troca todas as colunas
da linha

			0	1	2	3	4					
arrNomes	Berta	0	arrNotas	19	19	17	17	18	0	arrMedias	18	0
	Elsa	1		15	15	15	15	15	1		15	1
	Ana	2		12	15	17	11	8	2		12.6	2
	Carla	3		7	8	9	10	11	3		9	3
	Daniela	4		5	6	7	8	9	4		7	4
		5							5			5
		6							6			6



Ficheiros de Texto

```
public static void ordenarInformacao(String[] arrNomes, int[][] arrNotas, double[] arrMedias, int totalAlunos) {
```

```
for (int idx1 = 0; idx1 < totalAlunos - 1; idx1++) {
    for (int idx2 = idx1 + 1; idx2 < totalAlunos; idx2++) {
        if (arrMedias[idx2] > arrMedias[idx1]) {
```

```
//.....[troca media]
```

```
double auxMedia = arrMedias[idx1];
arrMedias[idx1] = arrMedias[idx2];
arrMedias[idx2] = auxMedia;
```


```
//.....[troca nome]
```

```
String auxNome = arrNomes[idx1];
arrNomes[idx1] = arrNomes[idx2];
arrNomes[idx2] = auxNome;
```

```
//.....[troca notas]
```

```
int[] auxNotas = arrNotas[idx1];
arrNotas[idx1] = arrNotas[idx2];
arrNotas[idx2] = auxNotas;
```

$$\left. \begin{array}{l} \{ \\ \{ \\ \{ \\ \{ \end{array} \right\}$$


		0	1	2	3	4		
	Ana	0	12	15	17	11	8	12.6
	Berta	1	19	19	17	17	18	18
	Carla	2	7	8	9	10	11	9
	Daniela	3	5	6	7	8	9	7
	Elsa	4	15	15	15	15	15	15
		5						
		6						
arrNomes		arrNotas					arrMedias	

troca linha inteira

			0	1	2	3	4					
arrNomes	Berta	0	arrNotas	19	19	17	17	18	0	arrMedias	18	0
	Elsa	1		15	15	15	15	15	1		15	1
	Ana	2		12	15	17	11	8	2		12.6	2
	Carla	3		7	8	9	10	11	3		9	3
	Daniela	4		5	6	7	8	9	4		7	4
		5							5			5
		6							6			6

Ficheiros de Texto

```
public static void visualizarMedias(String[] arrNomes, double[] arrMedias, int totalAlunos) {  
  
    for (int aluno = 0; aluno < totalAlunos; aluno++) {  
        System.out.printf("%-10s : %5.2f %n", arrNomes[aluno], arrMedias[aluno]);  
    }  
}
```

arrNomes	Berta	0	arrNotas	19	19	17	17	18	0	arrMedias	18	0	Berta	:	18.00
	Elsa	1		15	15	15	15	15	1		15	1		:	15.00
	Ana	2		12	15	17	11	8	2		12.6	2		:	12.60
	Carla	3		7	8	9	10	11	3		9	3		:	9.00
	Daniela	4		5	6	7	8	9	4		7	4		:	7.00
		5							5			5			
		6							6			6			

Ficheiros de Texto

Problema6: Adicionar à resolução do problema5 a funcionalidade de gravar as médias num ficheiro de texto (mediasDosAlunos.txt). Cada linha do ficheiro deve conter o nome de um aluno e respetiva média, separados por vírgula.

Que estruturas de dados utilizar?

arrNomes	Berta	0	arrNotas	19	19	17	17	18	0	arrMedias	18	0
	Elsa	1		15	15	15	15	15	1		15	1
	Ana	2		12	15	17	11	8	2		12.6	2
	Carla	3		7	8	9	10	11	3		9	3
	Daniela	4		5	6	7	8	9	4		7	4
		5							5			5
		6							6			6

Que estrutura modular utilizar?

- **gravarMedias()**
 - gravar o nome dos alunos (arrNomes) e respetivas médias (arrMedias)

Ficheiros de Texto

```
public static void gravarMedias(String[] arrNomes, double[] arrMedias, int totalAlunos)
    throws FileNotFoundException {
```

```
    PrintWriter out = new PrintWriter(new File("mediasDosAlunos.txt"));
```

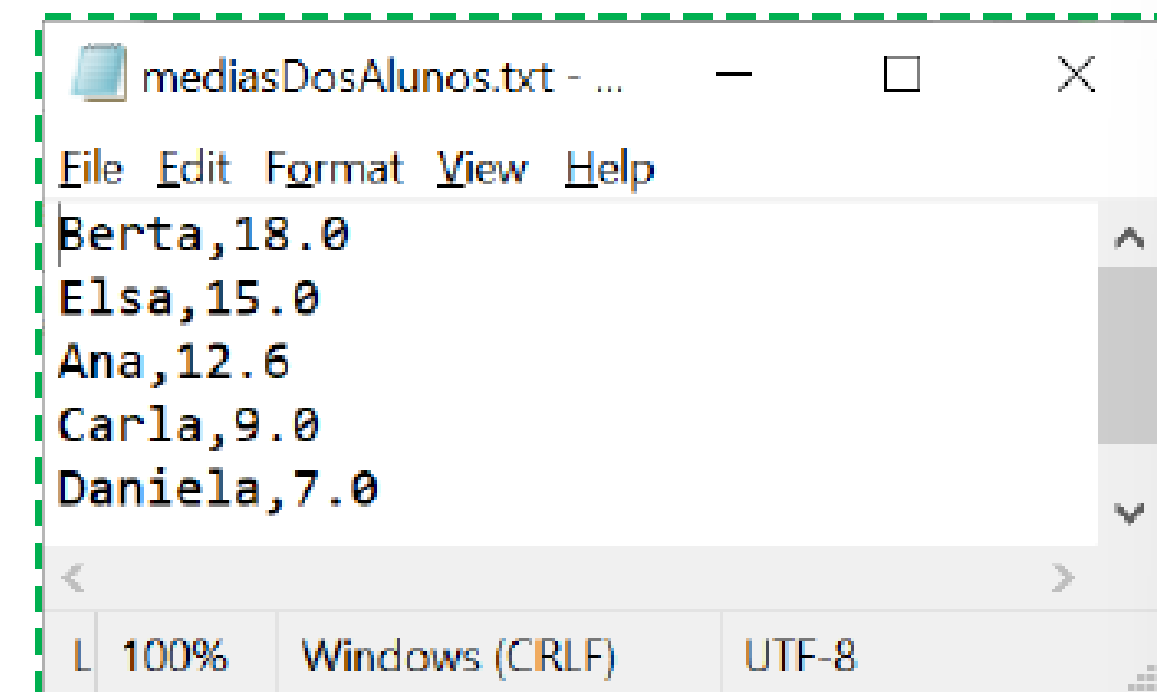
```
    for (int aluno = 0; aluno < totalAlunos; aluno++) {
        out.println(arrNomes[aluno] + "," + arrMedias[aluno]);
    }
```

```
    out.close();
```

```
}
```

Evitar literais.
Preferível usar constantes

arrNomes	Berta	0	arrNotas	19	19	17	17	18	0	arrMedias	18	0
	Elsa	1		15	15	15	15	15	1		15	1
	Ana	2		12	15	17	11	8	2		12.6	2
	Carla	3		7	8	9	10	11	3		9	3
	Daniela	4		5	6	7	8	9	4		7	4
		5							5			5
		6							6			6





#R4E

Software Developer

Algoritmia e Programação

Ficheiros

