Programação I
19-janeiro-2018

Exame final **sessão 2** **duração 2H:30m**

NMEC:_____ Nome: _____

Informações prévias:

- O ficheiro deve ter um nome do tipo EF12345.java, em que 12345 deve ser substituído pelo seu número mecanográfico. Coloque o seu nome e número mecanográfico no início do ficheiro como comentário.
- A prova é realizada sem consulta. Pode consultar a documentação de Java disponível no próprio computador através do comando view-javadoc "classe" (exemplo: view-javadoc Math)

Considere o ficheiro PaisesHumid.txt que se encontra no desktop. Este ficheiro tem uma lista de humidades de vários países com a seguinte estrutura: na primeira linha tem um valor inteiro com o número de humidades a ler; cada uma das restantes linhas começa com um número inteiro positivo que indica o dia, um segundo valor que indica a humidade média do dia, em % (entre 10 e 100), e o resto da linha inclui o nome do país, de acordo com o exemplo:

```
7
1 76 Portugal
156 23 Estados Unidos
1303 56 Alemanha
24 23 Marrocos
119 95 Costa Rica
45 89 Noruega
1 66 Portugal
```

Cada país é representado pela classe **Pais {int dia; int humidade; String nome}**.

Complete o programa base dado EF12345.java (que está no desktop) implementando cada uma das funções indicadas:

1) Função para ler os dados do ficheiro para um array do tipo Pais, devolvendo o array. Tem como argumento o nome do ficheiro. Não precisa de verificar se o ficheiro existe. Em alternativa fazer função para ler do teclado (50% da cotação); **(3 valores)**

   static Pais[] lerFichHum(String nomeF) throws IOException{}

2) Função, que tendo um array de países e uma frase como argumentos imprime todos os países que contenham a frase (procura no dia, humidade e nome – crie um string com os 3 campos). **(3 valores)**

   static void procurarListar(Pais [] p, String frase){}

Exemplo da procura de "**56**":	Exemplo da procura de "**or**":
1**56** 23 Estados Unidos 1303 **56** Alemanha	1 76 P**or**tugal 45 89 N**or**uega 1 66 P**or**tugal

3) Função para calcular a humidade mínima, devolvendo um array com os índices de todos os países que têm essa humidade. O argumento é o array com os países. Se devolver só um mínimo tem 50% da cotação). **(2 valores)**

   static int[] humMin(Pais [] p){}

4) Função para gravar num ficheiro de texto o array de países, de acordo com o formato indicado abaixo nos resultados. Os argumentos são o array de países e o nome do ficheiro; **(2 valores)**

   static void gravarPais (Pais [] p, String nome) throws IOException{}

5) Função para calcular e devolver um array com a frequência de humidades (número de vezes que uma determinada humidade aparece). Deve ignorar humidades fora do intervalo [10,100]. O argumento é o array de países. **(3 valores)**

   static int[] freqHum(Pais [] p){}

6) Função para imprimir o array da frequência de humidades, dado como argumento. Só imprime os valores das frequências > 0. **(1 valores)**

   static void printFreq(int[] f){}

7) Função para verificar se um dia e país já existe. Os argumentos são um array de países, o dia e país a verificar. Devolve **true** se existir e **false** se não existir. **(1 valores)**

   static boolean diaPaisExiste(Pais[] p, int comp, int dia, String pais) {}

8) Função para criar um novo array de países sem valores repetidos de dia e país. Devolve array com o número exato de países e dias não repetidos. Deve usar a função da alínea 7). **(2 valores)**

   static Pais[] removerRepetidos(Pais [] p) {}

9) Função para ordenar por ordem crescente do dia o array de países, pelo método bubble sort. **(3 valores)**

   static void bubbleSort(Pais a[]) {}

Sugestão: vá implementando e testando as várias funções (alíneas) passo a passo.
Soluções que não usem o programa, as funções e as estruturas de dados propostas podem ser penalizadas até 40%.
*A não formatação (indentação) adequada, ou a falta de comentários apropriados podem ser penalizadas **até 2 valores**.*
*Um programa que não compile pode ser penalizado **até 10 valores**, dependendo da gravidade dos erros.*
Funções úteis: String s1,s2; s1.**indexOf(s2)** *-> posição de s2 em s1. System.arraycopy(origem, inicioOrig, destino, inicioDest, tamanho)*

Resultados da execução do programa:

```
Dia   Hum Pais
    1  76 Portugal
  156  23 Estados Unidos
 1303  56 Alemanha
   24  23 Marrocos
  119  95 Costa Rica
   45  89 Noruega
    1  66 Portugal
Dia   Hum Pais
  156  23 Estados Unidos
 1303  56 Alemanha
Minimo =   156  23 Estados Unidos
Minimo =    24  23 Marrocos
Hum. Freq
  23    2
  56    1
  66    1
  76    1
  89    1
  95    1
true
```

PaisesHumidOrd.txt

```
6
    1  76 Portugal
   24  23 Marrocos
   45  89 Noruega
  119  95 Costa Rica
  156  23 Estados Unidos
 1303  56 Alemanha
```

```
import java.util.Scanner;
import java.io.*;

public class EF12345 {

    public static void main(String[] args) throws IOException{
        Pais[] paises;
        paises = lerFichHum("PaisesHumid.txt");
        procurarListar(paises," ");  // espaço lista tudo
        procurarListar(paises,"56"); // lista paises onde apareca "56"
        int [] mins = humMin(paises);
        for(int i=0; i<mins.length; i++)
            System.out.printf("Minimo = %5d %3d %s%n", paises[mins[i]].dia,
              paises[mins[i]].humidade, paises[mins[i]].nome);
        int[] freq = freqHum(paises);
        printFreq(freq);
        System.out.println(diaPaisExiste(paises,paises.length,24,"Marrocos"));
        paises=removerRepetidos(paises);
        bubbleSort(paises);
        gravarPais(paises, "PaisesHumidOrd.txt");
    }
// Implementar funções pedidas aqui
// …
}
class Pais {
    int dia;
    int humidade;
    String nome;
}
```