

## Grado en Ingeniería Informática y Grado en Estadística Fundamentos de Programación

Examen convocatoria extraordinaria. 25 de enero de 2019.

Apellidos \_\_\_\_\_

Nombre \_\_\_\_\_ Grupo \_\_\_\_\_

--	--	--	--	--	--	--

DNI y Firma

Duración del examen: 3,5 horas.

Empezar cada problema en una cara distinta.

Poner nombre y apellidos en todas las páginas.

Las hojas del enunciado también deben entregarse.

Se valorará la presentación y la claridad en la exposición.

Se valorará la adecuación de las estructuras utilizadas al problema a resolver.

No se calificarán las respuestas escritas a lápiz.

1. Un fotógrafo dispone de 20 tipos diferentes de fotografías, etiquetadas de 0 a 19, cada una de ellas con un precio diferente. En el vector **precio** almacena el valor de cada tipo de foto (en la posición 0 el valor de la foto 0, en la posición 1 el valor de la foto 1, y así sucesivamente).

Dispone de otro vector, **ventas**, que almacena registros de tipo **Rventa**. El tipo **Rventa** se puede suponer ya definido. Los registros **Rventa** contienen dos campos, uno con el nombre (**nombre**) del cliente y otro con el tipo de fotografía (**tipo**) que ha comprado.

El vector **ventas** esta ordenado por el nombre de los clientes, aunque puede haber más de un registro con el mismo nombre, ya que ese cliente puede haber realizado más de una compra (incluso del mismo tipo de foto).

Por ejemplo, una situación particular podría ser:

Vector **ventas**

0	1	2	3	4	5
nombre: Luis tipo: 1	nombre: Luis tipo: 1	nombre: Luis tipo: 3	nombre: Oscar tipo: 0	nombre: Oscar tipo: 2	nombre: Pedro tipo: 4

Vector **precio**

0	1	2	3	4	5	6	7	8	...	19
12,4	4	2,99	3,75	4,25	6	15	19,99	9,99		12


A partir de esta información, se desea generar una matriz, **fotos**, que almacene en cada casilla el número de fotos de un determinado tipo de un cliente particular. Las filas representan el tipo de foto. Las columnas representan el cliente aunque sin almacenar su nombre: en la posición 0 estaría la información del primer cliente del vector **ventas** (sea cual sea su nombre), en la posición 1 se guarda la información del siguiente cliente distinto al anterior del vector **ventas**, y así sucesivamente. De tal forma, que la posición  $[i][j]$  de la matriz almacena el número de fotos del tipo  $i$  que ha comprado el cliente  $j$ .

Por ejemplo, para los datos dados, la matriz deseada sería la siguiente:

	0	1	2
0	0	1	0
1	2	0	0
2	0	1	0
3	1	0	0
4	0	0	1
...	0	0	0
19	0	0	0

Matriz **fotos**

Se pide:


-  **[1 pto]** Elaborar un método Java que, a partir del vector **ventas** devuelva cuántos clientes diferentes han realizado una compra. Para los datos del ejemplo, el valor devuelto sería **3**.

```
public static int clientes(Rventa[] ventas) {
    int cont = 0;
    if (ventas.length > 0){
        String actual = ventas[0].nombre;
        cont = 1;
        for (int i = 1; i < ventas.length; i++)
            if (!actual.equals(ventas[i].nombre)) {
                actual = ventas[i].nombre;
                cont++;
            }
    }
    return cont;
}
```

- b) **[2 ptos]** Elaborar un método Java que, tomando como entrada el vector **ventas** y el vector **precio**, construya y devuelva la matriz **fotos**.

**NOTA:** Puede utilizarse, si se desea, el método del apartado anterior aunque no se haya implementado.

```
public static int[][] matrizFotos (Rventa[] ventas, double[] precio) {
    int fil = precio.length;
    int col = clientes(ventas);
    int[][] fotos = new int[fil][col];
    for (int i = 0; i < fil; i++)
        for (int j = 0; j < col; j++)
            fotos[i][j] = 0;
    if (ventas.length > 0){
        String actual = ventas[0].nombre;
        int columna = 0;
        fotos[ventas[0].tipo][columna]++;
        for (int i = 1; i < ventas.length; i++) {
            if (!actual.equals(ventas[i].cliente))
                columna++;
            actual = ventas[i].nombre;
        }
        fotos[ventas[i].tipo][columna]++;
    }
    return fotos;
}
```

-  **[1,5 ptos]** Elaborar un método Java que tome como entrada la matriz **fotos** y el vector **precio** y escriba en pantalla cuánto ha ganado el fotógrafo con todas sus ventas.

```
public static double ganancia (int[][] fotos, double[] precio) {
    double total = 0;
    for (int i = 0; i < fotos.length; i++) {
        for (int j = 0; j < fotos[0].length; j++)
            total = total + fotos[i][j]*precio[i];
    }
    return total;
}
```



**[2,5 ptos]** La codificación llamada "clave de César" consiste en desplazar las letras un número fijo de posiciones, empezando de nuevo cuando se llega al final. Por ejemplo, para el desplazamiento 3, la palabra **axb** se traduciría por **dae**.

En un determinado sistema se utiliza este tipo de codificación pero, para no tener que enviar dos datos, el valor del desplazamiento viene dado por el número de letras anteriores al único punto que hay en el mensaje (que no hay que considerar para la clave). Por ejemplo, si el mensaje recibido es **axb.cd**, el número para el desplazamiento es 3, y la clave codificada en el mensaje que ha recibido es **daefg**.

Escriba una función Java que obtenga la clave a partir de un mensaje codificado con estas características. Se trabaja sobre el alfabeto inglés.

**PRECONDICIÓN:** La cadena de entrada está formada exclusivamente por letras minúsculas del alfabeto inglés y un único punto.

```
public static char carDesplazado (char car, int desplaz){
    char nuevo;
    if ((car + desplaz) <= 'z')
        nuevo = (char)(car + desplaz);
    else
        nuevo = (char)('a' + (desplaz - ('z'-car)-1));
    return nuevo;
}

public static String clave(String mensaje) {
    int posPunto = mensaje.indexOf('.');
    String nueva = "";

    for (int i=0; i<posPunto; i++)
        nueva = nueva + carDesplazado(mensaje.charAt(i), posPunto);

    for (int i= (posPunto + 1); i<mensaje.length(); i++)
        nueva = nueva + carDesplazado(mensaje.charAt(i), posPunto);
    return nueva;
}
```



**3. [1,25 ptos]** Escribir una función Java **recursiva** que devuelva la suma de los **n** primeros números enteros tomando **n** como parámetro de entrada.

```
public static int suma(int n) {
    if (n == 0)
        return n;
    else
        return n + suma(n-1);
}
```



[1,75 ptos] Un fichero de texto **entrada.txt** contiene números enteros, a razón de uno por línea. Crear un programa Java que escriba en el fichero de texto **salida.txt** un mensaje que indique cual es la suma de los números pares que ocupan una línea par en el fichero de entrada. Supóngase que el fichero de entrada existe y que están importados los paquetes necesarios.

**RECORDATORIO:** Para declarar y abrir en modo escritura un fichero de texto:

```
PrintWriter <id_fich> = new PrintWriter (new FileWriter (<nombre_fich>));
```

Al abrir el fichero se puede producir la excepción **IOException**.

Para declarar y abrir en modo lectura un fichero de texto:

```
Scanner <id_fich> = new Scanner (new File(<nombre_fich>));
```

Al abrir el fichero se puede producir la excepción **FileNotFoundException**.

```
public class Ficheros {  
    public static void main(String[] args) {  
        Scanner datos;  
        PrintWriter result;  
  
        try{  
            datos = new Scanner (new File ("entrada.txt"));  
        }  
        catch (FileNotFoundException e){  
            System.out.println("No se encontró el fichero de entrada");  
            return;  
        }  
        try{  
            result = new PrintWriter (new FileWriter ("salida.txt"));  
        }  
        catch (IOException e) {  
            System.out.println("No se puede abrir fichero de salida");  
            return;  
        }  
  
        int suma = 0;  
        int linea = 0;  
        int num;  
        while (datos.hasNextInt()) {  
            linea ++;  
            num = datos.nextInt();  
            if (n%2 == 0 && linea%2 == 0)  
                suma = suma + num;  
        }  
        result.println("La suma es " + suma);  
        datos.close();  
        result.close();  
    }  
}
```