Introducción al Software, Curso 2017-2018

Problema 8: Practicar con arrays, ArrayList, recursión y tipos enumerados

Datos personales	
Apellidos:	
Nombre:	

1 Lectura de datos de empleados

Objetivos

Practicar con la lectura de datos, los tipos enumerados y ArrayList

Descripción

Se dispone de la siguiente clase que contiene los datos de un empleado:

```
public class Empleado {
   private String nombre;
   private TipoEmpleado tipo;
   public Empleado(String nombre, TipoEmpleado tipo) {
      this.nombre=nombre;
      this.tipo=tipo;
   }
   public String toString() {
      return tipo+“: ”+nombre;
   }
}
```

El tipo de empleado se define en la clase TipoEmpleado, que contiene los valores Administrativo, Comercial, Operario, Ingeniero, Directivo.

Se desea añadir a la clase Empleado un método para leer una lista de empleados usando la clase CajaTexto del paquete fundamentos. En la caja de texto se escribirán los empleados uno por línea usando el formato del siguiente ejemplo. El método debe retornar un ArrayList de objetos de la clase Empleado.

```
Antonio González, Adminstrativo
Jesús Pozo, Comercial
Laura Pérez, Directivo
```

Recordar que es posible convertir un String a valor enumerado usando el método estático valueOf(String s)

Respuesta:

<poner aquí la definición de la clase TipoEmpleado>

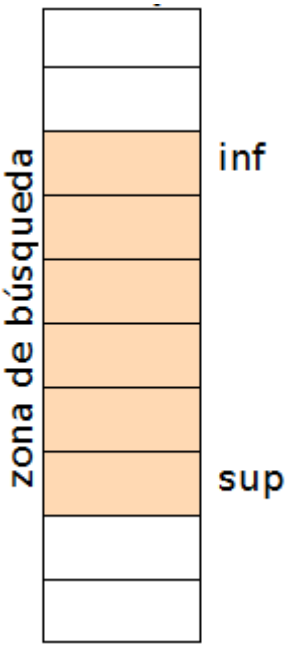
<poner aquí el código del método que lee la lista de empleados>

2 Búsqueda binaria

Objetivos

Practicar con ArrayList y recursión

Descripción



Escribir un método recursivo llamado busquedaBinaria que busca el empleado de nombre nom dentro de una parte de un ArrayList de objetos de la clase Empleado que se pasa como parámetro y cuyos valores están ordenados por orden alfabético de nombres. La zona de búsqueda está delimitada por los índices inf y sup, ambos incluidos. Si se encuentra el valor en la zona de búsqueda, se devuelve el índice de la casilla en que se encuentra. Si no se encuentra, se devuelve -1. El método tiene por tanto como parámetros el ArrayList, el nombre buscado nom, y los límites de la zona de búsqueda: inf y sup.

En el algoritmo de búsqueda binaria se divide la zona de búsqueda en dos mitades. En la versión recursiva, el caso directo se da cuando inf y sup son iguales, y en ese caso se retorna inf si el nombre del empleado de esa casilla del ArrayList vale nom, y -1 en caso contrario. En el caso recursivo se calcula el punto medio med=(inf+sup)/2 y si nom es mayor que el nombre del empleado de la casilla med del ArrayList se retorna el valor retornado por la búsqueda binaria para el mismo ArrayList y valor nom pero entre los límites med+1 y sup, y en caso contrario se retorna el valor retornado por la búsqueda binaria para el mismo ArrayList y valor nom pero entre los límites inf y med.

Se supone que los índices inf y sup son correctos, inf<=sup, y el ArrayList es correcto estando siempre ordenado.

Respuesta:

<poner aquí el código del método busquedaBinaria>

3 Retenciones del IRPF

Objetivos

Practicar con arrays y el pseudocódigo

Descripción

TABLA TRAMOS IRPF§	§	§
de§	a§	Retención§
0,00 €	12.450,00 €	19,00%
12.450,00 €	20.200,00 €	24,00%
20.200,00 €	35.200,00 €	30,00%
35.200,00 €	60.000,00 €	37,00%
60.000,00 €		45,00%

La tabla siguiente muestra la tabla de retenciones del IRPF para 2017. Una persona que gane entre 0 y 12.450€ de salario bruto tendrá una retención del 19%

de su salario. Si gana más, por ejemplo 13.450€ se le retendrá el 19% por los primeros 12.450€ y un 24% por los siguientes 1.000 euros. Y así sucesivamente. Escribir un método Java que, dado el salario bruto retorne la retención a aplicar. El método seguirá este pseudocódigo, que está casi completo:

```
método retencionIRPF (real salarioBruto) retorna real
   tablaTramo= array con los datos de la columna izquierda de la tabla
   tablaRetencion= array con datos de la col. dcha. de la tabla divididos entre 100
   real retencion=0
   para i desde 0 hasta tamaño de tablaTramo-2 hacer
      si salarioBruto>tablaTramo[i+1] entonces
         // la retención es sobre el tramo completo y hay que seguir en el bucle
         retencion=retencion+(tablaTramo[i+1]-tablaTramo[i])*tablaRetencion[i]
      si no
         // la retención es sobre un tramo final parcial y hay que salir del bucle
         retencion=retencion+(salarioBruto-tablaTramo[i])*tablaRetencion[i]
         retorna retencion
      fin si
   fin para
   // falta calcular el último tramo del IRPF, el del 45%
   retencion=retencion+...
   retorna retencion
fin método
```

En este pseudocódigo falta la penúltima instrucción, en la que se añade el último tramo de la retención (el que lleva el 45%) a la variable retencion. Pensar cómo se hace este último cálculo y ponerlo en el código.

Respuesta:

<poner aquí el código del método retencionIRPF>