

Programaci3n II - Grupo 1M: Ejercicio

1. Descripci3n del Problema

El objetivo de este ejercicio es la implementaci3n de varios elementos necesarios para realizar diferentes estudios sociol3gicos sobre una comunidad aut3noma.

El archivo de datos tipo es `2022cam.csv`

2. La Abstracci3n de Datos Localidad

Empezaremos implementando una clase de nombre `Localidad` definida mediante la siguiente especificaci3n:

```
class Localidad
{
    È /* Nombre de la localidad */
    È private String nombre;

    È /* Poblaci3n */
    È private int poblacion;

    È /** Constructor de Localidad */
    È Localidad (String nombre, int poblacion)

    È /** Constructor de Localidad a partir de una l3nea de texto */
    È Localidad (String linea)

    È /** Localidad como texto */
    È public String toString ()

    È /** Nombre de esta localidad */
    È String nombre ()

    È /** Poblaci3n total de esta localidad */
    È int poblacion ()

    È /** Comparador de Localidad por igualdad */
    È public boolean equals (Object o)
}
```

3. La Abstracci3n de Datos Comunidad

Esta *Abstracci3n de Datos* representa una Comunidad Aut3noma, definida como una colecci3n de `Localidad(es)`.

Su especificaci3n es:

```
public class Comunidad
{
    È /* Colecci3n de localidades */
    È private Localidad[] ldlloc;
```

```

Ê /* Número de localidades guardadas en la colección */
Ê private int size;

Ê /** Constructor de Comunidad */
Ê public Comunidad ()

Ê /** "¿loc está en esta Comunidad" */
Ê private boolean esta (Localidad loc)

Ê /** Comunidad como texto */
Ê public String toString ()

Ê /** Añade loc a esta Comunidad poniéndola al final */
Ê public void add (Localidad loc)

Ê /**
Ê * POST: resultado: listado de los nombres de Localidad(es)
Ê *           que están en esta Comunidad, cada nombre en una línea.
Ê */
Ê public String nombresLocalidades ()

Ê /*
Ê * POST: Devuelve la posición que ocupa la Localidad con este nombre.
Ê *           Si no se encuentra, devuelve -1.
Ê */
Ê public int posicion (String nombre)

Ê /*
Ê * POST: Devuelve la población de la Localidad con este nombre.
Ê *           Si no se encuentra, devuelve -1.
Ê */
Ê public int poblacion (String nombre)

Ê /*
Ê * POST: resultado: población total del grupo de localidades
Ê *           cuyos nombres vienen en llocs.
Ê */
Ê public int poblacionConjunta (String[] llocs)

}

```

4. Formato de los datos del fichero

El fichero de texto de nombre **2022cam.csv** tiene los datos de población de los municipios de la comunidad autónoma de Madrid.

Cada línea del fichero tiene el formato:

```
Sexo; Provincia; Municipio; Fecha; Total
```

Cada campo representa lo siguiente:

¥ **Provincia**: nombre de la provincia

¥ **Municipio**: nombre del municipio

¥ **Fecha**: fecha de los datos

¥ **Sexo y Total**: El campo **Sexo** es un string que puede tener los valores:

! "Total": indica que el campo **Total** tiene la población total del municipio.

! "Hombres": indica que el campo **Total** tiene el total de hombres total del municipio.

! "Mujeres": indica que el campo **Total** tiene el total de mujeres total del municipio.

Así pues, cada municipio aparece en tres líneas diferentes del fichero. La primera línea tiene una descripción de los campos.

```
Sexo; Provincia; Municipio; Fecha; Total
...
Total; Madrid; Alcobendas; 1 de enero de 2022; 117041
Total; Madrid; Alcorcon; 1 de enero de 2022; 170296
...
Hombres; Madrid; Alcobendas; 1 de enero de 2022; 55907
Hombres; Madrid; Alcorcon; 1 de enero de 2022; 81673
...
Mujeres; Madrid; Alcobendas; 1 de enero de 2022; 61134
Mujeres; Madrid; Alcorcon; 1 de enero de 2022; 88623
...
```

Se trata de escribir un método que cree un objeto de la clase Comunidad a partir de un fichero como el descrito.