OBJETIVOS

- Implementación de constructores a partir de `String`.
- Implementación de tipos inmutables.

CONSTRUCTOR A PARTIR DE STRING

Vamos a añadir a nuestros tipos un nuevo constructor que permita construir un objeto a partir de una representación del mismo en forma de cadena de caracteres.

Para cada tipo se proporciona el formato de la cadena a partir de la cual debe construirse el objeto. Para hacerlo, añada a la clase que implementa el tipo un constructor que reciba como parámetro un `String` con el formato que se indica y que asigne un valor a cada uno de los atributos del tipo. Para aquellos atributos que no estén presentes en la cadena, asígneles un valor por defecto tal como hizo en el constructor con parámetros.

- **Alumno**: "`12345678Z,Juan,López García,20/1/1998,juan@alum.us.es`". Para la fecha también se admite la forma 05/02/1998, completando con ceros los campos día y mes.
- **Asignatura**: "`Fundamentos de Programación#1234567#12.0#ANUAL#1`". En este caso se utiliza el carácter '#' como separador en lugar de la coma, ya que ésta puede aparecer en el nombre de la asignatura.
- **Beca**: "`ABC1234,6000.0,12,ORDINARIA`"
- **Despacho**: "`F1.43,1,3`"
- **Espacio**: "`A0.10,0,100,TEORIA`"
- **Nota**: "`Fundamentos de Programación#1234567#12.0#ANUAL#1;2014;PRIMERA;10.0;true`"
- **Persona**: "`12345678Z,Juan,López García,20/01/1998,juan@acmemail.com`"
- **Tutoria**: "`L,15:30,17:30`"

Tenga en cuenta que en la representación como cadena pueden aparecer espacios en blanco entre los separadores y los valores de las propiedades, que deberá eliminar antes de almacenar el valor en el atributo correspondiente. Por ejemplo, en el caso de Alumno, una representación válida sería:

```
"12345678Z, Juan, López García, 20/01/1998, juan@alum.us.es"
```

En este caso, debe asegurarse de que al extraer el nombre, por ejemplo, elimina el espacio en blanco inicial de la cadena "` Juan`" para quedarse con el valor "`Juan`", que es el que debe almacenarse en el atributo.

TIPOS INMUTABLES

Defina el nuevo tipo **BecaInmutable** con las mismas propiedades que el tipo Beca, pero de forma que sea inmutable. Para ello, cree una nueva interfaz `BecaInmutable` y una clase `BecaInmutableImpl` que implemente dicha interfaz.

El tipo **Nota** es en parte inmutable, ya que no dispone de métodos modificadores. Sin embargo, la implementación que se ha hecho de este tipo en la clase `NotaImpl` no asegura del todo la

inmutabilidad del tipo. Aplique sus nuevos conocimientos sobre tipos inmutables para crear una nueva clase `NotaInmutableImpl` que implemente la interfaz `Nota` haciendo que el tipo sea inmutable.

Puede copiar el código de las clases `BecaImpl` y `NotaImpl` y luego realizar las modificaciones adecuadas en ellas para que los nuevos tipos sean inmutables. **Antes de hacerlo, asegúrese de que las clases están bien construidas y pasan todos los test**.

TEST

Añada casos de prueba para los constructores a partir de `String` en las clases de test de los respectivos tipos. Para ello, utilice el método `leeFichero` que se le proporciona en la clase de utilidad `Grados`, que debe copiar en el paquete `fp.grados.utiles`. Este método lee un fichero de texto formado por líneas que contienen cada una la representación como cadena de un objeto, y construye a partir de él una lista con dichos objetos.

Por ejemplo, dado el fichero `personas.txt`, situado en una carpeta de nombre `res` dentro del proyecto, y con el siguiente contenido,

```
12345678Z,Juan,López García,20/07/1998,juan@acmemail.com
12345678Z,Antonio,López López,03/11/1997,antonio@acmemail.com
12345678Z,Sonia Estefanía,Amor Gena,10/12/1988,sonia@acmemail.com
12345678Z,María,Lora Santa,26/07/2002,maria@acmemail.com
```

la llamada al método[1]

```
List<Persona> personas = Grados.leeFichero("res/personas.txt",
      s->new PersonaImpl(s));
```

crea la lista `personas` con el siguiente contenido:

```
[12345678Z - López García, Juan - 20/07/1998, 12345678Z - López López, Antonio -
03/11/1997, 12345678Z - Amor Gena, Sonia Estefanía - 10/12/1988, 12345678Z - Lora
Santa, María - 26/07/2002]
```

Realice este mismo test con el resto de tipos, utilizando los ficheros de texto que se le proporcionan para cada uno de ellos.

En cuanto a los tipos inmutables, para probar cada uno de ellos realice un test en el que construya objetos del tipo y posteriormente intente modificarlos. Compruebe que los objetos se comportan de la forma esperada.

[1] No se preocupe si no entiende ahora la sintaxis empleada en la llamada. Esto se explicará más adelante.