Proyecto mdBuses (listas, interfaces, IO)

Se va a crear una aplicación para controlar los autobuses de línea que hay en una ciudad. Para ello se crearán las clases `Bus`, `Servicio`, `EnMatricula`, `PorLinea` y la interfaz `Criterio`, todas en el paquete `buses`.

- Mientras no se indique lo contrario, las variables de instancia serán privadas y los métodos públicos.
- Pueden añadirse los métodos privados que se consideren.

Primera parte

Clase `Bus`

- Crea la clase `Bus` que mantiene información de un autobús de línea del cual se conocen el código del autobús (`int codBus`), la matrícula (`String matricula`) y la línea a la que pertenece (`int codLinea`). La clase tendrá un constructor cuyos argumentos son el código del autobús y la matrícula.
- Crea el método `void setCodLinea(int codLinea)` que asigna el código de la línea a la que pertenece el autobús. Crea los métodos `int getCodBus()`, `String getMatricula()` y `int getCodLinea()` que devuelven el código del autobús, la matrícula y el código de la línea a la que pertenece respectivamente .
- Dos autobuses serán considerados iguales si coinciden su código de autobús y su matrícula. La letra de la matricula podrá estar indistintamente en mayúsculas o minúsculas.
- La representación de un autobús (método `toString()`) debe mostrar las tres variables de instancia. Por ejemplo:

```
Bus(675,2959CDN,25)
```

Clase `Servicio`

- Crea la clase `Servicio` que colecciona los autobuses que hay en una ciudad. Esta clase contiene un `String` que indica el nombre de la ciudad y una colección (una lista llamada `buses`) de autobuses. Define un constructor que tiene como argumento el nombre de la ciudad del servicio. Este constructor debe crear la lista para almacenar los autobuses.
- Define el método `String getCiudad()` que devuelve el nombre de la ciudad y otro `List<Bus> getBuses()` que devuelve la lista de buses.
- Para incluir autobuses en la lista define el método público `void leeBuses(String file) throws IOException,` que lee los datos de los autobuses del fichero `file`. El fichero de entrada contiene los datos de cada autobús en una línea del fichero con el siguiente formato (ver el fichero `buses.txt`):

```
<codBus>,<matricula>,<codLinea>
```

  Cuando en la lectura de los datos de un autobús se produzca algún tipo de error, bien porque falten datos o porque alguno sea incorrecto, se deberá mostrar en consola el error y se sigue con el resto del fichero de entrada.

Clase `MainPrueba`

- Crea una aplicación `MainPrueba` en el paquete por defecto que cree un servicio para la ciudad de Málaga con los autobuses del fichero `buses.txt`. Esta aplicación mostrará por consola la ciudad y todos los autobuses, uno a uno.

Segunda parte

Interfaz `Criterio`

Se desea obtener una selección de autobuses atendiendo a diferentes criterios de selección. Así, nos podrán interesar los autobuses que pertenecen a una línea determinada, es decir, que tienen un código de línea dado, los que contienen una cadena de caracteres en su matrícula, etc.

- Define la interfaz `Criterio` que contenga el método `boolean esSeleccionable(Bus bus)` que indica si el autobús pasado como argumento se debe seleccionar según este criterio.

Clase `EnMatricula` (quizás te la puedas ahorrar)

- Crea la clase `EnMatricula` que implementa un `Criterio` y mantiene una cadena de caracteres que se le pasa en el constructor.
- Implementa el método de selección de manera que el autobús argumento será seleccionable si su matrícula contiene la cadena dada en el constructor.

Clase `PorLinea` (quizás te la puedas ahorrar)

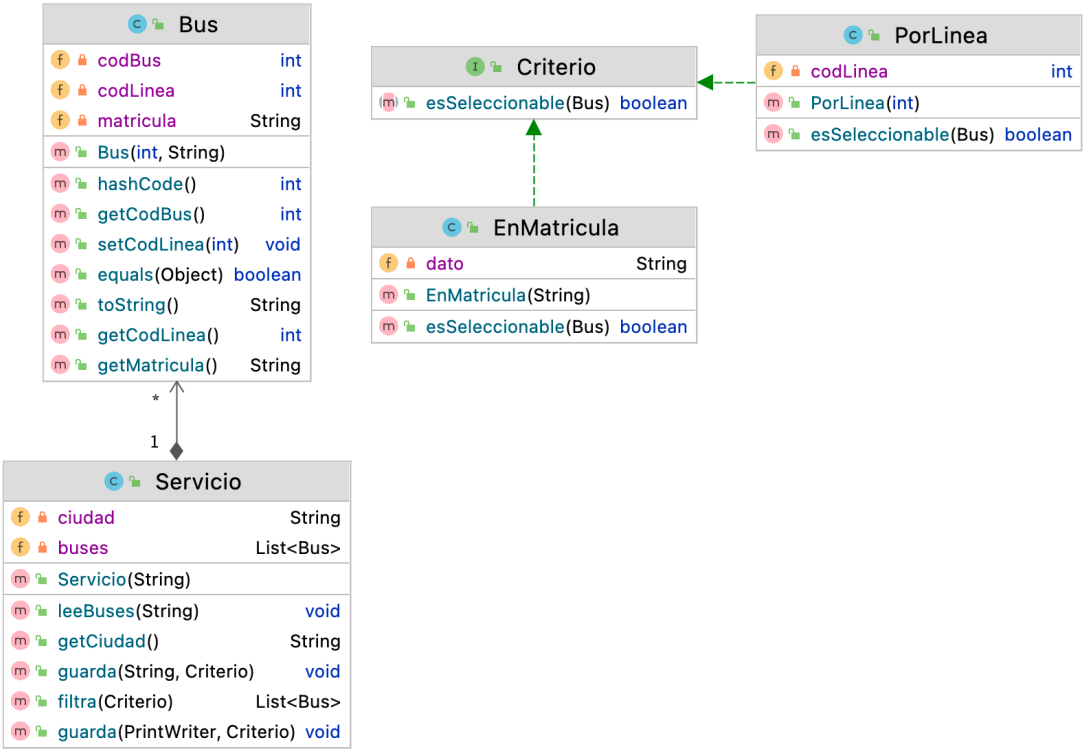
- Crea la clase `PorLinea` que implementa un `Criterio` y mantiene un código de línea que se le pasa en el constructor.
- Implementa el método de selección de manera que el autobús argumento será seleccionable si su código de línea coincide con el dado en el constructor.

Clase `Servicio`

Continuamos con la clase `Servicio` para añadir nuevos métodos.

- Crea el método `List<Bus> filtra(Criterio criterio)` que devuelve una lista con aquellos autobuses del servicio que cumplen el criterio que se pasa como argumento.
- Se pretende guardar en un fichero los autobuses que seleccionemos con el método anterior. Para ello se van a crear dos métodos. El primero `void guarda(String file, Criterio criterio) throws FileNotFoundException`, guarda en el fichero `file` los autobuses que cumplan el criterio dado como segundo argumento. Para ellos se ayudará de otro método `void guarda(PrintWriter pw, Criterio criterio)` que guarda los autobuses seleccionados por el criterio en el dispositivo de salida proporcionado como primer argumento.

A continuación, se muestra un diagrama de clases del ejercicio completo.



Se proporciona la clase `MainBuses` que hace una prueba de todo lo anterior. La salida del programa `MainBuses` debe ser la mostrada abajo. Además, debe crear dos ficheros, el primero, `linea21.txt`, con los autobuses de la línea 21, el segundo, `contiene29.txt`, con los buses que contiene el 29 en la matrícula:

```
Malaga
Error, faltan datos en 636,1338RFE
Error en dato numérico en 714,4956XFU,7l
Autobuses de la linea 21
Bus(653,1540HIH,21)
Bus(652,7658PDM,21)
Bus(590,3774ARP,21)
Bus(654,3513FZV,21)
Bus(523,8297WHJ,21)
Bus(634,5009KVE,21)
Bus(575,9452NKW,21)
Bus(645,6675NFF,21)
Autobuses cuya matricula contiene 29
Bus(463,2934IAC,3)
Bus(619,2298VGL,10)
Bus(656,2965JGB,20)
Bus(523,8297WHJ,21)
Bus(477,2906HCE,22)
Bus(675,2959CDN,25)
Bus(593,3729MMI,32)
Bus(713,0293PVU,71)
```