Método que muestra el contenido del catálogo

Este método regresa de manera compacta el contenido del catálogo. Esto es, para cada una de las referencias distintas de `null`, cuyo total está registrado en `numDiscos`, muestra el contenido general de cada disco junto con la posición que ocupa, para permitir al usuario elegir un disco por la posición que ocupa en el catálogo.

En la tarjeta de responsabilidades tienes la siguiente descripción:

Nombre	Salida	Entradas	Descripción
mstraCatalogo	Cadena con lista de discos	encabezado	Muestra la lista de discos dados de alta

Puedes ya poner la documentación de Javadoc y el encabezado del método:

```
278   /**
279    * Regresa una cadena con el catalogo de discos. Cada disco
280    * muestra su contenido en forma compacta.
281    * @param encabezado El titulo del listado.
282    * @return La cadena con un disco por renglon y la posicion que ocupa.
283    */
284   public String mstraCatalogo(String encabezado) {
```

Antes que nada debes verificar que el catálogo existe, para que el programa no aborte por tratar de usar una referencia nula:

```
285     if (catalogo == null)
286       return encabezado + "\nEl␣catalogo␣no␣existe";
```

También puede suceder que el catálogo (y todos los arreglos asociados) existan, pero que no haya ningún disco registrado. Eso lo sabes si el contador `numDiscos` está en 0, en cuyo caso también sales del método con un mensaje de error.

```
287     if (numDiscos <= 0)
288       return encabezado + "\nNo␣hay␣discos␣en␣el␣catalogo";
```

Una vez iniciada la cadena con el encabezado en el parámetro, el que verificas para no usar una referencia nula, vas a recorrer el catálogo listando tantos discos como te indique el contador `numDiscos`, recordando que las posiciones en los arreglos empiezan en cero, por la que `numDiscos` apunta a la primera posición vacía, donde ya no hay discos. Aunque has tenido cuidado para que no haya referencias nulas mezcladas con las referencias válidas a discos, verificas, antes de usar la referencia, que no sea nula. Si lo es, reportas que la referencia es nula y si no lo es, reportas el contenido del disco en esa posición. Si la referencia es nula usas el enunciado `continue` para regresar al bloque de actualización del `for` y seguir con la siguiente posición en el catálogo.

```
289     String cadena = encabezado == null ? "" : encabezado + "\n";
290     for (int i=0; i < numDiscos; i++) {
291       cadena += "[" + i + "]␣"; // posicion que ocupa el disco
292       if ( catalogo[i] == null) {
293         cadena += "no␣hay␣disco␣en␣esta␣posicion\n";
294         continue;
295       }
```

Si la referencia al disco es válida, sigues adelante en el bloque del `for` para el i–ésimo disco. Vas a dar una descripción compacta del disco, indicando el lugar que ocupa en el catálogo.

```
296       cadena += "(" + catalogo[i].getTIPO_DISCO() + ")␣"
297         + catalogo[i].getNOMBRE() + "\n"
298         + "\t␣Num.␣de␣transmisiones␣permitidas:␣"
299         + catalogo[i].getPermitidas() + "\n"
300         + "\t␣Num.␣de␣transmisiones␣activas:␣"
301         + catalogo[i].getActivas() + "\n"
302         + "\n";
303     } // for
304     return cadena;
305   } // fin de mstraCatalogo
```

Una modificación que podrías hacer es que en lugar de que te dé el tipo de disco como un entero, te diga si se trata de un CD, un DVD o un Bluray. Usando una condicional aritmética o un método auxiliar, se ve fácil de implementar.

Método que lista aquellos discos que tienen transmisiones activas

En la tarjeta de responsabilidades tienes el siguiente registro para este método:

Nombre	Salida	Entradas	Descripción
mstraActivos	Cadena con la lista de discos en transmisión	encabezado	Da una lista de los discos en transmisión junto con la lista de hora de inicio

Puedes ya dar la documentación y el encabezado del método:

```
307   /**
308    * Muestra en una lista aquellos discos que tienen
309    * transmisiones activas. Identifica al disco por
310    * la posicion que ocupa y, para cada disco, identifica
311    * a las transmisiones activas por la columna que ocupan
312    * en el renglon correspondiente al disco en el arreglo
313    * fechas.
314    * @param encabezado Para que el listado tenga un
315    *                   titulo.
316    * @return una lista de los discos que tienen transmisiones
317    *         activas con las transmisiones identificadas por
318    *         la posicion que ocupa cada una.
319    */
320   public String mstraActivos(String encabezado) {
```

Nuevamente verificas que la referencia al catálogo no sea nula y si es así, que el catálogo no esté vacío:

```
321     if (catalogo == null || numDiscos <= 0)
322       return encabezado + "\nNo␣hay␣discos␣en␣el␣catalogo";
```

Si alguna de estas dos situaciones se presenta se sale del método con un mensaje adecuado. Si no sales del método, recorres el catálogo procesando aquellos discos que tengan una o más transmisiones activas. Inicias la lista con el encabezado dado, verificando que no sea una referencia nula:

```
323     String cadena = encabezado == null ? "" : encabezado + "\n";
```

Recorres los discos, obteniendo en cada uno de ellos el número de transmisiones activas:

```
324     for( int i = 0; i < numDiscos; i++)   {
325       if (catalogo[i] == null) continue;
326       int numActivas = catalogo[i].getActivas();
```

Si no tiene transiciones activas saltas ese disco y sigues con el siguiente:

```
327       if ( numActivas <= 0 ) continue;
```

Para cada disco en la posición i, como su atributo `activas` es mayor que cero, recorres el arreglo `fechas` hasta la posición marcada por el atributo `activas`. Para eso usas otro enunciado `for`, anidado en el primero, usando, por supuesto, otra variable para el `for` anidado, mostrando el lugar ocupado.

```
328       cadena += "catalogo[" + i + "]:␣" + catalogo[i].getNOMBRE()
329           + "\n";
330       for (int j = 0; j < numActivas; j++) {
331         // la j da la posicion de fecha en el disco posicion i
332         cadena += "[" + j + "]␣" + daCalndrio(fechas[i][j]) + "\n";
333       } // fin de for j
```

También en este caso estás suponiendo que las fechas se encuentran en posiciones consecutivas. Usas un método auxiliar, `daCalndrio`, que implementarás una vez termines este método, para obtener una cadena que nos dé la hora.

Una vez agregadas a la cadena todas las fechas de transmisión precedidas por el lugar que ocupan, agregas una cadena de fin de línea y regresas la cadena construida.

```
334       cadena += "\n";
335     } // fin de for i
336     return cadena;
337   } // fin de mstraActvos
```

El método auxiliar `daCalndrio`

Como este es un método auxiliar no tienes su descripción en la tarjeta de responsabilidades. Pero por cómo se está usando sabes que recibe un objeto de la clase `GregorianCalendar` y regresa una cadena. No usa ninguno de los atributos del objeto por lo que puedes declararlo público de la clase. A continuación está su documentación y el encabezado que coincide con su uso:

```
339   /**
340    * Convierte una fecha de GregorianCalendar en una cadena con la
341    * fecha y la hora representada en el parametro.
342    * @param fecha un GregorianCalendar que se desea "descifrar".
343    * @return una cadena que corresponde al parametro.
344    */
345   public static String daCalndrio(GregorianCalendar fecha) {
```

Lo primero que haces en el método, como en todos aquellos que reciben referencias, es verificar que la referencia sea válida. Si no lo es porque es nula, sales del método con el mensaje adecuado.

```
346     if (fecha == null) return "fecha␣invalida";
```

Si la fecha es válida usa los métodos que viste en la clase `Disco` para interpretar objetos de la clase `GregorianCalendar` y armas una cadena, teniendo cuidado de distinguir entre la hora 1 y la hora distinta de 1:

```
347     String laFecha = Disco.daFecha(fecha)
348       + ( fecha.get (fecha.HOUR) != 1
349           ?  "␣a␣las␣" :  "a␣la␣" )
350       + Disco.daHora(fecha);
```

Una vez armada la fecha en una cadena, el método la regresa.

```
351     return laFecha;
352   } // fin de daCalndrio
```

Nos vemos en el próximo video donde seguirás programando métodos de la clase `Catalogo`.