Métodos de acceso y actualización

Tienes la siguiente lista de atributos en la clase para manejar el catálogo de discos:

```
48  /* Variables de objeto */
49  private Disco [ ] catalogo; // ...
50  private int numDiscos = 0;  // ...
51  private GregorianCalendar [ ] [ ] fechas;
51                              // ...
58  private GregorianCalendar [ ][ ][ ] historico;
59                              // ...
62  private int [ ] numHist;    // ...
```

Si bien todos los atributos deben poder obtenerse mediante un `get`, ese no es el caso para los métodos `set`, ya que muchos de los arreglos dependen del disco dado de alta en el catálogo, conforme se dan de alta. Por ello tendrás todos los métodos de consulta pero sólo uno de actualización, para sustituir un catálogo por otro, inicializando el resto de los arreglos de acuerdo al contenido del arreglo que pasa como argumento. Los métodos `get` simplemente regresan el atributo solicitado y tienen como tipo de regreso, cuando se trata de un arreglo, el tipo de los elementos y el número de dimensiones del arreglo:

```
150 /**
151  * Regresa el catalogo de discos, incluyendo
152  * aquellas posiciones que no tienen disco.
153  * @return Arreglo con el catalogo de discos.
154  */
155 public Disco [ ] getCatalogo ( ) {
156   return catalogo;
157 }
158 /**
159  * Regresa el numero de discos registrados en el
160  * Catalogo.
161  * @return el numero de discos registrados en
162  * la empresa.
163  */
164 public int getNumDiscos ( ) {
165   return numDiscos;
166 }
167 /**
168  * Regresa un arreglo de dos dimensiones donde en cada
169  * renglon registra las fechas de prestamo para cada
170  * disco.
171  * @return el arreglo con las fechas de los prestamos
172  * para cada disco.
173  */
174 public GregorianCalendar [ ][ ] getFechas ( ) {
175   return fechas;
176 }
```

```
177   /**
178    * Regresa el arreglo historico de transmisiones
179    * iniciadas y terminadas para cada disco. Hay una
180    * pareja de renglones con fechas para cada disco,
181    * por lo que el arreglo es de tres dimensiones.
182    * @return el historico de todos los discos.
183    */
184   public GregorianCalendar [ ][ ][ ] getHistorico ( ) {
185     return historico;
186   }
```

El método `setCalendario` no es tan sencillo. Como en el constructor de dos parámetros, hay que copiar el catalogo `nuevos` al atributo `catalogo`, pero en este caso este último es *sustituido* por el primero, manteniendo el tamaño de `nuevos`.

```
187   /**
188    * Se copia el catalogo del parametro, si no es
189    * una referencia nula. Si nuevos es una referencia
190    * nula, se "borran" todos los arreglos asociados.
191    * Para cada disco que existe en nuevos se
192    * inicializan los arreglos asociados y se anota cuantos
193    * discos hay. No se copian las referencias nulas (ni se
194    * cuentan) para que no haya huecos en el arreglo.
195    * @param nuevos El arreglo con el que se va a inicializar el
196    *               catalogo y todos los arreglos asociados.
197    */
198   public void setCatalogo( Disco [ ] nuevos ) {
```

Sin embargo hay que vigilar que la referencia que te están pasando no sea nula. Si lo es, equivale a "borrar" el catálogo, junto con todos los arreglos asociados y así lo haces, saliendo inmediatamente después del método.

```
199     int cuantos = nuevos == null ? 0 : nuevos.length;
200     if (cuantos == 0 ) { // "Borrar" todos los arreglos
201       catalogo = null;
202       fechas = null;
203       historico = null;
204       numDiscos = 0;
205       numHist = null;
206       return; // Recuerda que regresa void
207     }
```

Si la referencia no es nula, procedes a construir los arreglos asociados del mismo tamaño que `nuevos`.

```
208     // Llegas aca si la referencia no es nula (cuantos != 0)
209     catalogo = new Disco[cuantos]; // Reconstruyes el catalogo
210                                    // y el resto de los arreglos
211     fechas   = new GregorianCalendar[cuantos][ ];
212     historico = new GregorianCalendar[cuantos][2][ ];
213     numHist  = new int [ cuantos];
```

```
214     // Quieres copiar los discos vivos.
215     // catalogo empieza con todas sus referencias en null.
216     numDiscos = 0; // No has copiado ningun disco "vivo"
```

Posteriormente para los discos que están dados de alta en `nuevos`, construyes las columnas correspondientes en los arreglos. Si encuentras una posición en el catálogo nuevo donde su referencia sea nula simplemente no lo copias al catálogo definitivo. Con esto garantizas que en el catálogo definitivo todos los discos ocupen posiciones consecutivas. Cada disco que copies, como no sabes su historia de cuándo dio transmisiones y cuándo las terminó, mejor supones que tiene cero transmisiones activas y empiezas como si el catálogo fuera nuevo. No es lo mismo que construir uno nuevo porque el catálogo que te pasan como parámetro puede tener discos registrados pero no en posiciones contiguas.

```
217     for ( int i = 0; i < cuantos; i++) {
218       if (nuevos[i] == null) continue; // Regresas al encabezado del for
219       // Llegas aca si hay un disco vivo en esa posicion
220       catalogo[numDiscos] = nuevos[i]; // lo copias al catalogo
221       int numPerm = catalogo[i].getPermitidas(); // el disco no es nulo
222       catalogo[numDiscos].setActivas(0); // Inicializas transmisiones
223       fechas[numDiscos] = new GregorianCalendar[numPerm];
224       historico[numDiscos][0] = new GregorianCalendar[2 * numPerm];
225       historico[numDiscos][1] = new GregorianCalendar[2 * numPerm];
226       numDiscos ++;
227     }// for
228   } // setCatalogo
```

Como los arreglos de objetos se inician con `null` en todas sus posiciones, las que no fueron explícitamente copiadas tienen ese valor.

Métodos de implementación

Regresas a la tarjeta de responsabilidades para ver los métodos de implementación que anotaste en ella. Irás uno por uno.

Método que agrega un disco al catálogo

Nombre	Salida	Entradas	Descripción
addCatalogo	boolean	Disco	Agrega un disco al catalogo si es que hay lugar. Dice si lo agregó o no

Puedes empezar con la documentación de Javadoc y el encabezado del método:

```
230   /**
231    * Agrega un disco al catalogo, si es que hay lugar.
232    * @param nuevo El disco a agregar.
233    * @return false si el disco es nulo o ya no hay lugar;
234    * verdadero si lo pudo agregar.
235    */
236   public boolean addCatalogo ( Disco nuevo ) {
```

Como ya es costumbre, primero tienes que saber si la referencia que te están pasando es válida. Si es una referencia nula o si ya no hay espacio en el catálogo sales del método regresando `false`, ya que no se agregó ningún disco.

```
237     if (nuevo == null || numDiscos >= catalogo.length)
238       return false; // No agrega nada
```

Si la referencia es válida, la acomodas en el primer lugar disponible, indicado por `numDiscos`.

```
239     // Llegas aca porque hay lugar y la referencia es valida
240     catalogo[numDiscos] = nuevo;
```

A continuación, igual que en el tercer constructor, obtienes el valor del atributo `permitidas` de ese disco para construir las columnas de los arreglos asociados. Sabes que la referencia no es nula.

```
241     int numPerm = nuevo.getPermitidas(); // transmisiones permitidas
242     fechas[numDiscos] = new GregorianCalendar[numPerm];
243     historico[numDiscos][0] = new GregorianCalendar[2 * numPerm];
244     historico[numDiscos][1] = new GregorianCalendar[2 * numPerm];
```

Por último, incrementas el contador de discos registrados `numDiscos` y regresas el valor `true`.

```
245     numDiscos ++;
246     return true;
247   } // fin de addCatalogo
```

Método que da una transmisión de un disco en el catálogo

Nombre	Salida	Entradas	Descripción
daTransmision	boolean	posición del disco	Registra la transmisión del disco elegido a una cierta hora. Avisa si pudo o no dar la transmisión.

Este método debe pedirle al disco en la posición dada, si es que existe, una transmisión. Para poderla registrar necesita saber el lugar que ocupa el disco, que le pasan como parámetro, y el lugar que va a ocupar en fechas, por el que puede preguntar directamente al disco. Si el disco no existe o no tiene transmisiones disponibles devuelve `false`. Con este algoritmo puedes empezar codificando la documentación y el encabezado:

```
249   /**
250    * Otorga una transmision de un disco dado, determinado
251    * por la posicion que ocupa en el catalogo, de donde el
252    * usuario elige la posicion del disco.
253    * Obtiene la hora y fecha de la transmision y la registra
254    * para el disco en el arreglo fechas del disco elegido.
255    * @param cualDisco Posicion del disco que va a transmitir.
256    * @return falso Si las posiciones del disco es incorrecta;
257    * si ya no hay lugar para prestamos en ese disco;
258    * regresa true si se pudo dar la transmision.
259    */
260   public boolean daTransmision ( int cualDisco ) {
```

Empiezas por verificar que el número de disco sea válido:

```
261     if ( cualDisco >= catalogo.length || cualDisco < 0 )
262       return false;
```

Una vez que sabes que la posición es válida, verificas si en esa posición hay una referencia nula. Si es así, se sale del método entregando `false`.

```
263     if ( catalogo[ cualDisco ] == null )
264       return false;
```

Una vez que sabes que tienes una referencia válida, averiguas cuántas transmisiones tiene activas:

```
265     int numDato = catalogo[cualDisco].getActivas();
```

Comparas directamente con sus transmisiones permitidas y si ya están completas escribes un mensaje de que ya no hay lugar, diciendo el disco del que se trata, y sales con `false`.

```
266     if ( numDato >= catalogo[cualDisco].getPermitidas() ) {
267       System.out.println("El␣disco␣"+ catalogo[cualDisco].getNOMBRE()
268                          + "␣ya␣no␣tiene␣transmisiones␣disponibles");
269       return false;
270     }
```

Si no has salido del método, en este punto puedes registrar la transmisión en el disco elegido. Construyes una fecha, la registras en el catálogo y le pides al disco una transmisión con esa fecha. Te regresa un mensaje de si pudo o no dar esa transmisión. Comunicas al usuario (escribiéndole) el mensaje que te regresa el método de la clase `Disco`. Terminas el método regresando `true`.

```
271     // Llegas aca porque puedes dar una transmision
272     GregorianCalendar ahora = new GregorianCalendar();
273     fechas[cualDisco][numDato] = ahora; // registro en el catalogo
274     System.out.println( catalogo[cualDisco] . daTransmision(ahora) ) ;
275        // Registro en el disco en la posicion cualDisco
276     return true;
277   } // fin de daTransmision
```

En el siguiente video seguirás con la implementación de otros métodos de la clase `Catalogo`.