Método que lista las transmisiones activas para un disco dado

Este método es muy similar al que muestra todos los discos, excepto que únicamente lista las transmisiones activas, si es que las tiene, de un disco dado. La especificación en la tarjeta de responsabilidades es como sigue:

Nombre	Salida	Entradas	Descripción
mstraActivas	Cadena con la posición de cada transmisión	Posición del disco seleccionado	Lista las transmisiones activas junto con la posición que ocupa cada una de ellas

La documentación de Javadoc se presenta de la siguiente forma:

```
354  /**
355   * Muestra las transmisiones activas de un disco determinado.
356   * @param cualDisco el disco elegido en el catalogo.
357   * @return cadena con una lista de las transmisiones activas.
358   */
```

La única diferencia realmente con el método que muestra a *todos* los discos activos es que en este método estás eligiendo un disco particular, sin tener que recorrer todo el catálogo. El encabezado sigue directamente de la especificación en la tarjeta de responsabilidades. El método es público (aparece en la tarjeta de responsabilidades), regresa una cadena con una transmisión activa por renglón, se llama `mstraActivas` y recibe como parámetro la posición del disco que se desea revisar:

```
359  public String mstraActivas ( int cualDisco ) {
```

Lo primero que debes hacer es verificar que el catálogo contiene algo; que la posición dada existe y tiene un disco válido. Si alguna de estas condiciones no se da, sale del método regresando una cadena nula, después de escribir un mensaje al usuario.

```
360    if ( catalogo == null // el catalogo exista
361         /* la posicion sea valida */
362         || cualDisco < 0 || cualDisco >= catalogo.length
363         /* el disco en esa posicion existe */
364         || catalogo[cualDisco] == null) {
365      System.out.println("Este␣disco␣no␣existe␣en␣el␣catalogo");
366      return null;
367    } // del if
```

El orden en la expresión booleana es importante ya que Java, en cuanto tenga alguna de las subexpresiones verdadera ejecutará la línea 366. Para que brinque este enunciado y siga dentro del método es necesario que **todas** las subexpresiones sean falsas.

Obtienes el número de transmisiones activas de disco elegido, que ya sabes que existe, pidiéndole este valor al disco. Si no es mayor que cero, das el mensaje adecuado y sales del método.

```
368    int cuantas = catalogo[cualDisco].getActivas(); // peticion al disco elegido
369    if ( cuantas <= 0 ) {
370      System.out.println( "Este␣disco␣no␣tiene␣transmisiones␣activas" ) ;
371      return null ;
372    }
```

Si sigues dentro del método y llegas a la línea 373 es porque el disco tiene transmisiones activas, que se encuentran en el renglón `fechas[cualDisco]`. Inicias la cadena con la identificación del disco y recorres el renglón en la posición `cualDisco` desde la columna cero hasta la `cuantas − 1`, donde obtienes todas las transmisiones que no han sido terminadas. Esto lo haces con un `for`.

```
373     String cadena =
374       catalogo[cualDisco].muestraDisco ("Transmisiones␣activas:")
375       + "\n";
376     for (int i = 0; i < cuantas; i++) {// numero de transmisiones activas
```

Para cada columna verificas que no sea una referencia nula. Si no lo es, procesas la fecha y la agregas a la cadena a regresar, volviendo a usar el método estático `daCalndrio`

```
377       if (fechas[cualDisco][i] != null)
378         cadena += "[" + i + "]\t" + daCalndrio(fechas[cualDisco][i])
379         + "\n";
380       else
381         cadena += "fecha␣no␣registrada\n";
382     }  // del for
```

Finalmente regresas la cadena armada y terminas el método:

```
383     return cadena;
384   }  // fin de mstraActivas
```

Terminar una transmisión activa

La descripción de este método en la tarjeta de responsabilidades es la siguiente:

Nombre	Salida	Entradas	Descripción
terminaTrans	boolean	posición del disco y consola	Pregunta, de este disco, cuál es la transmisión a eliminar. Transcribe el inicio y final al histórico. Avisa si pudo o no.

De este encabezado puedes obtener la documentación de Javadoc y el encabezado del método. El método es público, de objeto, regresa un valor booleano, se llama `terminaTrans` y recibe como parámetros la posición del disco a modificar y una consola (`Scanner`):

```
386   /**
387    * Termina una transmision activa y la coloca en el historico
388    * de transmisiones para ese disco.
389    * @param cualDisco la posicion del disco cuya transmision se
390    * desea terminar.
391    * @param cons la consola a traves de la cual se hace la
392    * comunicacion.
393    * @return si pudo (true) o no (false) terminar la transmision.
394    */
395   public boolean terminaTrans(int cualDisco, Scanner cons) {
```

Para terminar una transmisión el usuario debe decir la posición del disco del que quiere terminar la transmisión. Las acciones a realizar son:

1. Verificar que los parámetros sean correctos.
2. Mostrar las transmisiones activas del disco seleccionado, incluyendo la posición de cada transmisión.
3. Pedirle al usuario que elija la posición de una transmisión.
4. Obtener la fecha y hora en que se está terminando la transmisión con un nuevo objeto de GregorianCalendar.
5. Colocar en el histórico, en la posición que le corresponde, el registro de inicio y fin de la transmisión, incrementando el contador de transmisiones terminadas para ese disco.
6. Recorrer las transmisiones que se encuentran a la derecha de la eliminada, un lugar a la izquierda, para que las transmisiones activas se encuentren consecutivas en el arreglo `fechas`.
7. Actualizar el número de transmisiones activas en el disco.

Para implementar el primero punto ya sabes qué hacer. Antes que nada hay que verificar que los parámetros que te están pasando sean válidos.

```
396     if (cualDisco < 0 || cualDisco >= numDiscos
397        || catalogo[cualDisco] == null) { // verificar disco
398       System.out.println("El␣disco␣"+ cualDisco + "␣no␣existe");
399       return false;
400     }
401     if (cons == null ) {     // verificar consola
402       System.out.println("No␣es␣una␣consola␣valida");
403       return false;
404     }
```

Para los puntos 2 al 4 también sabes qué hacer y cómo hacerlo. Muestras al usuario el contenido del disco solicitado. Verificas si tiene transmisiones activas y si las tiene le muestras las transmisiones activas en el orden en que están registradas, usando para esto último el método que acabas de programar, registrando el número de transmisiones activas:

```
405     System.out.println("Para␣el␣disco\n"
406              + catalogo[cualDisco].muestraDisco("Disco␣num.␣["
407                                                 + cualDisco + "]"));
408     System.out.println("\nTenemos␣las␣siguientes␣transmisiones␣activas:");
409     String cadena = mstraActivas(cualDisco);
410     if (cadena == null) {
411       System.out.println("Este␣disco␣no␣tiene␣transmisiones␣activas");
412       return false;
413     }
414     System.out.println(cadena); // cadena != null
415     // Pides al disco sus transmisiones activas
416     int numActivas = catalogo[cualDisco].getActivas();
```

Lo siguiente que tienes que hacer es obtener la posición de la transmisión que se desea eliminar, vigilando que la posición sea correcta. Para ello vas a programar un método `pideNum` cuyo encabezado es:

```
    public static int pideNum( String msg, Scanner cons, int minVal, int maxVal) {
```

Como es un método que interacciona con el usuario y con este tipo de interacciones tienes que tener mucho cuidado y prevenir errores de todo tipo, pospones su programación. Sabes que si el usuario no elige de manera adecuada, el método regresará un -1, que no puede ser una selección válida. Para poder compilar y que lo puedas programar después, simplemente regresas el valor `0` (cero) y sigues adelante como si tuvieras el método completo. No vas a poder probar el proyecto, pero por el momento no importa.

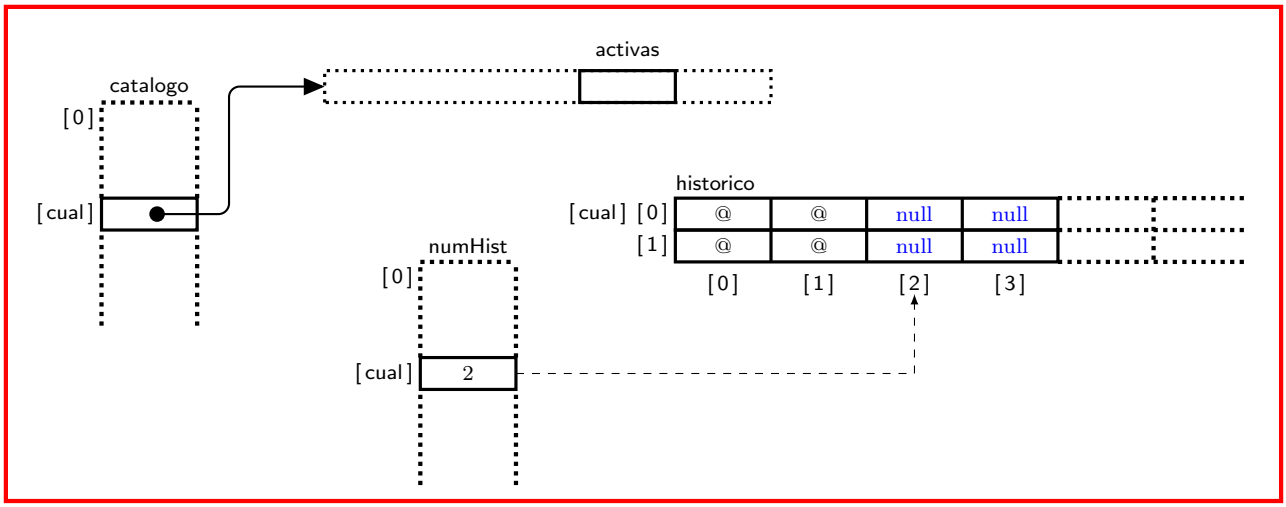
Sigues con el método que termina una transmisión en un disco dado.

```
416     int cualTrans = pideNum("Elige␣el␣numero␣de␣transmision"
417                             + "␣a␣terminar,␣", cons, 0, numActivas - 1);
418     if (cualTrans == -1) { // el numero dado esta fuera de rango
419       System.out.println("Esta␣transmision␣no␣existe");
420       return false;
421     }
```

Obtienes la fecha de inicio del arreglo `fechas[cualDisco]` y construyes el fin de la transmisión seleccionada, construyendo un `GregorianCalendar` en el momento.

```
422     GregorianCalendar inicio = fechas[cualDisco][cualTrans];
423     GregorianCalendar fechaFin = new GregorianCalendar();
```

Tienes la siguiente organización para los registros históricos:



Para registrar en el arreglo `historico` la transmisión terminada, eliges la posición del disco para que te diga el arreglo `numHist` la posición en la que tienes que registrar la transmisión:

```
424     int donde = numHist[cualDisco];
```

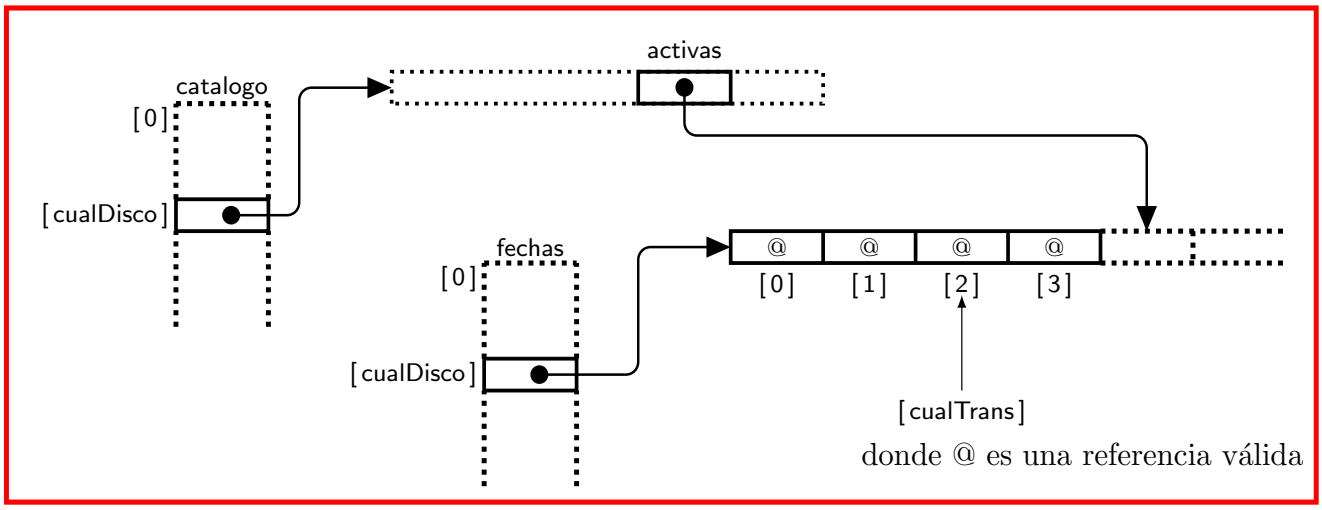
Una vez obtenida la información de la posición en `historico` donde se deben registrar las fechas, procedes a registrarlas e incrementas el contador para ese disco.

```
425     historico[cualDisco][0][donde] = inicio;
426     historico[cualDisco][1][donde] = fechaFin;
427      numHist[cualDisco]++;
```

Das el mensaje de la fecha y hora en que se termina la transmisión, que ya tienes construida:

```
428     System.out.println("Transmision terminada: "
429                        + daCalndrio(fechaFin));
```

Para programar el inciso 6 hay que tener cuidado al dar de baja una transmisión, porque estás suponiendo que las transmisiones activas se encuentran en posiciones consecutivas en la tabla `fechas`; el atributo `activas` apunta al primer lugar vacío en la tabla de fechas para el disco seleccionado:



Usarás un método de clase que recorre una celda dada de un arreglo de una dimensión. El método es público, de clase, regresa verdadero si pudo eliminar y falso si no; se llama `eliminaCelda` y recibe como parámetros el arreglo en el que va a eliminar y la posición a eliminar.

Método auxiliar `eliminaCelda`

La celda que quieres eliminar es en el arreglo de una dimensión `fechas[cualDisco]`, al mismo tiempo que vas a agregar registros a `historico[cualDisco][0]` e `historico[cualDisco][1]`, que dan el inicio y final de una transmisión que el usuario pide terminar. Por lo que la eliminación de la celda es, como ya se mencionó, en un arreglo de una dimensión. Ya agregaste al arreglo histórico y ahora hay que eliminar la celda del arreglo de fechas de inicio.

Con lo que ya se ha mencionado puedes ver la documentación de Javadoc y el encabezado del método:

```
437   /**
438    * Elimina la celda solicitada, si existe, en el arreglo
439    * correspondiente. Despues de llamar a este metodo,
440    * se tienen que hacer los decrementos correspondientes
441    * a los arreglos que se esten usando.
442    * @param arreglo de una dimension del que se va a eliminar
443    * la celda.
444    * @param cual el indice del elemento que se va a eliminar.
445    * @return si pudo (true) o no (false) hacer la eliminacion.
446    */
447   public static boolean eliminaCelda(Object[] arreglo, int cual) {
```

El primer parámetro es un arreglo de una dimensión de objetos, porque puede acomodar a cualquier arreglo de discos, fechas o de cualquier otra clase de objetos, ya que toda clase hereda de (extiende a) la clase `Object`. El segundo parámetro es, simplemente, la posición del objeto que se quiere eliminar. Con este encabezado puedes usarlo en el método `terminaTrans`, una vez obtenida la posición en `fechas` de la fecha que deseas eliminar: Debes usarlo en este método, aunque lo programarás después. Lo invocas y si regresa verdadero, eliminas la transmisión directamente en el disco y sales del método regresando verdadero (todo esto en el método `terminaTrans` de la clase `Catalogo`).

```
430     if ( eliminaCelda(fechas[cualDisco], cualTrans)) {
431       catalogo[cualDisco].terminaTransmision();
432       return true;
433     }
```

Si no lo puedes eliminar y no has salido del método, lo haces regresando falso y terminando el método.

```
434     return false; // si no pudo eliminar y no ha salido
435   } // fin de eliminaTrans
```

Hay que notar que `fechas[cualDisco]` es un arreglo de una dimensión de objetos de la clase `GregorianCalendar`.

Trabajarás ahora con el método que recorre una posición en un arreglo de objetos. Tienes ya el encabezado en la línea 447 (ver la página anterior).

Como en todos los métodos que has programado, lo primero que debes hacer es verificar que los argumentos que llegan al método son correctos. Primero que la referencia al arreglo no es nula:

```
448     if (arreglo == null) { // verificar si el arreglo existe
449       System.out.println("El␣arreglo␣no␣existe");
450       return false;
451     }
```

A continuación aseguras que la posición esté en rangos:

```
452     // Si la posicion en el arreglo esta fuera de rango
453     if ( cual < 0 || cual >= arreglo.length) {
454       System.out.println("La␣posicion␣a␣eliminar␣no␣existe");
455       return false;
456    }
```

En ambas verificaciones, si la condición se evalúa a verdadera sales del método regresando falso y emitiendo un mensaje de error.

Una vez que llegas a la línea 457 es porque los argumentos son correctos. Cada vez que te encuentres en una posición del arreglo vas a copiar la posición siguiente a la actual y anular la siguiente. Vas a recorrer el arreglo con un `for` a partir de la posición indicada:

```
457     for (int i = cual + 1; // empiezas uno despues de la posicion dada
```

Quieres parar el recorrido por alguna de estas dos causas:

(a) Se acaba el arreglo porque estás en la última posición

(b) La posición contiene una referencia nula. Como estás suponiendo que todas las referencias válidas se encuentran consecutivas, eso quiere decir que ya no hay referencias válidas a la derecha de la siguiente.

Dado esto, la condición para que el `for` siga ejecutándose es:

```
458         i < arreglo.length && arreglo[i] != null; // sin salirse del arreglo
```

Al terminar cada iteración, pasas al siguiente elemento:

```
459         i++) {
```

Como ya se mencionó, lo que vas a hacer es copiar el contenido de la siguiente posición a la actual y anular la siguiente posición:

```
462           arreglo[i - 1] = arreglo[i]; // i al menos vale 1
463           arreglo[i] = null;          // ya esta copiado
464         } // for
```

Si llegas a la línea 463 es porque sí se pudo recorrer el arreglo, por lo que regresas verdadero.

```
463     return true;
464   } // fin de eliminaTrans
```

¡Te esperamos en el siguiente video para seguir programando los métodos de esta clase, incluyendo el método `pideNum`!