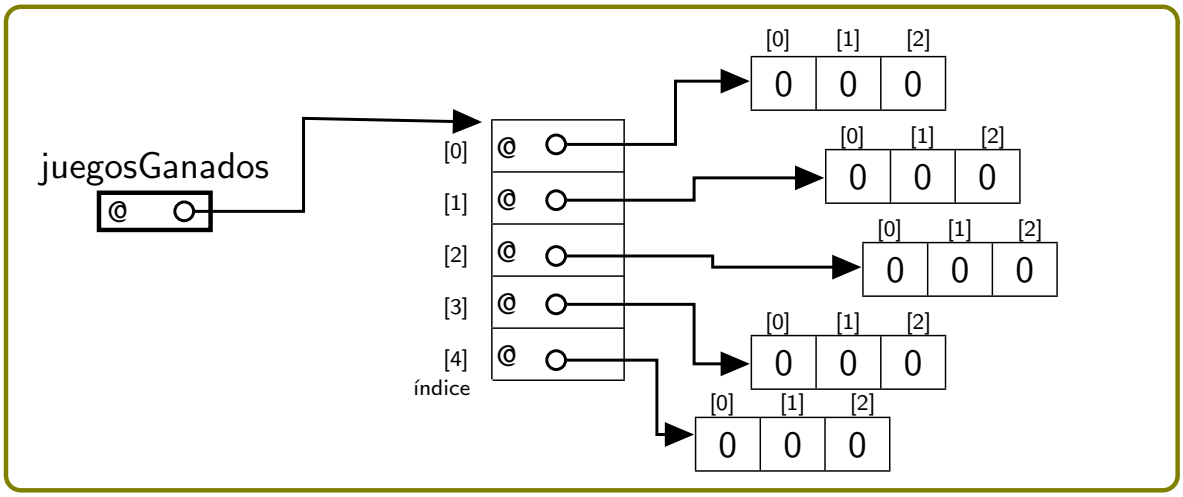
Más sobre arreglos

También puedes construir directamente el arreglo de dos dimensiones con el mismo número de columnas en cada renglón:

```
int[][] juegosGanados = new int[5][3]; // 5 renglones, cada renglon con 3 columnas
```

que queda en memoria como se muestra en la siguiente figura:



Asignación y uso (de elementos) de arreglos

Para hacer referencia a alguno de los elementos de un arreglo, lo hacemos de la siguiente manera:

- Cuando hablamos de *elementos* de un arreglo podremos estar hablando de una cara cuando el arreglo es de tres dimensiones, un renglón si tiene al menos dos dimensiones o una celda específica del arreglo.
- Todos los elementos de un arreglo comparten el identificador.
- Los elementos se distinguen entre sí porque ocupan posiciones distintas en el arreglo.
- Para obtener su valor basta elegir la posición del elemento en el arreglo y usarlo en una expresión.
- Para asignarle valor a un elemento, aparece del lado izquierdo de una asignación.
- La sintaxis de la selección de un elemento se hace también poniendo una expresión entera entre cada pareja de corchetes que tiene la declaración:

```
juegosGanados[i + 1]     // Elige el arreglo en la posicion i+1
juegosGanados[3][j]      // Elige cuarto renglon columna j
NDIAS[numDia]            // Elige el elemento en la posicion correspondiente
                         // al valor de numDia
catalogo[1]              // Elige al disco en la segunda posicion
```

- Si se usa un arreglo como parámetro formal, como tipo del parámetro aparecerá únicamente el tipo de los elementos y el número de dimensiones:

```
public int buscaMayor ( int[] enteros )
```

y el método que recibe el parámetro no tiene que conocer los tamaños de cada dimensión, que puede obtener con el atributo length dentro del método.

Insistimos:

- Las posiciones en cada dimensión **siempre** empiezan 0 (cero)
- Si tam es el tamaño del arreglo, el mayor índice es (tam - 1)
- Para obtener el número de elementos de un arreglo se usa el **atributo** length
- Si el arreglo es, por ejemplo, de tres dimensiones:

```
int[][][] juegos;
  /* conastruccion del arreglo */
juegos.length     // Da el numero de tablas
juegos[i].length  // Da el numero de renglones de la tabla en la posicion i
juegos[2][4].length // Da el numero de columnas en el renglon 4, tabla 2
```

- El identificador, como ya mencionamos, lo comparten todos los elementos.
- Si el elemento seleccionado aparece a la izquierda de una asignación se trata de la ubicación de un elemento (una variable). El lado derecho tiene que coincidir en tipo con el lado izquierdo. Si se eligió un renglón de un arreglo de dos dimensiones, en el lado derecho deberá haber un arreglo del mismo tipo de elementos de una dimensión.
- Si el elemento seleccionado aparece en una expresión, se trata del valor del argumento, en caso de que se trate de la selección de una celda, o de la referencia a un arreglo si se trata de un arreglo. Cuando se pasan arreglos como argumentos se evaluará de la misma forma que acabamos de mencionar.
- Como los arreglos son objetos, aunque Java los pase "por valor", se podrá modificar el contenido del arreglo, como con cualquier objeto, pero no se podrá cambiar "al arreglo" (su referencia).

Implementación de la clase Catalogo

Tenemos ya todos los conocimientos que requerimos para declarar los atributos de clase y objeto de la clase Catalogo por lo que estamos listos para empezar a programar la clase principal de esta aplicación, que se llama Catalogo

Indicamos que pertenece al paquete discos y también que queremos importar las clases GregorianCalendar y Scanner. Asimismo escribimos los comentarios necesarios para Javadoc y el encabezado de la clase.

```
1  package discos;
2  import java.util.Scanner;
3  import java.util.GregorianCalendar;
4  /**
5   * Simula el uso de un catalogo de discos.
6   * En cada sesion puede construir su base de discos,
7   * agregar discos, consultar el catalogo, iniciar
8   * transmisiones y terminar transmisiones, registrando
9   *  todas las actividades en el catalogo.
10  * @author: Elisa Viso.
11  * @version 2017.
12  */
13 public class Catalogo {
```

Constantes de clase para Catalogo

Requerimos, al igual que con el número de transmisiones permitidas o activas de la clase Disco, de una constante que nos de el tamaño máximo del catálogo de discos:

```
15  /** Numero maximo de discos en el catalogo */
16   public static final MAX_DISCOS = 50;
```

También, como el catálogo va a ofrecer un menú, del cual el usuario va a elegir la opción a ejecutar, vamos a declarar un arreglo de cadenas, donde cada cadena en la posición `cual` va a representar a la opción `cual` a ejecutarse.

```
17  /** Menu del catalogo */
18  public static final String[] MENU_CATALOGO = {
19    "Salir",                                  // 0
20    "Agregar␣disco",                          // 1
21    "Mostrar␣discos",                         // 2
22    "Mostrar␣discos␣activos",                 // 3
23    "Pedir␣transmision",                      // 4
24    "Terminar␣transmision",                   // 5
25    "Mostrar␣Disco",                          // 6
26    "Mostrar␣historico␣de␣un␣disco",          // 7
27    "Mostrar␣historico␣de␣todos␣los␣discos"   // 8
28  };
```

No queremos usar "números mágicos" al implementar el menú. Por eso declaramos lo que se conocen como constantes simbólicas de clase para que dentro del código podamos, si es necesario, cambiar las cadenas de lugar o agregar nuevas. Si deseas cambiar o agregar nuevas constantes, los cambios los haces únicamente en el lugar en que están estas declaraciones. Es conveniente **siempre** tener como primera opción salir del menú, porque siempre permanecerá en el primer lugar (0), aun cuando agreguemos más opciones al menú.

```
29  /** Accion de salir del menu. */
30  public final static int SALIR          = 0;
31  /** Accion de agregar un disco al catalogo. */
32  public final static int AGRGA_DSCO     = 1;
33  /** Accion de mostrar el catalogo. */
34  public final static int MSTRA_DISCOS   = 2;
35  /** Accion de mostrar el catalogo de discos activos.*/
36  public final static int MSTRA_DSCS_ACTVS = 3;
37  /** Accion de pedir una transmision. */
38  public final static int PIDE_TRNSMSN   = 4;
39  /** Accion de terminar una transmision. */
40  public final static int TRMINA_TRNSMSN = 5;
41  /** Accion de terminar una transmision. */
42  public final static int MSTRA_UNDISCO  = 6;
43  /** Mostrar los histroricos de un disco */
44  public final static int MSTRA_HIST     = 7;
45  /** Mostrar los histroricos de todos los discos */
46  public final static int MSTRA_HISTRS   = 8;
```

Atributos de objeto de la clase Catalogo

Las constantes simbólicas públicas las comentamos con Javadoc para que aparezcan en la página de la clase. Pero los atributos de un objeto son, en general, privados. Lo que es privado no aparece, de todos modos, en la página de la clase, por lo que usaremos comentarios sencillos

Recordemos que la clase Catalogo cuenta con varias colecciones:

- El *catálogo* propiamente dicho que es un arreglo de objetos de la clase Disco. La construcción del arreglo se va a dar en los distintos constructores, pero la declaración es:

```
50  /* Variables de objeto */
51  private Disco[] catalogo; // Catalogo de discos
```

  - Una cosa es el número total de posiciones en el arreglo, que se dan al construirlo y otra el número de posiciones ocupadas
  - Siempre, cuando tenemos colecciones (o arreglos) es conveniente llevar un contador de las posiciones ocupadas:

```
51  private int numDiscos = 0;  // Numero de discos en el catalogo de discos
```

- En la tarjeta de responsabilidades aparece también un arreglo por disco (dos dimensiones) con las fechas de inicio de cada transmisión del disco en esa posición.
  - Declaras un arreglo de dos dimensiones, con el número de renglones el mismo que el número de discos
  - El número de columnas del renglón en la posición i está dado por el número de transmisiones simultáneas permitidas para el disco en la posición i
  - El número de renglones se conoce en el constructor, pero el número de columnas hasta que se registre el disco

```
54  private GregorianCalendar [ ][ ] fechas;
55                                // Un renglon por disco, una columna
56                                // por transmision. Cada renglon tiene
57                                // un numero de columnas dada por
58                                // el disco en ese renglon. El numero
59                                // de fachas registradas esta dado
60                                // por el atributo activas del disco.
```

- También mencionamos un arreglo para guardar el histórico de las fechas de transmisiones iniciadas y terminadas.
  - Cada disco debe tener su propia tabla de histórico, lo que nos da una primera dimensión con un número de tablas igual al tamaño del catálogo.
  - En cada tabla debemos tener dos renglones, uno para las fechas de inicio de la transmisión y otro para las fechas de final de esa transmisión para ese disco.
  - Cada renglón debe tener un número suficiente de celdas para ir guardando las transmisiones que se van terminando. Como `permitidas` da el número máximo de transmisiones *simultáneas*, podemos pensar en el doble de estos lugares para el histórico.
  - El tamaño de la primera dimensión va a estar dado por el tamaño del catálogo, por lo que lo construiremos al mismo tiempo que al catálogo, en los constructores.
  - La segunda dimensión tiene tamaño 2, pero no se puede construir hasta que la primera dimensión esté definida, lo que haremos también en los constructores.

- La tercera dimensión va a ser dos veces el número de transmisiones simultáneas permitidas, y se va a construir cuando se asigne un disco al catálogo.

```
61  private GregorianCalendar [ ] [ ] [ ] historico;
62                          // Una pareja de renglones por cada disco
63                          // Fecha de inicio y fin de una transmision
64                          // para el disco elegido
```

- Como hemos hecho hasta ahora, una cosa es el tamaño de un arreglo (número de lugares disponibles) y otra el número de lugares *ocupados* en un momento dado. Para el arreglo `historico` el número de tablas que ocupan lugar es el mismo que el número de discos en el catálogo; el *máximo* número de lugares para transmisiones (fechas de inicio y fin) también queda definido con la construcción de un objeto de la clase `Disco`, pero el número de posiciones ocupadas no está dado y hay que irlas contando conforme se van ocupando los lugares. Para eso utilizaremos un arreglo de enteros con el mismo tamaño que el catálogo de discos, donde anotaremos, para cada disco, cuantos registros hemos llenado en el histórico.

```
65  private int [ ] numHist; // Da, para cada disco, el numero de
66                          // transmisiones iniciadas y terminadas
67                          // para ese disco
```

  Su construcción se hará en cuanto sepamos el número de lugares que hay para el catálogo, en los constructores, e indicarán, cada celda, el número de registros históricos que tiene ese disco.

Vista general de las relaciones entre los arreglos

Una vez construido el catalogo y todos los arreglos asociados, se da una relación como la que se ve en la figura en la siguiente página. Todos los arreglos relacionados tienen el mismo tamaño en el índice más externo, que es el mismo tamaño del `catalogo`.

El índice que selecciona un objeto `Disco` en el `catalogo`, elige las fechas de inicio y terminación de las transmisiones para ese disco, elige la posición para el histórico de ese disco. Los tres arreglos se manejan de forma paralela.

Otra forma de lograr esto mismo hubiese sido codificando una clase para cada renglón de los tres arreglos, pero el ansia de terminar pudo más.

Figura 1 Relación entre los arreglos que conforma a la clase `Catalogo`