El método `main` de la clase `Catalogo`

Quieres probar (ejecutar) tu clase para ver que todo funciones bien. Eso lo harás con un método `main` en esta misma clase. Empiezas con el encabezado de este método:

```
680   public static void main(String[] args) {
```

Vas a ir compilando conforme vayas codificando para verificar que todo está bien, al menos en la sintaxis.

Declaras un objeto de la clase `Catalogo`, usando el constructor al que le pasas el tamaño final del catálogo y un arreglo inicial con discos, cuyo encabezado es:

```
    public Catalogo (int numDscs, Disco[] nuevos)
```

Construyes el arreglo de objetos de la clase `Disco` "al vuelo" como viste en la lección correspondiente, dando el tipo y número de dimensiones del arreglo, que en este caso es una, y listando entre llaves los elementos. Cada uno de los elementos lo construyes explícitamente objetos de la clase `Disco`, al invocar al constructor de un objeto de la clase `Catalogo`.

```
682   Catalogo elMio =
683      new Catalogo (10,
684                  new Disco[]{new Disco(Disco.DVD,
685                                        "Ahora␣los␣ves,␣ahora␣no",
686                                        1999,5),
687                  new Disco(Disco.BR,"Billions",2015,4),
688                  new Disco(Disco.BR,"Outlander",2016,3),
689                  new Disco( Disco.CD,"Frank␣Sinatra",1992, 2)});
```

Usas las constantes simbólicas de la clase `Disco` que corresponden al tipo de disco. Compilas y te marca el error de que no conoce a las constantes `DVD`, `BR` y `CD` de la clase `Disco`, así que abres el archivo de esta clase para verificar cómo esxtán declaradas estas constantes.

De la clase `Disco`

```
private static final short CD = 1,
                           DVD = 2,
                           BR = 3;
```

Ves que estas constantes están declaradas como `private` en la clase `Disco` y les cambias el acceso a `public`. Compilas la clase `Disco`, cierras el archivo, vuelves a compilar la clase `Catalogo` y ves que se resolvió el problema.

Una vez que cuentas con un objeto de esta clase le pides que interaccione con el usuario usando el método que acabas de programar.

```
691    // Inicia la comunicacion desde el catalogo con el usuario
692    elMio.conectaCatlgo();
```

En este punto puedes construir más catálogos y conectarlos al usuario. Recuerda que los catálogos que construyas después (o antes), la comunicación con el usuario va a ser una después de la otra, pues el flujo de la ejecución no va a continuar hasta que salga del método `conectaCatlgo` invocado desde algún objeto de la clase `Catalogo`.

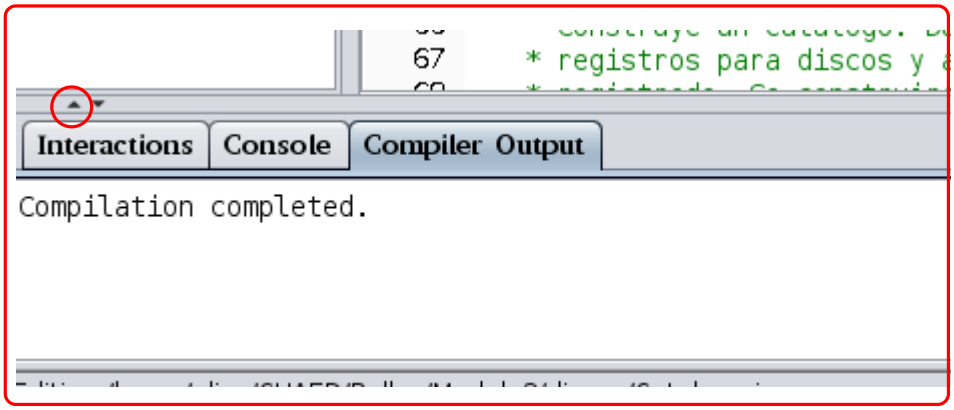
Puedes invocar más de una vez a este método con el mismo objeto o con objetos distintos. Si se trata del mismo objeto dentro del mismo método `main`, el contenido del catálogo se mantendrá a

través de invocaciones de este método, a menos que vuelvas a construir un catálogo nuevo usando el mismo identificador.

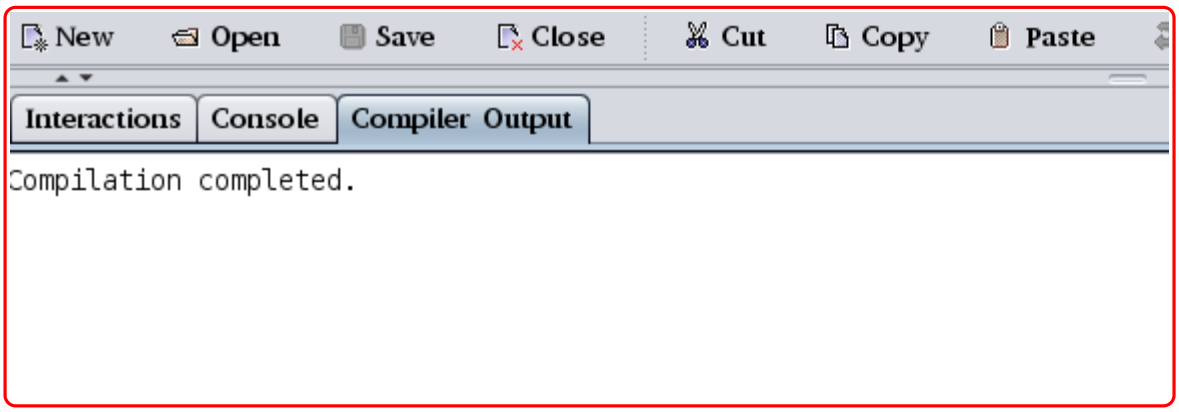
Terminas la clase cerrando el método `main` y a la clase misma.

```
693   } // main
694 } // Catalogo
```

Para poder observar la ejecución tenemos que agrandar el panel de interacciones, lo que conseguimos haciendo click, arriba de la pestaña correspondiente, a la flecha que apunta hacia arriba:



Con este click la pantalla completa se vuelve de interacción.



Se hace click sobre la pestaña Run y se empieza a ejecutar el programa, mostrando el siguiente texto:

```
Bienvenido al Catalogo de discos
Menu de opciones de trabajo
============================
 [0] Salir
 [1] Agregar disco
 [2] Mostrar discos
 [3] Mostrar discos activos
 [4] Pedir transmision
 [5] Terminar transmision
 [6] Mostrar dsco
 [7] Mostrar historico de un disco
 [8] Mostrar historico de todos los discos
Elige una opcion [0-8]
terminando con [Enter]: -->
```

Supón que se elige la opción 2. Pero vuelve a esperar un [Enter], por lo que tendrás que revisar como pide la opción, así que al volver a elegir tecleas 0 (cero) para salir de la ejecución. Regresas a revisar la clase `Catalogo`:

```
550      // Al terminar de mostrar el menu, pedirle al usuario la opcion
551      opcion = pideNum (cons, "Elige una opcion ",
552                        0, MENU_CATALOGO.length - 1);
553      cons.nextLine(); // Para comerse el [Enter]
```

Recuerdas que `pideNum` ya se come el [Enter] y que tú te lo vuelves a comer en la línea 553. Borras esa línea y vuelves a compilar y ejecutar. Ya se resolvió este problema.

En la siguiente ejecución pides la opción 3, de mostrar discos activos y te dice que no hay discos activos. Vuelves a elegir, y esta vez eliges que te muestre todos los discos y la ejecución te dice que esta opción no está implementada. Regresas a ver completo el archivo de la clase `Catalogo`, pero al buscar la constante simbólica `MSTRA_DISCOS` ves que sólo aparece en la declaración, pero no como etiqueta en el `switch`, así que hay que agregar esa opción. La colocas entre `AGRGA_DISCOS` y `MSTRA_DSCS_ACTVS`.

Como siempres, empiezas preguntando si hay algo en el cartálogo y si no es válido o no hay nada, le das el mensaje adecuado al usuario y sales del `switch`:

```
568 case MSTRA_DISCOS:
569          if (catalogo == null || numDiscos <= 0) {
570            System.out.println("No hay discos registrados "
571                               + "en el catalogo");
572            break;
573          }
```

Si no has salido del `switch` quiere decir que hay algo en el catálogo, por lo que simplemente lo vas a recorrer, pidiéndole a cada disco que se muestre. Es conveniente ponerle un encabezado a este listado, para luego recorrer el cat álogo de discos, pidiéndole a cada disco que se muestre:

```
574     System.out.println("Discos disponibles");
575     for (int i=0; i < numDiscos; i++)
576        catalogo[i].muestraDisco();
577     break;
```

Este código no compila porque el método `mstraDisco` de la clase `Disco` espera una cadena como parámetro. La cadena que eliges es el número de disco que está mostrando y cambias la llamada del método:

```
574     for (int i=0; i < numDiscos; i++)
575        catalogo[i].muestraDisco("Disco[" + i + "]:");
576     break;
```

Te escribe el encabezado y ya no te da mensaje de opción errónea. Lo que pasa es que `muestraDisco` regresa una cadena, pero la estás desperdiciendo. Hay que poner esta invocación dentro de un `println`, por lo que el código queda como se muestra a continuación y ya funciona.

```
574     System.out.println("Discos␣disponibles");
575     for (int i=0; i < numDiscos; i++)
576        System.out.println(catalogo[i].muestraDisco("Disco[" + i + "]:"));
577     break;
```

Recuerdas que la clase `Catalogo` tiene un método que muestra todo su contenido, así que podrías sustituir todo este código con la llamada a ese método:

```
574          System.out.println(mstraCatalogo("Discos␣disponibles") );
575          break;
```

Regresas ahora a compilar y ejecutar tu clase y al elegir la opción 2 se muestra todo el catálogo de discos, e inmediatamente después se vuelve a mostrar el menú:

```
Discos disponibles
[0] (DVD) Ahora los ves, ahora no
        Num. de transmisiones permitidas: 5
        Num. de transmisiones activas: 0

[1] (BR) Billions
        Num. de transmisiones permitidas: 4
        Num. de transmisiones activas: 0

[2] (BR) Outlander
        Num. de transmisiones permitidas: 3
        Num. de transmisiones activas: 0

[3] (CD) Frank Sinatra
        Num. de transmisiones permitidas: 2
        Num. de transmisiones activas: 0


Menu de opciones de trabajo
===========================
 [0] Salir
 [1] Agregar disco
 [2] Mostrar discos
 [3] Mostrar discos activos
 [4] Pedir transmision
 [5] Terminar transmision
 [6] Mostrar disco
 [7] Mostrar historico de un disco
 [8] Mostrar historico de todos los discos
Elige una opcion (terminando con [Enter]): [0-8] -->
```

Resulta interesante que pruebes cada una de las opciones, como es agregar un disco, eliminar un disco, pedir una transmisión (o varias) y después terminar una o varias transmisiones. Primero pides transmisiones al último disco, ya que tiene pocas permitidas. Pides una transmisión y te das cuenta que la hora que te está poniendo es siempre la misma. Regresas al archivo de la clase `Disco`, que es la que te da la fecha y revisas bien estos métodos. En realidad lo hiciste cuando terminaste la clase `Disco`, pero por alguna razón se regresó a un archivo anterior. Para corregir tienes que distinguir bien, en la clase `GregorianCalendar`, cuáles atributos son constantes de clase, como `AM` o `PM`, y cuáles identifican a campos que coontienen la información, como son `AM_PM`, `MINUTED` o `YEAR`.

El método `daHora` de la clase `Disco` recibe un `GregorianCalendar` como parámetro y finalmente queda, agregando que muestre los segundos, como sigue:

```
/**
 * Edita la hora para que salga en singular o plural.
 * @param cal un GregorianCalendar.
 * @return Una cadena que dice la hora con AM o PM.
 */
public static String daHora (GregorianCalendar cal) {
  int hora = cal.get (cal.HOUR);
  int minutos = cal.get (cal.MINUTE);
  int segundos = cal.get (cal.SECOND);
  String ampm = cal.get(cal.AM_PM) == cal.AM
    ? "AM" : "PM";
  hora = (hora == 0 && cal.get(cal.AM_PM) == cal.PM) ? 12 : hora;
  return "␣" + (hora < 10 ? "0" + hora : hora) + ":"
    + (minutos < 10 ? "0" + minutos : minutos)
    + ":" + (segundos < 10 ? "0" + segundos : segundos)
    + "␣" + ampm + "␣";
}
```

Haces algunas pruebas más, terminando transmisiones y revisando el histórico, y ves que ya todo funciona como querías.

Al elegir la opción para `Salir`, la ejecución se despide y termina:

```
Elige una opcion (terminando con [Enter]): [0-8] --> 0
Termina la sesion.
Hasta luego.
>
```

Con esto termina este primer curso de introducción a Java. Esperamos que lo hayas disfrutado y aprendido mucho.