SESIÓN 7: Clases y Objetos II. Herencia, Clases Abstractas e Interfaces.

Objetivos

- Profundizar y repasar el diseño de clases a través de la clase *Complejo*.
- Reescribir la clase *Fraccion* que implemente la interface *Comparable.*
- Desarrollar clases derivadas o *subclases* a partir de una clase base o *superclase*.
- Saber invocar al constructor de la *superclase* usando la palabra reservada *super.*
- Saber sobrescribir métodos de objeto en las *subclases.*

Nota importante: Siga el esquema de nombrado de paquetes que se indicó en la sesión 01 es decir: **org.ip.sesion07**. En ese paquete se crearán todos los programas que se proponen en la sesión dándoles un nombre al programa y que se indica en cada ejercicio entre **paréntesis y en negrita**.
Al final de la sesión, el alumno deberá cargar el trabajo realizado a su repositorio personal indicando la clave correspondiente a la sesión.

Ejercicios propuestos

1. Implementa en este ejercicio la jerarquía de clases que se expone a continuación.

La clase `Hora` **(Hora)** que representa un instante de tiempo concreto, compuesto por una hora ($0 \leq$ hora ≤ 23) y un minuto ($0 \leq$ minuto ≤ 59). Y dispone de los siguientes métodos:

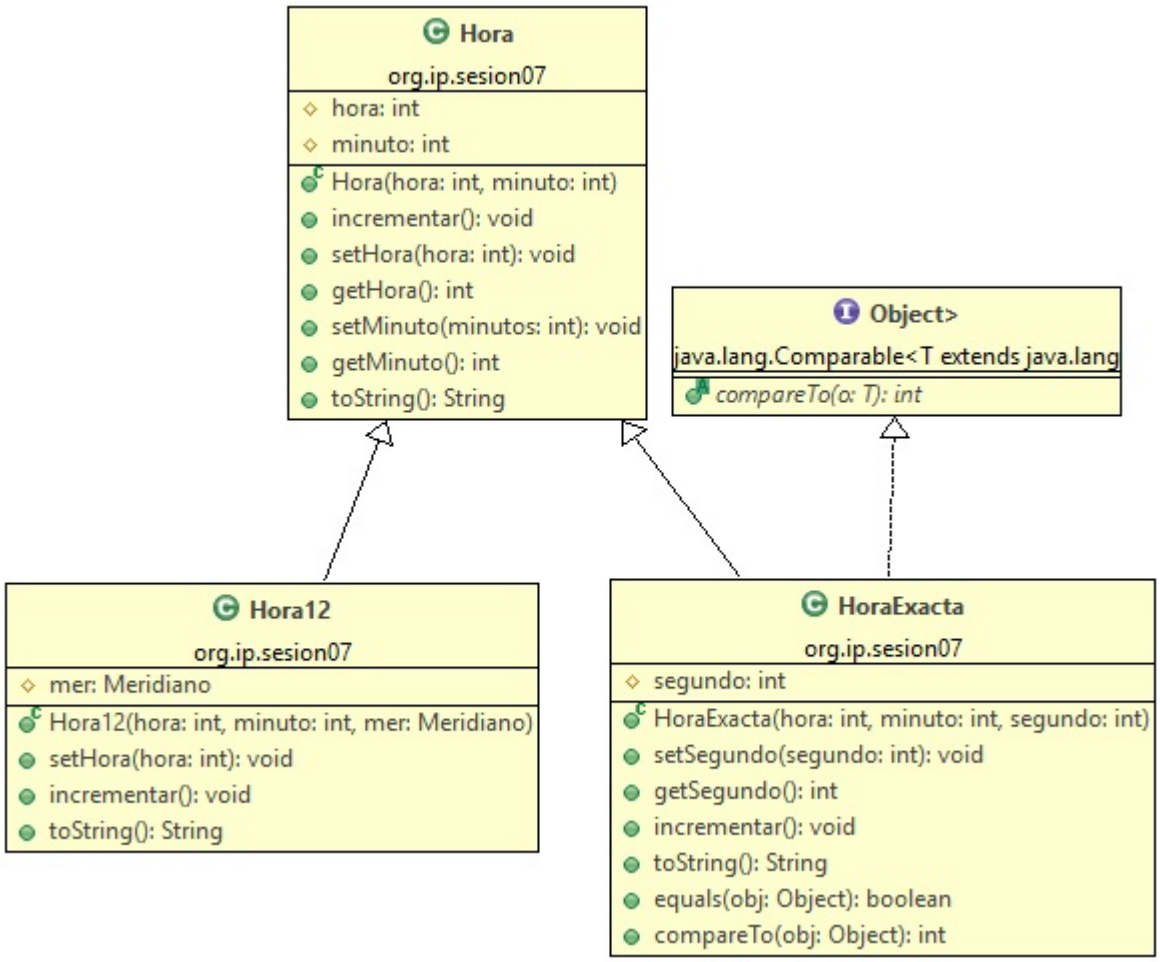
- `Hora(hora, minuto)`: construye un objeto con los datos que se pasan como parámetros.
- `incrementar()`: incrementa el instante de tiempo actual en un minuto.
- `setHora(hora)`: asigna un valor a la hora, si éste tiene sentido $0 \leq$ hora ≤ 23).
- `getHora()`: obtiene el valor de la hora.
- `setMinuto(minuto)`: signa un valor al minuto, si éste tiene sentido ($0 \leq$ minuto ≤ 59).
- `getMinuto()`: obtiene el valor del minuto.
- `toString()`: devuelve un `String` con la representación del instante de tiempo (reloj)

Ahora, se debe implementar una nueva clase, `Hora12` **(Hora12)**, que funciona de forma similar a la clase `Hora`, con la diferencia de que las horas sólo pueden tomar un valor entre 1 y las 12 ($1 \leq$ hora ≤ 12) y se distingue la mañana de la tarde mediante "**AM**" y "**PM**". Para estos dos valores se pueden utilizar los **enumerados**, que permiten definir grupos de constantes como valores posibles de una variable y se usa para definirlos la palabra reservada **`enum`**. Para acceder a los valores constantes de un enumerado se utiliza la sintaxis `NombreEnumerado.ValorConstante`. En nuestro caso podremos definir

```
public enum Meridiano { AM, PM }
protected Meridiano mer;
```

Por último, a partir de la clase `Hora`, implementar la clase `HoraExacta` **(HoraExacta)**, (que a su vez implementa la interface `Comparable` para obligar a implementar el método `compareTo`), que incluye en el instante de tiempo los segundos ($0 \leq$ segundo ≤ 59). Además de re-implementar, caso de ser necesario, los métodos visibles de la clase `Hora`, dispondrá de los siguientes métodos:

- `HoraExacta(hora, minuto, segundo)`: construye un objeto de la clase `HoraExacta` con los datos pasados como parámetros.
- `setSegundo(segundo)`: que asigna un valor pasado como parámetro al segundo, siempre y cuando tenga sentido ($0 \leq$ segundo ≤ 59).
- `getSegundo()`: obtiene el valor del segundo.
- `incrementar()`:incrementa el instante de tiempo actual en un segundo.
- `equals(Object)`: compara si dos instantes de tiempo son iguales o no.
- `compareTo(Object)`: Comparar dos instantes de tiempo (hora, minuto, segundo).



Implementar un programa **(TestHora)** que pruebe el funcionamiento de la jerarquía de clases anteriormente definida. A modo de ejemplo, se proporciona una salida que te permitirá comprobar si las operaciones que se han implementado en la jerarquía de clases son correctas.

Algunas indicaciones. Para incrementar minutos o segundos se utilizarán bucles `for` como los siguientes:

```
for (int i = 1; i <= 61; i++) {
     reloj.incrementar();
}

for (int i = 1; i <= 4000; i++) {
     relojExacto.incrementar();
}
```

Los momentos exactos para comparar con `equals` o `compareTo` son los siguientes:

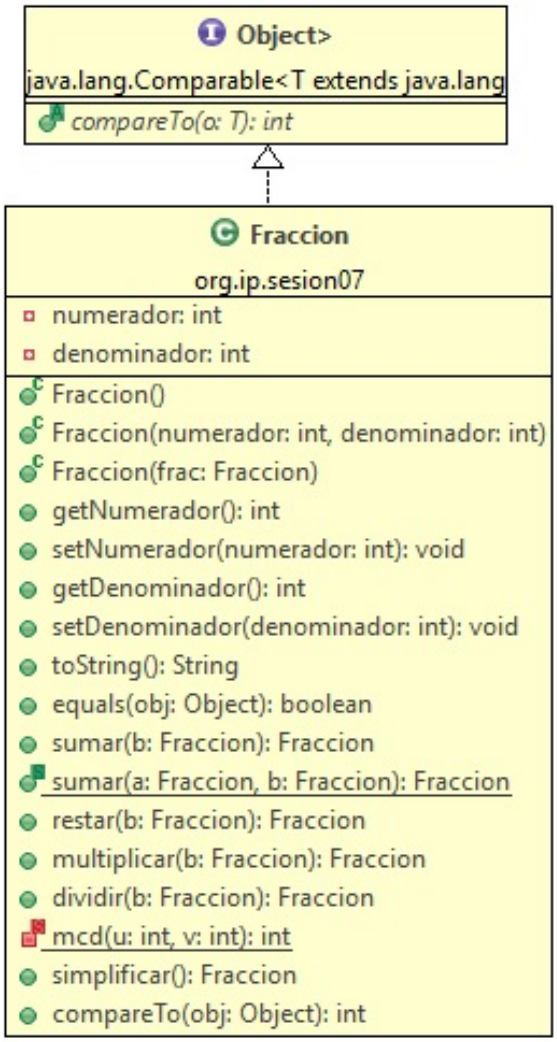
```
HoraExacta momentoExacto1 = new HoraExacta (1, 2, 3);
HoraExacta momentoExacto2 = new HoraExacta (1, 2, 3);
HoraExacta momentoExacto3 = new HoraExacta (10, 20, 30);
```

Ejemplo de ejecución:

```
Hora del reloj: 11:30
Hora del reloj: 12:31
Hora del reloj: 23:31
Hora del reloj: 0:32

Hora del reloj12: 11:10 AM
Hora del reloj12: 12:11 AM
Hora del reloj12: 12:11 AM
Hora del reloj12: 1:12 PM

Hora exacta del reloj: 0:0:31
Hora exacta del reloj: 0:1:32
Hora exacta del reloj: 23:1:32
Hora exacta del reloj: 0:8:12

COMPROBRACION DE MOMENTOS EXACTOS
Los momentos exactos son: 1:2:3, 1:2:3 y 10:20:30
1:2:3 ES IGUAL A 1:2:3
1:2:3 NO ES IGUAL A 10:20:30
1:2:3 ES MENOR QUE 10:20:30
10:20:30 ES MAYOR QUE 1:2:3
1:2:3 ES IGUAL A 1:2:3
```

Trabajo autónomo

2. Implementa una nueva clase `Fraccion` **(Fraccion)** que implemente la interface `Comparable`. Para ello, utiliza la clase que ya tienes diseñada de la sesión 06 añadiéndole que implemente la interface anterior (deberás implementar el método `compareTo`*)*.



Implementar un programa **(TestFraccion)** que pruebe el funcionamiento de la clase anterior.

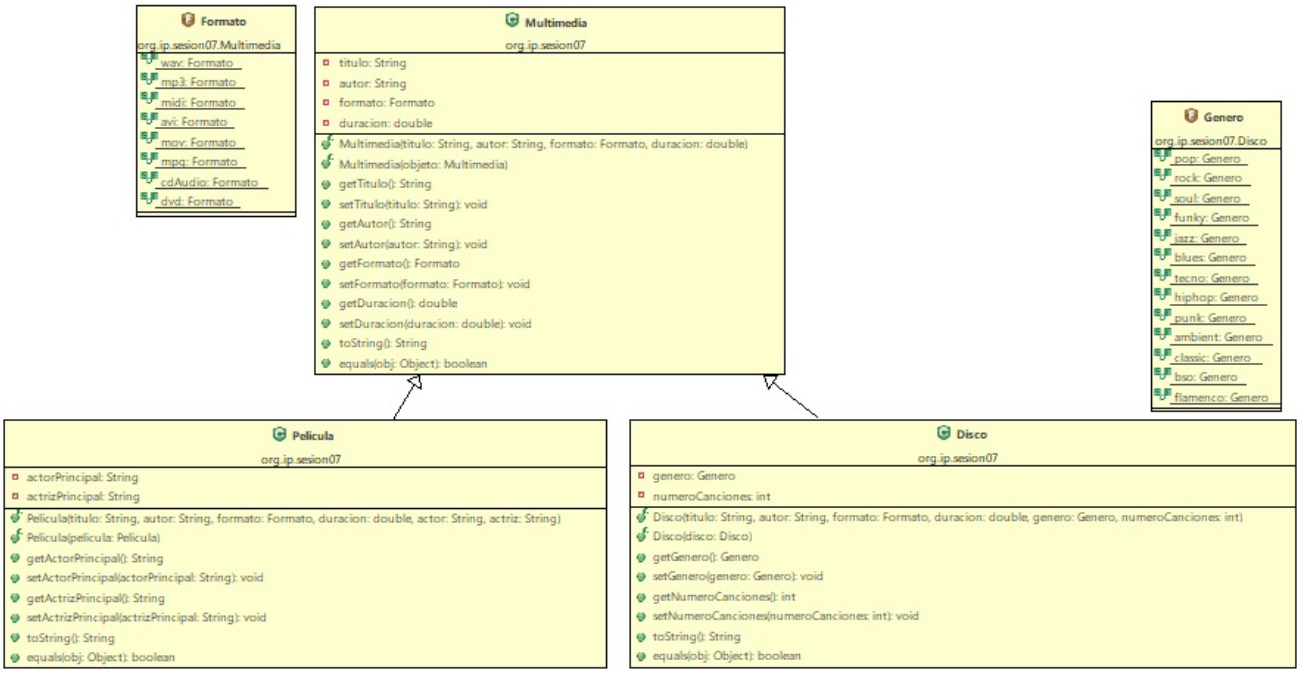
Ejemplo de ejecución:

```
Cuantas comparaciones de fracciones deseas realizar?
4
Comparacion 1
Introduce numerador y denominador de la primera fraccion
1 5
Introduce numerador y denominador de la segunda fraccion
1 5
1/5 es igual que 1/5

Comparacion 2
Introduce numerador y denominador de la primera fraccion
-1 2
Introduce numerador y denominador de la segunda fraccion
1 2
-1/2 es menor que 1/2

Comparacion 3
Introduce numerador y denominador de la primera fraccion
1 5
Introduce numerador y denominador de la segunda fraccion
1 8
1/5 es mayor que 1/8

Comparacion 4
Introduce numerador y denominador de la primera fraccion
1 8
Introduce numerador y denominador de la segunda fraccion
1 5
1/8 es menor que 1/5
```

3. Implementa una clase `Multimedia` **(Multimedia)** para almacenar objetos de tipo *multimedia* (películas, discos, mp3, etc.). Esta clase contiene atributos (`titulo`, `autor`, `formato` y `duracion`) tal y como se muestran en el diagrama de clases que se indica a continuación. Destacar que el atributo `formato` es un tipo enumerado (`enum`) y que puede almacenar uno de los siguientes valores: `wav`, `mp3`, `midi`, `avi`, `mov`, `mpg`, `cdAudio` y `dvd`. Esta clase tiene, además de constructores, *getters* y *setters*, un método `toString()` para devolver la información del objeto, y un método `equals()` que recibe un objeto de tipo `Multimedia` y devuelve `true` en caso de que el `titulo` y el `autor` sean iguales a los del objeto que llama al método y `false` en caso contrario.

A continuación, implemente una clase `Pelicula` **(Pelicula)** que herede de la clase `Multimedia`. La clase `Pelicula` debe tener, además de los atributos heredados, un actor principal (`actorPrincipal`) y una actriz principal (`actrizPrincipal`). Si la película no tuviera actor principal o actriz principal, ese campo debe de introducirse como la cadena vacía (“”). Esta clase debe tener los métodos que se indican en el diagrama de clases (constructores, *getters* y *setters*), destacando que se debe de sobrescribir el método `toString()` para que devuelva, además de los datos heredados, los nuevos datos, y también el método `equals()` que compararía si dos objetos de la clase `Pelicula` son iguales o no.

De la misma forma se pide implementar una clase `Disco` **(Disco)** que herede de la clase `Multimedia`. La clase `Disco` debe tener, además de los atributos heredados, un `genero` (es un tipo enumerado (`enum`) que puede tomar los siguientes valores: `pop`, `rock`, `soul`, `funky`, `jazz`, `blues`, `tecno`, `hiphop`, `punk`, `ambient`, `classic`, `bso` y `flamenco`) y un número de canciones (`numeroCanciones`). Esta clase debe tener los métodos que se indican en el diagrama de clases (constructores, *getters* y *setters*), destacando que se debe de sobrescribir el método `toString()` para que devuelva, además de los datos heredados, los nuevos datos, y también el método `equals()` que compararía si dos objetos de la clase `Disco` son iguales o no.



Implementar un programa **(TestMultimedia)** que permita probar las clases anteriores. A modo de ejemplo, se proporciona una salida que te permitirá comprobar si las operaciones que se han implementado en las diferentes clases son correctas. A modo de ejemplo, los objetos con los que trabajaría nuestro test serían los siguientes:

```
Pelicula("Million Dollar Baby", "Clint Eastwood", Formato.dvd, 137.30, "Clint Eastwood",
"Hillary Swank")
Pelicula("The Bridges of Madison County", "Clint Eastwood", Formato.dvd, 134.10, "Clint
Eastwood", "Meryl Streep");
Pelicula("Gladiator", "Ridley Scott", Formato.avi, 155.15, "Russell Crowe", "Connie
Nielsen");
Pelicula("The Lord of the Rings: The Fellowship of the Ring", "Peter Jackson",
Formato.dvd, 155.15, "Elijah Wood", "Liv Tyler");
Pelicula("Mar adentro", "Alejandro Amenabar", Formato.mpg, 125.05, "Javier Bardem",
"Belen Rueda");
Pelicula pelicula6 = new Pelicula(pelicula5);

Disco("Sin Mirar Atras", "David Bisbal", Formato.cdAudio, 46.02, Genero.pop, 13);
Disco("How to Dismantle an Atomic Bomb", "U2", Formato.cdAudio, 49.20, Genero.rock, 11);
Disco("Quitate las Gafas", "Melendi", Formato.mp3, 47.24, Genero.pop, 12);
Disco("Southside", "Texas", Formato.mp3, 44.53, Genero.rock, 11);
Disco("Soy Gitano", "Camaron", Formato.cdAudio, 32.21, Genero.flamenco, 8);
Disco disco6 = new Disco(disco5);
```

Ejemplo de ejecución:

```
Objeto Multimedia [titulo: The Lord of the Rings: The Fellowship of the Ring, de: Peter Jackson, con formato: dvd, duracion: 155.15 min.seg]
  Pelicula {protagonizada por: {Elijah Wood y Liv Tyler}

Objeto Multimedia [titulo: Million Dollar Baby, de: Clint Eastwood, con formato: dvd, duracion: 137.3 min.seg]
  Pelicula {protagonizada por: {Clint Eastwood y Hillary Swank}
 Y
Objeto Multimedia [titulo: The Bridges of Madison County, de: Clint Eastwood, con formato: dvd, duracion: 134.1 min.seg]
  Pelicula {protagonizada por: {Clint Eastwood y Meryl Streep}
SON DIFERENTES XXX

Objeto Multimedia [titulo: Mar adentro, de: Alejandro Amenabar, con formato: mpg, duracion: 125.05 min.seg]
  Pelicula {protagonizada por: {Javier Bardem y Belen Rueda}
 Y
Objeto Multimedia [titulo: Mar adentro, de: Alejandro Amenabar, con formato: mpg, duracion: 125.05 min.seg]
  Pelicula {protagonizada por: {Javier Bardem y Belen Rueda}
SON IGUALES ===

Objeto Multimedia [titulo: Southside, de: Texas, con formato: mp3, duracion: 44.53 min.seg]
  Disco {genero: rock y numero de canciones: 11}

Objeto Multimedia [titulo: How to Dismantle an Atomic Bomb, de: U2, con formato: cdAudio, duracion: 49.2 min.seg]
  Disco {genero: rock y numero de canciones: 11}
 Y
Objeto Multimedia [titulo: Quitate las Gafas, de: Melendi, con formato: mp3, duracion: 47.24 min.seg]
  Disco {genero: pop y numero de canciones: 12}
SON DIFERENTES XXX

Objeto Multimedia [titulo: Soy Gitano, de: Camaron, con formato: cdAudio, duracion: 32.21 min.seg]
  Disco {genero: flamenco y numero de canciones: 8}
 Y
Objeto Multimedia [titulo: Soy Gitano, de: Camaron, con formato: cdAudio, duracion: 32.21 min.seg]
  Disco {genero: flamenco y numero de canciones: 8}
SON IGUALES ===
```