Clase #4

Profesor Gonzalo Abel Benoffi
FRUTN - MDP - AÑO 2023

Previously on POO

- Programación **Orientada** a Objetos
- Fundamentos de Programación **Orientada** a Objetos
- ¿Qué es una clase?

La unión hace la fuerza

Ejemplo práctico: DoME

DoME* es una aplicación que nos permite almacenar información sobre discos compactos de música (en CD) y de películas (en DVD). La idea es crear un catálogo de todos los CD y DVD que tenemos, o todos los que hemos visto o escuchado.

*Database of Multimedia Entretainment (Base de Datos de Entretenimientos Multimediales).

Funcionalidades de DoME

- Debe permitirnos ingresar información sobre los CD y los DVD.
- Debe almacenar esta información de manera permanente, de tal modo que pueda ser usada más adelante.
- Debe brindar una función de búsqueda que nos permita por ejemplo, encontrar todos los CD de un cierto intérprete que hay en la base, o todos los DVD de determinado director.
- Debe permitirnos imprimir listados como por ejemplo: listado de todos los DVD que hay en la base o un listado de todos los CD de música.
- Debe permitirnos eliminar información.
- Debe permitir prestar o devolver elementos.


PARLOPHONE
EMI
THE BEATLES stereo
HELP!


A RON HOWARD FILM
EIGHT DAYS A WEEK
THE TOURING YEARS
THE BAND YOU KNOW. THE STORY YOU DON'T.
THE BEATLES

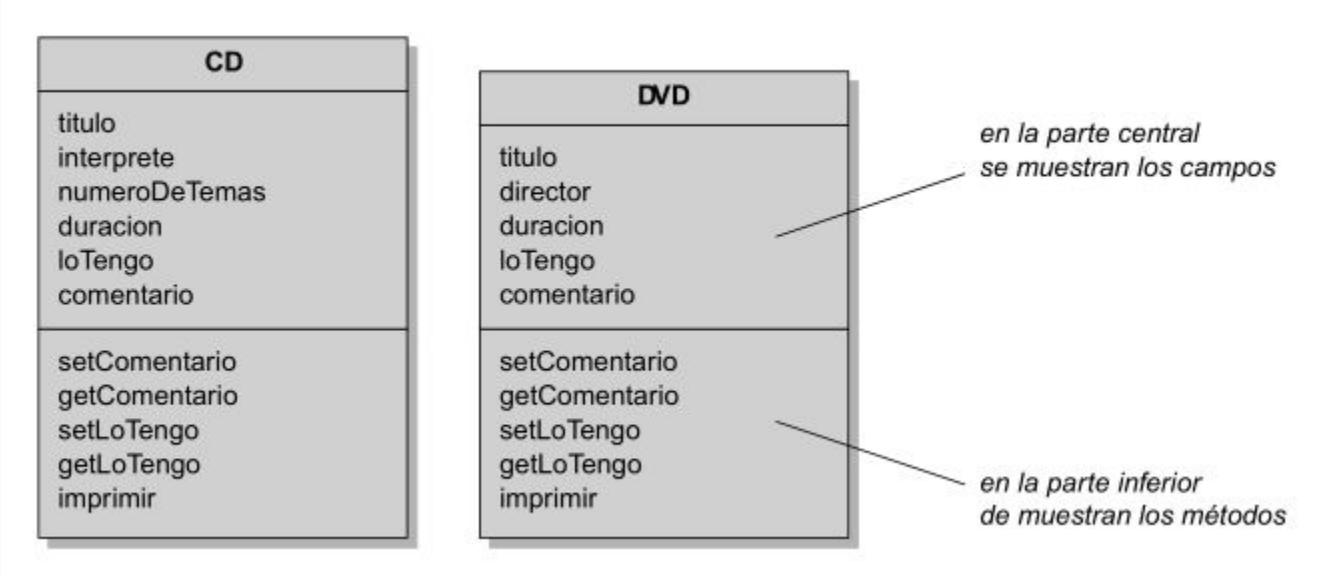
Información de DoME - CD

- El título del álbum
- El intérprete (el nombre de la banda o del cantante)
- El número de temas que tiene el CD
- El tiempo de duración del CD
- Una bandera que indique si tenemos una copia de este CD («lo tengo»)
- Un comentario (un texto arbitrario).

Información de DoME - DVD

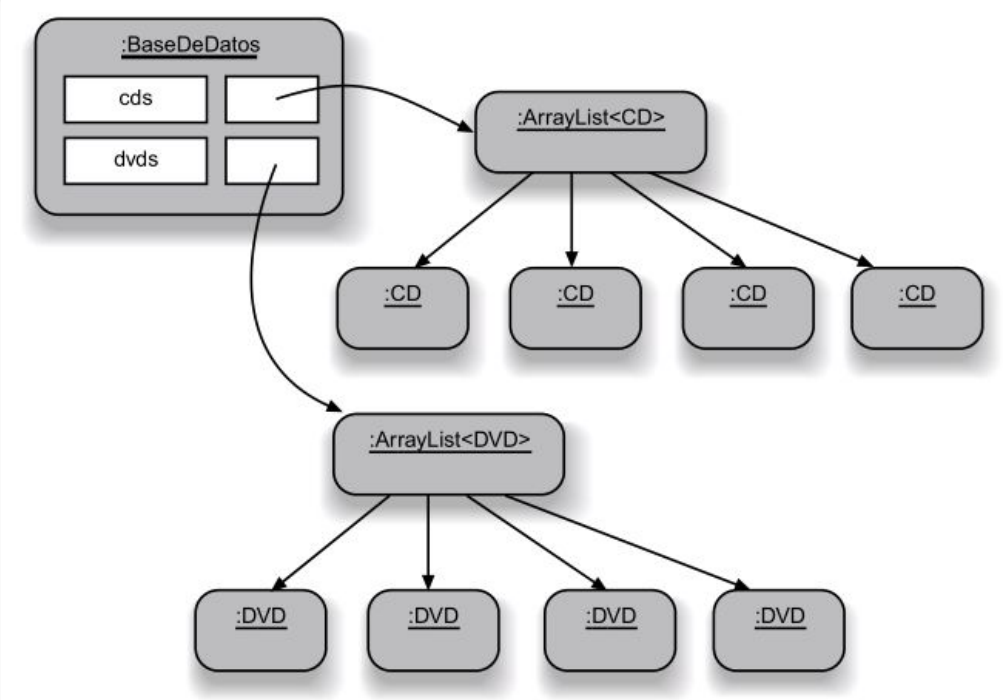
- El título del DVD
- El nombre del director
- El tiempo de duración
- Una bandera que indique si tenemos una copia de este DVD («lo tengo»)
- Un comentario (un texto arbitrario).

Modelo de las clases.



¿Algo más?

Modelo de clases v2



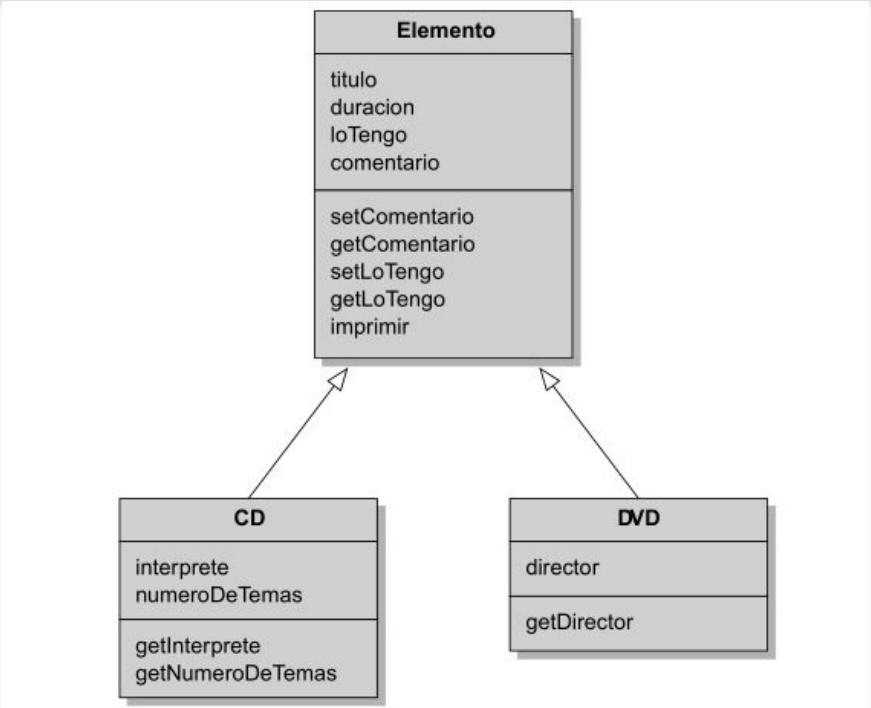
¿Problemas?

¿Problemas?

- Duplicación de código
- Modificación de atributos
- Nuevos atributos
- Nuevos elementos

Esta la Herencia y la Herencia...

Modelo de clases v3



Jerarquías de herencia

- La herencia puede usarse en forma mucho más general. Se pueden heredar más de dos subclases a partir de la misma superclase y una subclase puede convertirse en la superclase de otras subclases.
- El ejemplo más conocido de una jerarquía de herencia es la clasificación de las especies que usan los biólogos.
- La herencia es una técnica de abstracción que nos permite categorizar las clases de objetos bajo cierto criterio y nos ayuda a especificar las características de estas clases.

Herencia en JAVA

```
public class CD extends Elemento
{
    private String interprete;
    private int numeroDeTemas;
    // constructor de CD, getter-setter de sus atributos
}
```

Herencia y derechos de acceso

- Los miembros definidos como públicos, ya sea en la superclase o en la subclase, serán accesibles para los objetos de otras clases, pero los miembros definidos como privados serán inaccesibles
- La regla de la privacidad también se aplica en la herencia
  - Una subclase no puede acceder a los miembros privados de su superclase.
  - Si un método de una subclase necesita acceder o modificar campos privados de su superclase, entonces la superclase necesitará ofrecer los métodos de acceso y/o métodos de modificación apropiados.
  - Una subclase puede invocar a cualquier método público de su superclase como si fuera propio, no se necesita ninguna variable.

Herencia e inicialización

- La inicialización siempre depende de la clase/objeto en cuestión pero que pasa cuando hay herencia y por ende otros constructores?
- Invocamos al constructor de la clase padre en el constructor de las clases hijas mediante **super (parametro1, parametro2)**
- En Java, un constructor de una subclase siempre debe invocar en su primer sentencia al constructor de la superclase. Si no se escribe una llamada al constructor de una superclase, el compilador de Java insertará automáticamente una llamada a la superclase, para asegurar que los campos de la superclase se inicialicen adecuadamente.

Agregando otros tipos de elementos.

- Videojuego
- Libro
- Serie
- Temporada
- ????

Ventajas

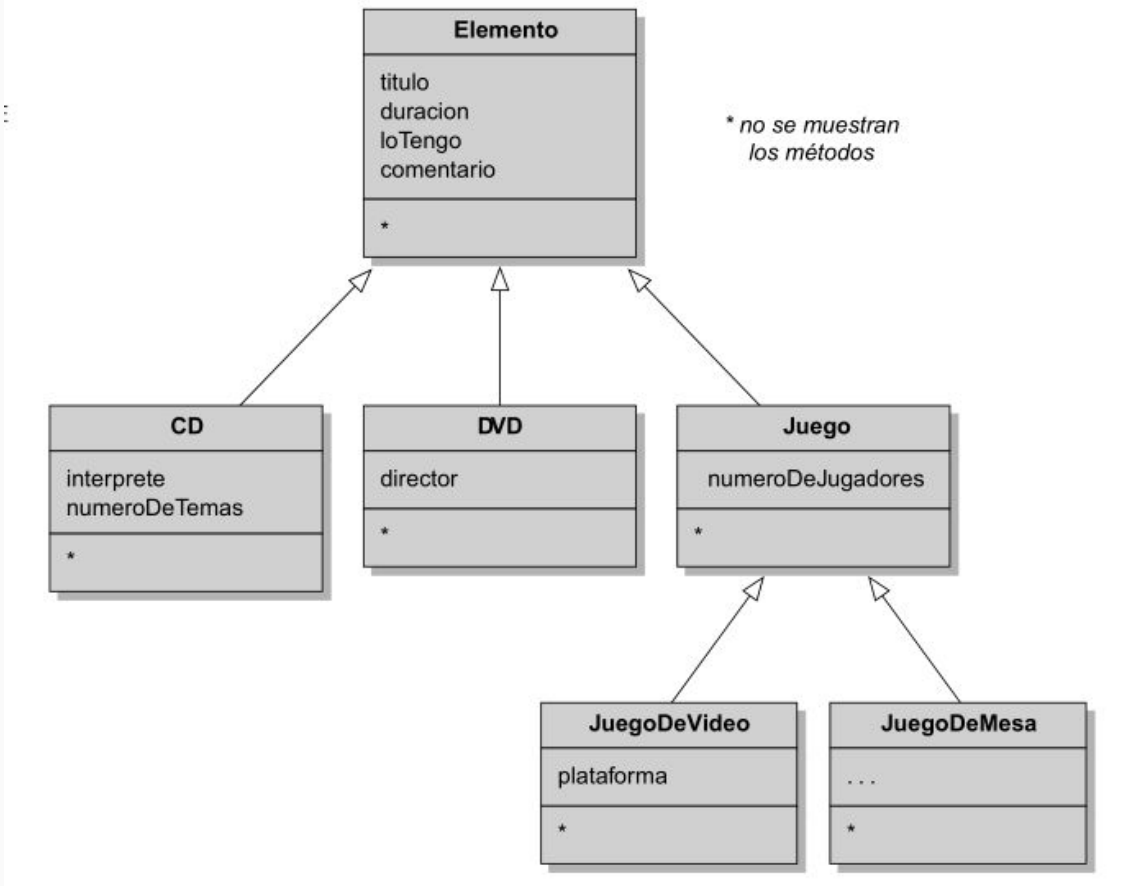
- Reutilizar el trabajo existente
- Facilidad de extender nuestro sistema
- Facilidad para modificar nuestro sistema
- +portable +flexible +sencillo
- Promueve modularización, compartimiento, privacidad

Creando más superclases

La clase **JuegoDeMesa**

- Una cuarta subclase de Elemento?
- Subclase de VideoJuego?
- ???

Modelo de Clases v4



Sustitución.

En los lenguajes orientados a objetos podemos sustituir por un objeto de una subclase en el lugar donde se espera un objeto de una superclase porque el objeto de la subclase es un caso especial de la superclase.

Mejorando la clase BaseDeDatos

```
Coche miCoche = new Coche(); // es válido!

Vehiculo v1 = new Vehiculo();  // es válido!

Vehiculo v2 = new Coche(); // es válido!

Vehiculo v3 = new Bicicleta(); // es válido!
```

Subtipos y pasaje de parámetros.

```
public void agregarElemento(Elemento elElemento)
{
    //agregar a una lista
}
```

Variables polimórficas

```
for(Elemento elemento : elementos)

{

    elemento.imprimir();

}
```

Enmascaramiento de tipos

```
Vehiculo v;

Coche a = new Coche();

v = a; // es correcta

a = v; // es un error

a = (Coche) v; // correcto
```

Enmascaramiento de tipos

```
Vehiculo v;

Coche a;

Bicicleta b;

a = new Coche();

v = new Coche(); // correcta

b = (Bicicleta) a; // ¡error en tiempo de compilación!

b = (Bicicleta) v; // ¡error en tiempo de ejecución!
```