Prueba 1 de Evaluación Continua

Análisis y Diseño de Software (2021/2022)

Entrega cada ejercicio en hojas separadas

Ejercicio 1 ***(4.5 puntos)***

Se quiere desarrollar una aplicación para músicos que permita incluir información sobre la discografía de artistas, y sirva como foro de discusión para la publicación de mensajes, estructurados en categorías musicales (p.ej., Heavy, Indie).

La aplicación considera grupos musicales y músicos individuales. Ambos tienen un nombre, pueden pertenecer a una o más categorías musicales, incluir una biografía, así como una discografía. Cada disco de la discografía tiene un nombre y año de publicación, y una o más canciones. Cada canción viene descrita por un nombre, una duración en segundos, y puede incluir opcionalmente un fichero de audio. Los grupos musicales tienen miembros, que son músicos, y un músico puede pertenecer a varios grupos, o a ninguno.

Los usuarios de la aplicación son músicos individuales, y acceden a ésta con un usuario y una contraseña. No obstante, la aplicación puede incluir información de músicos individuales que no tengan cuenta (p.e., Mozart). Un usuario puede añadir categorías musicales al foro de discusión, así como mensajes. Los mensajes deben incluirse en una categoría musical, y tienen un título, texto, y fecha de creación. Opcionalmente, pueden ser respuesta a otro mensaje, e incluir ficheros de audio como adjuntos. Un fichero de audio tiene un nombre, un tamaño en bytes, y su formato puede ser WAV, MP3 o FLAC. Por otro lado, las categorías musicales tienen un título y una fecha de creación. Una categoría musical (p.ej. heavy) puede incluir mensajes, así como otras subcategorías musicales (p.ej. black metal) de manera jerárquica.

Se pide:

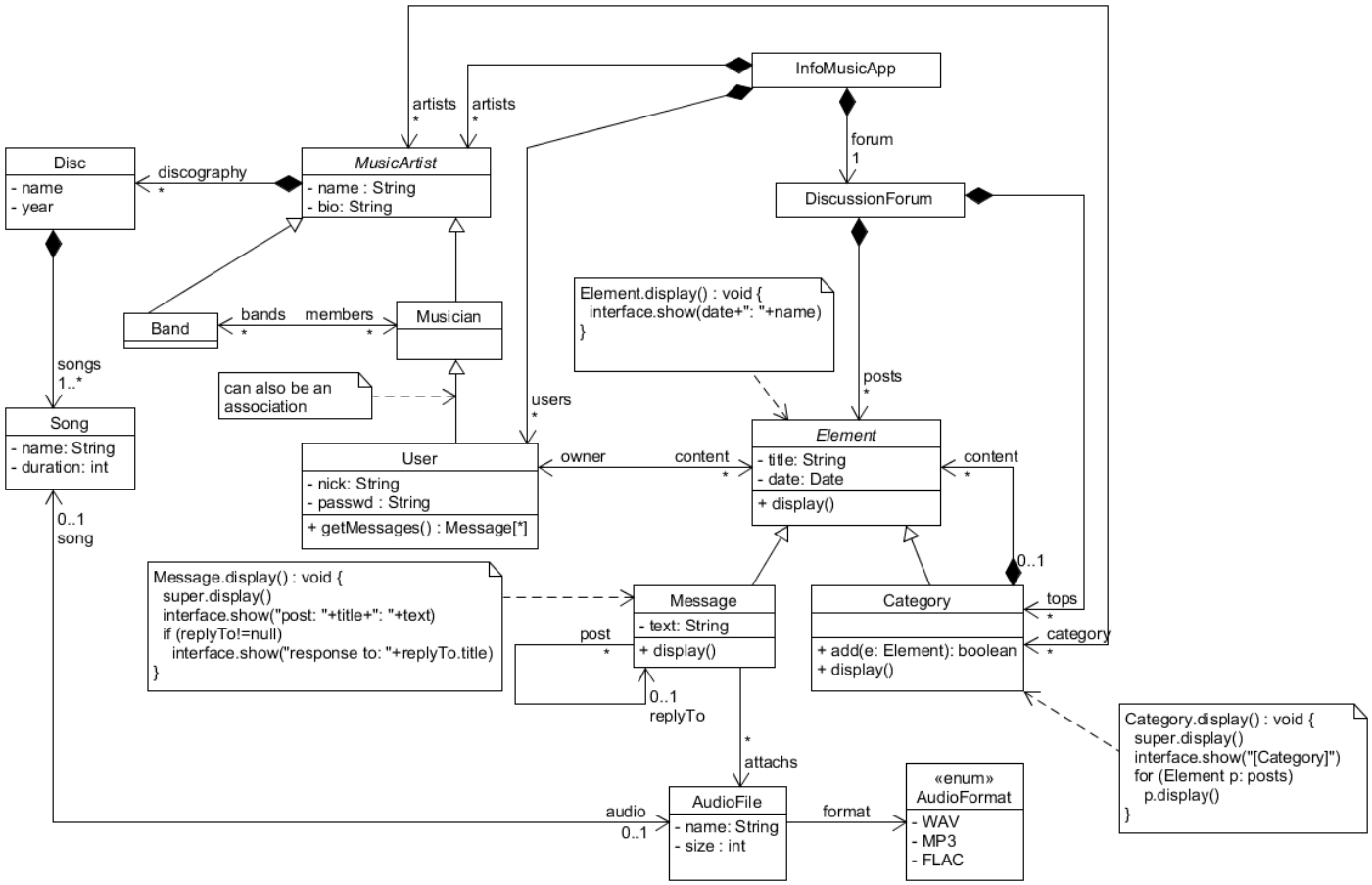
a) Realiza un diseño orientado a objetos que refleje los requisitos del sistema. Se valorará especialmente el uso de principios de calidad en el diseño orientado a objetos. No es necesario incluir métodos en este apartado. **(3.5 puntos)**

b) Incluye métodos en tu diagrama para (no es necesario pseudocódigo):

1) Añadir un mensaje o una categoría dentro de una categoría. **(0,25 puntos)**
2) Obtener todos los mensajes publicados por un usuario **(0,25 puntos)**

c) Incluye en el diagrama un método para visualizar toda la información relevante de una categoría dada (que debe incluir la información de categorías incluidas en ella directa o indirectamente). Incluye pseudocódigo (o puedes usar Java), así como métodos auxiliares, si son necesarios. **(0,5 puntos)**

Solución:

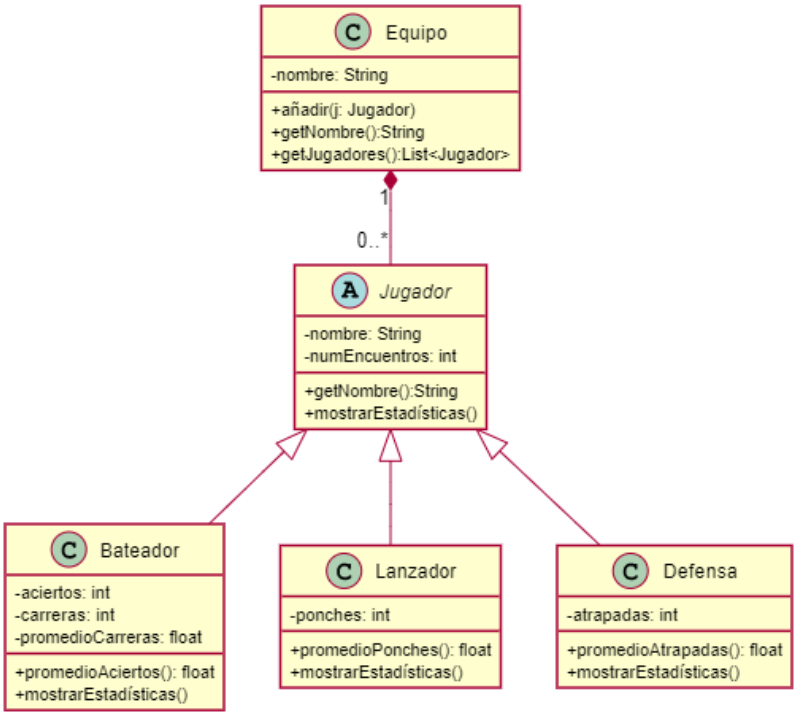

InfoMusicApp
artists
*
artists
*
forum
1
Disc
- name
- year
discography
*
MusicArtist
- name : String
- bio: String
DiscussionForum
Element.display() : void {
interface.show(date+": "+name)
}
Band
bands
*
members
*
Musician
songs
1..*
can also be an
association
users
*
posts
*
Song
- name: String
- duration: int
User
- nick: String
- passwd : String
+ getMessages() : Message[*]
owner
content
*
Element
- title: String
- date: Date
+ display()
content
*
0..1
song
Message.display() : void {
super.display()
interface.show("post: "+title+": "+text)
if (replyTo!=null)
interface.show("response to: "+replyTo.title)
}
Message
- text: String
+ display()
post
*
0..1
replyTo
Category
+ add(e: Element): boolean
+ display()
tops
*
category
*
Category.display() : void {
super.display()
interface.show("[Category]")
for (Element p: posts)
p.display()
}
*
attachs
audio
0..1
AudioFile
- name: String
- size : int
format
«enum»
AudioFormat
- WAV
- MP3
- FLAC

Prueba 1 de Evaluación Continua
Análisis y Diseño de Software (2021/2022)

Entrega cada ejercicio en hojas separadas

Ejercicio 2 ***(2.0 puntos)***

El diagrama de clases de la figura muestra un diseño para el cálculo de estadísticas de los jugadores (bateadores, pitchers y defensas) de los equipos de las Grandes Ligas de Béisbol.



Analiza el diagrama y contesta a las siguientes cuestiones:

1. (0.5 ptos) Para cada una de las siguientes líneas de código, indique si son correctas o incorrectas y señale por qué.
   a) (0.25 ptos) `Jugador j1 = new Bateador();`
   b) (0.25 ptos) `Lanzador j2 = new Jugador();`

   Solución:
   a) CORRECTA: se instancia un objeto de clase "Bateador" y se le asigna a una referencia a un objeto de clase Jugador (clase padre).
   c) INCORRECTA: la clase Jugador es abstracta.

2. (1.0 ptos) Partiendo del siguiente fragmento de código Java para instanciar objetos de diferentes clases:

```
Jugador j1 = new Lanzador();
Bateador j2 = new Bateador();
Defensa j3 = new Defensa();
Jugador j4 = new Bateador();
```

   Para cada una de las siguientes asignaciones, indique si es correcta o incorrecta y por qué.
   a) (0.25 ptos) j2 = j1; // INCORRECTO: Una referencia a Jugador no puede considerarse Bateador.
   b) (0.25 ptos) j1 = j2; // CORRECTO: Una referencia a Bateador es un Jugador
   c) (0.25 ptos) j2 = j4; // INCORRECTO: Una referencia a Jugador no puede ser Bateador
   d) (0.25 ptos) j1 = j3; // CORRECTO: Una referencia a Defensa es un Jugador

3. (0.25 ptos) Teniendo en cuenta que Java emplea ligadura dinámica (o linkado dinámico), ¿a qué clase pertenece el método que se ejecuta en el siguiente fragmento de código y por qué?

```
Bateador j1 = new Bateador();
Jugador j2 = j1;
j2.mostrarEstadísticas();
```

Solución: Se ejecuta el método mostrarEstadísticas() de la clase "Bateador" porque el linkado dinámico hace que al asignar j1 a j2, la referencia j2 apunte al área de memoria reservada para el objeto de clase "Bateador".

4. (0,25 ptos) Señale cuál de los dos siguientes fragmentos de código es correcto y cuál incorrecto y por qué.

a)
```
Jugador j1 = new Bateador();
j1.promedioAtrapadas();
```

b)
```
Lanzador j1 = new Lanzador();
j1.getNombre();
```

Solución:

a) No es correcto porque la referencia j1 es de tipo "Bateador" y no conoce el método "promedioAtrapadas".
b) Es correcto porque se está empleando la implementación que se encuentra en la clase padre "Jugador".

Prueba 1 de Evaluación Continua

Análisis y Diseño de Software (2021/2022)

Entrega cada ejercicio en hojas separadas

Ejercicio 3 ***(3 puntos)***

Se quiere desarrollar un software controlador del tráfico de los aviones de pasajeros que llegan y salen de un aeropuerto. Para ello, los aviones se dan de alta y baja del controlador según corresponda.

Durante el control de un avión, éste en todo momento ha de tener una localización (longitud, latitud, altitud) y uno de los siguientes estados: embarcando, desembarcando, en puerta, en pista, en espera, y en vuelo.

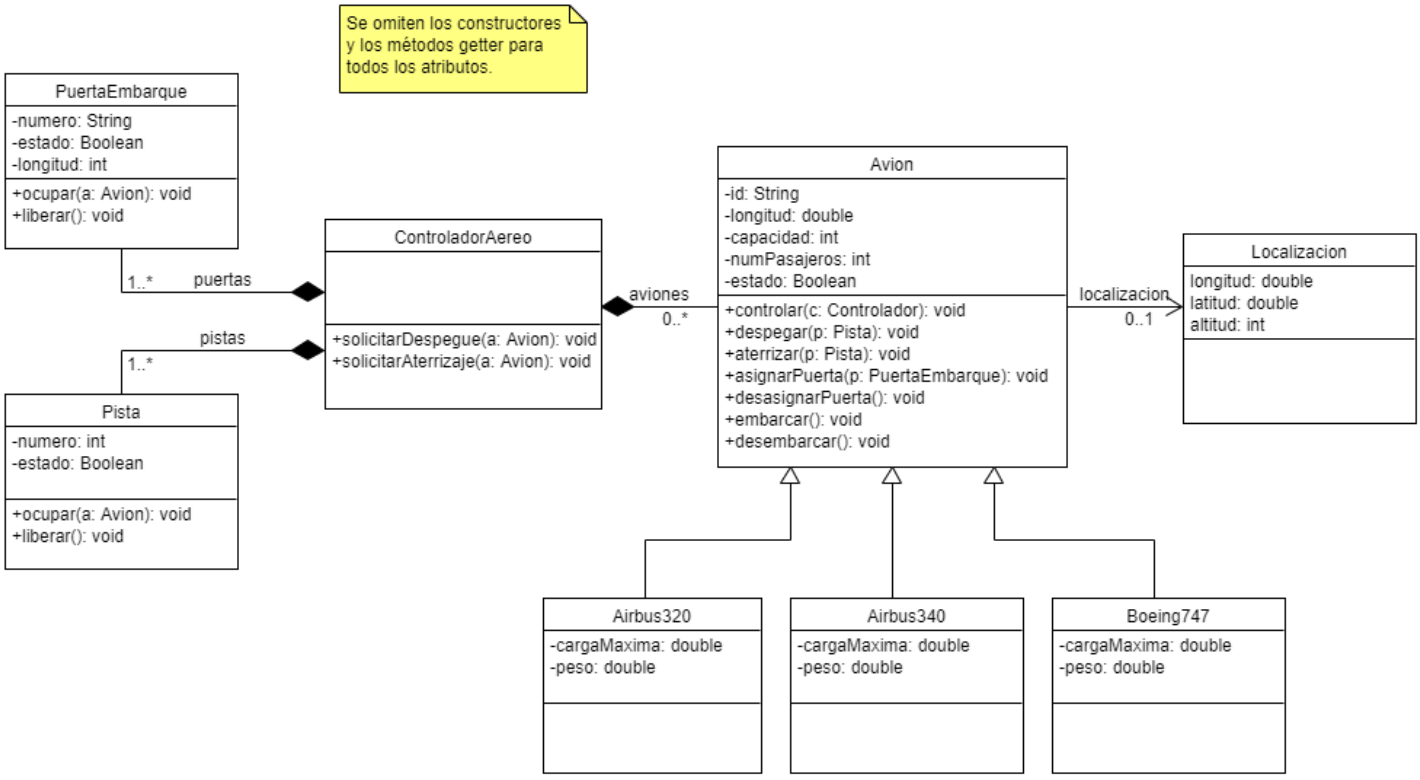
Los aviones (caracterizados por un id, longitud, capacidad y número de pasajeros, carga máxima y peso) solicitan al controlador despegues y aterrizajes. Atendiendo al estado (disponible o no disponible) de pistas y puertas de embarque, cuando lo considere oportuno, el controlador da permiso a los aviones para despegar y aterrizar en una pista, o les asigna y desasigna una puerta de embarque.

El controlador gestiona 3 modelos de avión: Airbus 320, Airbus 340 y Boeing 747. Todo modelo de avión tiene unos protocolos de embarque y desembarque particulares. Además, algunos modelos, como el Airbus340 y el Boeing 747, tienen un protocolo interno (gestionado desde cabina) de apertura y cierre de bahías para carga-descarga de mercancías.

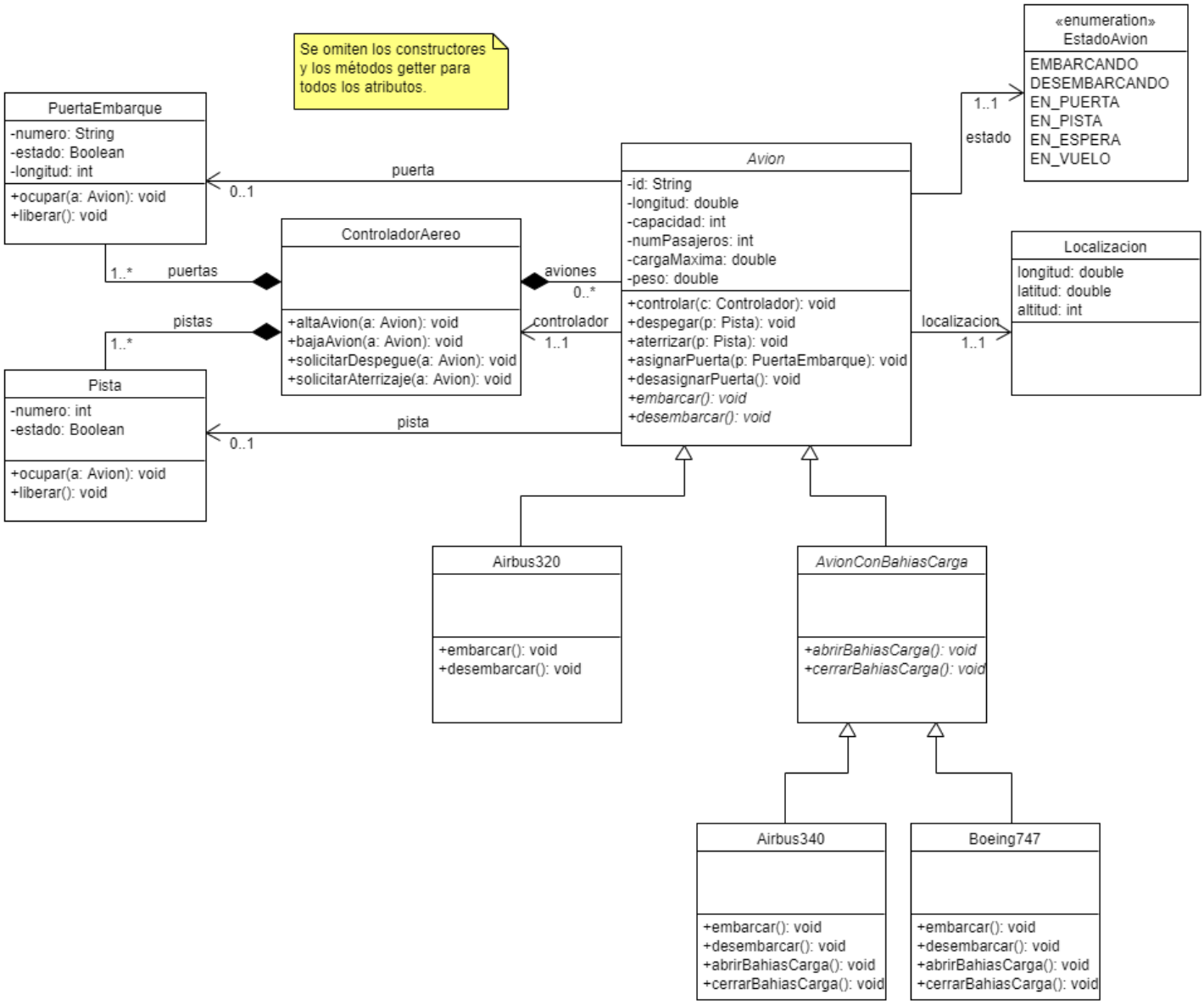
Atendiendo a estos requisitos, se ha realizado el diagrama de clases de abajo.

Se pide:

Modificar y completar si es necesario el diagrama para que refleje un buen diseño orientado a objetos. No hace falta dibujar el nuevo diagrama entero; sólo con aquellas partes que hayan sufrido cambios.



Solución:



Cambios realizados en el diseño del enunciado:

1. Se añade la enumeración `EstadoAvion`, y se añade el atributo `estado` de `Avion` por una asociación de cardinalidad 1..1 con `EstadoAvion`.
2. Se hace abstracta la clase `Avion`.
3. Se hacen abstractos los métodos `embarcar()` y `desembarcar()` en la clase `Avion`, y se añaden a sus subclases `Airbus320`, `Airbus340` y `Boeing747`.
4. Se declara la clase abstracta `AvionConBahiasCarga` con los métodos `abrirBahiasCarga()` y `cerrarBahiasCarga()` abstractos.
5. Las clases `Airbus340` y `Boeing747` heredan de `AvionConBahiasCarga` y sobreescriben/especializan los métodos `abrirBahiasCarga()` y `cerrarBahiasCarga()`.
6. Se añade una asociación de cardinalidad 1..1 entre `Avion` y `ControladorAereo`.
7. Se añaden los métodos `altaAvion(a: Avion)` y `bajaAvion(a: Avion)` en `ControladorAereo`.
8. Se cambia la cardinalidad de `localizacion` a 1..1.
9. Se añaden dos asociaciones `puerta` y `pista` de cardinalidad 0..1 en la clase `Avion`.
10. Se mueven los atributos `cargaMaxima` y `peso` a la clase `Avion` desde sus subclases.

Cada cambio se califica con 0.3 puntos.