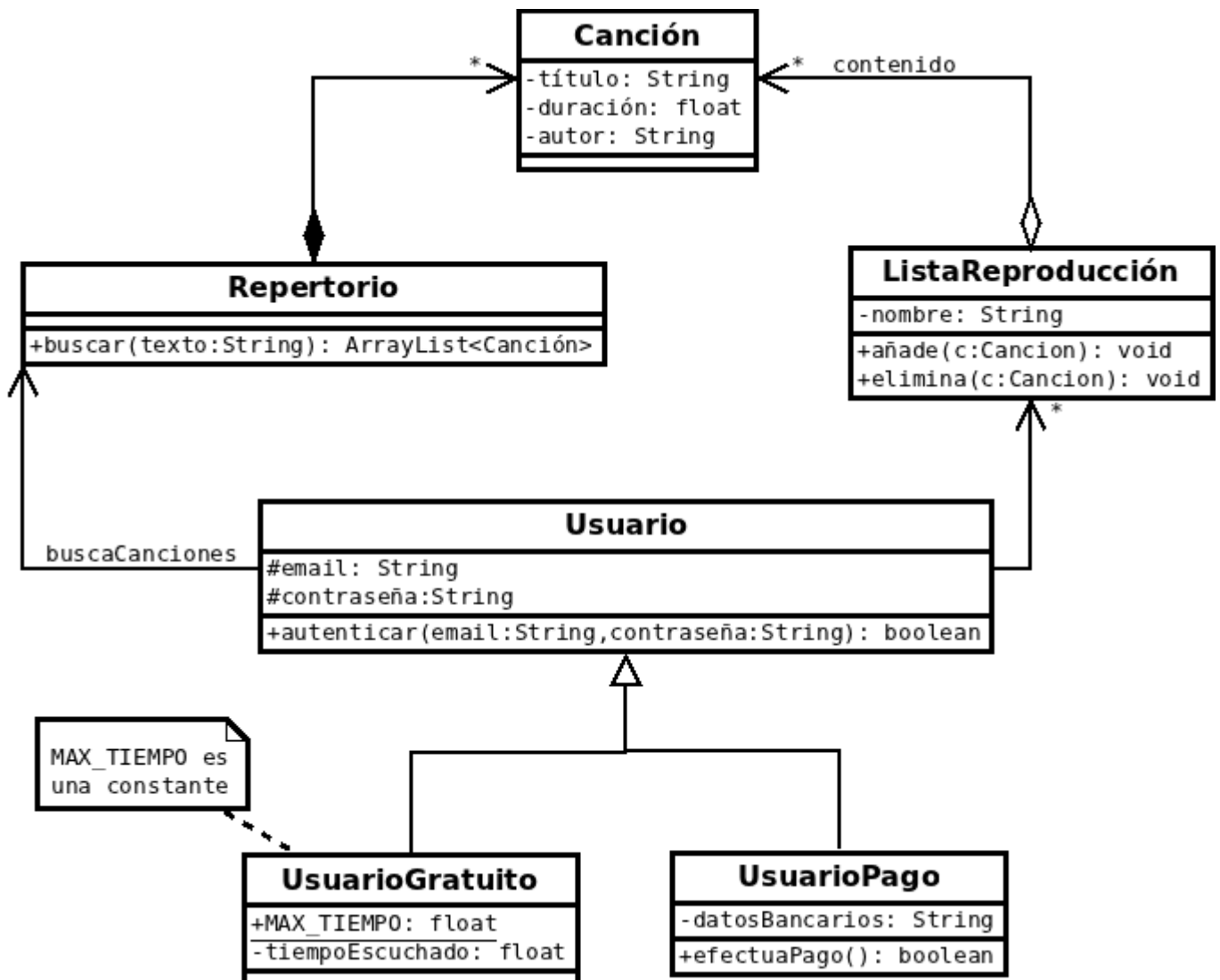


El reproductor de música

Estamos creando un reproductor de música online (a semejanza de otros como Spotify, Grooveshark...). Esta aplicación está formada por los siguientes componentes:

- Una clase **Usuario**, que representa a los usuarios que pueden acceder a la aplicación. Los usuarios se identifican mediante su dirección de email y su contraseña. Hay dos tipos de usuario:
 - Los **Usuarios Gratuitos**, que no pagan por el servicio pero pueden escuchar música por un tiempo limitado (definido por la constante MAX_TIEMPO)
 - Los **Usuarios de Pago**, para los cuales se deben guardar los datos bancarios.
- La clase **ListaReproducción** guarda las listas de reproducción que los usuarios pueden crear, y a las cuales se puede dar un nombre, y añadir o eliminar **Canciones**.
- El usuario puede buscar nuevas canciones en un **Repertorio** a partir de un texto de búsqueda.
- Las canciones se identifican según el título, la duración y el autor/intérprete.

A continuación se muestra el diagrama de clases UML de la aplicación:



Preguntas y Ejercicios

1. (2 PUNTOS) Responde a las siguientes preguntas:

- A diferencia de la clase `Usuario`, en la clase `UsuarioPago` no se definen explícitamente atributos para guardar la dirección de correo electrónico, ni la contraseña, ni el método `autenticar`. ¿Significa eso que los usuarios de pago no necesitan autenticarse para entrar en el sistema?
- Según el diagrama de clases UML, ¿Una lista de reproducción puede estar vacía? ¿Y el repertorio?
- La relación entre la clase `Repertorio` y la clase `Canción` tiene un aspecto ligeramente diferente a la relación entre la clase `ListaReproducción` y la clase `Canción`. ¿Qué tipo de relación es cada una de ellas? ¿En qué se diferencian la una de la otra?
- Según el diagrama de clases UML, dada una instancia de `ListaReproducción`, ¿se puede conocer directamente a qué usuario pertenece? ¿Por qué?

2. (2,5 PUNTOS) Para todas las clases del UML, escribe el código Java correspondiente a la definición de las clases, sus atributos y la implementación de las relaciones entre dichas clases. **No** es necesario que escribas cabeceras ni código alguno, ni de los constructores ni de ningún método (de momento).

3. (1 PUNTO) El único constructor de la clase `Usuario` tiene la siguiente cabecera:

```
public Usuario(String mail, String password)
```

Teniendo esto en cuenta, escribid el código del constructor de la clase `UsuarioPago`.

4. (1 PUNTO) El método de la clase `Usuario` cuya notación UML es:

```
+autenticar(email:String, contraseña:String):boolean
```

comprueba si el email y la contraseña que se le pasan por parámetros coinciden con el email y la contraseña guardados como atributos del usuario, y devuelve true en caso de que coincidan (es decir, la contraseña es correcta y por tanto el usuario puede entrar en el sistema); y false en caso contrario. Escribe el mencionado método en lenguaje Java.

5. (2,5 PUNTOS) El método de la clase `Repertorio` cuya notación UML es:

```
+buscar(texto:String):ArrayList<Canción>
```

busca el texto que se le pasa por parámetro en todas las canciones del repertorio, y devuelve una lista con todas las canciones para las cuales el texto está contenido en el título de la canción o con el autor. Escribe el mencionado método en lenguaje Java. Puedes asumir que los `get` y `set` para los atributos de la clase `Canción` ya están creados.

AYUDA: el siguiente método de la clase `String`:

```
public int indexOf(String str)
```

devuelve un número igual o mayor que 0 si el texto del parámetro "str" está contenido en el `String` sobre el cual se llama, o -1 si no se encuentra. Ejemplo:

```
String texto = "Gatos y perros";  
int a = texto.indexOf("blabla"); // devuelve -1  
int b = texto.indexOf("atos");   // devuelve un número mayor o igual que 0
```

6. (1 PUNTO) Implementa los métodos `toString()` de las clases `Canción` y `ListaReproducción`. Recordad que la cabecera de esos métodos es:

```
public String toString();
```

Solución

1. a) Sí es necesario que UsuarioPago también se autentifique en el sistema. La dirección de correo electrónico, la contraseña y el método autenticar no se definen en UsuarioPago, pero son heredados desde la superclase Usuario y por tanto, UsuarioPago también tiene los mencionados atributos y método.
b) El asterisco en ambas relaciones indican una cardinalidad de 0 o varios. Por tanto, tanto las listas de reproducción como el repertorio podrían estar vacíos.
c) Entre las clases ListaReproducción y Canción hay una relación de agregación. Esto significa que las canciones forman parte de la lista de reproducción. Si una lista de reproducción se elimina, las canciones podrían seguir formando parte de otras listas de reproducción.
Entre las clases Repertorio y Canción hay una relación de composición (o agregación fuerte). Esto también significa que las canciones forman parte del repertorio, pero si el repertorio se elimina, las canciones contenidas en él también serán eliminadas (puesto que no tienen sentido fuera del repertorio).
d) No se puede saber, ya que la navegabilidad es unidireccional de Usuario a ListaReproducción. Dado un usuario, podemos saber cuáles son sus listas de reproducción, pero no a la inversa.

```
2. public class Usuario {
    protected String email, contraseña;
    private Repertorio repertorio;
    private List<ListaReproducción> listas;
}
public class UsuarioGratuito extends Usuario {
    public static final float MAX_TIEMPO = 30; //o cualquier otro valor
    private float tiempoEscuchado;
}
public class UsuarioPago extends Usuario {
    private String datosBancarios;
}
public class ListaReproducción {
    private String nombre;
    private List<Canción> contenido;
}
public class Repertorio {
    private List<Canción> repertorioCanciones;
}
public class Canción {
    private String autor, título;
    private float duración;
}

3. public UsuarioPago(String mail, String password) {
    super(mail, password);
}
```

Solución también válida:

```
public UsuarioPago(String mail, String password, String datosBanco) {
    super(mail, password);
    this.datosBancarios = datosBanco;
}
```

```

4. public boolean autenticar(String email, String contraseña) {
    if(this.email.equals(email) && this.contraseña.equals(contraseña)) {
        return true;
    } else {
        return false;
    }
}

5. public ArrayList<Canción> buscar(String texto) {
    ArrayList<Canción> coincidentes = new ArrayList<Canción>();
    Iterator<Canción> it = repertorio.iterator();
    while(it.hasNext()) {
        Cancion c = it.next();
        if(c.getAutor().indexOf(texto) >=0 || c.getTitulo().indexOf(texto) >=0) {
            coincidentes.add(c);
        }
    }
    return coincidentes;
}

6. toString de la clase Canción:
public String toString() {
    return autor + " - " + título;
}

toString de la clase ListaReproducción:
public String toString() {
    StringBuilder ret = new StringBuilder(nombre).append(":\n");
    Iterator<Canción> it = contenido.iterator();
    while(it.hasNext()) {
        ret.append(it.next().toString()).append("\n");
    }
    return ret.toString();
}

Solución también válida para la clase ListaReproducción:
public String toString() {
    String ret = nombre + ":\n";
    Iterator<Canción> it = contenido.iterator();
    while(it.hasNext()) {
        ret += it.next().toString() + "\n";
    }
    return ret;
}

```