INGENIERIA INFORMATICA
Escuela Politécnica Superior
Universidad Autónoma De Madrid

# Apartado 3

## Práctica 2

David Teófilo Garitagoitia Romero-Javier Fernandez de Alarcon Gervas

**3/8/2021**

# Tabla de contenido

# PLANTEAMIENTO

Lo primero es ver que clases vamos a requerir, según dice el enunciado parece evidente que va a ser necesario una clase reunion (abstracta), también será necesaria la clase sala (abstracta), Sistema, usuario (abstracta) de la que heredan usuario registrado y sin registrar

Se omiten constructores y métodos get y set.

# CLASES

## 1. Usuario

### Atributos

-salas: Las salas en las que pudo entrar el usuario

-reuniones: las reuniones en las que pudo entrar el usuario

### Métodos

-entrarSala(s:Sala): Tanto los usuarios no registrados como los registrados pueden entrar a salas por lo que es un método común, se pasa por argumento la sala a la que se desea entrar, dentro de la sala se llama al método comprobarEntradaUsuario(u:Usuario) que se encargará de comprobar que el usuario no se encuentra entre los usuarios bloqueados de la sala, en caso de entrar se a devuelve un booleano en función de sip udo entrar o no.

-entrarReunionSala(r:Reunion): Método similar al anterior, dentro de la reunion, llama al método comprobarEntrada para verificar que el usuario pueda entrar en la reunión.

- addMensaje(s:Sala, t:String, p:boolean): Método para añadir un mensaje en una sala de chat

- addVideo(s:sala, n:Nombre, fi: Date, ff:Date): Método para añadir video en una sala de video

## 2. UsuarioRegistrado

Debe ser una clase abstracta y será padre de los distintos tipos de preguntas.

### Atributos

-nombre: Nombre del usuario

-email: Email del usuario

-password: Contraseña del usuario

### Métodos

-crearSala(s: String, p:Boolean, t:TipoSala): Método para que el usuario registrado pueda crear una sala, este, en función de TipoSala, dentro de la función se piden las variables restantes y después llama al constructor de sala pasándole los argumentos de crearSala y añade esa nueva sala a las salas creadas por el usuario

-prohibirEntrada(s: Sala, u: Usuario): método para prohibir la entrada del usuario u en la sala s

-+programarReunion(n:String, f:Date, d:double, s:Sala, t:TipoReunion): método para añadir una nueva reunion a la sala s del usuario, llamará al constructor de reunión solicitando los argumentos faltantes y despues llamará al método addReunion de la sala, que comprobará que ambos sean compatibles, en caso contrario, eliminará la reunión y devolverá false para indicar que no se ha podido completer la acción satisfactoriamente.

### 3. UsuarioSinRegistrar

#### Atributos

-nick

### 4. Salas

#### Atributos

-nombre

-isPrivate; Booleano que define si la sala es privada o pública

-usuariosBloqueados: Lista de usuarios que no pueden entrar en la sala

#### Métodos

-addReunion(r:Reunion): método para añadir una reunión a la sala, comprobará el tipo de reunion, y si es compatible, la añadirá al conjunto de reuniones de la sala

### 5. Mixta

#### Atributos

-subSalas: sub salas contenidas en la sala mixta

### 6. Chat

#### Atributos

-moderada: Booleano para saber si la sala es moderada

-idioma: enum para indicar el idioma de la sala

-mensajes: conjunto de mensajes contenidos en el chat

### 7. Videoconferencia

#### Atributos

-imagen: Imagen opcional de fondo durante las video llamadas

### 8. Mensajes

#### Atributos

fecha: Fecha en la que se envió el mensaje

texto: contenido del mensaje

## 9. Video

### Atributos

-nombre

-fechaIni

-fechaFin

## 10. MensajePrivado

### Atributos

-usuariosPermitidos: Lista de usuarios que pueden ver el mensaje

## 11. Reunion

### Atributos

-nombre

-fechaInicio

-duracion

### Métodos

-comprobarEntrada(u:Usuario): método para comprobar si un usuario puede entrar en esa reunión.

# CÓDIGO

## Método 1

```
public boolean comprobarEntradaReunion(Usuario user) {
        if(usuariosBloqueados.contains(user)) {
                return false;
        } else {
                return true;
        }
}
```

## Método 2

```
public Usuario usuariosProhibidos(Sala sala) {
        Usuario bloqueados[];

        for Sala s : getSalasInteriores(sala) {
                bloqueados += s.usuariosBloqueados;
        }
        return bloqueados;
}
```

En este método se emplea otro método auxiliar getSalasInteriores() para obtener la lista de salas que están dentro de la sala pasada por argumento. En caso de no tener salas mixtas se devuelve la propia sala.

[FINAL DE DOCUMENTO]