Ejercicio 6

1. Requisitos funcionales
    a. Identificación del id de la conferencia
    b. Contraseña de la conferencia
    c. Usuarios y dos tipos de usuarios, normales y anfitriones.
    d. Los anfitriones pueden iniciar o terminar la conferencia.
    e. En caso de que no inicie la conferencia los usuarios normales no se pueden conectar.
    f. Varios usuarios conectados a la conferencia.
    g. Los usuarios pueden elegir iniciar su cámara o su micrófono, pueden chatear, por privado y en público, el privado se muestra en la opción de mensajes privados y pueden conectarse y desconectarse de la conferencia.
2. Análisis
    a. Main: sirve para correr el programa y hacer de vista en el MVC.
        i. Atributos:
            1. No aplica
        ii. Métodos:
            1. Main: sirve para correr el programa, es público y es estático.
    b. Controller: sirve como controlador del programa en el MVC.
        i. Atributos:
            1. Users: ArrayList<User> que almacena los usuarios de las meetings, privado.
            2. Meetings: ArrayList<Meeting> que almacena las meetings, privado.
        ii. Métodos:
            1. getUsers: retorna un ArrayList<User>, sin argumentos, público.
            2. setUsers: retorna void, pide un ArrayList<User>, público.
            3. getMeetings: retorna un ArrayList<Meeting>, público.
            4. setMeetings: retorna void, pide un ArrayList<Meeting>, público.
            5. showUOptions: retorna un string, sin argumentos, público.
            6. showAOptions: retorna un string, sin argumentos, público.
    c. Meeting:
        i. Atributos:
            1. Id, entero, público, es público debido a que cualquiera puede acceder a la conferencia, es el id de la meeting.
            2. Password, entero, es privado, es la password de la meeting.
            3. HostId, entero, privado, es el id del host de la meeting.
            4. UsersId ,ArrayList<Integer>, son los ids’s de los que participan en la meeting.
        ii. Métodos: Se debe denotar que tanto el getId como setId no es necesario dado que es completamente público.

1. getPassword: retorna un entero, sin argumentos, es público y retorna la password de la meeting.
2. setPasssword: pide un entero, es público y retorna void.
3. getHostId: retorna un entero, sin argumentos, es público y retorna el id del host.
4. setHostId: pide un entero, es público y retorna void.
5. getUsersId: retorna un ArrayLitst<Integer> con los id's de los usuarios, sin argumentos, es público.
6. setUsersId: pide un ArrayList<Integer> con los id's de los usuarios, retorna void, público.

d. User: interfaz que permite definir los métodos generales de un usuario.
   i. Atributos: No aplica
   ii. Métodos:
      1. getName: retorna un string con el nombre, sin argumentos, público.
      2. setName: retorna void, pide un string con el nombre, público.
      3. getId: retorna un entero con el id del usuario, sin argumentos, público.
      4. setId: retorna un void, pide un entero con el id del usuario, público.
      5. getDevice: retorna un device, sin argumentos, público.
      6. setDevice: retorna void, pide un device, público.
      7. getMicro: retorna booleano, sin argumentos, público.
      8. setMicro: retorna void, pide un booleano, público.
      9. getCamera: retorna booleano, sin argumentos, público.
      10. setCamera: retorna void, pide un booleano, público.
      11. isInMeeting: retorna booleano, sin argumentos, público.
      12. setMeeting: retorna void, pide un booleano, público.

e. NormalUser: clase que implementa User y que es el caso en que sea un usuario normal.
   i. Atributos:
      1. Device: es un device, privado.
      2. Name: es un string con el nombre, privado.
      3. Id: es un entero con el id del usuario, privado.
      4. Micro: es un booleano con el micrófono apagado o encendido, privado.
      5. Camera: es un booleano con la cámara encendida o apagada, privado.
      6. InMeeting: es un booleano sí está o no en una reunión, privado.
   ii. Métodos: Los métodos son overrides de la interfaz user entonces no los escribiré.

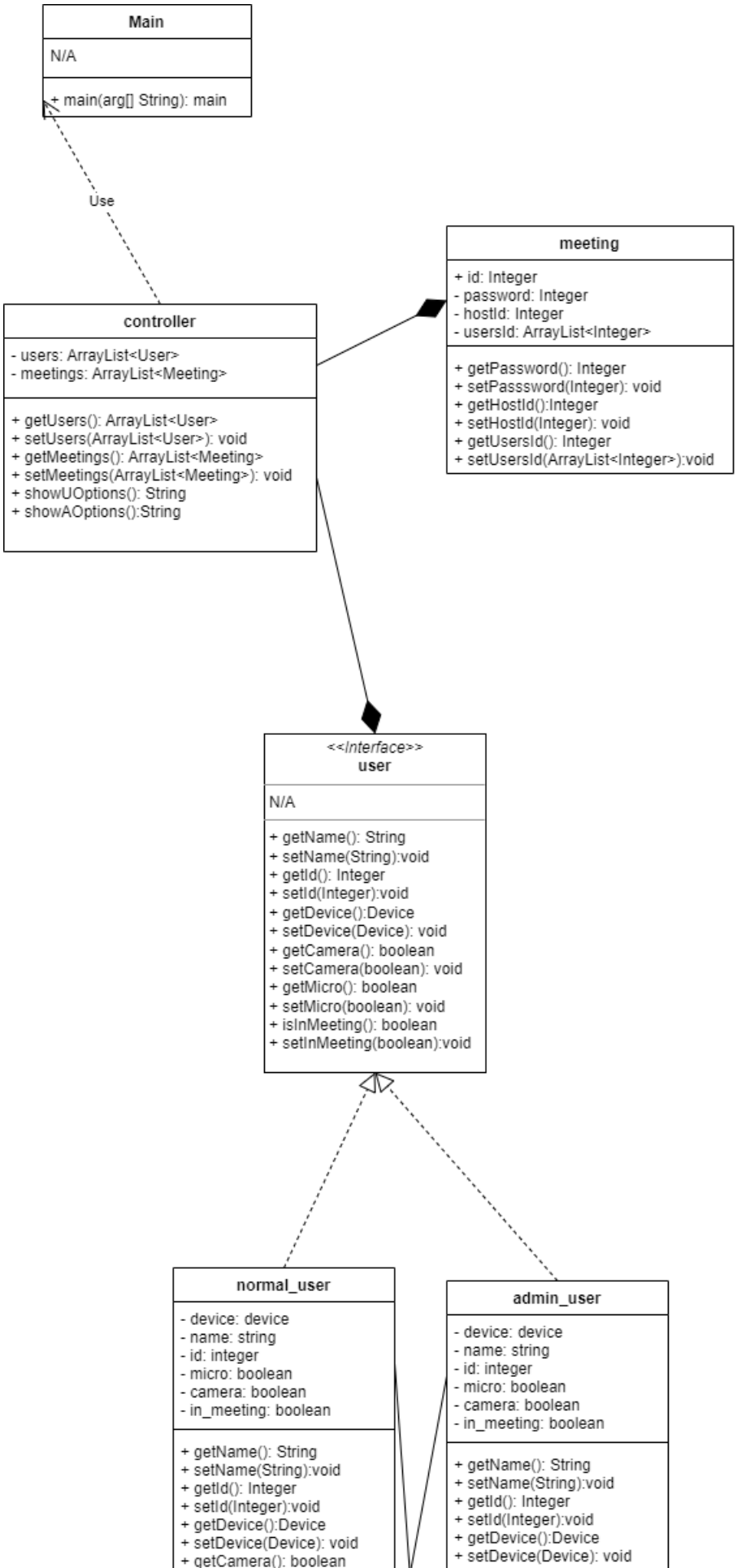
f. AdminUser: clase que implementa User y que es el caso de que sea un usuario anfitrión.
    i. Atributos:
        1. Device: es un device, privado.
        2. Name: es un string con el nombre, privado.
        3. Id: es un entero con el id del usuario, privado.
        4. Micro: es un booleano con el micrófono apagado o encendido, privado.
        5. Camera: es un booleano con la cámara encendida o apagada, privado.
        6. InMeeting: es un booleano sí está o no en una reunión, privado.
    ii. Métodos: clase implementa User por ende los métodos a usar son overrides que no escribiré aquí.
g. Device: interfaz que permite definir los métodos generales de un dispositivo.
    i. Atributos: No aplica
    ii. Métodos:
        1. getOnOff: retorna un booleano, sin argumentos, público.
        2. setOnOff: retorna un void, pide un booleano, público.
h. Cel: clase que implementa device, por lo que varios métodos no se escribirán dado que están en Override, así mismo modela un celular.
    i. Atributos:
        1. OnOff: booleano que indica sí está prendido u apagado un dispositivo, privado.
        2. Calling: sí está llamando o no, privado.
    ii. Métodos:
        1. getCalling: retorna booleano, sin argumentos, público.
        2. setCalling: retorna void, pide un booleano, público.
i. Car: clase que implementa device, por lo que varios métodos no se escribirán dado que están en Override, así mismo modela un carro.
    i. Atributos:
        1. OnOff: booleano que indica sí está prendido u apagado un dispositivo, privado.
        2. Driving: booleano que indica sí está manejando o está frenado un vehículo, privado.
    ii. Métodos:
        1. getDriving: retorna un booleano, no pide argumentos, público.
        2. setDriving: retorna void, pide un booleano, es público.
j. Tablet: clase que implementa device, por lo que varios métodos no se escribirán dado que están en Override, así mismo modela una tablet.
    i. Atributos:

1. OnOff: booleano que indica sí está prendido u apagado un dispositivo, privado.
2. RunningApps: booleano que indica sí se está corriendo una aplicación en la Tablet, privado.

ii. Métodos

1. getRunningApps: retorna un booleano, sin argumentos, público.
2. setRunningApps: retorna un void, pide un booleano, público.

3. Diseño


Main
N/A
+ main(arg[] String): main
Use
meeting
+ id: Integer
- password: Integer
- hostId: Integer
- usersId: ArrayList<Integer>
+ getPassword(): Integer
+ setPasssword(Integer): void
+ getHostId():Integer
+ setHostId(Integer): void
+ getUsersId(): Integer
+ setUsersId(ArrayList<Integer>):void
controller
- users: ArrayList<User>
- meetings: ArrayList<Meeting>
+ getUsers(): ArrayList<User>
+ setUsers(ArrayList<User>): void
+ getMeetings(): ArrayList<Meeting>
+ setMeetings(ArrayList<Meeting>): void
+ showUOptions(): String
+ showAOptions():String
<<Interface>>
user
N/A
+ getName(): String
+ setName(String):void
+ getId(): Integer
+ setId(Integer):void
+ getDevice():Device
+ setDevice(Device): void
+ getCamera(): boolean
+ setCamera(boolean): void
+ getMicro(): boolean
+ setMicro(boolean): void
+ isInMeeting(): boolean
+ setInMeeting(boolean):void
normal_user
- device: device
- name: string
- id: integer
- micro: boolean
- camera: boolean
- in_meeting: boolean
+ getName(): String
+ setName(String):void
+ getId(): Integer
+ setId(Integer):void
+ getDevice():Device
+ setDevice(Device): void
+ getCamera(): boolean
admin_user
- device: device
- name: string
- id: integer
- micro: boolean
- camera: boolean
- in_meeting: boolean
+ getName(): String
+ setName(String):void
+ getId(): Integer
+ setId(Integer):void
+ getDevice():Device
+ setDevice(Device): void