Second Assignment

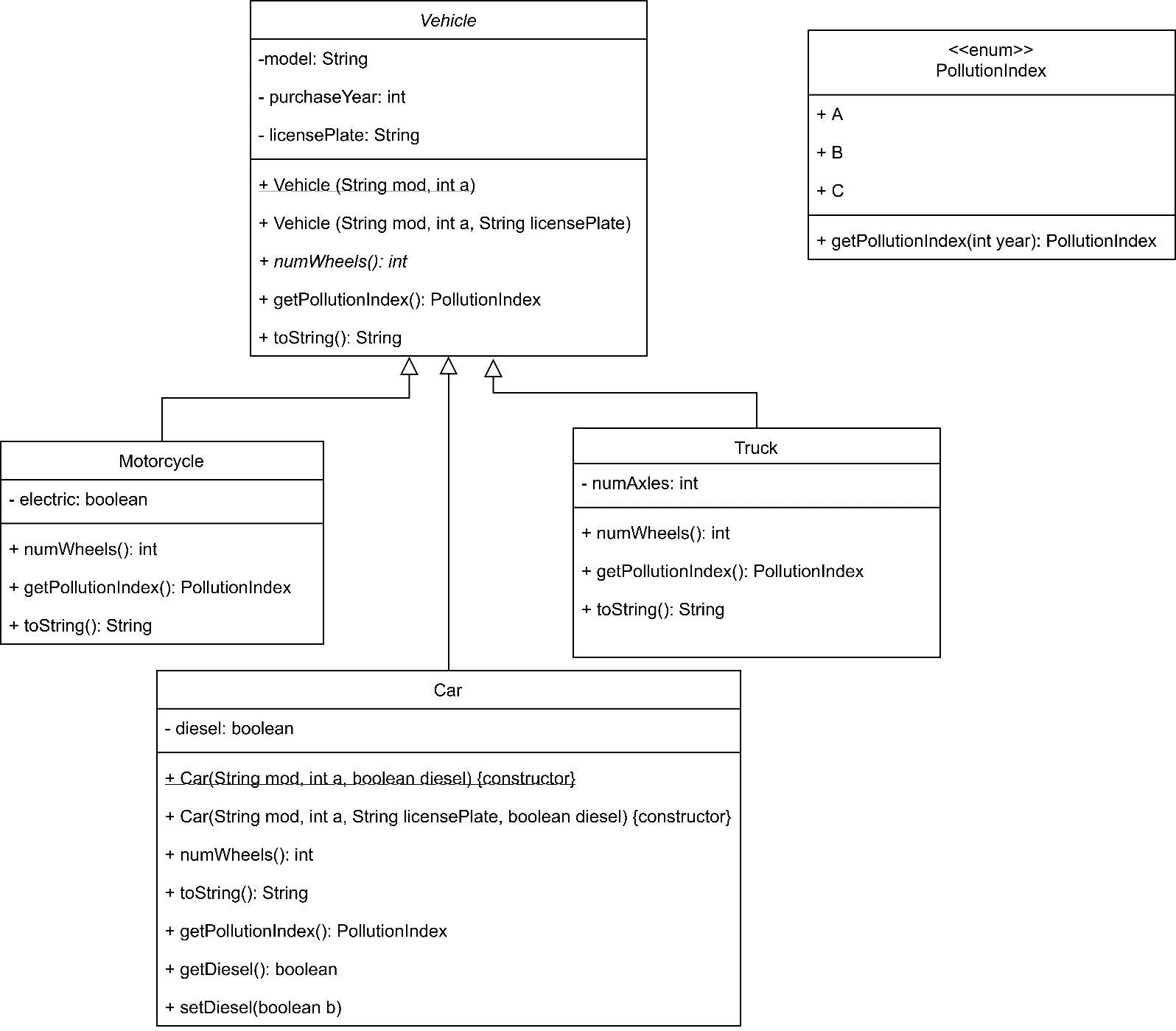
Object oriented design

Authors:

Kevin de la Coba Malam

Marcos Aarón Bernuy

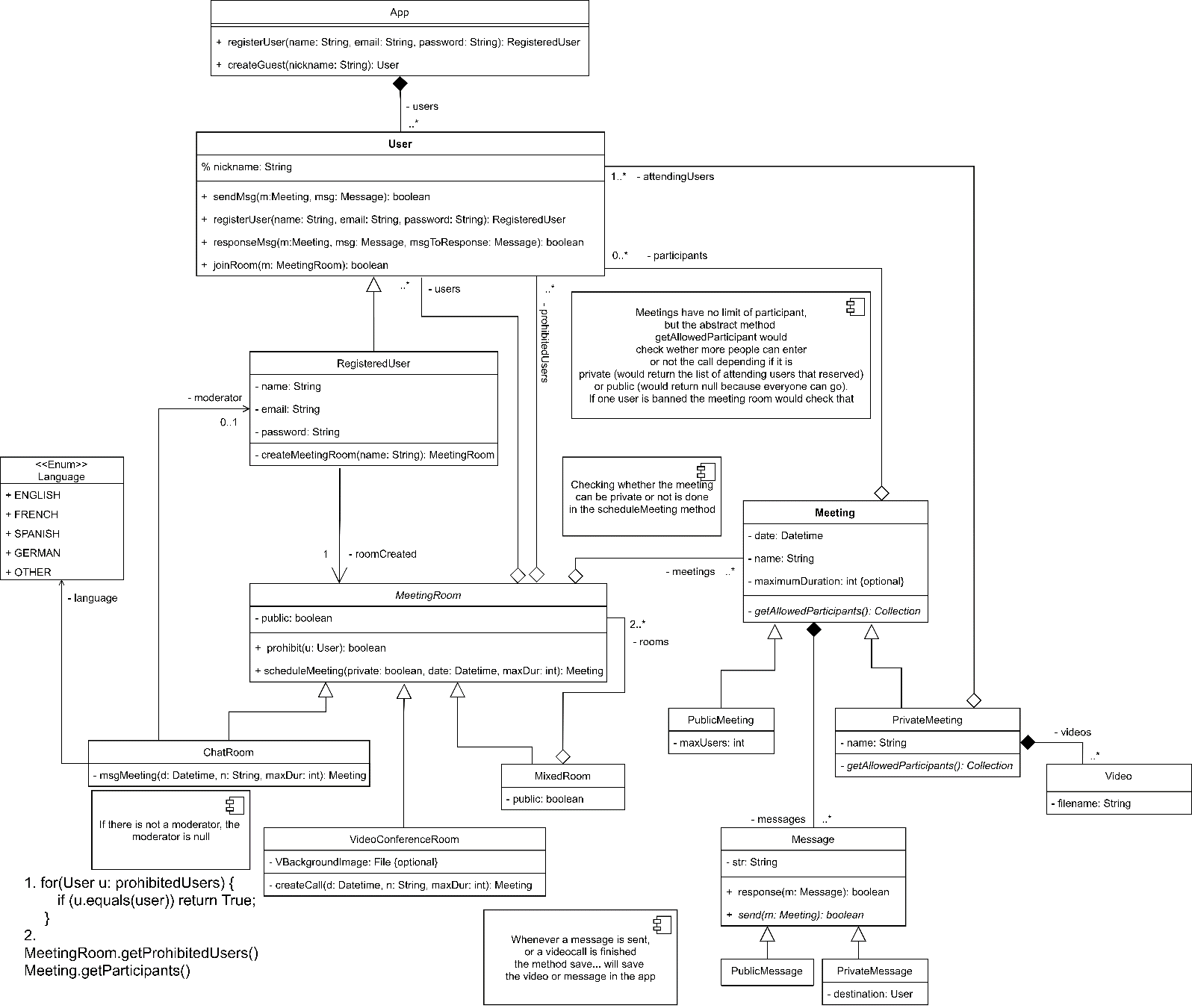
**Part 1**



This is the class diagram of the part 1. We can see that we added 2 more classes **Motorcycle** and **Truck**. As we can see the motorcycle has an attribute to check if it is electric or not, the truck has an attribute to know the number of Axles, and both classes implement the abstract methods in vehicle.

In our classes we did not write the constructors and getters.

**Part 3**



This is the class diagram of the part 3. We can see that we have the main class **App**, this class allows users to register or join as a guest. So, we can differentiate the users by **User** and **RegisteredUser**, the user can only join meetings and interact via messages, the registered-user inherits all these methods and can also create one meeting.

There are 3 types of meetings **ChatRoom**, **VideoConferenceRoom** and **MixedRoom**, the chat-room has a language and a registered-user as a moderator, if the moderator is null then there is not a moderator in the chat-room. We also have video-conference-rooms, these ones have an optional attribute which is the background. Finally, we have a mixed room which contains 2 or more rooms of any type (including mixed). The rooms have users in it and have prohibited users, which cannot join the room, these data is stored in 2 lists, one with the users in the room and one with the prohibited users.

These rooms can create several **Meeting** objects, they need to specify the date, name, and maximum duration of the meeting. The meetings can be **public** or **private**, if they are publicit is specified the maximum number of users that can be in it, if it is private there is a list of users that can be in the meeting, in both cases we the meeting stores the actual participants.

In order to save the content, meetings have **Messages**, the messages can be **public** or **private** but in both cases the messages are sent by a user and then they are stored in the meeting (if the meeting allows messages). Private rooms have videos which are stored in the actual private room.

Pseudocode:

1. *Check that a user can access a meeting.*

**for**(User **u:** prohibitedUsers) {

**if** (u.equals(user)) **return** True;

}

1. *Obtain the users with prohibited entry into a room, and all those contained in it.*

MeetingRoom.getProhibitedUsers()

MeetingRoom.getUsers()