

notes

Maksym Planeta

June 6, 2014

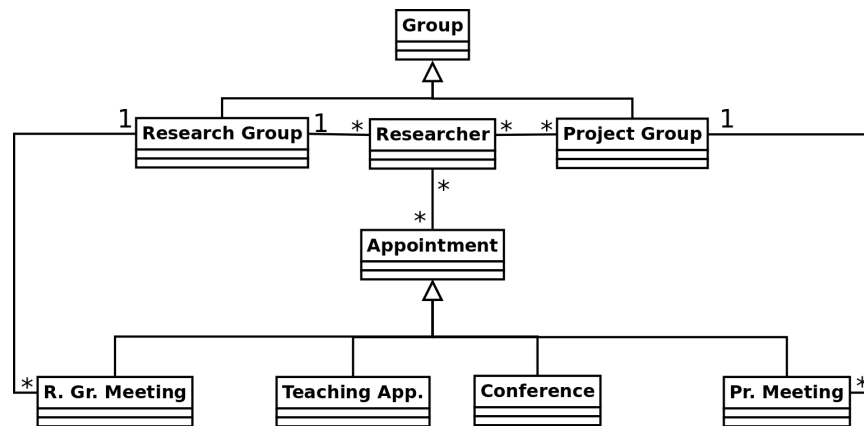
Contents

1	DONE Use Case scenarios	3
1.1	Business Concept Model	3
1.2	Use Case Model	5
1.2.1	Register	6
1.2.2	Login	6
1.2.3	Logout	7
1.2.4	Create appointment	7
1.2.5	Enter type-specific appointment information	8
1.2.6	Create time field	9
1.2.7	Enter generic appointment information	9
1.2.8	Delete appointment	9
1.2.9	Leave appointment	10
1.2.10	Add participant to appointment	10
1.2.11	Create group	11
1.2.12	Join group	11
1.2.13	Leave group	12
1.2.14	Delete group	12
1.2.15	Change Group password	12
1.2.16	Change appointment	12
1.2.17	View appointment details	12
1.2.18	View schedule	13
1.2.19	View groups	13
2	DONE Identify system interfaces and operations	14
2.1	Component Identification	14
2.1.1	Business Type Model	14
2.1.2	System Interfaces	15

2.1.3	Business Type Interface Model	21
2.1.4	Initial Component Specifications	21
3	DONE Component Interaction	21
3.1	Collaboration Diagrams	21
3.2	Auxiliary types	21
3.2.1	Auxiliary Types	21
3.3	Business Interfaces	24
3.3.1	Group Management	24
3.3.2	Researcher Management	24
3.3.3	Appointment Management	25
3.3.4	Time Management	25
3.4	System Interfaces	25
3.4.1	Register	25
3.4.2	Login	25
3.4.3	Logout	26
3.4.4	Create appointment	26
3.4.5	Time management	26
3.4.6	Delete appointment	26
3.4.7	Add researcher to an appointment	27
3.4.8	Create group	27
3.4.9	Join group	27
3.4.10	Leave group	27
3.4.11	Conflicts	27
3.4.12	Search appointment	28
3.4.13	Search participant	28
3.4.14	Search researcher	28
3.4.15	Search group	28
4	DONE Component Specification	29
4.1	Business interface specification diagrams	29
4.1.1	General notes:	29
4.1.2	IGroupManagement	29
4.1.3	IResearcherManagement	29
4.1.4	IAppointmentManagement	30
4.2	System interface specification diagrams	30
4.3	Pre- and Post- Conditions	30
Quark project :quark:cbse:		

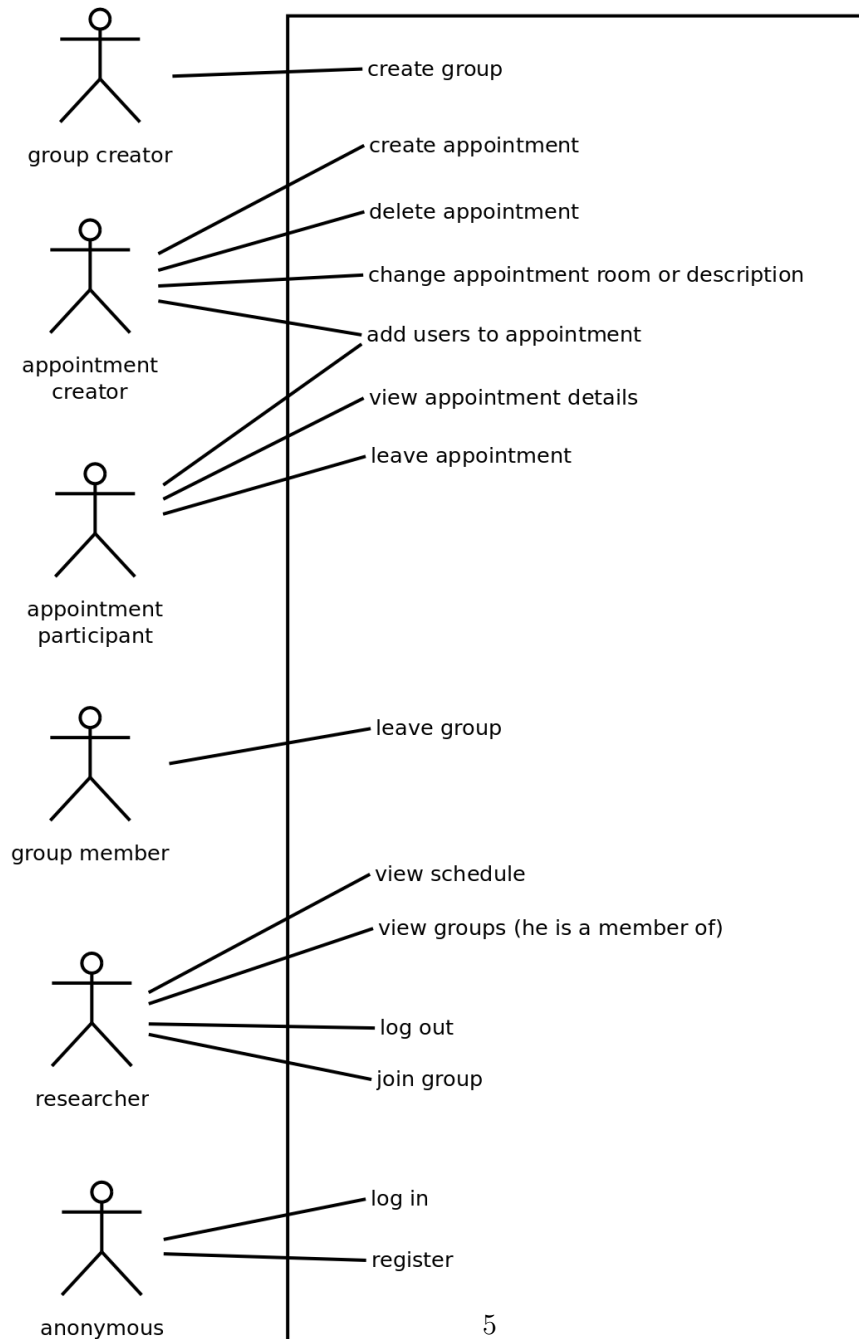
1 DONE Use Case scenarios

1.1 Business Concept Model



1.2 Use Case Model

Use Case Model



1.2.1 Register

Initiator: Anonymous

Goal: Create new user

Main Success Scenario:

1. Anonymous gets to sign in page
2. Anonymous enters credentials (name, email, password)
3. System checks credentials for correctness
4. System creates new user
5. System logs in new user
6. Anonymous role is changed to "researcher"

Extensions:

1. Anonymous does not enter all the data
 - (a) System asks user to enter all the data
 - (b) Anonymous enters all the data
2. Email already in use
 - (a) System asks user to enter another email
 - (b) Anonymous enters another email
3. Password does not meet security requirements
 - (a) System asks user to enter another password
 - (b) Anonymous enters another password

1.2.2 Login

Initiator: Anonymous

Goal: Log in

Main Success Scenario:

1. Anonymous gets to log in page
2. Anonymous enters credentials (email, password)

3. System checks credentials for correctness
4. System logs in new user
5. Anonymous role is changed to "researcher"

Extensions:

1. Anonymous does not enter all the data
 - (a) System asks user to enter all the data
 - (b) Anonymous enters all the data
2. Email and password combination is not known to the system
 - (a) System asks user to enter another email and password
 - (b) Anonymous enters another email and password

1.2.3 Logout

Initiator: Researcher

Goal: Log out

Main Success Scenario:

1. Researcher asks system to log him out
2. System terminates connection with user
3. Researcher changes his role to Anonymous

1.2.4 Create appointment

Initiator: Researcher

Goal: Create new appointment

Main Success Scenario:

1. Researcher asks system to create new appointment
2. Researcher selects appointment type
3. Include Enter type specific appointment information
4. Include Enter generic appointment information
5. Researcher is shown conflicts with his schedule (OPT)

6. Researcher is shown conflicts with all other schedules (OPT)
7. Researches acknowledges appointment creation
8. System creates the appointment
9. System assigns researcher as the creator for newly created appointment

Extensions:

1. Researcher decides to avoid conflicts
 - (a) Researcher changes appointment time
 - (b) Goto 5.

1.2.5 Enter type-specific appointment information

Initiator: Included only

Goal: Get type specific information

Main success scenario:

1. System creates form with type-specific data field
2. Researcher enters data

Extensions:

1. Appointment type is "Project group meeting"
 - (a) Create field for group selection
 - (b) Include Create time field
2. Appointment type is "Research group meeting"
 - (a) Create field for group selection
 - (b) Include Create time field
3. Appointment type is "Teaching appointment"
 - (a) Create field for group selection
 - (b) Include Create time field
4. Appointment type is "Conference appointment"
 - (a) Create field for group selection
 - (b) Include Create time field

1.2.6 Create time field

Initiator: Included only

Goal: Create time field for requesting date/time information

Main success scenario:

1. Researcher tells when appointment takes place
2. Researcher tells that appointment is regular
3. Researcher tells period within which appointment takes place
4. Researcher tells time range in which appointment takes place

Extensions:

1. Researcher tells that appointment is one-shot
 - (a) Done

1.2.7 Enter generic appointment information

Initiator: Included only

Goal: Get type generic information

Main success scenario:

1. Researchers enters location
2. Researchers enters description

1.2.8 Delete appointment

Initiator: Appointment creator

Goal: Delete appointment from all schedules

Main success scenario:

1. Creator finds appointment in his schedule
2. Creator asks system to delete appointment
3. System asks for acknowledgment
4. Creator acknowledges
5. System deletes appointments from all schedules

1.2.9 Leave appointment

Initiator: Appointment participant

Goal: Delete appointment from personal schedule

Main success scenario:

1. Creator finds appointment in his schedule
2. Creator asks system to delete appointment
3. System asks for acknowledgment
4. Creator acknowledges
5. System deletes appointments from creator's schedule

1.2.10 Add participant to appointment

Initiator: Appointment participant

Goal: Add other participant to appointment

Main success scenario:

1. Participant finds appointment in his schedule
2. Participant asks system to invite another participant
3. System asks for another participant information
4. Participant enters another participant information (email, other inf is OPT)
5. System shows list of found participants
6. Participant chooses one or more (OPT) other participants
7. System asks for acknowledgment
8. Participant acknowledges
9. System invites chosen participants to chosen appointment

Extensions:

1. System does not find any participant that matches entered information
 - (a) Participant enters another information
 - (b) System makes another search

1.2.11 Create group

Initiator: Participant

Goal: Create project or research group

Main success scenario:

1. Participant chooses group name and type
2. System checks that group with specified name and is possible to create
3. System creates group
4. System generates group password
5. System assign participant as a group creator
6. System shows password to group creator

Extensions:

1. Group name of such type already in use
 - (a) System asks participant to enter another name and type
 - (b) Participant enters another name and type

1.2.12 Join group

Initiator: Participant

Goal: Join new project or research group

Main success scenario:

1. Participant chooses group from list of the groups
2. Enters group password and decides to join
3. System checks name, type and password
4. System assign participant to new group

Extensions:

1. User already takes part in research group
 - (a) Deny joining another research group

1.2.13 Leave group

Initiator: Participant

Goal: Leave group

Main success scenario:

1. Participant enters group name and type
2. System removes user from the group

1.2.14 Delete group

Impossible?

1.2.15 Change Group password

Impossible

1.2.16 Change appointment

Initiator: Appointment creator

Goal: Change appointment details

Main success scenario:

1. Creator chooses appointment from the schedule
2. Creator changes appointment details (details and location)
3. Creator sends new details to the system
4. System saves the changes

1.2.17 View appointment details

Initiator: Appointment participant

Goal: View appointment details

Main success scenario:

1. Participant chooses appointment from schedule
2. Participant passes appointment handler to system
3. System finds matching appointment in the list of appointments
4. System returns matching appointment with details to the researcher

1.2.18 View schedule

Initiator: Researcher

Goal: View researcher's schedule

Main success scenario:

1. Participant tells the system data range
2. System finds matching appointments in the list of participant's appointments
3. System returns list of matching appointments to the researcher

1.2.19 View groups

Initiator: Researcher

Goal: View group participant

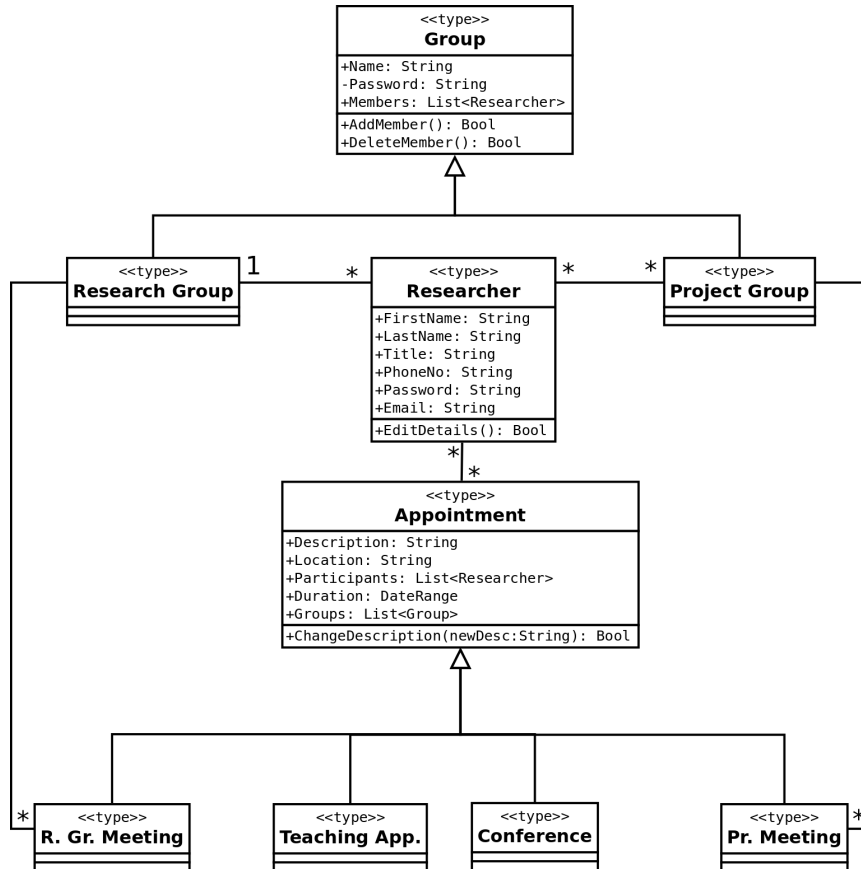
Main success scenario:

1. Participant tells the system group search key
2. System finds matching groups in the list of all groups
3. System returns list of matching groups to the user

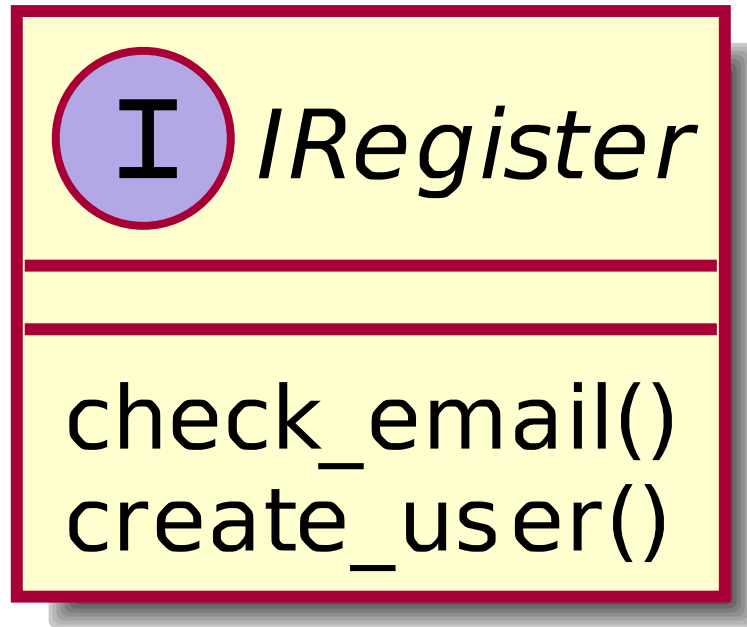
2 DONE Identify system interfaces and operations

2.1 Component Identification

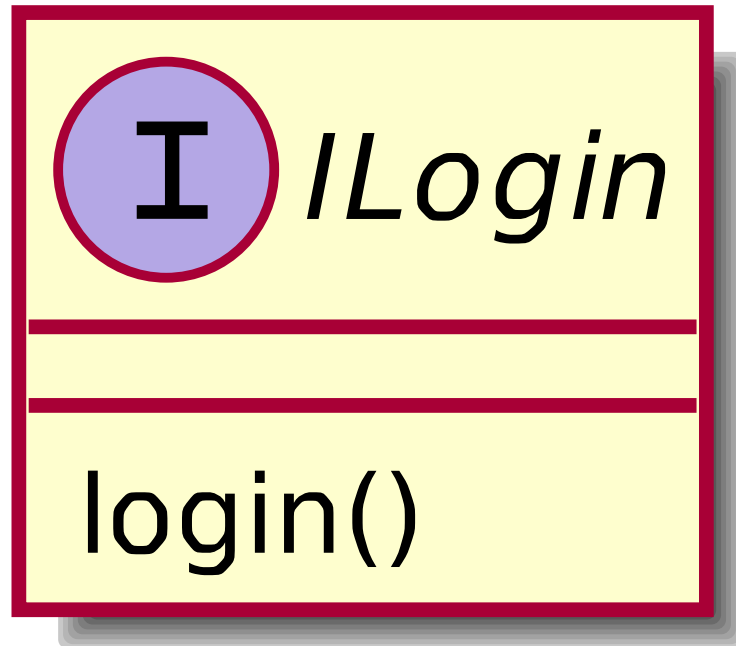
2.1.1 Business Type Model



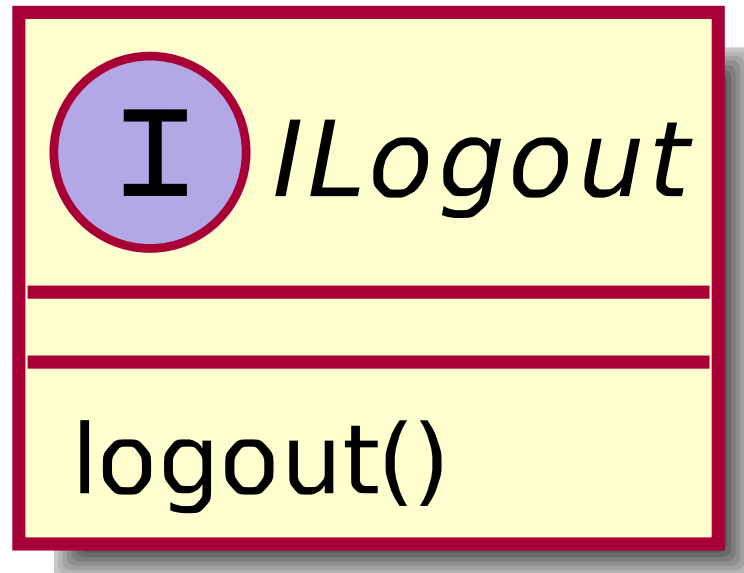
2.1.2 System Interfaces




1. Register



2. Login



3. Logout



I *ICreateAppointment*

The diagram shows a yellow rectangular box with a red border. Inside the box, the letter 'I' is enclosed in a purple circle. To the right of the circle is the text *ICreateAppointment*. Below this, there is a horizontal line, followed by another horizontal line, and then the method name `create_appointment()`.

`create_appointment()`

4. Create appointment

5. Time management

We do this on the client side by constructing an object that encapsulates time range information.




I *ITime*

The diagram shows a yellow rectangular box with a red border. Inside the box, the letter 'I' is enclosed in a purple circle. To the right of the circle is the text *ITime*. Below this, there is a horizontal line, followed by another horizontal line, and then the method name `create_time_information()`.

`create_time_information()`

6. Delete appointment

Delete operation deletes appointment entity if the invoker of this command is the creator of the appointment. And appointment is deleted only from ivoker's schedule if he is just participant of the appointment.

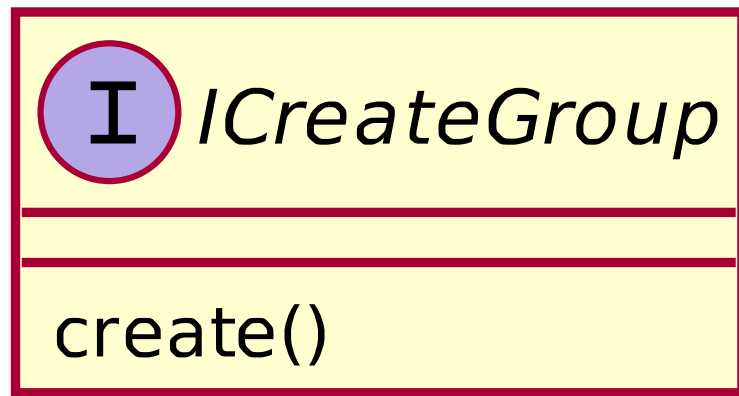
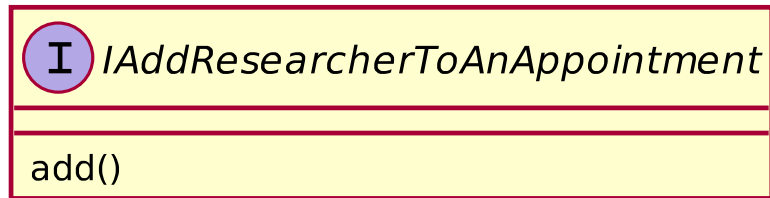


I *IDeleteAppointment*

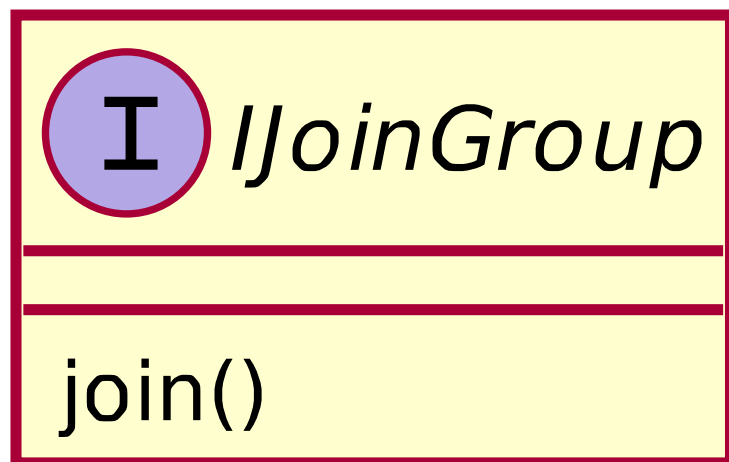
The diagram shows a yellow rectangular box with a red border. Inside the box, the letter 'I' is enclosed in a purple circle. To the right of the circle is the text *IDeleteAppointment*. Below this, there is a horizontal line, followed by another horizontal line, and then the method name `delete()`.

`delete()`

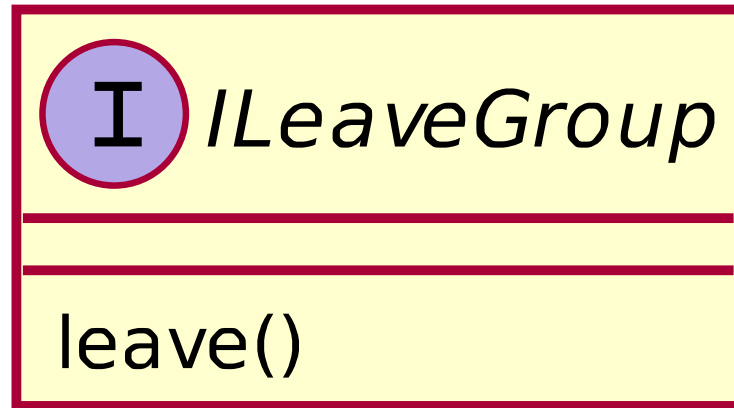
7. Add researcher to an appointment



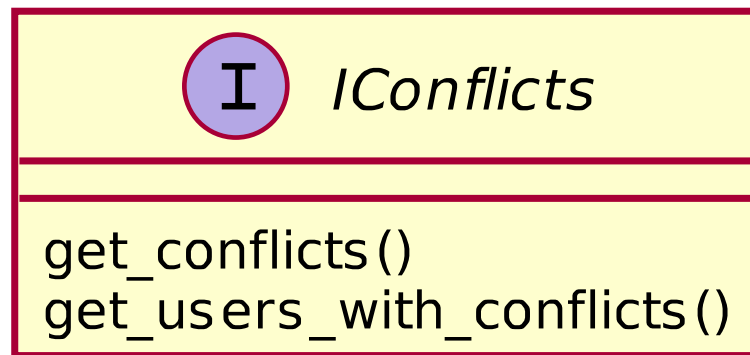
8. Create group



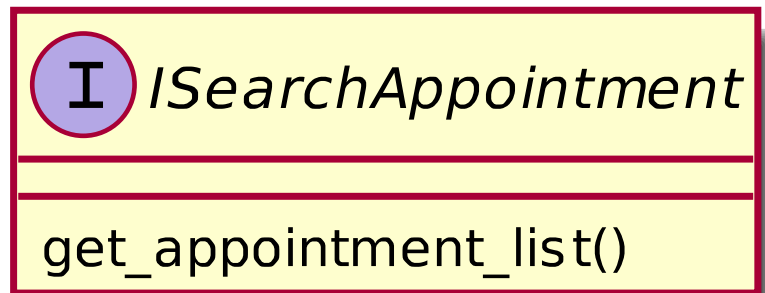
9. Join group



10. Leave group

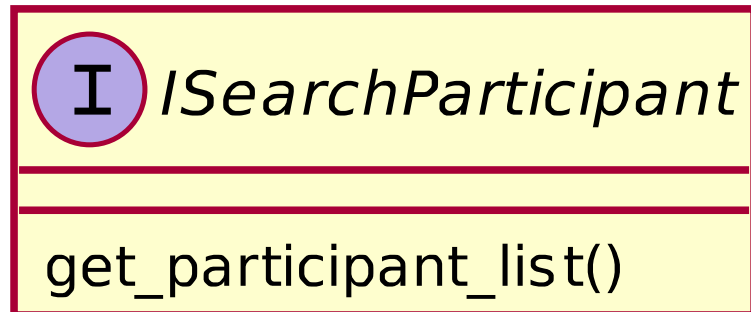


11. Conflicts

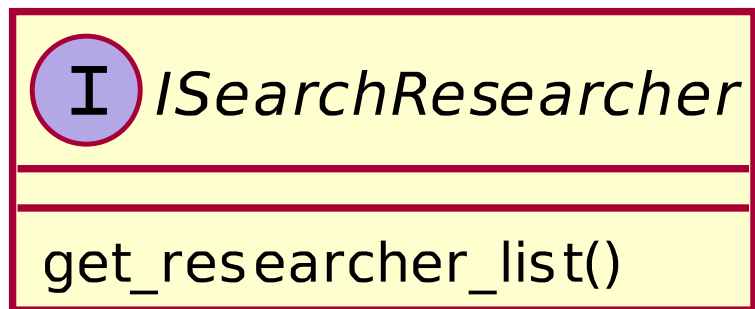


12. Search appointment

13. Search participant

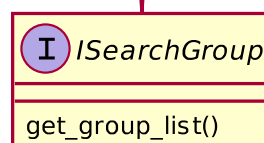


14. Search researcher

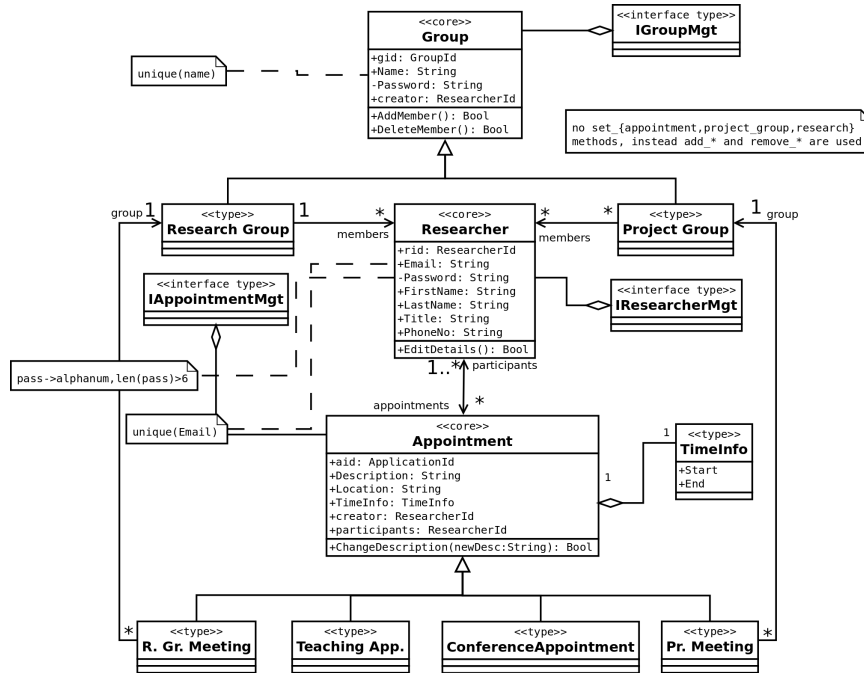


Get list of all groups where participant is a member, if participant is specified, return all groups otherwise

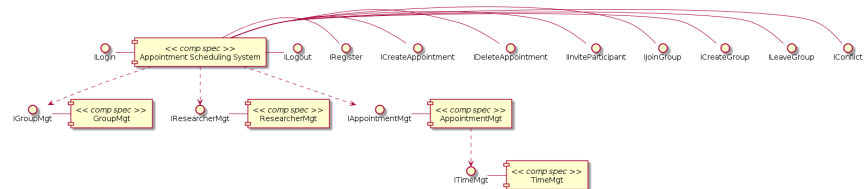
15. Search group



2.1.3 Business Type Interface Model



2.1.4 Initial Component Specifications



3 DONE Component Interaction

3.1 Collaboration Diagrams

3.2 Auxiliary types

3.2.1 Auxiliary Types

1. Types for referring to objects

- ResearcherId, AppointmentId, GroupId



Figure 1: IConflicts

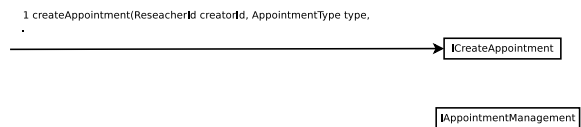


Figure 2: ICreateAppointment

2. Types for communicating categories

GroupType -> enum {PROJECT_GROUP, RESEARCH_GROUP}

AppointmentType ->
 enum {PROJECT_GROUP_MEETING, RESEARCH_GROUP_MEETING,
 TEACHING_APPOINTMENT, CONFERENCE_APPOINTMENT, GENERIC_APPOINTMENT}

3. Types for passing around data

GroupDetails

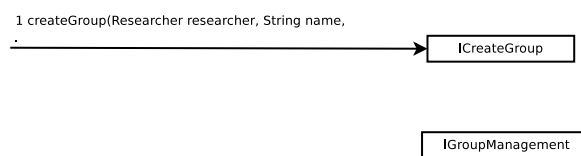


Figure 3: ICreateGroup

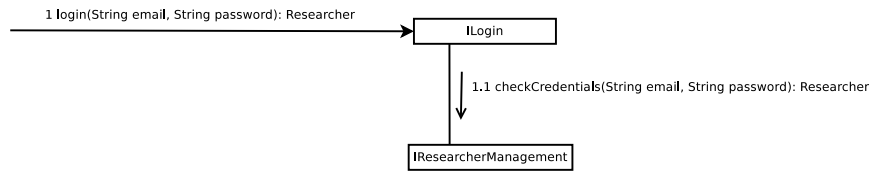


Figure 4: ILogin

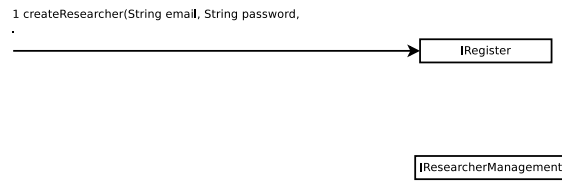


Figure 5: IRegister

- gid
- name
- type
- members (ids) We do not pass password around in open data structures, rather we have a checkPassword method in Group Management interface - it should be more secure when interacting with third party components.

ResearcherDetails

- rid
- emailAddress
- firstname
- lastname
- title

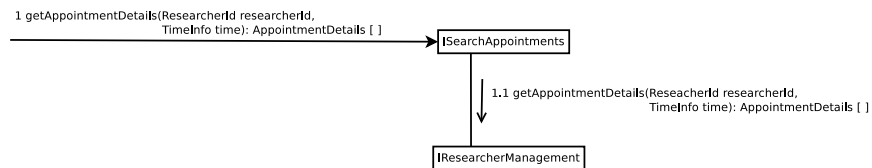


Figure 6: ISearchAppointment

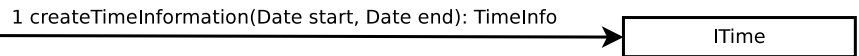


Figure 7: ITime

- phoneNbr

AppointmentDetails

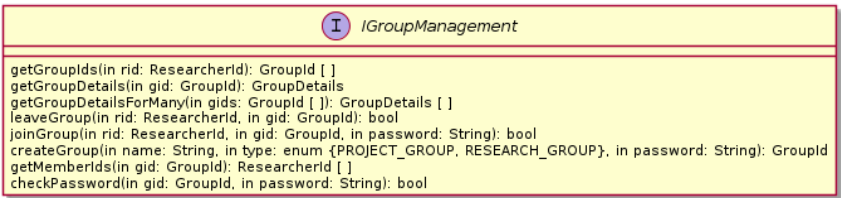
- aid
- type
- timeInterval
- location
- description
- participants (ids)

TimeInterval

- start
- end

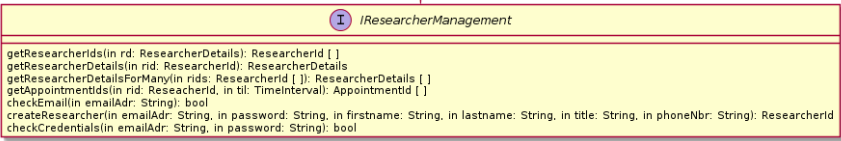
3.3 Business Interfaces

3.3.1 Group Management

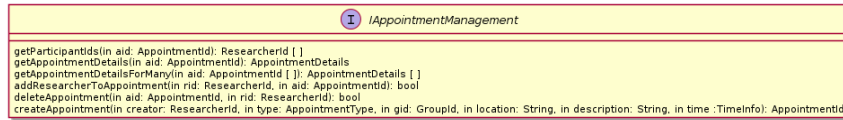


3.3.2 Researcher Management

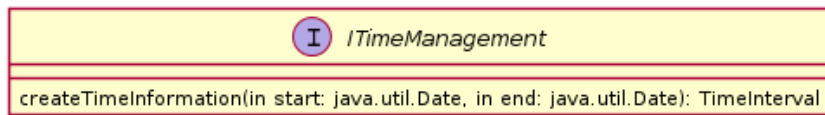
getResearcherIds(): In most cases the parameter rd will be a *partially* specified ResearcherDetails-object. Therefore this operation is pretty general and can also be used to retrieve the IDs of all researchers (-> see collaboration diagram for check_email)
 checkEmail(): check if given email address is unique.



3.3.3 Appointment Management

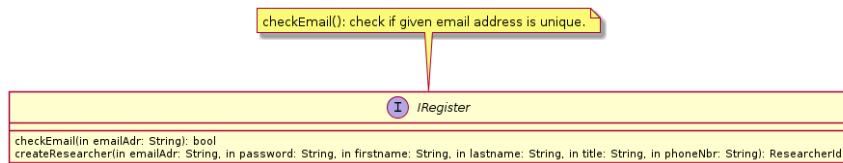


3.3.4 Time Management

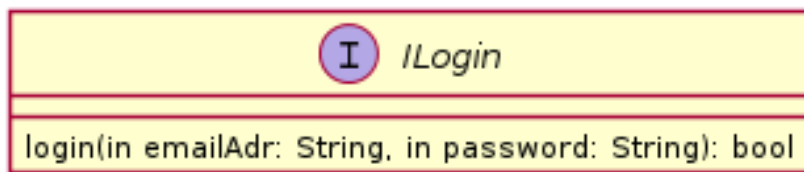


3.4 System Interfaces

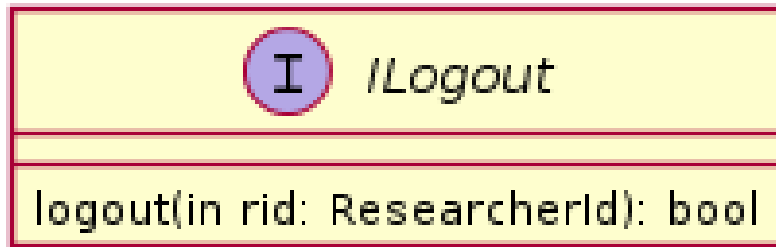
3.4.1 Register



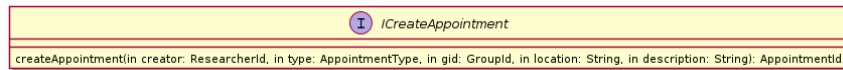
3.4.2 Login



3.4.3 Logout

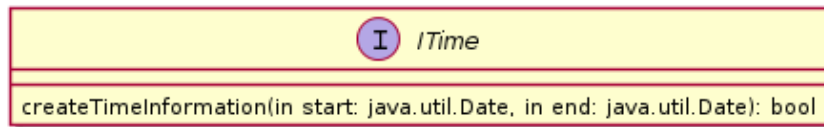


3.4.4 Create appointment



3.4.5 Time management

We do this on the client side by constructing an object that encapsulates time range information.




3.4.6 Delete appointment


Delete operation deletes appointment entity if the invoker of this command is the creator of the appointment. And appointment is deleted only from invoker's schedule if he is just participant of the appointment.




3.4.7 Add researcher to an appointment

 <i>IAddResearcherToAnAppointment</i>
<code>addResearcherToAppointment(in rid: ResearcherId, in aid: AppointmentId): bool</code>

3.4.8 Create group

 <i>ICreateGroup</i>
<code>createGroup(in creator: Researcher, in name: String, in type: GroupType, in password: String): GroupId</code>


3.4.9 Join group

 <i>IJoinGroup</i>
<code>joinGroup(in rid: ResearcherId, in gid: GroupId, in password: String): bool</code>

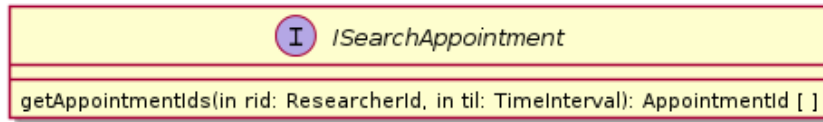
3.4.10 Leave group

 <i>ILeaveGroup</i>
<code>leaveGroup(in rid: ResearcherId, in gid: GroupId): bool</code>

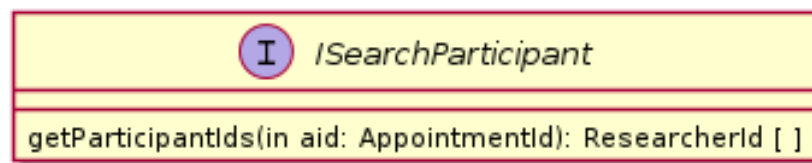
3.4.11 Conflicts

 <i>IConflicts</i>
<code>getAppointmentsWithConflicts(in rid: ResearcherId, in til: TimeInterval): AppointmentId [] []</code> <code>getResearchersWithConflicts(in aid: AppointmentId, in til: TimeInterval): ResearcherId []</code>

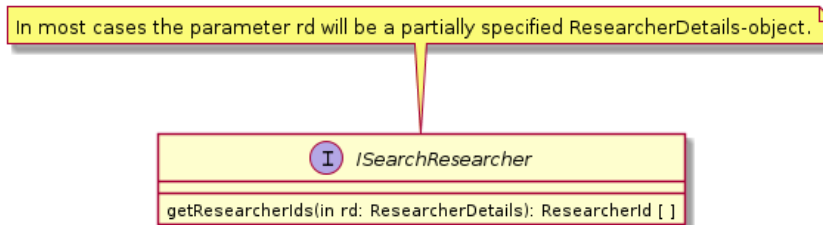
3.4.12 Search appointment



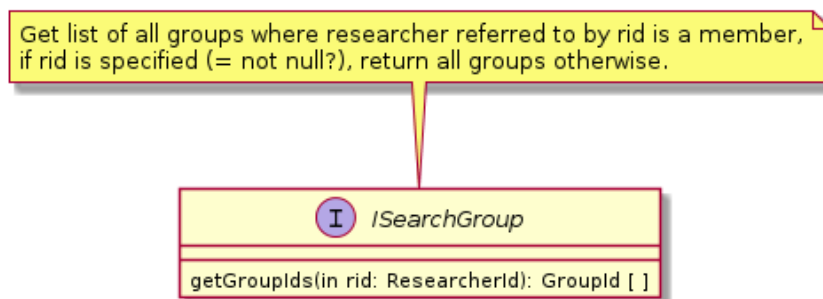
3.4.13 Search participant



3.4.14 Search researcher



3.4.15 Search group



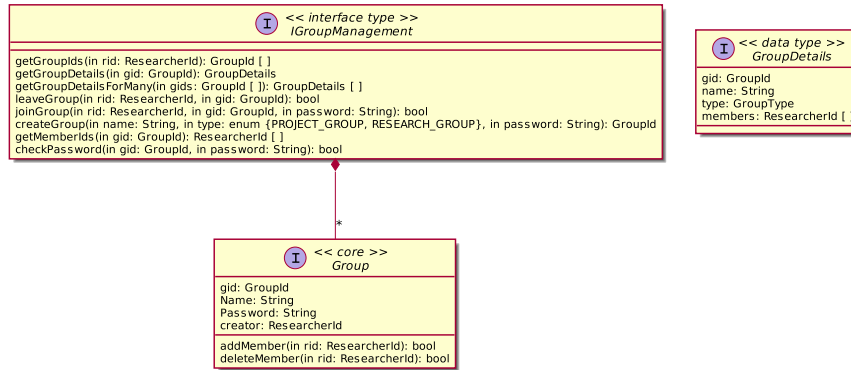
4 DONE Component Specification

4.1 Business interface specification diagrams

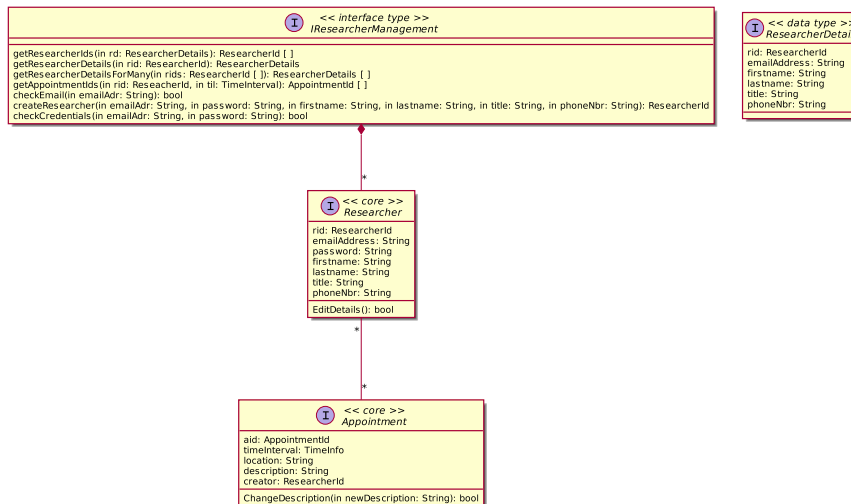
4.1.1 General notes:

- ResearcherId, AppointmentId, GroupId are all unsigned integers.
- GroupType is enum {PROJECT_{GROUP}, RESEARCH_{GROUP}}
- AppointmentType is enum {PROJECT_{GROUPMEETING}, RESEARCH_{GROUPMEETING}, TEACHING_{APPOINTMENT}, CONFERENCE_{APPOINTMENT}, GENERIC_{APPOINTMENT}}

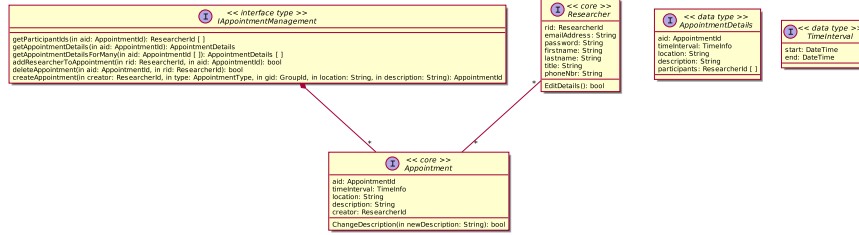
4.1.2 IGroupManagement



4.1.3 IResearcherManagement



4.1.4 IAppointmentManagement



4.2 System interface specification diagrams

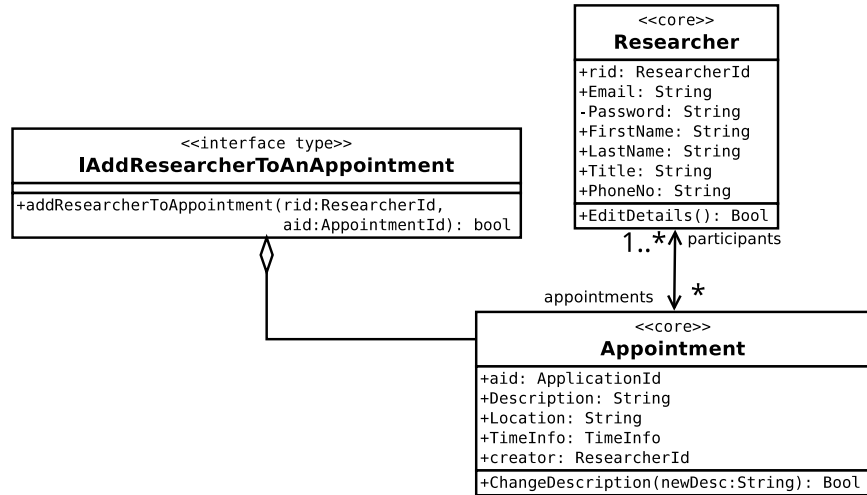


Figure 8: iAddResearcherToAnAppointment

4.3 Pre- and Post- Conditions

leaveGroup(in rid: ResearcherId, in gid: GroupId): bool

PRE: rid is in group members.

POST: rid is deleted from group members.

joinGroup(in rid: ResearcherId, in gid: GroupId, in password: String):
bool

PRE: rid is not a group member.

POST: rid becomes member of group.

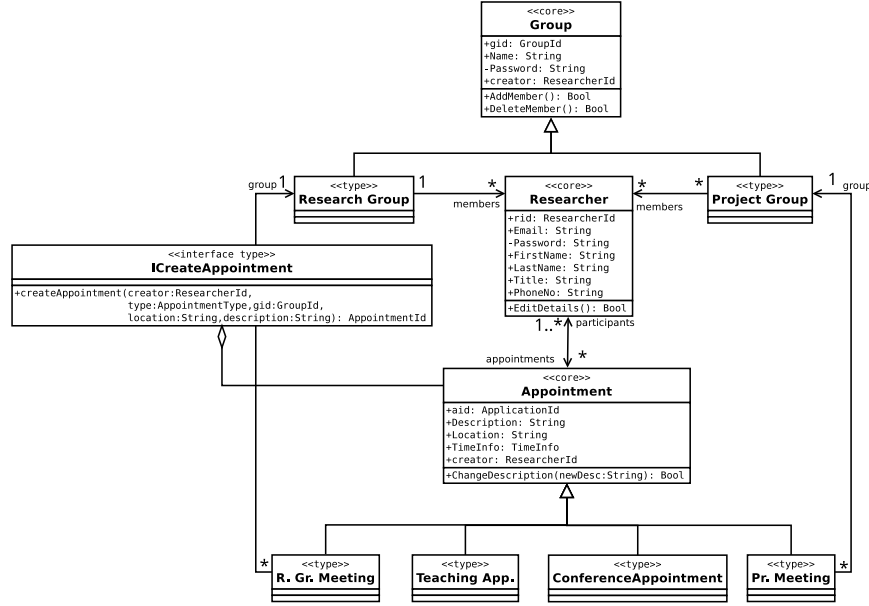


Figure 9: iCreateAppointment

createGroup(in name: String, in type: GroupType): GroupId

PRE: password < 6 symbols.

POST: group is not created.

PRE: group creator is not member of any research group and type is RESEARCH_GROUP.

POST: research group is created.

checkPassword(in gid: GroupId, in password: String): bool

PRE: gid must exist.

POST: password check is conducted.

createResearcher(in emailAdr: String, in password: String, in firstname: String, in lastname: String, in title: String, in phoneNbr: String): ResearcherId

PRE: email should be unique.

POST: user is created.

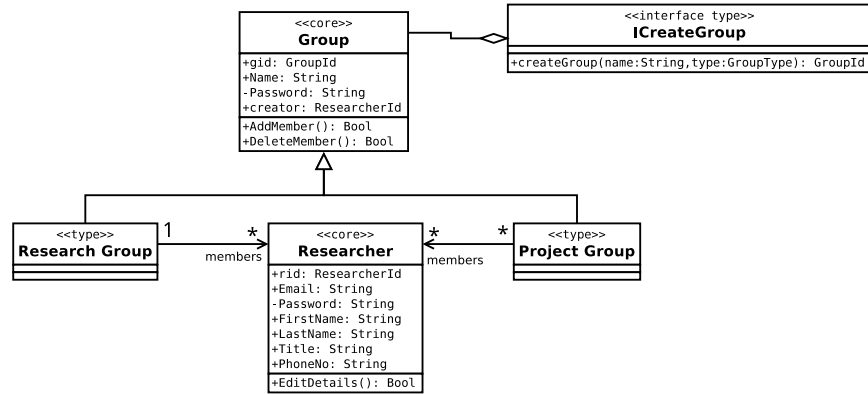


Figure 10: iCreateGroup

createAppointment(in creator: ResearcherId, in type: Appointment-
Type, in gid: GroupId, in location: String, in description: String): Ap-
pointmentId

PRE: researcher id must exist. groupid must exist.

POST: appointment is created.

createTimeInformation(in start: java.util.Date, in end: java.util.Date):
TimeInterval

PRE: start < end.

POST: TimeInterval is returned.

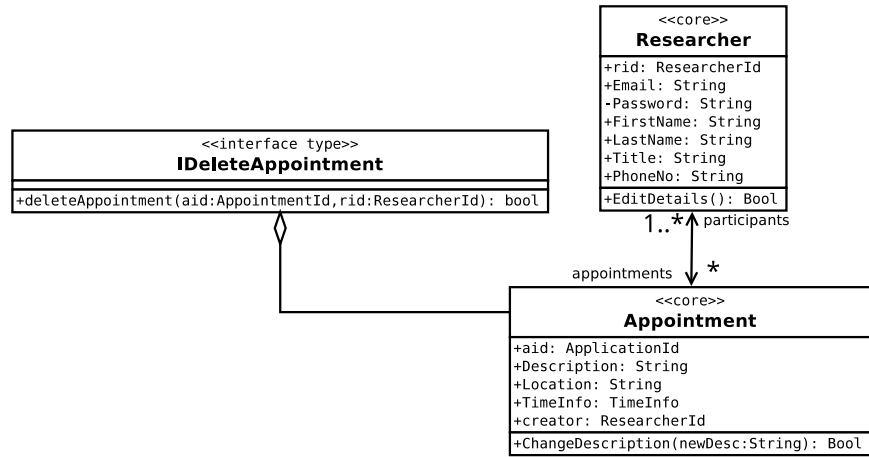


Figure 11: iDeleteAppointment

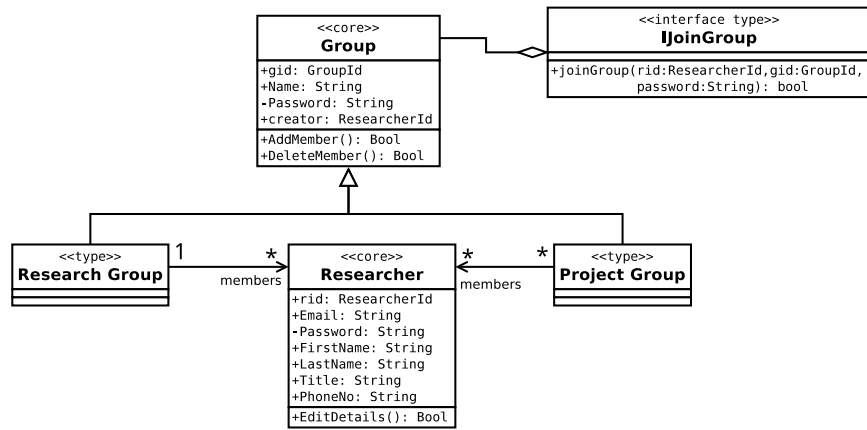


Figure 12: iJoinGroup

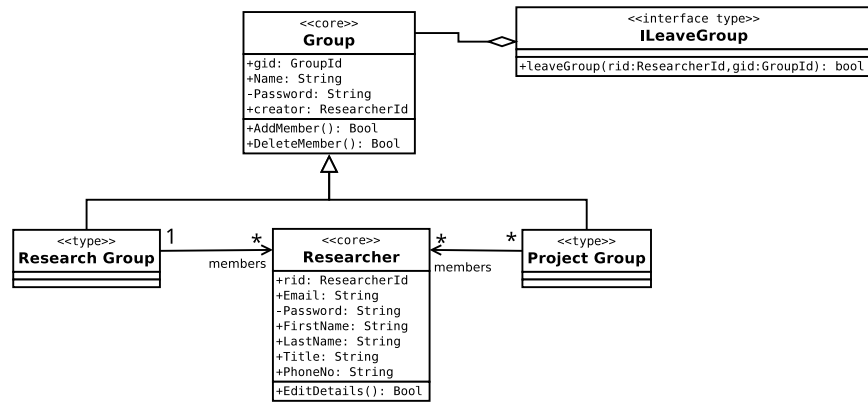


Figure 13: iLeaveGroup

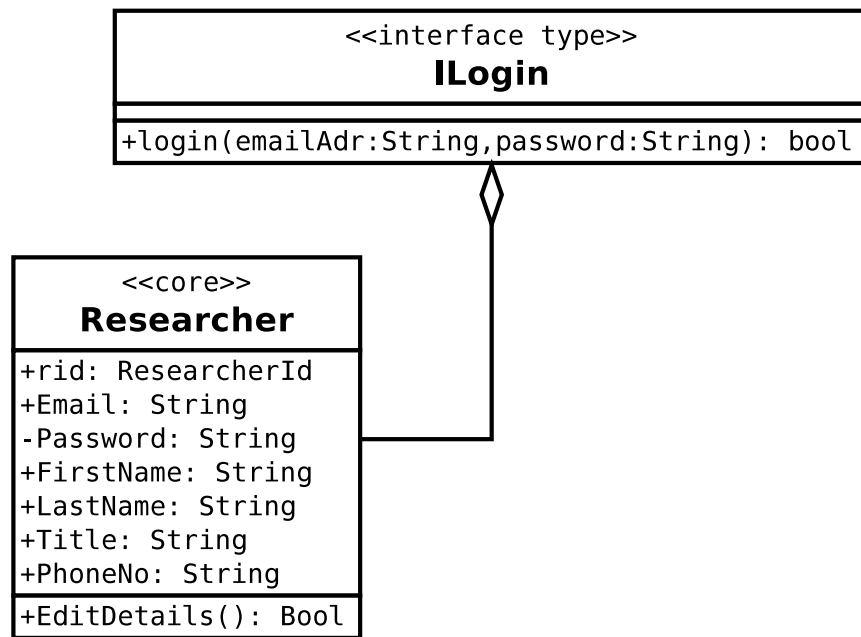


Figure 14: ilogin

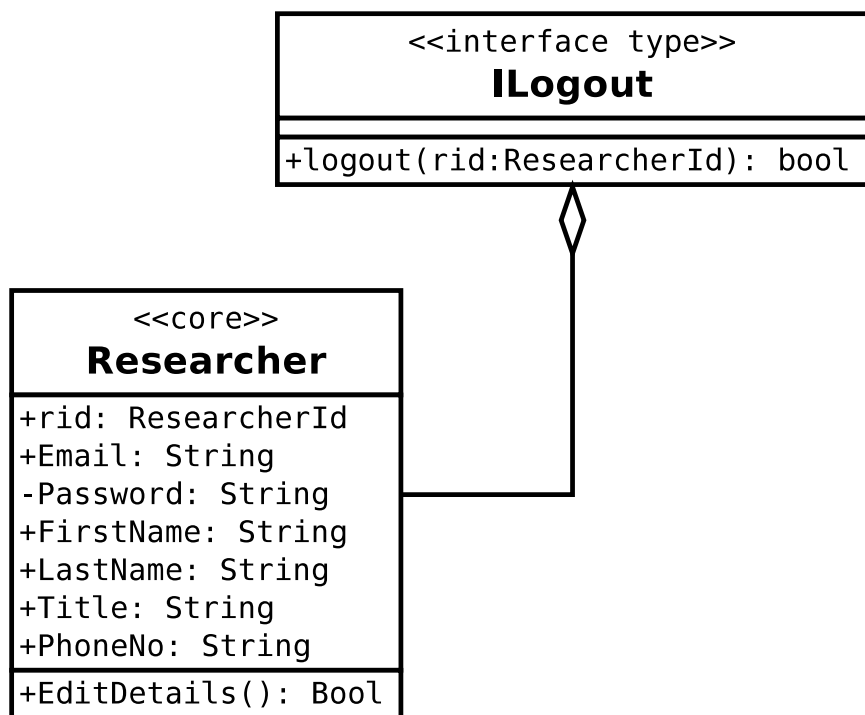


Figure 15: ilogout

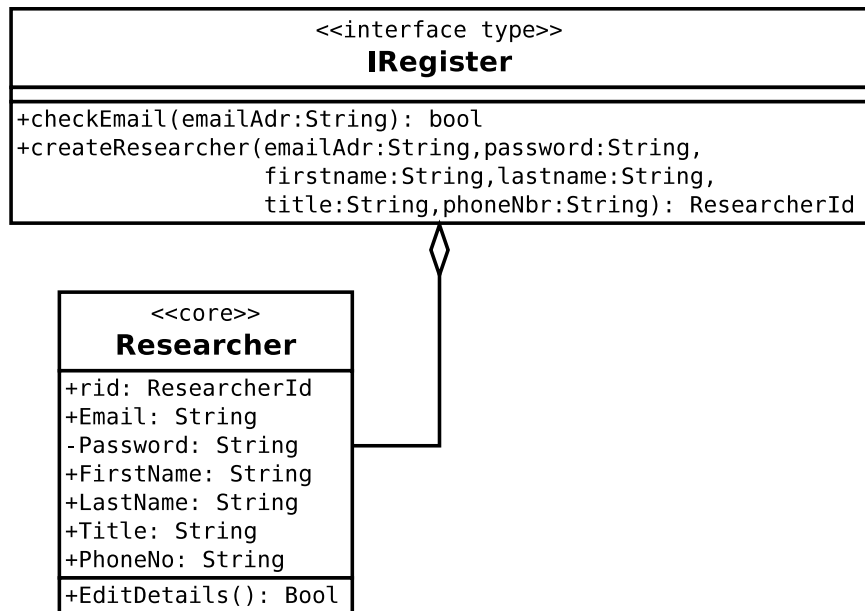


Figure 16: iregister