notes

Maksym Planeta

June 5, 2014

Contents

1 Quark project **QUARK:CBSE**

1.1 **DONE** Use Case scenarios

1.1.1 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

1.1.2 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

1.1.3 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

1.1.4 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.

1.1.5 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

1.1.6 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

1.1.7 Enter generic appointment information

Initiator: Included only
Goal: Get type generic information
Main success scenario:

1. Researchers enters location
2. Researchers enters description

1.1.8 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

1.1.9 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

1.1.10 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

1.1.11 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

1.1.12 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

1.1.13 Leave group

Initiator: Participant
Goal: Leave group
Main success scenario:

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

1.1.14 Delete group

Impossible?

1.1.15 Change Group password

Impossible

1.1.16 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

1.1.17 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

1.1.18 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

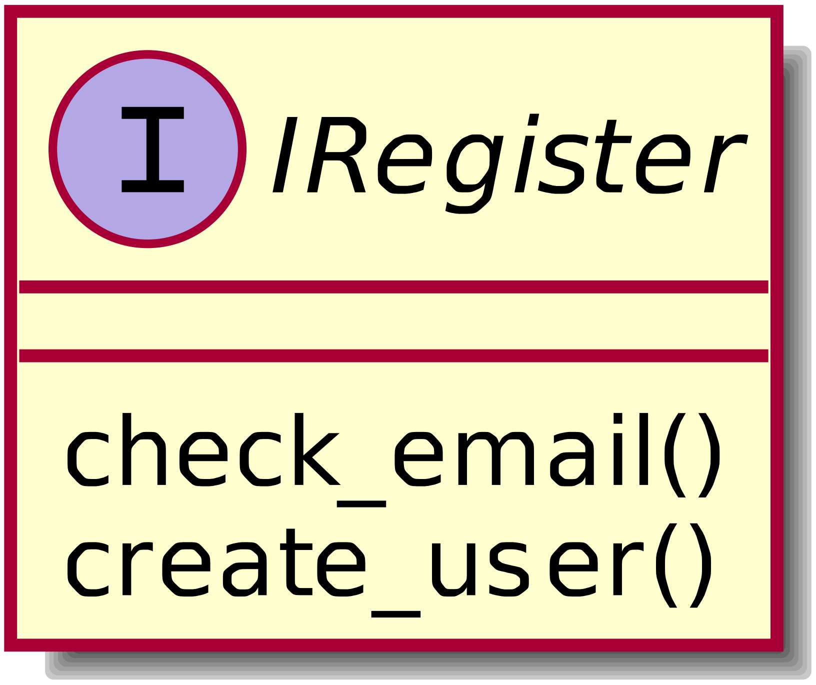
1.1.19 View groups

Initiator: Researcher
Goal: View group participant
Main success scenario:

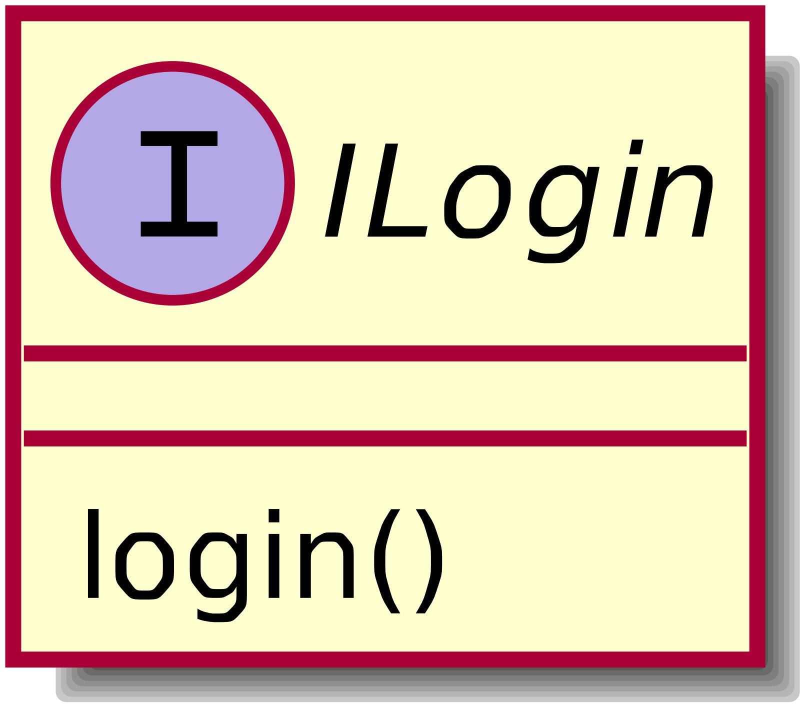
1. Participant tells the system group search key
2. System finds matching groups in the list of all groups
3. System returns list of matching groups to the user

1.2 DONE Identify system interfaces and operations

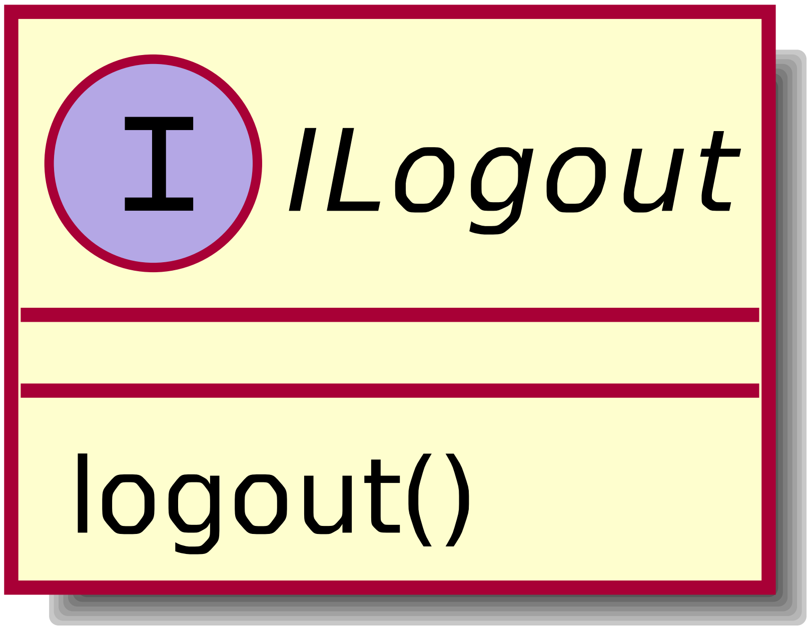
1.2.1 Register



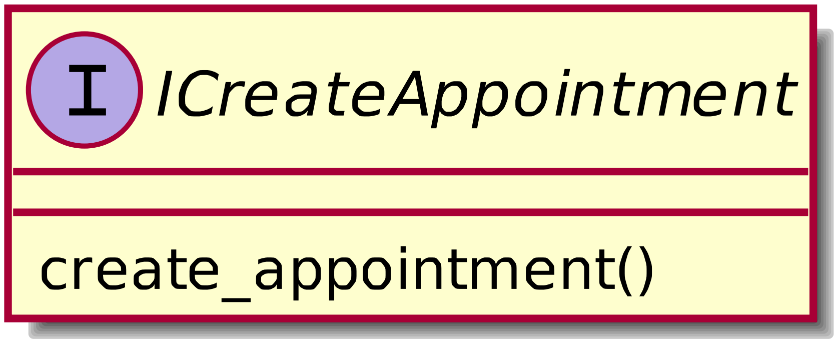
1.2.2 Login



1.2.3 Logout

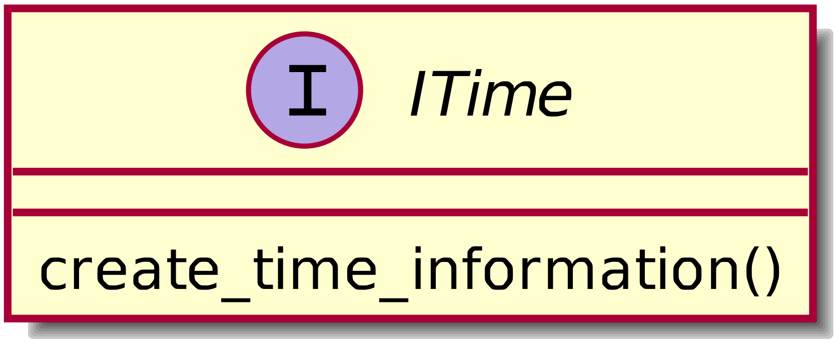


1.2.4 Create appointment



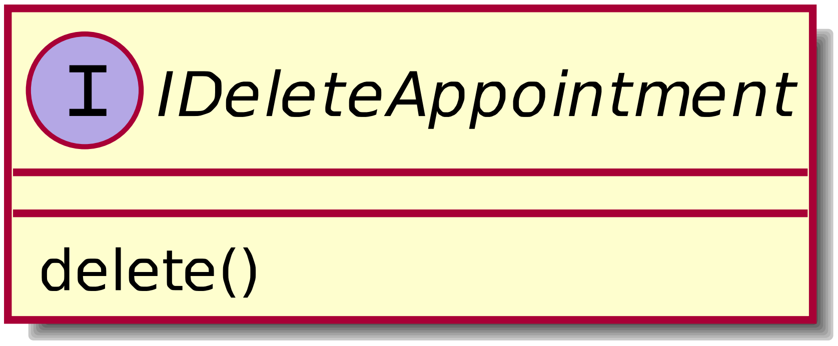
1.2.5 Time management

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

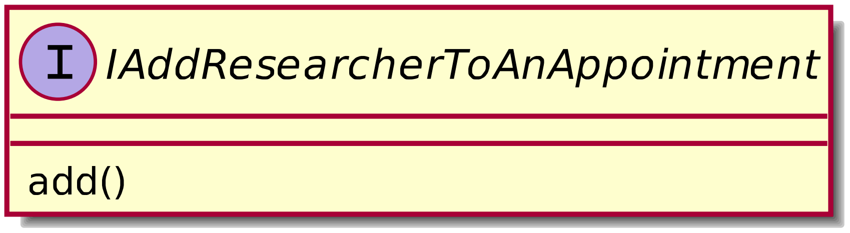


1.2.6 Delete appointment

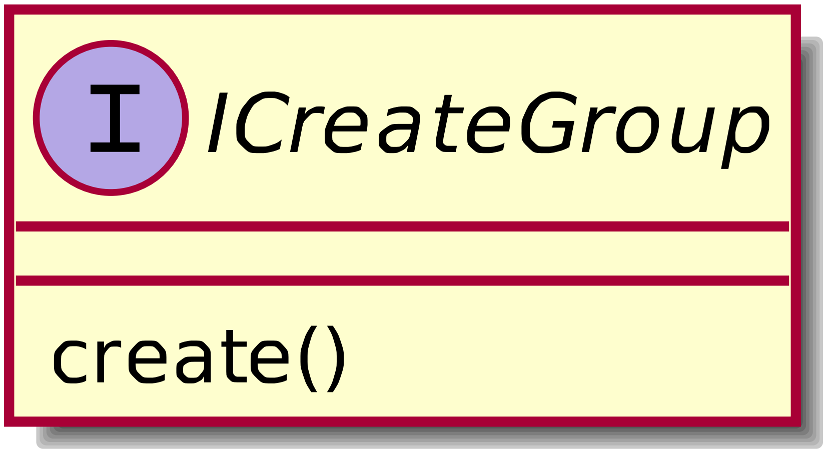
Delete operation deletes appointment entity if the invoker of this command is the creator of the appointment. And appointment is deleted only from ivoker's schedule if he is just participant of the appointment.



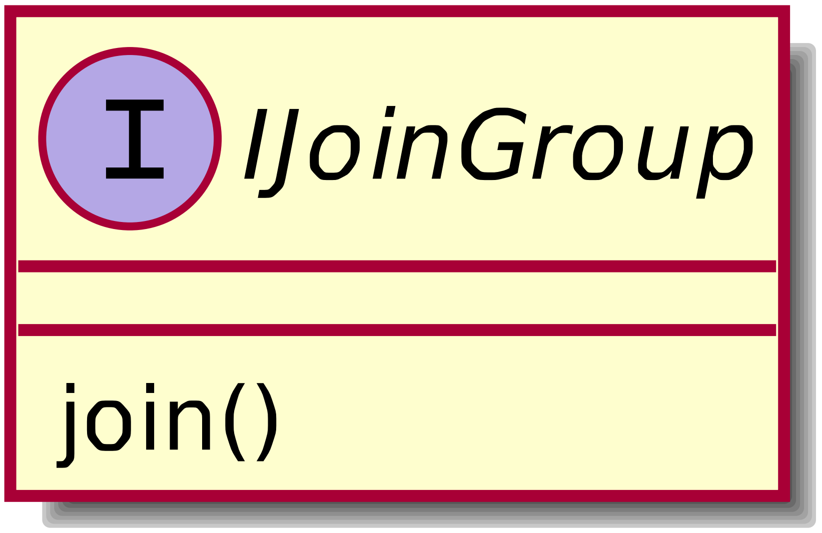
1.2.7 Add researcher to an appointment



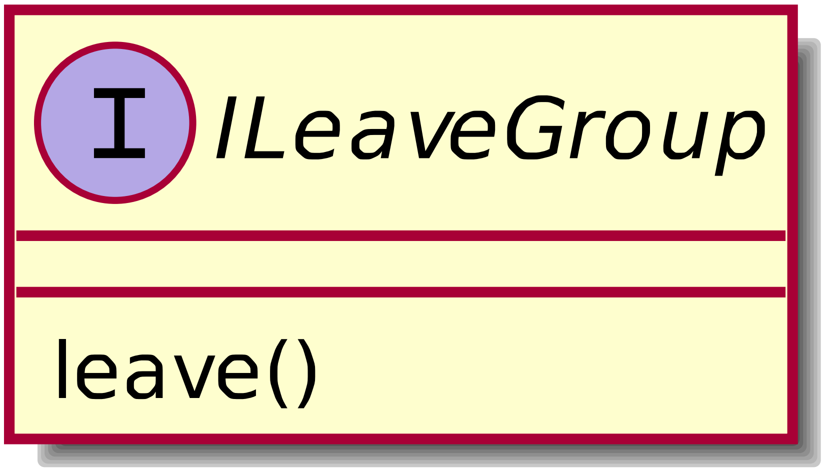
1.2.8 Create group



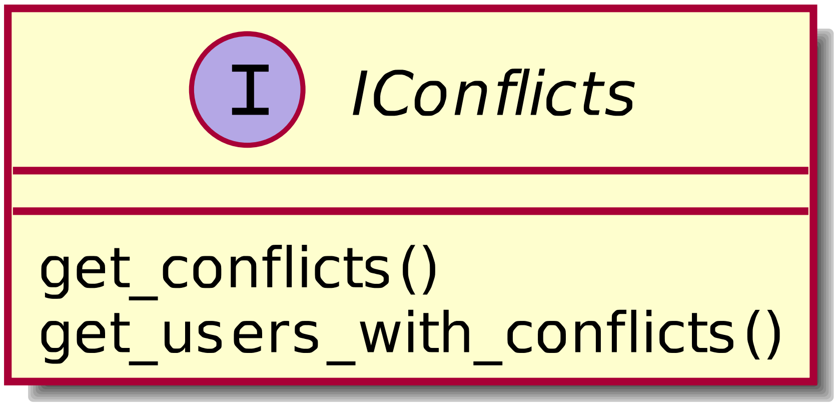
1.2.9 Join group



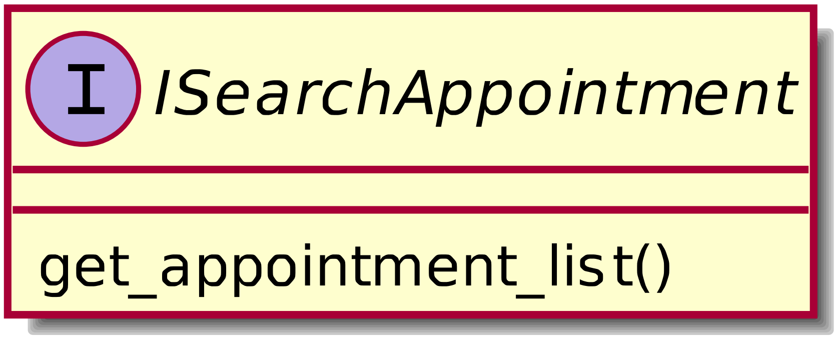
1.2.10 Leave group



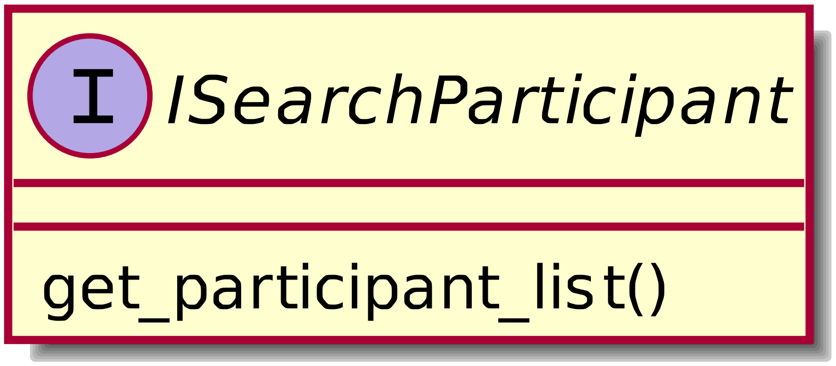
1.2.11 Conflicts



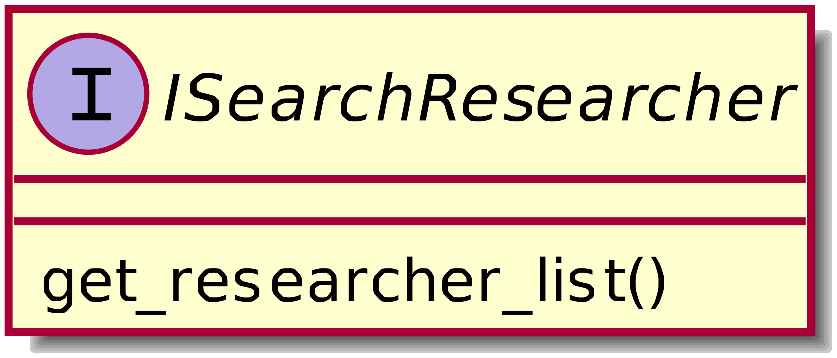
1.2.12 Search appointment



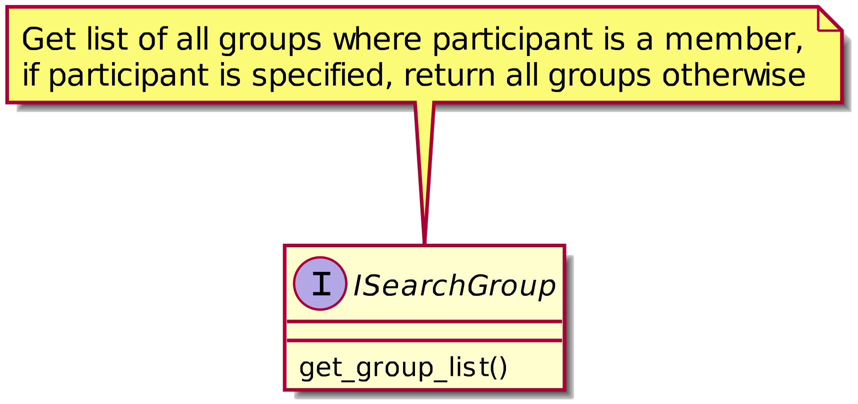
1.2.13 Search participant



1.2.14 Search researcher



1.2.15 Search group


Get list of all groups where participant is a member,
if participant is specified, return all groups otherwise
I
ISearchGroup
get_group_list()

1.3 DONE Component Interaction

1.3.1 Collaboration Diagrams

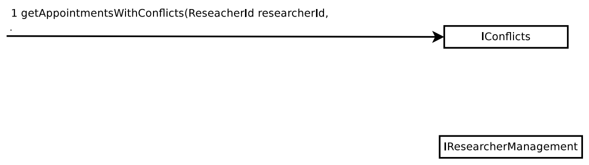


Figure 1: IConflicts

1.3.2 Auxiliary types

1. Auxiliary Types

   (a) Types for referring to objects

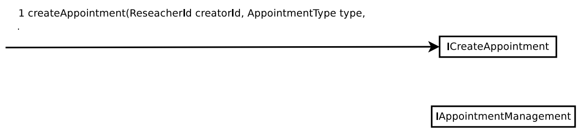

1 createAppointment(ReseacherId creatorId, AppointmentType type,
ICreateAppointment
IAppointmentManagement

Figure 2: ICreateAppointment

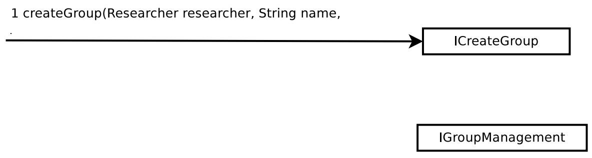


Figure 3: ICreateGroup

- ResearcherId, AppointmentId, GroupId

(b) Types for communicating categories

```
GroupType -> enum {PROJECT_GROUP, RESEARCH_GROUP}

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

(c) Types for passing around data
GroupDetails

- 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.

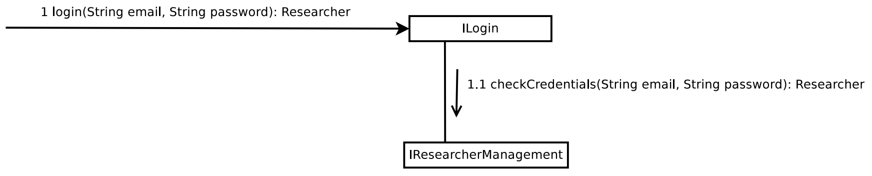

1 login(String email, String password): Researcher
ILogin
1.1 checkCredentials(String email, String password): Researcher
IResearcherManagement

Figure 4: ILogin

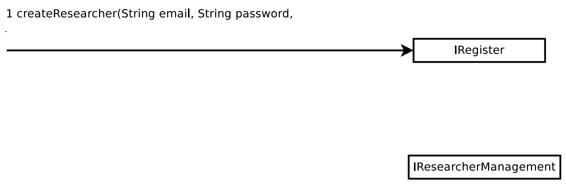


Figure 5: IRegister

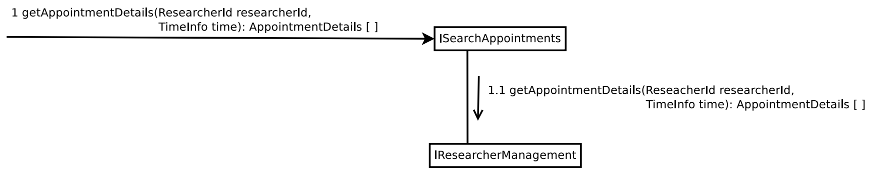

1 getAppointmentDetails(ResearcherId researcherId, TimeInfo time): AppointmentDetails [ ]
ISearchAppointments
1.1 getAppointmentDetails(ReseacherId researcherId, TimeInfo time): AppointmentDetails [ ]
IResearcherManagement

Figure 6: ISearchAppointment

ResearcherDetails

- rid
- emailAddress
- firstname
- lastname
- title
- phoneNbr

AppointmentDetails

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

TimeInterval

- start
- end

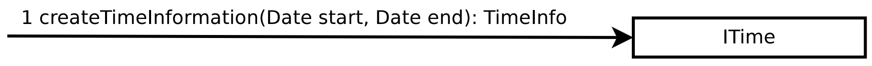
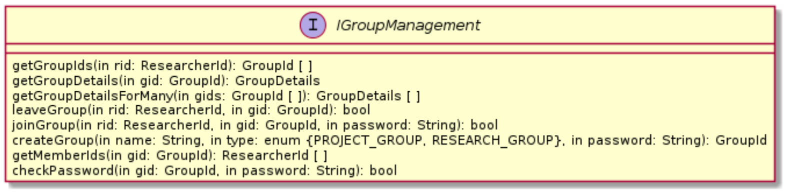


Figure 7: ITime

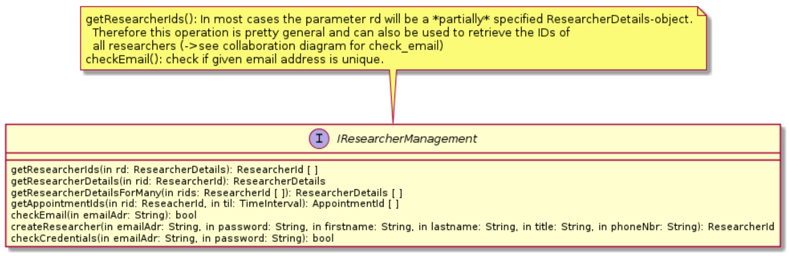
1.3.3 Business Interfaces

1. Group Management



```
IGroupManagement

getGroupIds(in rid: ResearcherId): GroupId [ ]
getGroupDetails(in gid: GroupId): GroupDetails
getGroupDetailsForMany(in gids: GroupId [ ]): GroupDetails [ ]
leaveGroup(in rid: ResearcherId, in gid: GroupId): bool
joinGroup(in rid: ResearcherId, in gid: GroupId, in password: String): bool
createGroup(in name: String, in type: enum {PROJECT_GROUP, RESEARCH_GROUP}, in password: String): GroupId
getMemberIds(in gid: GroupId): ResearcherId [ ]
checkPassword(in gid: GroupId, in password: String): bool
```

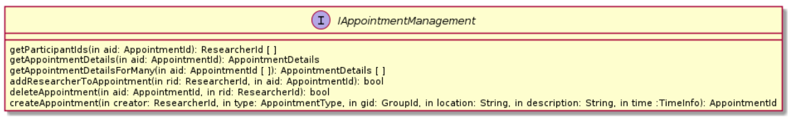
2. Researcher Management



```
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.
```

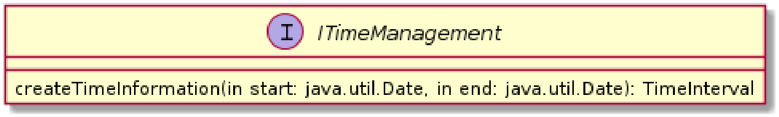
```
IResearcherManagement

getResearcherIds(in rd: ResearcherDetails): ResearcherId [ ]
getResearcherDetails(in rid: ResearcherId): ResearcherDetails
getResearcherDetailsForMany(in rids: ResearcherId [ ]): ResearcherDetails [ ]
getAppointmentIds(in rid: ReseacherId, in til: TimeInterval): AppointmentId [ ]
checkEmail(in emailAdr: String): bool
createResearcher(in emailAdr: String, in password: String, in firstname: String, in lastname: String, in title: String, in phoneNbr: String): ResearcherId
checkCredentials(in emailAdr: String, in password: String): bool
```

3. Appointment Management



```
IAppointmentManagement

getParticipantIds(in aid: AppointmentId): ResearcherId [ ]
getAppointmentDetails(in aid: AppointmentId): AppointmentDetails
getAppointmentDetailsForMany(in aid: AppointmentId [ ]): AppointmentDetails [ ]
addResearcherToAppointment(in rid: ResearcherId, in aid: AppointmentId): bool
deleteAppointment(in aid: AppointmentId, in rid: ResearcherId): bool
createAppointment(in creator: ResearcherId, in type: AppointmentType, in gid: GroupId, in location: String, in description: String, in time :TimeInfo): AppointmentId
```

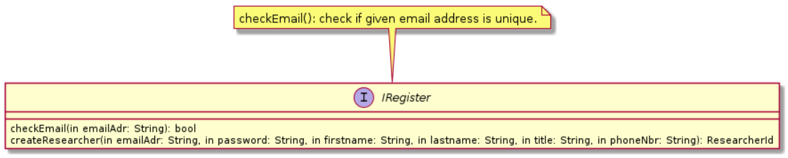
4. Time Management



```
ITimeManagement

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

1.3.4 System Interfaces

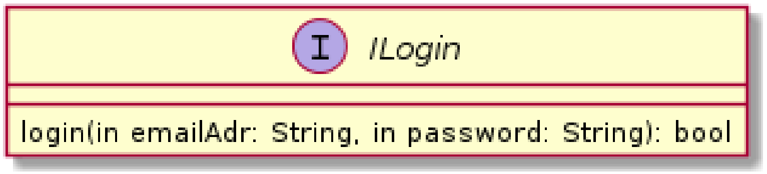
1. Register



```
checkEmail(): check if given email address is unique.
```

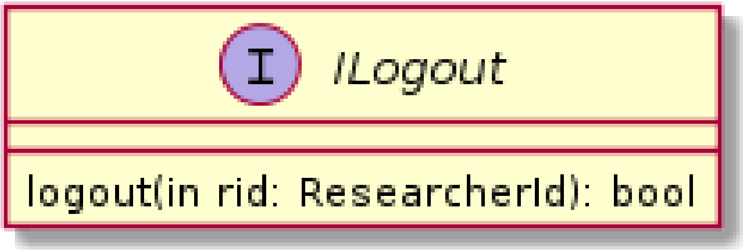
```
IRegister

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

2. Login



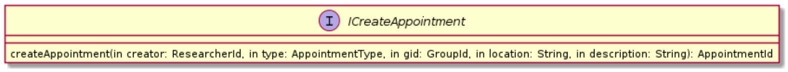
```
ILogin

login(in emailAdr: String, in password: String): bool
```

3. Logout



```
ILogout
logout(in rid: ResearcherId): bool
```

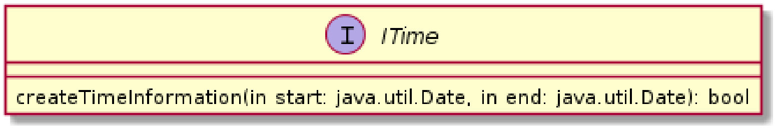
4. Create appointment



```
ICreateAppointment
createAppointment(in creator: ResearcherId, in type: AppointmentType, in gid: GroupId, in location: String, in description: String): AppointmentId
```

5. Time management

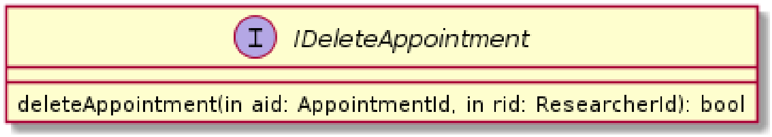
   We do this on the client side by constructing an object that encapsulates time range information.



```
ITime
createTimeInformation(in start: java.util.Date, in end: java.util.Date): bool
```

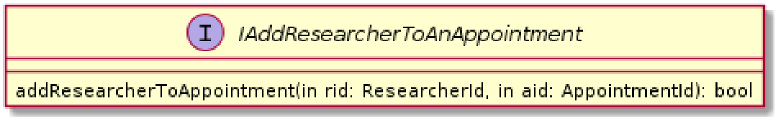
6. Delete appointment

   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.



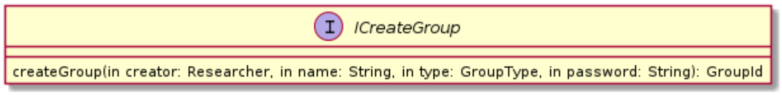
```
IDeleteAppointment
deleteAppointment(in aid: AppointmentId, in rid: ResearcherId): bool
```

7. Add researcher to an appointment



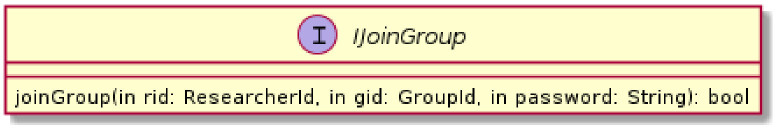
```
IAddResearcherToAnAppointment
addResearcherToAppointment(in rid: ResearcherId, in aid: AppointmentId): bool
```

8. Create group

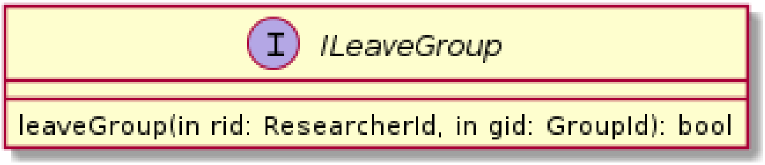


```
ICreateGroup
createGroup(in creator: Researcher, in name: String, in type: GroupType, in password: String): GroupId
```

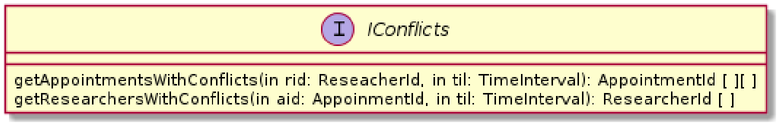
9. Join group


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

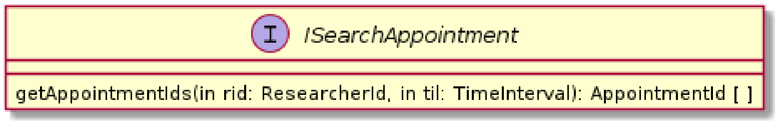
10. Leave group



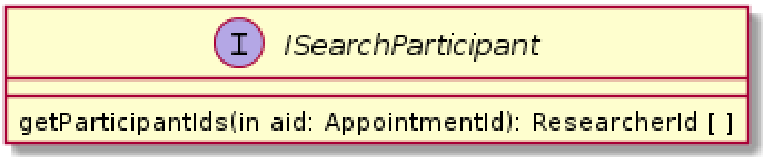
11. Conflicts


I
IConflicts
getAppointmentsWithConflicts(in rid: ReseacherId, in til: TimeInterval): AppointmentId [ ][ ]
getResearchersWithConflicts(in aid: AppoinmentId, in til: TimeInterval): ResearcherId [ ]

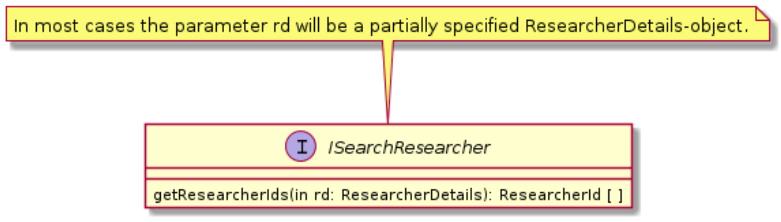
12. Search appointment

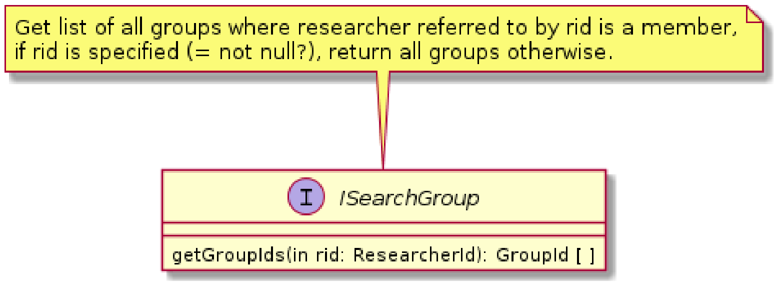

I
ISearchAppointment
getAppointmentIds(in rid: ResearcherId, in til: TimeInterval): AppointmentId [ ]

13. Search participant



14. Search researcher


In most cases the parameter rd will be a partially specified ResearcherDetails-object.
I
ISearchResearcher
getResearcherIds(in rd: ResearcherDetails): ResearcherId [ ]


Get list of all groups where researcher referred to by rid is a member,
if rid is specified (= not null?), return all groups otherwise.
ISearchGroup
getGroupIds(in rid: ResearcherId): GroupId [ ]

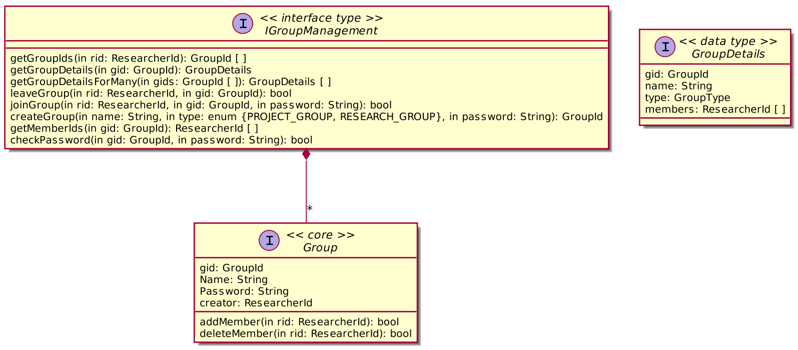
15. Search group

1.4 DONE Component Specification

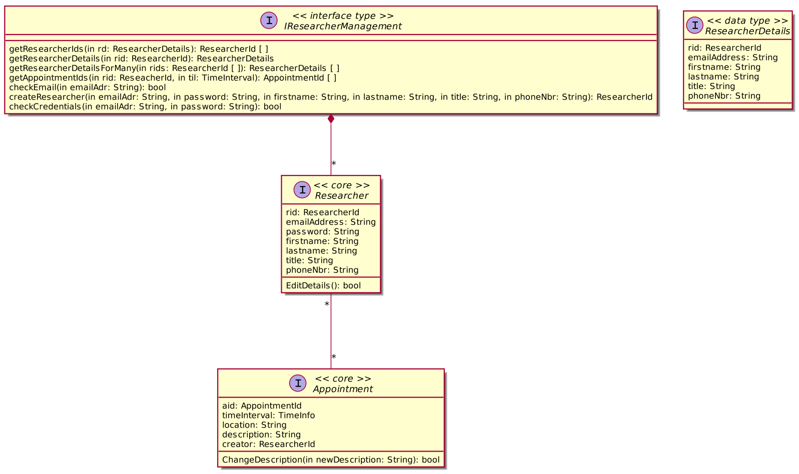
1.4.1 Business interface specification diagrams

1. General notes:

   - ResearcherId, AppointmentId, GroupId are all unsigned integers.
   - GroupType is enum $\{\text{PROJECT}_{\text{GROUP}}, \text{RESEARCH}_{\text{GROUP}}\}$
   - AppointmentType is enum $\{\text{PROJECT}_{\text{GROUPMEETING}}, \text{RESEARCH}_{\text{GROUPMEETING}}, \text{TEACHING}_{\text{APPOINTMENT}}, \text{CONFERENCE}_{\text{APPOINTMENT}}, \text{GENERIC}_{\text{APPOINTMENT}}\}$

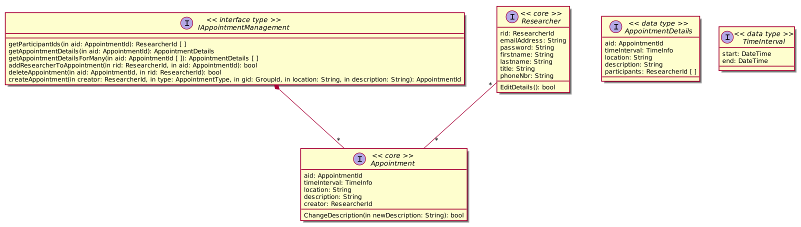

<< interface type >>
IGroupManagement
getGroupIds(in rid: ResearcherId): GroupId [ ]
getGroupDetails(in gid: GroupId): GroupDetails
getGroupDetailsForMany(in gids: GroupId [ ]): GroupDetails [ ]
leaveGroup(in rid: ResearcherId, in gid: GroupId): bool
joinGroup(in rid: ResearcherId, in gid: GroupId, in password: String): bool
createGroup(in name: String, in type: enum {PROJECT_GROUP, RESEARCH_GROUP}, in password: String): GroupId
getMemberIds(in gid: GroupId): ResearcherId [ ]
checkPassword(in gid: GroupId, in password: String): bool
<< data type >>
GroupDetails
gid: GroupId
name: String
type: GroupType
members: ResearcherId [ ]
*
<< core >>
Group
gid: GroupId
Name: String
Password: String
creator: ResearcherId
addMember(in rid: ResearcherId): bool
deleteMember(in rid: ResearcherId): bool

2. IGroupManagement



<< interface type >>
IResearcherManagement
getResearcherIds(in rd: ResearcherDetails): ResearcherId [ ]
getResearcherDetails(in rid: ResearcherId): ResearcherDetails
getResearcherDetailsForMany(in rids: ResearcherId [ ]): ResearcherDetails [ ]
getAppointmentIds(in rid: ReseacherId, in til: TimeInterval): AppointmentId [ ]
checkEmail(in emailAdr: String): bool
createResearcher(in emailAdr: String, in password: String, in firstname: String, in lastname: String, in title: String, in phoneNbr: String): ResearcherId
checkCredentials(in emailAdr: String, in password: String): bool
<< data type >>
ResearcherDetails
rid: ResearcherId
emailAddress: String
firstname: String
lastname: String
title: String
phoneNbr: String
*
<< core >>
Researcher
rid: ResearcherId
emailAddress: String
password: String
firstname: String
lastname: String
title: String
phoneNbr: String
EditDetails(): bool
*
*
<< core >>
Appointment
aid: AppointmentId
timeInterval: TimeInfo
location: String
description: String
creator: ResearcherId
ChangeDescription(in newDescription: String): bool

3. IResearcherManagement



<< interface type >>
IAppointmentManagement
getParticipantIds(in aid: AppointmentId): ResearcherId [ ]
getAppointmentDetails(in aid: AppointmentId): AppointmentDetails
getAppointmentDetailsForMany(in aid: AppointmentId [ ]): AppointmentDetails [ ]
addResearcherToAppointment(in rid: ResearcherId, in aid: AppointmentId): bool
deleteAppointment(in aid: AppointmentId, in rid: ResearcherId): bool
createAppointment(in creator: ResearcherId, in type: AppointmentType, in gid: GroupId, in location: String, in description: String): AppointmentId
<< core >>
Researcher
rid: ResearcherId
emailAddress: String
password: String
firstname: String
lastname: String
title: String
phoneNbr: String
EditDetails(): bool
<< data type >>
AppointmentDetails
aid: AppointmentId
timeInterval: TimeInfo
location: String
description: String
participants: ResearcherId [ ]
<< data type >>
TimeInterval
start: DateTime
end: DateTime
*
*
*
<< core >>
Appointment
aid: AppointmentId
timeInterval: TimeInfo
location: String
description: String
creator: ResearcherId
ChangeDescription(in newDescription: String): bool

4. IAppointmentManagement

1.4.2 System interface specification diagrams

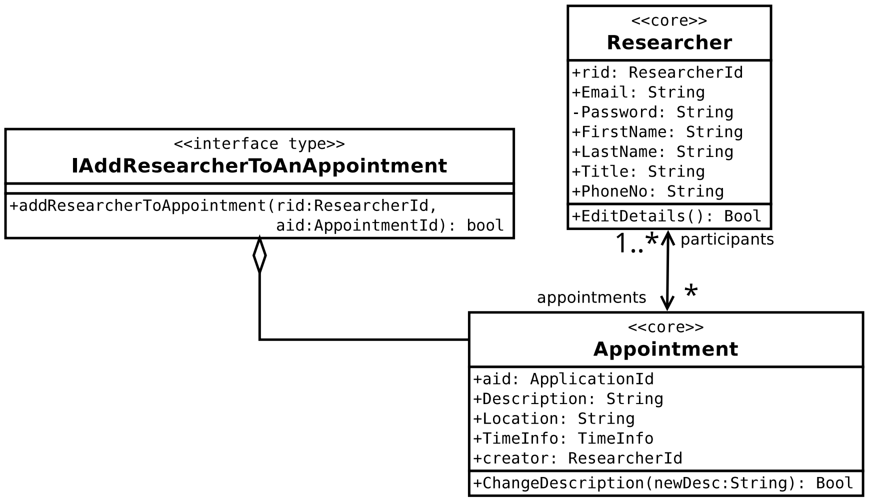


Figure 8: iAddResearcherToAnAppointment

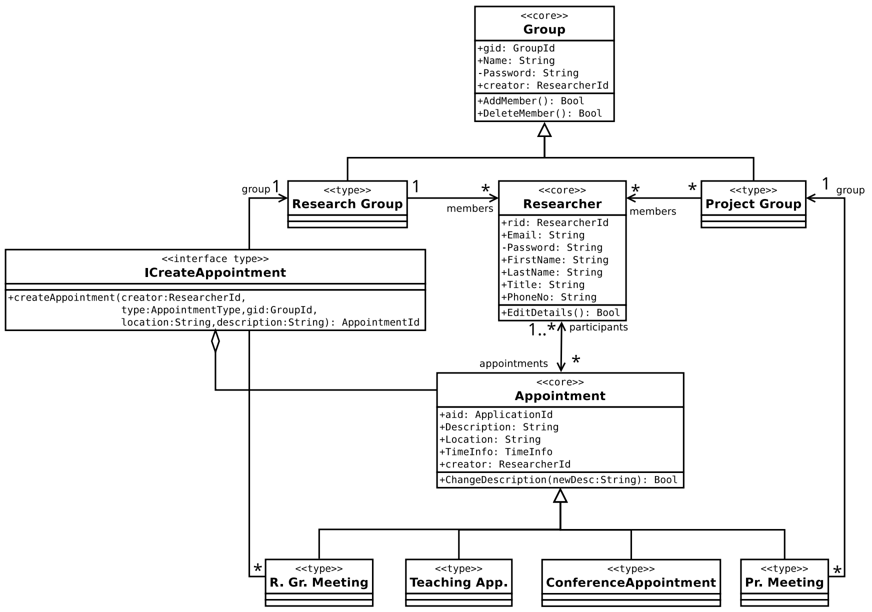


Figure 9: iCreateAppointment

1.4.3 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.
```

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

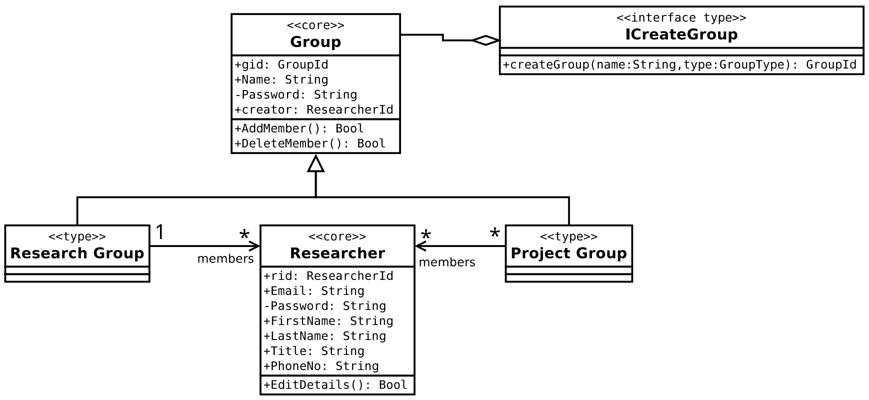


Figure 10: iCreateGroup

`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.`

createAppointment(in creator: ResearcherId, in type: AppointmentType, in gid: GroupId, in location: String, in description: String): AppointmentId

`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.`

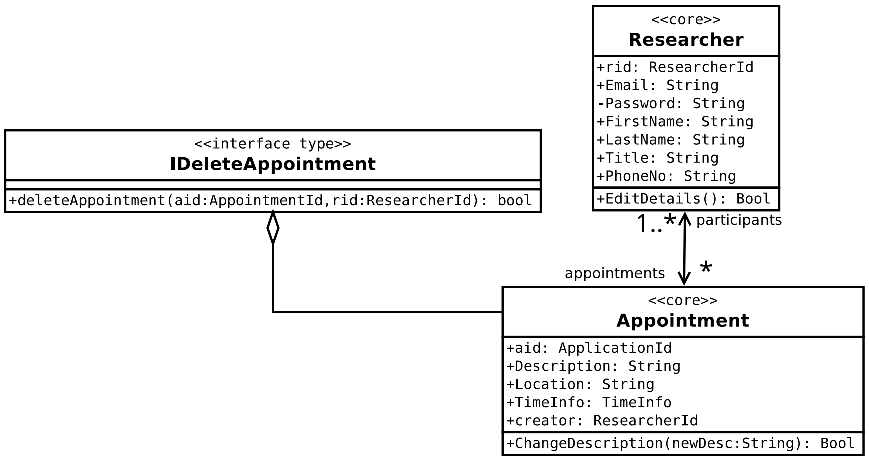


Figure 11: iDeleteAppointment

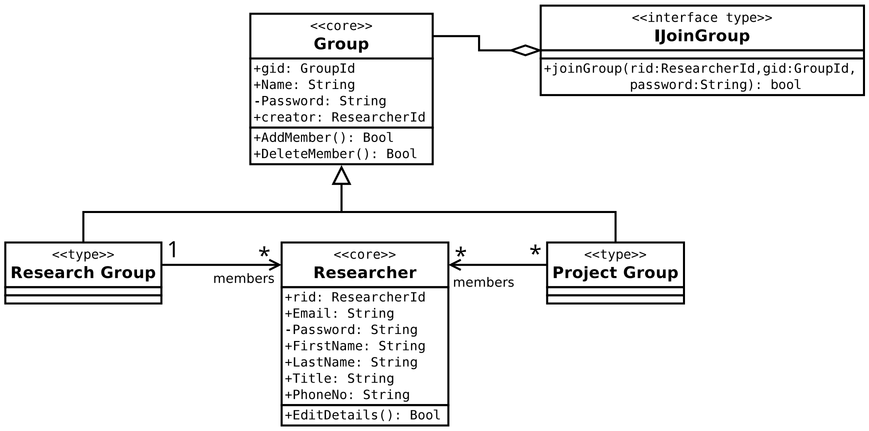


Figure 12: iJoinGroup

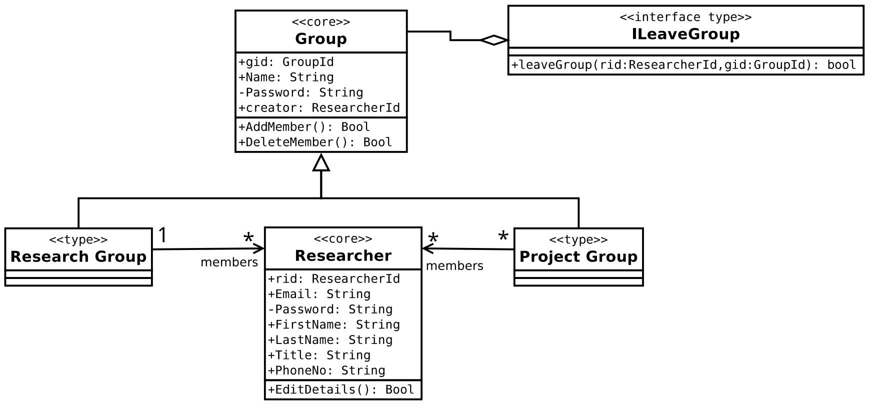


Figure 13: iLeaveGroup

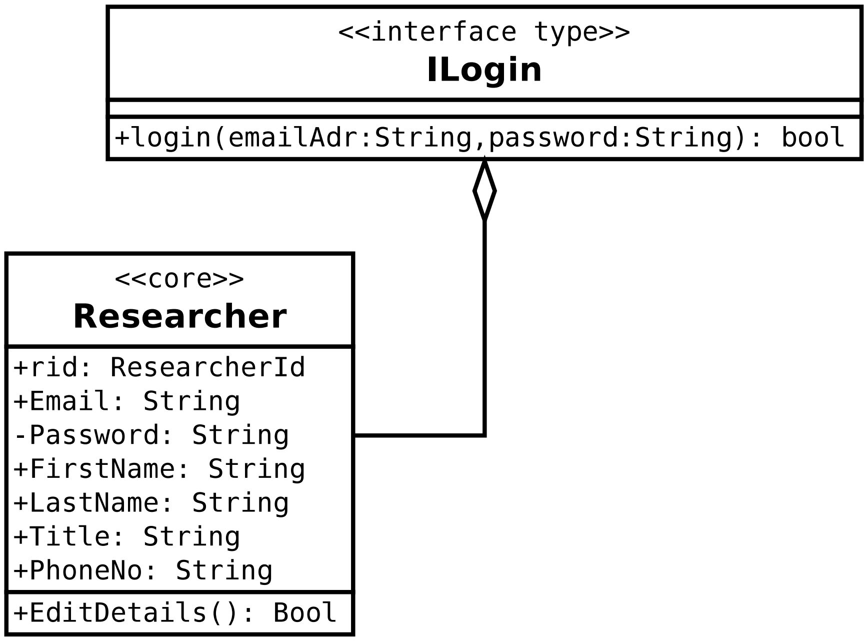


Figure 14: ilogin

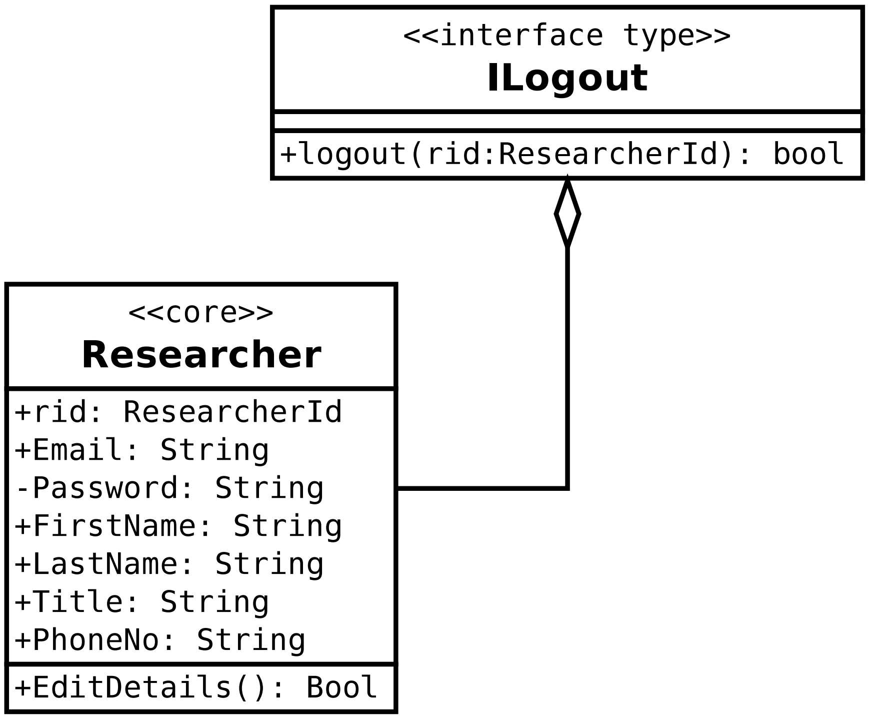


Figure 15: ilogout

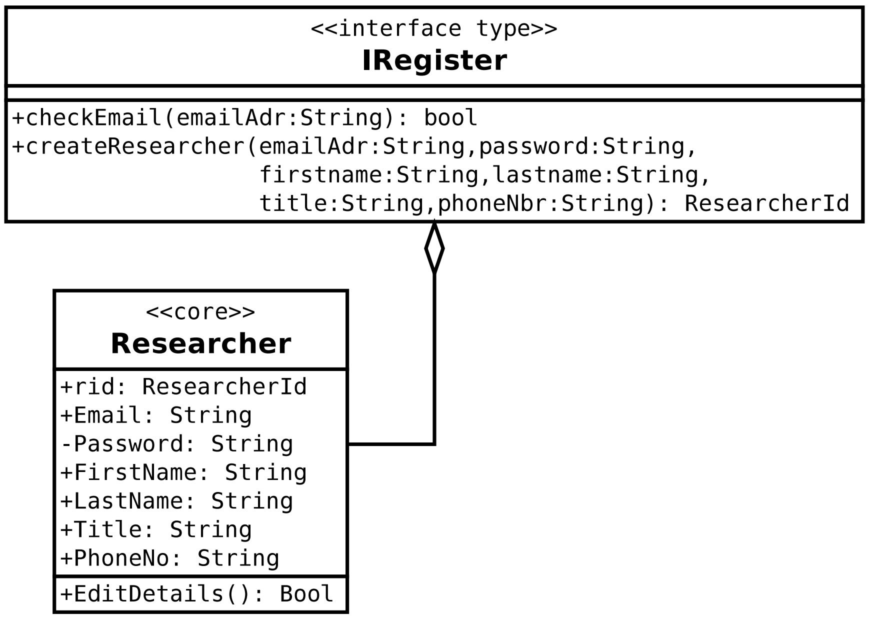


Figure 16: iregister