Note: We aknowledge that for deliverable #1 the goal was to "implement the student and tutor registration forms" (and set up Github users and repositories). Also, deliverable #2 was about implementing administrator functionalities, allowing admins to view their inbox of requests (of student and tutor registration froms developed in deliverable #1), accept requests, reject request, and view list of rejected requests allowing them to change their decision on rejected requests.

And how, the respective method functionalities for students, and tutors would be developed in later deliverables. Despite this, we still decided to include the respective methods in this UML diagram, as class diagrams typically include respective attributes, and methods.

Non-administrators only gain their functionality once their request has been approved by administrator

Why: Assuming only one administrator. This relation is a **many to one relation** (for non-adminstrators to administrators). Since an Administrator would administrate multiple non-administartors (students and tutors)

Keynote: The Administrator is a pre-registered user (their username and password are seeded in the database when the system is first launched).

Note: Decided to make the vast majority of methods and variables private in this UML diagram to follow good software engineering practices. It is best to keep class attributes at the lowest possible access level to promote modularity.

Non-Administrators

-firstName: String
-lastname: String
-emailAddress (username): String
-accoundPassword: String
-phoneNumber: int

*

Administrates

1

Administrators

-Username: String
-Password: String

-approveRegistration()
-dissaproveRegistration()

-viewInbox()
-viewRejectedRequests()

*Keynote: As discussed in deliverable two, administrators can look back at their list of rejected requests and change their status. However, when they accept an application, they can't change it back to a pending or rejected request. Citation from deliverable two: *"The Administrator can view the list of registrations they previously rejected. They can select any request from this list and change their decision to approve the registration. However, once approved, the decision can no longer be reversed."*

Why: Each **Student** is taught by **one or more Tutors** (hence the 1...* on the student side), and each **Tutor** teaches **one or more Students** (hence the 1...* on the tutor side).

Students

-programOfStudy: String

~viewUpcomingTeachingSessions()
~cancelExistingTeachingSession()
-viewPastSessions()
-cancelTutorSession()
-rateTutor()
-bookSession()

1...*

Teaches

1...*

Tutors

-highestDegree: String
-coursesOffered: String

~viewUpcomingTutorSessions()
~viewPastTutorSessions()
-cancelTeachingSessions()
-approveSessionRequest()
-rejectSessionRequest()

-toggleAutoApprove(): bol

*Keynote: Decided to add a 'toggle auto approve' function, since in project description it was stated that tutors can: "elect to auto-approve all session requests, so they do not have to act on each one individually. Requests are simply accepted automatically."