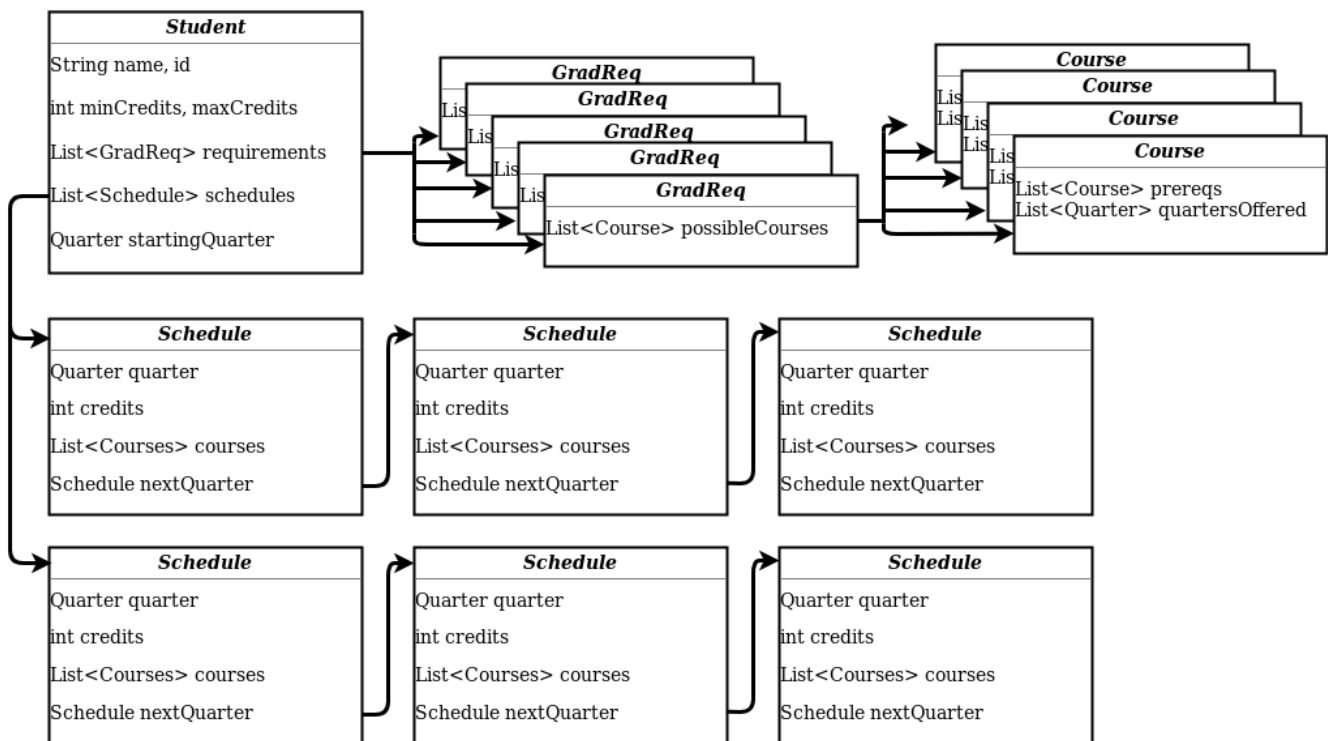# Data Types

A Student object will contain student attributes such as their name, student id, and starting quarter, and minimum and maximum quarterly credits. Their graduation requirements can be assigned to them using a list of GradReq objects.

GradReq represents a single graduation requirement, such as "English proficiency" or "Data Structures". It contains a list of Course objects representing the courses that can be used to fulfill that requirement. It can represent a single course, such as "Discrete Math", or a number of courses, such as "CS Elective".

Each Course object will contain both a list of the course's prerequisites as well as the quarters that the course is offered.

A generated student schedule will be represented by Schedule objects. Each schedule object will contain the quarter that the schedule represents, the number of credits taken, a list of courses, and a pointer to the schedule for the next quarter.

Student
String name, id
int minCredits, maxCredits
List<GradReq> requirements
List<Schedule> schedules
Quarter startingQuarter

GradReq
List<Course> possibleCourses

Course
List<Course> prereqs
List<Quarter> quartersOffered

Schedule
Quarter quarter
int credits
List<Courses> courses
Schedule nextQuarter

Schedule
Quarter quarter
int credits
List<Courses> courses
Schedule nextQuarter

Schedule
Quarter quarter
int credits
List<Courses> courses
Schedule nextQuarter

Schedule
Quarter quarter
int credits
List<Courses> courses
Schedule nextQuarter

Schedule
Quarter quarter
int credits
List<Courses> courses
Schedule nextQuarter

Schedule
Quarter quarter
int credits
List<Courses> courses
Schedule nextQuarter

Algorithm

To make a schedule, the algorithm will have to generate a list of possible courses that the student can take each quarter. To do this, the Student object should take a list of all Course objects within each of their GradReq's, and return a list of courses that meet the following requirements:

- Each course fulfills a graduation requirement.
- The student meets all prerequisites for each course.
- The course is offered during that quarter.
- The student is not currently taking the course, and has not already taken the course.

When returning a course, the algorithm should also specify which graduation requirement it is fulfilling. Therefore, if one course can fulfill multiple different graduation requirements, the algorithm would examine each possibility.

Because of the huge number of possibilities of course schedules, we must take as many measures as possible to trim calculations from the possibility tree. Because the algorithm should return a number of possible schedules, instead of just the one best schedule, the Student object should hold a list of Schedules of either a predetermined or user-inputted size, and then further Schedules should be compared against the worst (longest) schedule in the Student's list. For each iteration of the GenerateSchedule algorithm, the following scenarios should be checked before allowing a Schedule to be developed further:

- Check that the current schedule is not already longer than the longest schedule in the list.
- Check that the current schedule cannot be completed in less time than the longest schedule in the list. This can be accomplished by calculating:
  Credits remaining / Maximum credits per quarter = Minimum quarters remaining.
- Check that the current schedule is not too similar to another schedule in the list. This scenario is likely to occur when an early graduation requirement has many courses that can be used. For example, if "English proficiency" has 5 different courses that can be used, and the algorithm is instructed to generate 5 schedules, it is not particularly useful to generate 5 schedules with a different "English proficiency" course in the first quarter, but every subsequent quarter is identical.

Pseudocode

/*
Example pseudocode of the schedule generating algorithm
Given a student and a blank schedule object, it recursively generates permutations of course schedules.
*/

```
GenerateSchedule( Student, current Schedule )
        if student has no remaining graduation requirements
                add current schedule to the list of the student's schedules
                return
        else if this schedule is worse than the schedules already generated
                return
        else
                make list of all possible courses for current quarter
                if list is empty
                        GenerateSchedule( Student, next Quarter )
                else
                        for each course C in possible courses
                                if course fits in students current schedule
                                        add to current schedules
                                        GenerateSchedule( Student, current Schedule )
                                else
                                        GenerateSchedule( Student, next Schedule )
        return
```

Example scenario tree

Given scenario: Student can take max 10 credits per quarter
Courses / GradReqs: [CS101] [CS110] [ENG101 – Only fall qtr] [ENG102 – Requires ENG101]
Legend: →denotes current quarter, P denotes the courses that the student is able to take that quarter

```
                                    →F18:
                            P: CS101, CS110, ENG101
```

```
→F18: CS101              →F18: CS110              →F18: ENG101
P: CS110, ENG101         P: CS101, ENG101         P: CS101, CS110
```

```
                         →F18: CS110, CS101       →F18: ENG101, CS101
                         P: ENG101                P: CS110
```

```
→F18: CS101, CS110       Can't add                Identical to
P: ENG101                ENG101 to                another
                         current                  schedule
                         quarter
```

```
Can't add                F18: CS110, CS101        →F18: CS110. ENG101      →F18: ENG101, CS110
ENG101                   →W19:                    P: CS101                 P: CS101
to fall qtr              P:ENG101
```

```
F18: CS101, CS110        Identical to             Can't add                Identical to
→W19:                    another                  CS101 to                 another
P: None                  schedule                 current                  schedule
                                                  quarter
```

```
F18: CS101, CS110                                 F18: CS110, ENG101
W19: None                                         →W19:
→S19:                                             P: CS101, ENG102
P: None
```

```
F18: CS101, CS110     →F18: CS101, ENG101         F18: CS110, ENG101
W19:                  P: CS110                     →W19: ENG102
S19:                                               P: CS101
→F19:
P: ENG101
```

```
Worse than            Can't add                   Identical to
other                 CS110 to                    another
schedule              current                     schedule
                      quarter
```

```
                      F18: CS101, ENG101          F18: CS110, ENG101
                      →W19: CS110                 →W19: CS101
                      P: ENG102                   P: ENG102
```

```
                      F18: CS101, ENG101          F18: CS110, ENG101
                      →W19: CS110, ENG102         →W19: CS101, CS102
                                                  P: None
```

```
                      All                         All
                      requirements                requirements
                      completed                   completed
```