CLASSES, ATTRIBUTES and OPERATIONS:

Class #1: Students < The system will also be used to monitor and manage student..>

- uusid attr: <Each student will have a "UUSID" (University Unique Student Identifier) …>
- name attr: <The system must be able to track some basic identifying information for each of the students, to include names, …>
- address attr: <The system must be able to track some basic identifying information for each of the students, to include … addresses …>
- phoneNum attr: <The system must be able to track some basic identifying information for each of the students, to include … phone numbers …>
- enrolledDegree attr: <A student may only be enrolled in one degree program >
- arePreReqsSatisfied() op: < Some of the more advanced courses might also have prerequisite courses> and < course prerequisites and passing scores >
- selectDegree() op: < A student may only be enrolled in one-degree program … they will be allowed to switch between programs … >
- selectCourse() op: < Students will select courses from the most current Course Catalog > and < opportunity to take a wide variety of courses >
- requestCourse() op: < Allowing students to request a course >
- getRecords() op: < Report: The list of courses that a student has completed>
- getTermCost() op: < Report: The costs that a student has paid to take courses during a certain term, and overall>
- addRecord() op: < … maintain the records of courses taken … >
- getRecords() op: < The list of courses that a student has completed … >

Class #2: Record < … maintain the records of courses … >

- termId attr: < maintain the records of courses taken > and < the term and year must also be maintained for each attempt >
- courseId attr: < maintain the records of courses taken > and < the term and year must also be maintained for each attempt >
- termCourseId attr: < maintain the records of courses taken > and < the term and year must also be maintained for each attempt >
- Grade attr: < maintain the records of courses taken > and < the term and year must also be maintained for each attempt > and < measured by a final letter grade >
- getRecord() op: < The list of courses that a student has completed >

Class #3: Instructors < course can only be taken if one or more designated instructors >

- uuiid attr: < Instructors will also have a University Unique Instructor ID (UUIID) >
- name attr: < system must be able to store the names … for the instructors >
- officeHours attr: < system must be able to store the … office hours … for the instructors >
- email attr: < system must be able to store the … e-mail addresses for the instructors >
- qualifiedCourses attr: < a given course being offered by a qualified instructor >
- termCourseID attr: < a given course being offered by a qualified instructor > and < Qualified instructors must teach these courses >
- isQualifiedToTeach() op: < … course being offered by a qualified instructor > and < Qualified instructors must teach these courses >

- manageGrades() op: < to enter academic records and final grades >

Class #4: Courses < to offer online courses>

- courseID attr: < Each course must have a distinct Course ID>
- courseTitle attr: < Each course must have a … Course Title >
- courseDesc attr: < Each course must have a … Description>
- preReqs attr: < courses might also have prerequisite courses >
- costs attr: < courses also have different costs >
- managePreReq() op: < … courses might also have prerequisite courses … >
- getPreReqs() op: < … courses might also have prerequisite courses … >

Class #5: Grade < [Utility] final letter grade (e.g. A, B, C, D or F) >

- grade attr: < final letter grade: A, B, C, D or F >

Class #6: DegreeProgram < will establish degree programs for the students >

- name attr: < Each degree program will have a clear and distinct name >
- courses attr: < Each degree program will … consist of a set of courses >
- manageCourse() op: < Each degree program will … consist of a set of courses >

Class#7: Catalog < from the most current Course Catalog >

- courses attr: < will select courses from the most current Course Catalog >
- degrees attr: < A student may only be enrolled in one-degree program >

- manageCourse() op: < support the creation of new courses, along with the update and occasional removal of existing courses >
- manageDegree() op: < update the Course Catalog to make sure that they can address new topics, technologies and techniques >

Class#8: Term < enrollment in certain courses during each term >

- year attr: < the term and year must also be maintained for each attempt [of course] >
- semester attr: < the Fall, Spring and/or Summer terms >
- courseMap attr: < these courses might be offered during any (or all) of the Fall, Spring and/or Summer terms >
- manageTermCourse() op: < these courses might be offered during any (or all) of the Fall, Spring and/or Summer terms >
- getCoursesOffered() op: < Allowing students to request a course >

Class#9: TermCourse < these courses might be offered during any (or all) of the Fall, Spring and/or Summer terms >

- termCourseID attr: < [unique ID for] these courses might be offered during any (or all) of the Fall, Spring and/or Summer terms >
- termID attr: < these courses might be offered during any (or all) of the Fall, Spring and/or Summer terms >
- courseID attr: < these [catalog listed] courses might be offered … >

- instructorList attr: < a course can only be taken if one or more designated instructors are offering that course >
- requestList: attr: < … manage student course requests … >
- enrolledList attr: < all of the prerequisite courses must be completed successfully before that student is allowed to request and take [or enroll in] the main course >
- grades attr: < The student's performance will be measured by a final letter grade (e.g. A, B, C, D or F). >
- manageInstructor() op: < Academic Administrators [~~who~~] will assign instructors to courses from term to term >
- manageStudentRequests() op: < … manage student course requests … >
- manageStudentEnrollment () op: < all of the prerequisite courses must be completed successfully before that student is allowed to request and take [or enroll in] the main course >
- manageGrade() op: < The student's performance will be measured by a final letter grade >
- getFinalGrades(): op: < The student's performance will be measured by a final letter grade >

Class#10 {Role Client & System & Admin} <… an Administrator Role will be useful … >

- manageTerm() op: < from term to term [→ manage multiple terms] >
- manageTermCourse() op: < to courses from term to term >
- manageInstructor() op: < For each term, each instructor can be assigned to teach (at most) one course >

- manageStudentEnrollment() op: < Checking course requests to ensure they are valid (e.g. all prerequisites satisfied) >
- processTermGrades() op: < Recording a final grade for the student once the course has been completed >
- getStudentHistory() op: < The list of courses that a student has completed >
- getStudentCosts() op: < The costs that a student has paid to take courses during a certain term, and overall >
- getEnrollment() op: < The number of students enrolled in courses during the current term >
- manageDegree() op: < support the creation of new courses, along with the update and occasional removal of existing courses [assuming in potential new degrees] >
- manageCourse() op: < support the creation of new courses, along with the update and occasional removal of existing courses >
- manageCoursePreReqs() op: < support the creation of new courses, along with the update and occasional removal of existing courses >
- manageCourseInDegree() op: < support the creation of new courses, along with the update and occasional removal of existing courses >

Class#11: Semester <… offered during any (or all) of the Fall, Spring and/or Summer … >

- semester enum: <… any (or all) of the Fall, Spring and/or Summer … >

Class#12: Grade <… grade (e.g. A, B, C, D or F) … >

- grade enum: < grade (e.g. A, B, C, D or F) >

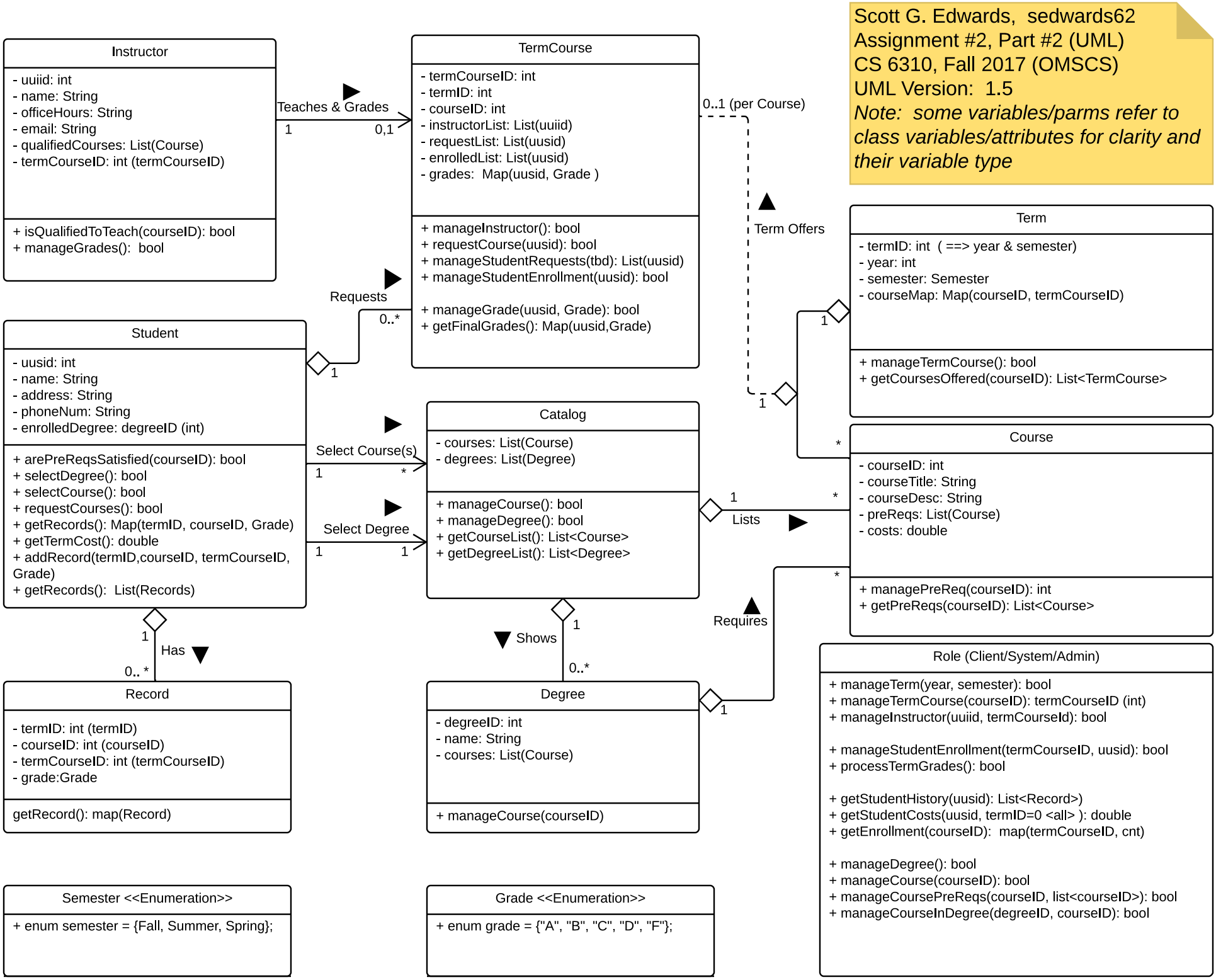
RELATIONSHIPS:

- Instructor Teaches & Grades: Directed Association between Instructor and TermCourse: < For each term, each instructor can be assigned to teach (at most) one course > and < Instructors must be able to enter academic records and final grades once the course has been completed at the end of the term >
- Requests: Aggregation between Student and TermCourse: < The system must also allow students to request enrollment in certain courses during each term >
- Select Courses(s): Directed Association between Student and Catalog < Students will select courses from the most current Course Catalog >
- Select Degree: Directed Association between Student and Catalog < they [students] will be allowed to switch between [degree] programs … >
- Has: Aggregation between Student and Record: < For each course that is taken by a student, an academic record must be maintained that records the student's performance during that session [term] of the course >
- Lists: Aggregation between Catalog and Courses: < courses from the most current Course Catalog >
- Requires: Aggregation between Degree and Courses: < Each degree program will … consist of a set of courses that must be successfully passed >
- Shows: Aggregation between Catalog and Degrees < [no reference – but Catalog is related to both Courses and Degrees, so Catalog chosen to relate both in same class] >

- Term Offers: Aggregation (for offered courses in a term) between Term and Course < these courses might be offered during any (or all) of the Fall, Spring and/or Summer terms >

Discussion/Notes:

1) TermCourse class present since want to separate the catalog's course definition from the term specific requirements of offering a changing set of classes on a semester basis.
2) The containing class for "Degree" is not discussed in problem statement. So, placed in the "Catalog" class since degrees and courses directly related, and changes curriculum typically involve changes in Catalog & Degree & Courses.
3) Specific class unique identifiers listed over & above the problem statement because:
    a. Just finished with DB/CS-6400.
    b. Use the unique identifiers to show interrelationship within the class attributes.
    c. Wanted to be consistent.
4) The Role class is a catch-all for the System Administrator functions that are described in the problem statement as "Client" and "System" and "Administrator". In a future version, these functions might have to be split apart due to client feedback.
5) NEITHER enrollment size nor Instructor-to-Student ratio discussed. UML is set up for one TermCourse section with multiple instructors, but could easily be expanded into multiple sections per Course per Term.


Scott G. Edwards, sedwards62
Assignment #2, Part #2 (UML)
CS 6310, Fall 2017 (OMSCS)
UML Version: 1.5
Note: some variables/parms refer to class variables/attributes for clarity and their variable type
Instructor
- uuiid: int
- name: String
- officeHours: String
- email: String
- qualifiedCourses: List(Course)
- termCourseID: int (termCourseID)
+ isQualifiedToTeach(courseID): bool
+ manageGrades(): bool
Teaches & Grades
1
0,1
TermCourse
- termCourseID: int
- termID: int
- courseID: int
- instructorList: List(uuiid)
- requestList: List(uusid)
- enrolledList: List(uusid)
- grades: Map(uusid, Grade)
+ manageInstructor(): bool
+ requestCourse(uusid): bool
+ manageStudentRequests(tbd): List(uusid)
+ manageStudentEnrollment(uusid): bool
+ manageGrade(uusid, Grade): bool
+ getFinalGrades(): Map(uusid,Grade)
0..1 (per Course)
Term Offers
1
Term
- termID: int (==> year & semester)
- year: int
- semester: Semester
- courseMap: Map(courseID, termCourseID)
+ manageTermCourse(): bool
+ getCoursesOffered(courseID): List<TermCourse>
Requests
0..*
1
Student
- uusid: int
- name: String
- address: String
- phoneNum: String
- enrolledDegree: degreeID (int)
+ arePreReqsSatisfied(courseID): bool
+ selectDegree(): bool
+ selectCourse(): bool
+ requestCourses(): bool
+ getRecords(): Map(termID, courseID, Grade)
+ getTermCost(): double
+ addRecord(termID,courseID, termCourseID, Grade)
+ getRecords(): List(Records)
Select Course(s)
1
*
Catalog
- courses: List(Course)
- degrees: List(Degree)
+ manageCourse(): bool
+ manageDegree(): bool
+ getCourseList(): List<Course>
+ getDegreeList(): List<Degree>
1
*
Lists
Course
- courseID: int
- courseTitle: String
- courseDesc: String
- preReqs: List(Course)
- costs: double
+ managePreReq(courseID): int
+ getPreReqs(courseID): List<Course>
Select Degree
1
1
1
Has
0..*
Record
- termID: int (termID)
- courseID: int (courseID)
- termCourseID: int (termCourseID)
- grade:Grade
getRecord(): map(Record)
Shows
1
0..*
Degree
- degreeID: int
- name: String
- courses: List(Course)
+ manageCourse(courseID)
Requires
1
*
Role (Client/System/Admin)
+ manageTerm(year, semester): bool
+ manageTermCourse(courseID): termCourseID (int)
+ manageInstructor(uuiid, termCourseId): bool
+ manageStudentEnrollment(termCourseID, uusid): bool
+ processTermGrades(): bool
+ getStudentHistory(uusid): List<Record>)
+ getStudentCosts(uusid, termID=0 <all>): double
+ getEnrollment(courseID): map(termCourseID, cnt)
+ manageDegree(): bool
+ manageCourse(courseID): bool
+ manageCoursePreReqs(courseID, list<courseID>): bool
+ manageCourseInDegree(degreeID, courseID): bool
Semester <<Enumeration>>
+ enum semester = {Fall, Summer, Spring};
Grade <<Enumeration>>
+ enum grade = {"A", "B", "C", "D", "F"};