OCL

Report Title:

Flex OCL Class Diagram Analysis for Term Project

Project Members:

- Jawad Shahid **(21L-5787)**
- Romesa Qadeer **(21L-5839)**
- Shaaf Salman **(21L-6083)**

1. Overview

- This document provides comprehensive details regarding the attributes of class diagrams and enumerations in the term project model. For visual representation of the class diagrams.

- please refer to the "Class Diagram Design" folder and navigate to the file located at

> 💡 "\Class Diagram Design\out\ClassDiagram\ClassDiagram.png"

- For OCL editing, please use the provided project.

---

2. Constraint Details

Course Registration System Overview

1. Course Viewing

- **Description:** Students can view the list of courses offered by any department in any semester.

2. Semester Details

- **Description:** Semesters are categorized as Fall, Spring, or Summer for any academic year.

3. Core and Elective Courses

- **Description:** Core courses are mandatory for degree completion, while elective courses offer flexibility in course selection.

4. Course Registration Limits

- **Description:** BS students can register for a maximum of 5 courses per semester, while MS students can register for a maximum of 3 courses, with exceptions for prerequisite courses.

5. Course Withdrawal

- **Description:** Students can withdraw from a course before the announced date.

6. Transcript Representation

- **Description:** Withdrawn courses are marked with a 'W' on the transcript.

7. Course Dropping

- **Description:** Students can drop a course within two weeks of registration, and dropped courses do not appear on the transcript.

8. CGPA Calculation

- **Description:** CGPA is calculated by aggregating all GPAs of semesters taken so far.

Repeat Courses

9. Course Repeat Requirements

- **Description:** Students must repeat all failed courses.

10. Repeat Course Criteria

- **Description:** Students under warning must repeat passed courses with a GPA below the minimum required for their degree.

11. Repeat Course Permissions

- **Description:** Students can repeat a course if not passed or to improve their grade.

12. Transcript Representation

- **Description:** Repeat courses are indicated on the transcript with a repeat count.

Warning System

13. Warning Status Viewing

- **Description:** Students can view the status of their warnings, if any.

14. Warning Issuance

- **Description:** Warnings are issued at the end of every semester if a student's CGPA is below the minimum required for their degree program.

15. Registration Restrictions

- **Description:** Students with a warning cannot register for subsequent semesters without approval from relevant faculty.

16. Warning Count Updates

- **Description:** Warning count increases by one if a student's CGPA is below the required minimum after each semester.

17. Admission Closure

- **Description:** If the warning count reaches three, admission to the university is closed unless the CGPA meets the required minimum.

18. Automatic Admission Cancellation

- **Description:** Admission is automatically canceled after the maximum duration allowed to earn a degree.

Fee Management

19. Fee Details Viewing

- **Description:** Students can view fee details.

20. Fee Payment Timeline

- **Description:** Fees are charged on a semester basis and are due two weeks before the start of the semester during course registration.

21. Payment Method

- **Description:** Fees are paid through bank challan available in the Accounts Office/Flex.

22. Non-Refundable Fees

- **Description:** All fees are non-refundable except for security deposits.

Loan Assistance

25. Loan Eligibility

- **Description:** Indigent students can apply for loans.

26. Loan Renewal

- **Description:** This assistance is subject to renewal every semester based on the student's academic performance and financial need.

27. Loan Coverage

- **Description:** The financial assistance is limited to tuition fees only.

28. Loan Discontinuation

- **Description:** The loan is discontinued if the student's CGPA falls below 2.00 for undergraduate degree and 2.5 for graduate degree.

29. Loan Repayment Period

- **Description:** The repayment of the loan starts three months after graduation or getting a job, whichever is earlier.

30. Loan Repayment Duration

- **Description:** The total amount has to be repaid within a period of four years after graduation. Students are required to sign to this effect.

31. Loan Application Process

- **Description:** All those who are admitted and are in real need of financial assistance should apply on the prescribed Financial Aid form. A notice to this effect will be posted on the notice board.

Merit Scholarship

32. Scholarship Awards

- **Description:** Merit scholarship is awarded to the Top Three position holders of each Examination Board.

33. Campus Merit Scholarship

- **Description:** Scholarship is also offered to top three position holders in the NU admission merit list of each campus.

34. Scholarship Maintenance Criteria

- **Description:** Students awarded merit Scholarship, 100% tuition fee for 8 regular semesters must maintain a CGPA of ≥ 3.00 in each semester.

Attendance Management

35. Attendance Viewing

- **Description:** A student can view the attendance of any course in which he/she has registered.

36. Attendance Criteria for Exam

- **Description:** In order for a student to appear in the Final exam of a course, he/she should have attendance more than 80%.

Results and Exam Requests

37. Marks Viewing

- **Description:** A student can view the marks of each course (detailed view).

38. Grade Change Request

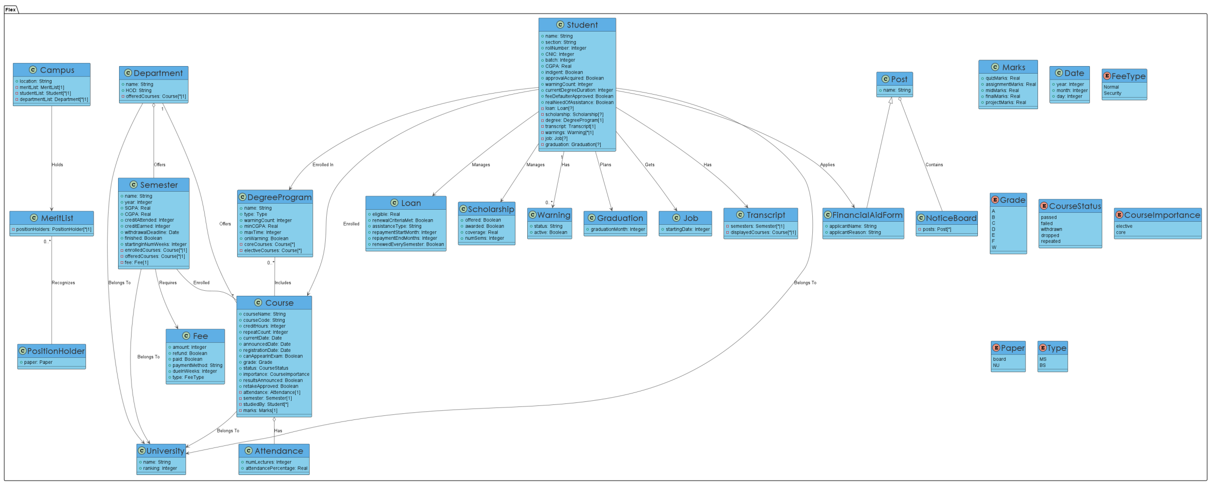
- **Description:** A student can apply for Change in Grade Request after the results are announced.

39. Exam Retake Request

- **Description:** A student’s request to retake any exam is approved by the respective HOD

---

3. Class Diagram Image



2. Classes

2.1. Post

- Attributes: `name`
- Description: Represents a generic post.

2.2. FinancialAidForm

- Attributes: `applicantName` , `applicantReason`
- Description: Extends the Post class to represent a financial aid application form.
- Constraints:
  - **Constraint 31:** Ensures that a student applying for financial aid meets the precondition of demonstrating real need.

2.3. NoticeBoard

- Property: `posts`
- Description: A notice board that aggregates posts.

2.4. Warning

- Attributes: `status` , `active`
- Description: Represents a warning issued to a student based on their academic status.

2.5. Graduation

- Attribute: `graduationMonth`
- Description: Represents the month in which a student graduates.

2.6. Job

- Attribute: `startingDate`
- Description: Represents the starting date of a job.

2.7. Student

- Attributes: Various attributes representing student information.
- Properties: Various properties representing relationships with other classes.
- Constraints:
    - **Constraint 37:** Ensures that all courses in a student's transcript have marks recorded.
    - **Constraint 35:** Ensures that all courses in a student's transcript have attendance records.
    - **Constraint 34:** Ensures that a student receiving a scholarship maintains a CGPA above a specified threshold.
    - **Constraint 30:** Defines the repayment period for a loan based on graduation.
    - **Constraint 29:** Defines the start of the loan repayment period based on graduation and job starting date.
    - **Constraint 28:** Discontinues a student's loan if their CGPA falls below a minimum threshold.
    - **Constraint 24:** Allows a student to give an exam if fees are paid or approved as a fee defaulter.
    - **Constraint 1:** Ensures that a student can view all courses offered by the university.
    - **Constraint 4:** Limits the number of registered courses based on the type of degree.
    - **Constraint 8:** Ensures that dropped courses are not included in the transcript.

- **Constraint 11:** Prohibits a student under warning from repeating low-grade courses.
- **Constraint 9:** Calculates the CGPA based on the student's transcript.
- **Constraint 14:** Ensures that warnings have meaningful status.
- **Constraint 15 and 18:** Issues warnings to students with a CGPA below the minimum threshold.
- **Constraint 17:** Updates the warning count based on finished semesters.
- **Constraint 20:** Ensures that fee details are available for all semesters.

2.8. DegreeProgram

- Attributes: `name`, `type`, `warningCount`, `minCGPA`, `maxTime`, `onWarning`
- Properties: `coreCourses`, `electiveCourses`
- Description: Represents an academic degree program.
- Constraints:
  - **Constraint 3:** Ensures that all core courses for a degree program are taken by students.

2.9. Transcript

- Properties: `semesters`, `displayedCourses`
- Description: Represents a student's academic transcript.
- Constraints:
  - **Constraint 6:** Ensures that withdrawn courses are marked with a grade of 'W'.
  - **Constraint 10:** Tracks the number of attempts for each course.
  - **Constraint 12:** Prohibits repeating a course without passing or indicating a desire to repeat.
  - **Constraint 13:** Indicates repeated courses on the transcript.

2.10. Semester

- Attributes: Various attributes representing semester information.
- Properties: `enrolledCourses`, `offeredCourses`, `fee`

- Description: Represents an academic semester.
- Constraints:
    - **Constraint 2:** Ensures that semester names are valid.
    - **Constraint 21:** Defines the due date for semester fees.

2.11. Department

- Attributes: `name` , `HOD`
- Properties: `offeredCourses`
- Description: Represents an academic department.
- Operations: Various operations related to departmental functions.

2.12. Marks

- Attributes: Various attributes representing marks for different assessments.
- Description: Represents the assessment marks for a course.

2.13. Date

- Attributes: `year` , `month` , `day`
- Description: Represents a date.

2.14. Course

- Attributes: Various attributes representing course information.
- Properties: `attendance` , `semester` , `studiedBy` , `marks`
- Description: Represents an academic course.
- Constraints:
    - **Constraint 38 and 39:** Define preconditions for requesting grade changes and course retakes.
    - **Constraint 16:** Allows a student to register for a course if certain conditions are met.
    - **Constraint 7:** Defines preconditions for dropping a course.
    - **Constraint 5:** Defines preconditions for withdrawing from a course.

2.15. Attendance

- Attributes: `numLectures`, `attendancePercentage`
- Description: Represents attendance details for a course.

2.16. University

- Attributes: `name`, `ranking`
- Description: Represents a university.

2.17. Campus

- Attributes: `location`
- Properties: `meritList`, `studentList`, `departmentList`
- Description: Represents a university campus.
- Constraints:
  - **Constraint 32 and 33:** Ensures scholarship for top-performing students.
  - **Constraint 18:** Limits student admissions based on warnings.
  - **Constraint 19:** Cancels admissions after the maximum duration.

2.18. MeritList

- Property: `positionHolders`
- Description: Represents a merit list of students.

2.19. PositionHolder

- Attribute: `paper`
- Description: Represents a position holder in a merit list.

2.20. Loan

- Attributes: Various attributes representing loan details.
- Description: Represents a financial loan for students.
- Constraints:
  - **Constraint 25:** Defines a precondition for applying for a loan.
  - **Constraint 26:** Ensures loans are renewed every semester.

    - **Constraint 27:** Limits loan assistance to fees.

2.21. Scholarship

- Attributes: Various attributes representing scholarship details.
- Description: Represents a scholarship awarded to students.

2.22. Fee

- Attributes: Various attributes representing fee details.
- Description: Represents a fee charged to students.
- Constraints:
    - **Constraint 22:** Ensures a specific payment method for fees.
    - **Constraint 23:** Defines refundability based on fee type.

2.23. Enums

- FeeType, Grade, CourseStatus, CourseImportance, Paper, Type
- Description: Enumerations representing various types and statuses used in the model.

---

3. Enumerations

3.1. FeeType

- **Description:** Represents the types of fees charged to students.
- **Values:**
    - **Normal:** Regular academic fees.
    - **Security:** Security or caution money fees.
- **Usage:** This enumeration categorizes fees within the system, distinguishing between regular academic fees and security fees.

3.2. Grade

- **Description:** Represents the grades assigned to students for courses.

- **Values:**
    - **A, B, C, D, E, F:** Represent standard letter grades.
    - **W:** Indicates withdrawal from a course.
- **Usage:** Used to assess and record student performance in courses, facilitating academic evaluation and transcript generation.

3.3. CourseStatus

- **Description:** Represents the status of a course for a student.
- **Values:**
    - **Passed:** Course completed successfully.
    - **Failed:** Course not completed satisfactorily.
    - **Withdrawn:** Course withdrawn before completion.
    - **Dropped:** Course dropped after registration.
    - **Repeated:** Course repeated due to failure or choice.
- **Usage:** Tracks the progress of students in individual courses, informing academic decisions and transcript generation.

3.4. CourseImportance

- **Description:** Indicates the importance or category of a course within a degree program.
- **Values:**
    - **Elective:** Optional courses within a program.
    - **Core:** Mandatory courses integral to a program's curriculum.
- **Usage:** Helps organize and prioritize courses within a degree program, guiding students in fulfilling degree requirements.

3.5. Paper

- **Description:** Represents the paper or examination board associated with a position holder in a merit list.
- **Values:**
    - **Board:** Examination conducted by a central board or authority.

    - **NU:** Examination conducted by the university.
- **Usage:** Identifies the examination board responsible for assessing and evaluating a student's academic performance.

3.6. Type

- **Description:** Indicates the type or level of an academic degree program.
- **Values:**
    - **MS:** Master's degree program.
    - **BS:** Bachelor's degree program.
- **Usage:** Distinguishes between different levels of academic programs, aiding in program categorization and student management.

---

4. Constriants Code Explaination

The following constraints define the behavior and rules governing various aspects of the University Management System:

Constraint 1: Student Can View Courses

```
invariant StudentCanViewCourses:
    Semester.allInstances()->forAll(s |
        Department.allInstances()->forAll(d |
            d.offeredCourses->includesAll(s.offeredCourses)
        )
    );
```

Constraint 2: Valid Semester Name

```
invariant ValidSemesterName:
    name = 'Fall' or name = 'Spring' or name = 'Summer';
```

Constraint 3: All Core Courses Taken

```
invariant AllCoreCoursesTaken:
    self.coreCourses->forAll(c |
        Student.allInstances()->exists(s |
            s.transcript.displayedCourses->includes(c)
        )
    );
```

Constraint 4: Max Registered Courses

```
invariant MaxRegisteredCourses:
    self.transcript.semesters->forAll(s |
        (self.degree.type = Type::BS implies
            s.enrolledCourses->size() <= 5
        ) and
        (self.degree.type = Type::MS implies
            s.enrolledCourses->size() <= 3
        )
    );
```

Constraint 5: Withdraw Course Operation

```
operation Withdraw() : Boolean[1] {
    precondition: self.status <> CourseStatus::withdrawn an
d currentDate.year < announcedDate.year or
        (currentDate.year = announcedDate.year and currentD
ate.month < announcedDate.month) or
        (currentDate.year = announcedDate.year and currentD
ate.month = announcedDate.month and currentDate.day < annou
ncedDate.day);
}
```

Constraint 6: Withdrawn Course Appears as 'W' on Transcript

```
invariant WithdrawnCourseW:
    self.displayedCourses->forAll(c |
        c.status = CourseStatus::withdrawn implies c.grade
= Grade::W
    );
```

Constraint 7: Drop Course Operation

```
operation Drop() {
    precondition: let diffInDays : Integer = (currentDate.y
ear - registrationDate.year) * 365 +
                    (currentDate.month - registrationDate.m
onth) * 30 +
                    (currentDate.day - registrationDate.da
y) in
                 diffInDays <= 14;
}
```

Constraint 8: Dropped Course Not on Transcript

```
invariant DroppedCourseNotOnTranscript:
    self.transcript.displayedCourses->forAll(c |
        c.status <> CourseStatus::dropped
    );
```

Constraint 9: CGPA Calculation Invariant

```
invariant CalculateCGPA:
    let totalCredits : Integer = self.transcript.semesters-
>collect(s | s.creditAttended)->sum() in
    let totalGradePoints : Real = self.transcript.semesters
->collect(s | s.SGPA * s.creditAttended)->sum() in
    self.CGPA = totalGradePoints / totalCredits;
```

Constraint 10: Track Course Attempts

```
invariant TrackCourseAttempts:
    self.displayedCourses->forAll(c |
        let failedAttempts : Integer = self.semesters->sele
ct(s | s.enrolledCourses->includes(c) and c.status = Course
Status::failed)->size() in
        let totalAttempts : Integer = self.semesters->selec
t(s | s.enrolledCourses->includes(c))->size() in
```

```
        totalAttempts = failedAttempts + 1
    );
```

Constraint 11: Repeat Courses Under Warning

```
invariant RepeatCoursesUnderWarning:
    let minCGPA : Real = if self.degree.type = Type::MS then 2.50 else 2.00 endif in
    let passedCourses : Set(Course) = self.transcript.semesters->select(s | s.CGPA >= minCGPA).enrolledCourses->asSet() in
    let lowGpaCourses : Set(Course) = passedCourses->select(c | c.grade <> Grade::A and c.grade <> Grade::B) in
    self.degree.warningCount > 0 implies self.transcript.displayedCourses->intersection(lowGpaCourses)->isEmpty();
```

Constraint 12: Repeat Course If Not Passed or Desired

```
invariant RepeatCourseIfNotPassedOrDesired:
    self.displayedCourses->forAll(c |
        let passedOrRepeated : Boolean = self.semesters->select(s | s.enrolledCourses->includes(c) and (c.status = CourseStatus::passed or c.status = CourseStatus::repeated))->notEmpty() in
        c.status <> CourseStatus::passed implies passedOrRepeated
    );
```

Constraint 13: Indicate Repeat Courses on Transcript

```
invariant IndicateRepeatCourses:
    let repeatCourses : Set(Course) = self.displayedCourses->select(c | c.repeatCount > 0) in
    repeatCourses->forAll(c |
        let totalRepeatCount : Integer = self.semesters->select(s | s.enrolledCourses->includes(c) and c.status = CourseStatus::repeated)->size() in
        c.repeatCount = totalRepeatCount
    );
```

Constraint 14: View Warnings

```
invariant ViewWarnings:
    self.warnings->notEmpty() implies self.warnings->exists
(w | w.status <> null);
```

Constraint 15 and later half of 18: Issue Warning at End of Semester

```
invariant IssueWarningAtEndOfSemester:
    let minCGPA : Real = if self.degree.type = Type::MS the
n 2.50 else 2.00 endif
    in
    if self.CGPA < minCGPA then
        self.warnings->exists(w | w.active = true)
    else
        self.warnings->isEmpty()
    endif;
```

Constraint 16: Register Course Operation

```
operation Register() {
    precondition: self.studiedBy->forAll(s | s.warnings->is
Empty() or s.approvalAcquired);
}
```

Constraint 17: Update Warning Count

```
invariant UpdateWarningCount:
    Semester.allInstances()->exists(s | s.finished = true)
implies
        let minCGPA : Real = if self.degree.type = Type::MS
then 2.50 else 2.00 endif in
```

```
        if self.CGPA < minCGPA then
            self.warningCount > self.transcript.semesters->
select(s | s.finished = true)->size()
        else
            self.warningCount = self.transcript.semesters->
select(s | s.finished = true)->size()
        endif;
```

Constraint 18 (first half): Close Student Admission

```
invariant CloseStudentAdmission:
    studentList->forAll(s | s.warningCount < 3 and s.warnin
gs->size() < 3);
```

Constraint 19: Cancel Admission After Max Duration

```
invariant CancelAdmissionAfterMaxDuration:
    studentList->forAll(s | s.degree.maxTime >= s.currentDe
greeDuration);
```

Constraint 20: View Fee Details

```
invariant ViewFeeDetails:
    transcript.semesters->forAll(s | s.fee <> null);
```

Constraint 21: Fee Due Date

```
invariant FeeDueDate: fee.dueInWeeks = self.startingInNumWe
eks - 2;
```

Constraint 22: Fee Payment Method

```
invariant FeePaymentMethod:
    paymentMethod = 'Challan';
```

Constraint 23: Non-Refundable Fee

```
invariant NonRefundable:
    if self.type = FeeType::Normal then
        refund = false
    else
        refund = true
    endif;
```

Constraint 24: Give Exam Operation

```
operation giveExam() {
    precondition: transcript.semesters->forAll(s |
        s.fee.paid or feeDefaulterApproved
    );
}
```

Constraint 25: Apply for Loan Operation

```
operation applyForLoan(s: Student) {
    precondition: s.indigent = true;
}
```

Constraint 26: Renew Loan Every Semester

```
invariant RenewEverySemester:
    renewedEverySemester = true;
```

Constraint 27: Financial Assistance Limited to Fees

```
invariant AssistanceLimitedToFees:
    assistanceType = 'Fees';
```

Constraint 28: Discontinue Loan

```
invariant DiscontinueLoan:
    let minCGPA : Real = if self.degree.type = Type::MS the
n 2.50 else 2.00 endif in
    self.CGPA < minCGPA implies self.loan = null;
```

Constraint 29: Loan Repayment Start

```
invariant LoanRepaymentStart:
    let earliestMonth : Integer =
        if graduation <> null and job <> null then
            if graduation.graduationMonth < job.startingDat
e then
                graduation.graduationMonth
            else
                job.startingDate
            endif
        else
            if graduation <> null then
                graduation.graduationMonth
            else
                job.startingDate
            endif
        endif
    in loan.repaymentStartMonth = earliestMonth + 3;
```

Constraint 30: Loan Repayment Period

```
invariant LoanRepaymentPeriod:
    loan <> null and graduation <> null implies loan.repaym
entEndMonths = graduation.graduationMonth + 48;
```

Constraint 31: Apply for Financial Assistance

```
operation apply(s: Student) {
    precondition: s.realNeedOfAssistance;
}
```

Constraint 32 and 33: Merit Scholarship for Top Three

```
invariant MeritScholarshipForTopThree:
    meritList.positionHolders->forAll(ph |
        ph.scholarship <> null
    );
```

Constraint 34: Maintain CGPA for Scholarship

```
invariant MaintainCGPAScholarship:
    self.scholarship <> null and
    self.scholarship.coverage = 100 and
    self.scholarship.numSems = 8 implies
    self.transcript.semesters->forAll(s |
        s.CGPA >= 3.00
    );
```

Constraint 35: View Course Attendance

```
invariant ViewCourseAttendance:
    self.transcript.semesters->forAll(s |
        s.enrolledCourses->forAll(c |
            c.attendance <> null
        )
    );
```

Constraint 36: Attendance for Final Exam

```
invariant Attendance80Appear:
    if attendance.attendancePercentage >= 80 then
        canAppearInExam = true
    else
        canAppearInExam = false
    endif;
```

Constraint 37: View Course Marks

```
invariant ViewCourseMarks:
    self.transcript.semesters->forAll(s |
        s.enrolledCourses->forAll(c |
            c.marks <> null
        )
    );
```

Constraint 38: Request Grade Change Operation

```
operation requestGradeChange() {
    precondition: resultsAnnounced;
}
```

Constraint 39: Request Exam Retake Operation

```
operation requestRetake() {
    precondition: retakeApproved;
}
```