New Uzbekistan University

(Assignment 1)

Implement micro-LMS program

CS211 Object Oriented Programming

New Uzbekistan University

New Uzbekistan University

Task: Implement micro-LMS program

- Implement a C++ program to administer university students and courses

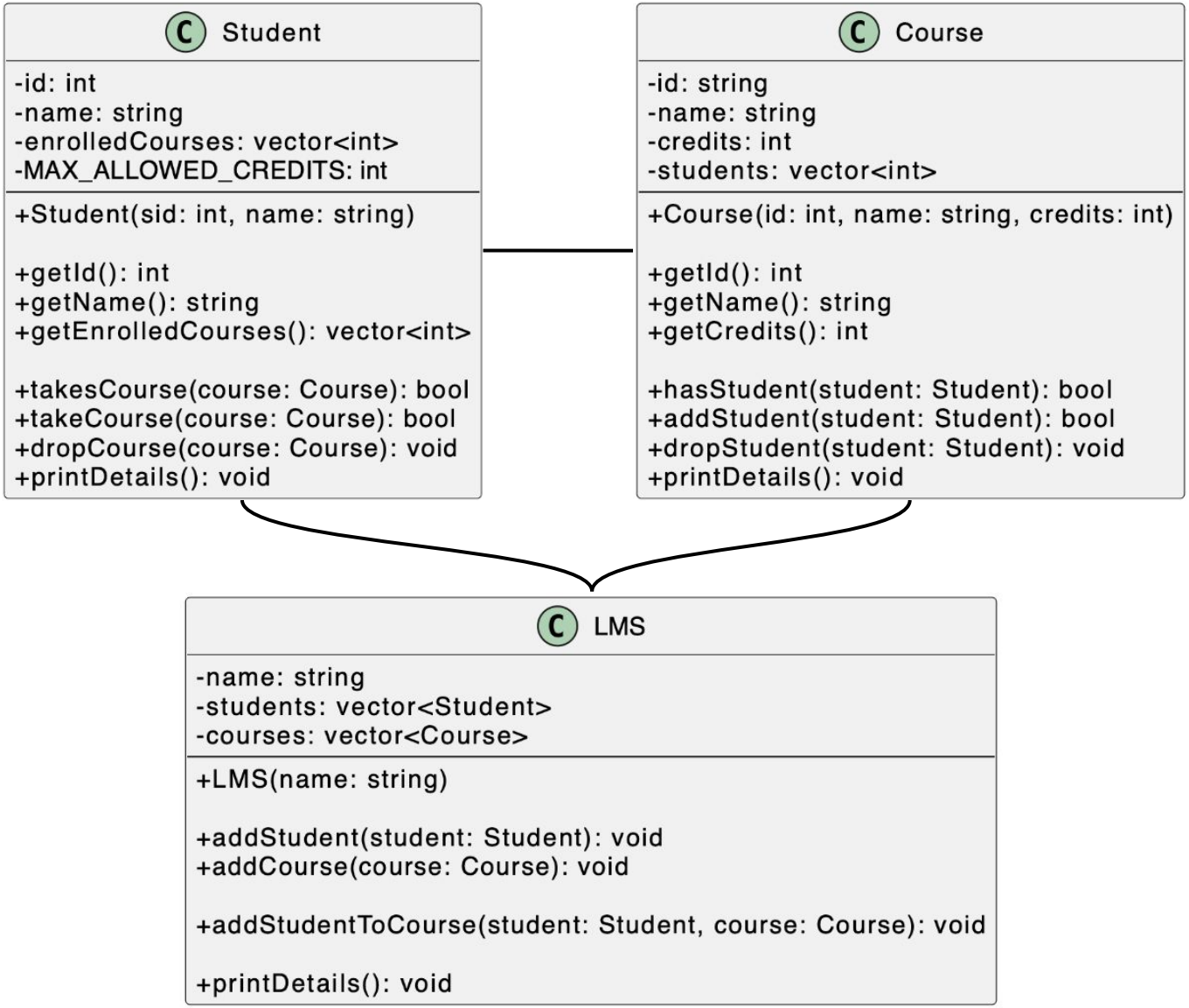
- Your LMS program must allow
  - Manage students
  - Manage courses
  - Analyze course enrollments

- Notes
  - You must implement your program using C++ classes

LMS: Learning Management System

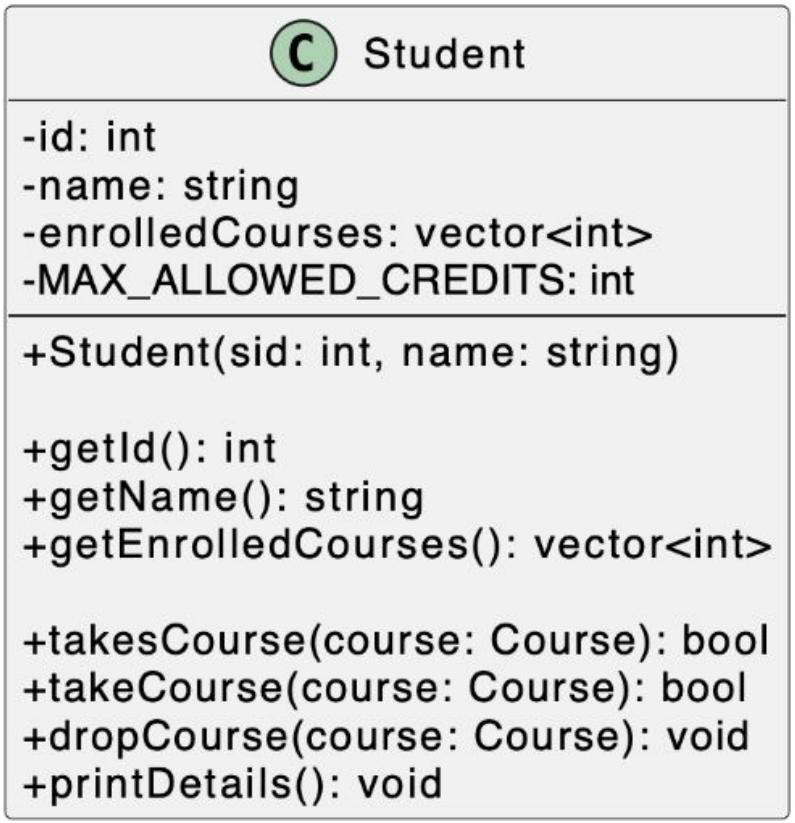
UML diagram

New Uzbekistan University



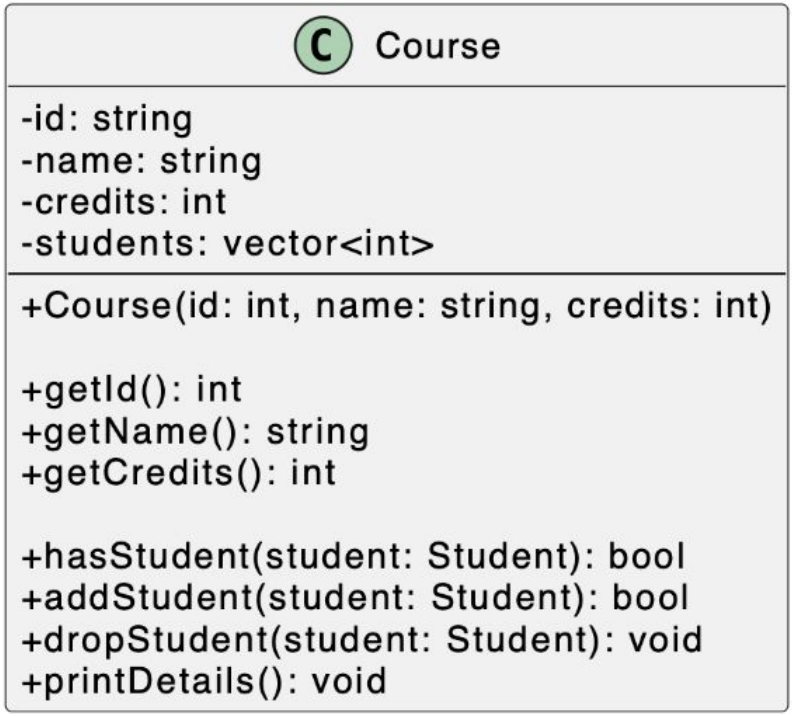
UML diagram of `Student` class

- **Member variables**
  - All variables (`id`, `name`, `enrolledCourses`, `MAX_ALLOWED_CREDITS`) must be **private** to ensure encapsulation.
  - `MAX_ALLOWED_CREDITS`: Constant defining the max credits (set it to constant value **18 credits**)
- **Member functions**
  - **Constructor**: Initializes `id`, `name`, and sets up defaults.
  - **Getters**: `getId()`, `getName()`, and `getEnrolledCourses()` return respective details (`const`).
  - **`takesCourse(Course)`**: Checks if the student is enrolled in the course.
  - **`takeCourse(Course)`**: Adds a course if not enrolled and within credit limit.
  - **`dropCourse(Course)`**: Removes a course from enrollment.
  - **`printDetails()`**: Displays student details and enrolled courses.



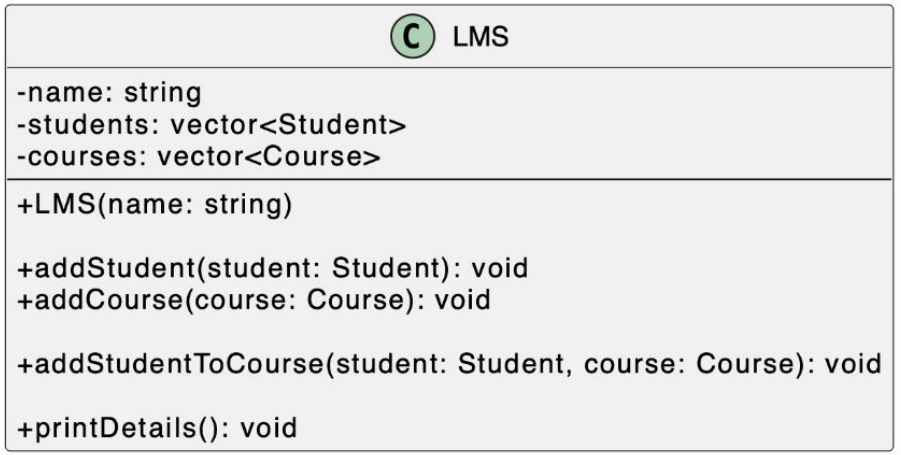
UML diagram of `Course` class

- **Member variables**
  - Store `id, name, credits,` and `students` securely.
  - `students` should store IDs of enrolled students.
  - `credits` should be an integer representing course credits
- **Member functions**
  - `Constructor`: Initializes `id, name,` and `credits`.
  - Getter functions: `getId(), getName(), getCredits()` - must be const and return respective values without modifications.
  - `hasStudent():` Check if a student is already enrolled. Return true if found.
  - `addStudent():` Add a student to the students list. Return true if the student was successfully added.
  - `dropStudent():` Remove a student from the students list.
  - `printDetails():` Print course details, including `id, name, credits,` and list of enrolled student IDs.



UML diagram of `LMS` class

- **Member variables**
  - Store `name`, `students`, and `courses` securely (i.e., private members)
  - Use a vector to store array of instances from `Student` and `Course` classes.
- **Member functions**
  - Constructor: Initialize `LMS` with a name.
  - `addStudent()`: Add a Student object to the students list.
  - `addCourse()`: Add a `Course` object to the `courses` list.
  - **addStudentToCourse():** Link a student to a course by updating both the `students` and `courses` lists. Perform necessary checks to ensure validity.
  - `printDetails()`: Display the LMS name, the list of students, and the list of courses with their details.

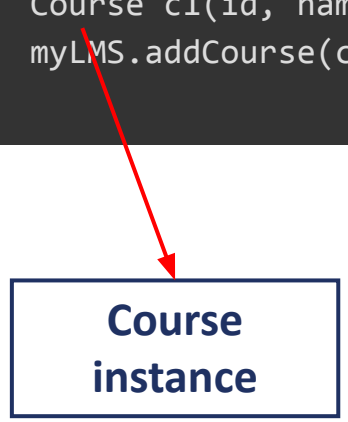


Example `main()` function

```
// Main Program
int main() {
    string nameLMS;
    getline(cin,nameLMS);
    LMS myLMS(nameLMS);
    int number_students;
    cin>>number_students;
    //Add students
    for(int i=0;i<number_students;i++){
        int id;
        string name;
        cin>>id>>name;
        Student s1(id, name);
        myLMS.addStudent(s1);
    }
```

Example `main()` function

```
// Main Program
// Add courses
    int number_courses;
    cin>>number_courses;
    for(int i=0;i<number_courses;i++){
        string id,name;
        int credit;
        cin>>id>>name>>credit;
        Course c1(id, name, credit);
        myLMS.addCourse(c1);
    }
```



Example `main()` function

```
// Main Program
// Enroll students in courses
    int n;
    cin >> n;
    for(int i=0;i<n;i++){
        int student_id;
        string course_id;
        cin>>student_id>>course_id;
        myLMS.addStudentToCourse(student_id, course_id);
    }
    // Print LMS details
    myLMS.printDetails();
    return 0;
}
```

Test case 1 (sample)

Input:

SmallLMS
1
1 Alice
1
101 Math 3
1
1 101

Output:

LMS Name: SmallLMS
Students:
Student ID: 1, Name: Alice
Enrolled Courses: 101~
Courses:
Course ID: 101, Name: Math, Credits: 3
Enrolled Students: 1

~ stands for space

Test case 2 (sample)

Input:

MediumLMS
5
1 Alice
2 Bob
3 Charlie
4 David
5 Eve
3
101 Math 3
102 Physics 3
103 Chemistry 3
5
1 101
2 101
3 102
4 103
5 103

Output:

LMS Name: MediumLMS
Students:
Student ID: 1, Name: Alice
Enrolled Courses: 101~
Student ID: 2, Name: Bob
Enrolled Courses: 101~
Student ID: 3, Name: Charlie
Enrolled Courses: 102~
Student ID: 4, Name: David
Enrolled Courses: 103~
Student ID: 5, Name: Eve
Enrolled Courses: 103~
Courses:
Course ID: 101, Name: Math, Credits: 3
Enrolled Students: 1 2
Course ID: 102, Name: Physics, Credits: 3
Enrolled Students: 3
Course ID: 103, Name: Chemistry, Credits: 3
Enrolled Students: 4 5

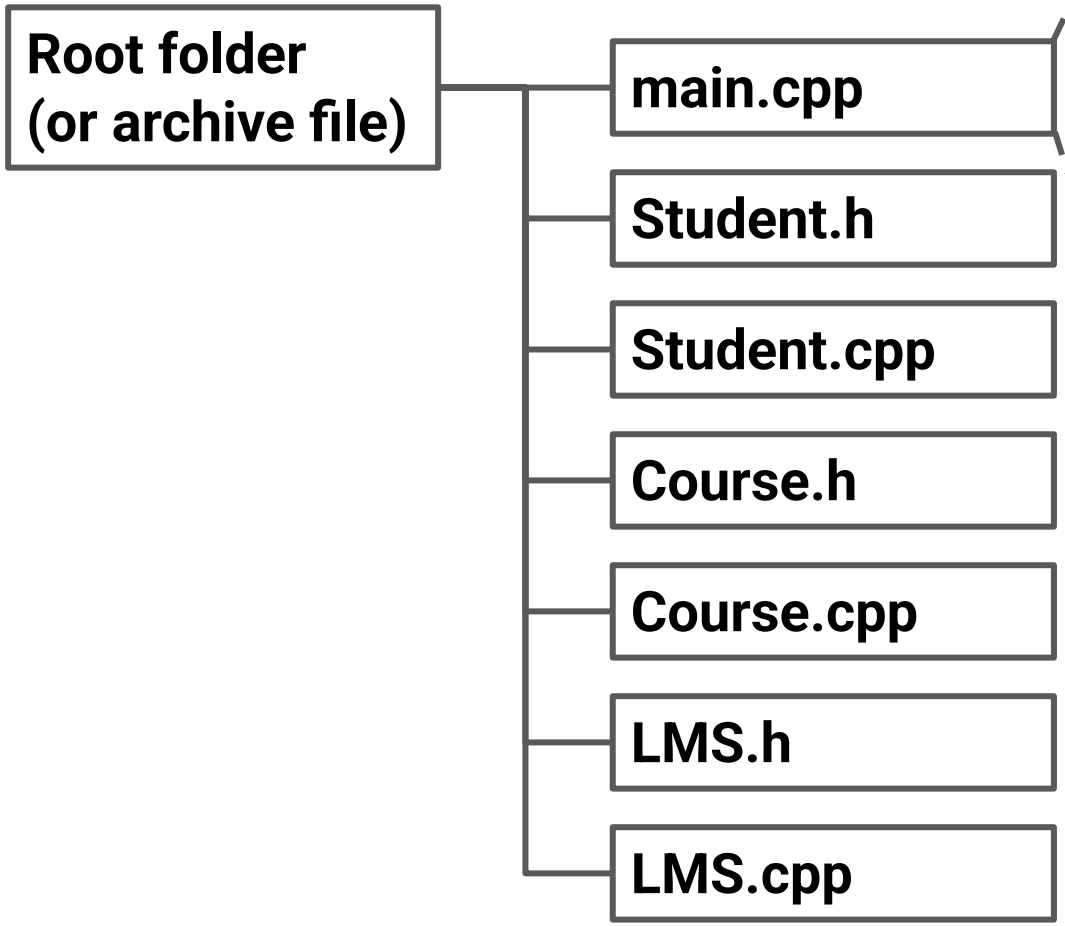
~ stands for space

New Uzbekistan University

Instructions for submission

- **Submit your code via gradescope platform**
  - Strictly follow the file structure is in the next slide (p11)
- **Deadline: by 23:59, 07.02.2025 (Fri)**
  - Penalty for late submissions
    - ➢ *On 08.02.2025 (Sat)*: max(0, assignment score - 10%)
    - ➢ *On 09.02.2025 (Sun)*: max(0, assignment score - 20%)
    - ➢ *On 10.02.2025 (Mon)*: max(0, assignment score - 30%)
  - Assignments NOT graded 3 days past the deadline
    - ➢ i.e., zero points after 10.02.2025

Expected code structure



Contents of main.cpp
as an example

New Uzbekistan University

```cpp
int main() {
    string nameLMS;
    getline(cin,nameLMS);
    LMS myLMS(nameLMS);
    int number_students;
    cin>>number_students;
    //Add students
    for(int i=0;i<number_students;i++){
        int id;
        string name;
        cin>>id>>name;
        Student s1(id, name);
        myLMS.addStudent(s1);
    }
    // Add courses
    int number_courses;
    cin>>number_courses;
    for(int i=0;i<number_courses;i++){
        string id,name;
        int credit;
        cin>>id>>name>>credit;
        Course c1(id, name, credit);
        myLMS.addCourse(c1);
    }
    // Enroll students in courses
    int n;
    cin >> n;
    for(int i=0;i<n;i++){
        int student_id;
        string course_id;
        cin>>student_id>>course_id;
  myLMS.addStudentToCourse(student_id, course_id);
    }
    // Print LMS details
    myLMS.printDetails();
    return 0;
}
```