C16-LAB. OO-Programming in C++ with Exception Handling

Objective:

1. Implement **classes** and **composition/aggregation** in C++.
2. Use **exception handling** to enforce constraints on student IDs and course room numbers.
3. Instantiate objects and associate students with courses.
4. Display results after handling exceptions.

What to do? Implement your solution using a minimalist approach (Lazy-Classes).

Step 1: Define the Student Class

- Attributes: name (string), id (int)
- Throw an exception if id < 5555
- Provide a constructor and a method to display student details

Step 2: Define the Course Class

- Attributes: name (string), room (string)
- Throw an exception if room is "MBA319" or "MBA312"
- Provide a constructor and a method to display course details

Step 3: Define an Aggregation Relationship

- Create a vector of Student objects inside the Course class
- Provide a method to **enroll students** in a course

Step 4: Implement Exception Handling

- Use try-catch blocks to catch and display error messages when adding students or courses.

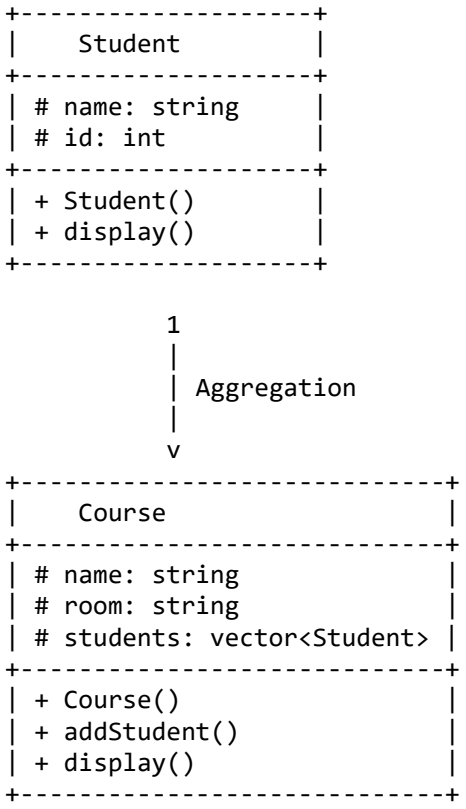
Step 5: Instantiate Objects (see sample main function)

- Students: Homer, Bart, Lisa, Marge, Maggie
- Courses: Math 101 in MBA320, CS102 in MBA311
- Attempt to enroll students in course Math101.
- Modify the main function to comply with the output requirements.

Step 6: Display the Course Roster

- Show course details and enrolled students

UML Diagram


Student
name: string
id: int
+ Student()
+ display()
1
Aggregation
Course
name: string
room: string
students: vector<Student>
+ Course()
+ addStudent()
+ display()

Sample main() function (Must be fixed to respond to exceptions and produce an output similar to the one shown below)

```
int main() {

        // SKELETON CODE

        // Creating students
        Student s1("Homer Simpson", 5678);
        Student s2("Bart Simpson", 4444); // Should cause an exception
        Student s3("Lisa Simpson", 8888);
        Student s4("Marge Simpson", 9000);
        Student s5("Maggie Simpson", 7777);

        // Creating courses
        Course c1("Math 101", "MBA320");
        Course c2("CS102", "MBA319"); // Should cause an exception

        // Enrolling students
        c1.addStudent(s1);
        c1.addStudent(s3);
        c1.addStudent(s4);

        // Displaying results
        c1.display();
        c2.display();
}
```

Expected Output

```
Error: Bart Simpson - Student ID must be at least 5555.
Error: Room MBA312 is not allowed.

Course: Math101 (Room: MBA320)
Enrolled Students:
Student: Homer Simpson (ID: 5678)
Student: Lisa Simpson (ID: 8888)
Student: Marge Simpson (ID: 9000)
Student: Maggie Simpson (ID: 7777)
All done!
```

Key Learning Outcomes:

1. **Encapsulation:** Defined protected attributes with public methods.
2. **Exception Handling:** Used try-catch to enforce constraints.
3. **Composition/Aggregation:** Associated students with courses.
4. **UML Modeling:** Visualized class relationships.