KLE Society's

KLE Technological University



**Open Ended (OE) Assessment Report**

**On**

**SCHOLERSYNC**

**Object Oriented Programming (20ECSC204)  
Object Oriented Programming Lab (20ECSP203)**

Submitted by

|  |  |  |
| --- | --- | --- |
| NAME | ROLL NO | SRN |
| MOHD.QADIR TERNIKAR | 16 | 02FE22BCS053 |
| HARSH NESARI | 13 | 02FE22BCS040 |
| MRUNALI BENNALKAR | 17 | 02FE22BCS055 |
| VANASHREE  NANDANGAVE | 65 | 02FE22BCS171 |

Team no: A-16

Faculty In-charge:

**Prof. Vaishali Y. Parab**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Academic year 2023-24

**problem statement**

Students face significant challenges in managing their academic information due to fragmented and disjointed access to course schedules, grades, and communication channels. This lack of a centralized platform results in inefficiencies, communication gaps, and difficulty in tracking academic progress, ultimately impacting their overall academic performance and experience.

**FEATURES OF APPLICATION**

**Concept of OOPs:**

Inheritance, Polymorphism, Encapsulation , Abstraction, Virtual functions , Templates , Exception handling and design patterns like **Singleton** and Observer

**CLASSES:  
 Person:**

 **Data Members**: name, id, password

 **Methods**: Constructors, getName(), getId(), checkPassword(), setPassword(), getPassword()

 **Features**: This is a base class for Student and Instructor. It provides basic functionalities like getting name, id, and password, and checking the password.

**Student:**

 **Data Members**: grades, enrolledCourses

 **Methods**: Constructors, enrollCourse(), addGrade(), getGrades(), getEnrolledCourses(), getGPA(), saveGrades(), loadGrades()

 **Features**: This class extends Person to include functionalities specific to a student such as enrolling in courses, managing grades, and calculating GPA.

* **Exception Handling in Student Menu:** Similar to the admin menu, the student menu handles invalid inputs by catching invalid\_argument exceptions, displaying an error message, and resetting the input stream.

**Instructor:**

* **Data Members**: department, assignedCourses
* **Methods**: Constructors, assignCourse(), addGradeToStudent(), getDepartment(), getAssignedCourses()
* **Features**: This class extends Person to include functionalities specific to an instructor such as assigning courses and adding grades to students

.

* **Exception Handling for instructor Menu:** The instructor menu also includes exception handling for invalid inputs in the same manner as the admin and student menus.

**Grade:**

 **Data Members**: course, grade, marks

 **Methods**: Constructors, getCourse(), getGrade(), getMarks(), save(), load()

 **Features**: This class encapsulates the grade details of a student for a specific course and provides methods to save and load grade data.

  **Exception handling in grade class:**

* In this part of the code, while reading grades from a file, if an error occurs (e.g., a course is not found during grade loading), a runtime\_error exception is thrown.
* The catch block catches this exception and breaks the loop, preventing the program from crashing and allowing it to handle the error gracefully.

**Course:**

 **Data Members**: courseName, courseCode, students, instructor

 **Methods**: Constructors, enrollStudent(), assignInstructor(), getCourseName(), getCourseCode(), getStudents(), getInstructor()

 **Features**: This class represents a course, managing enrolled students and the assigned instructor.

**University:**

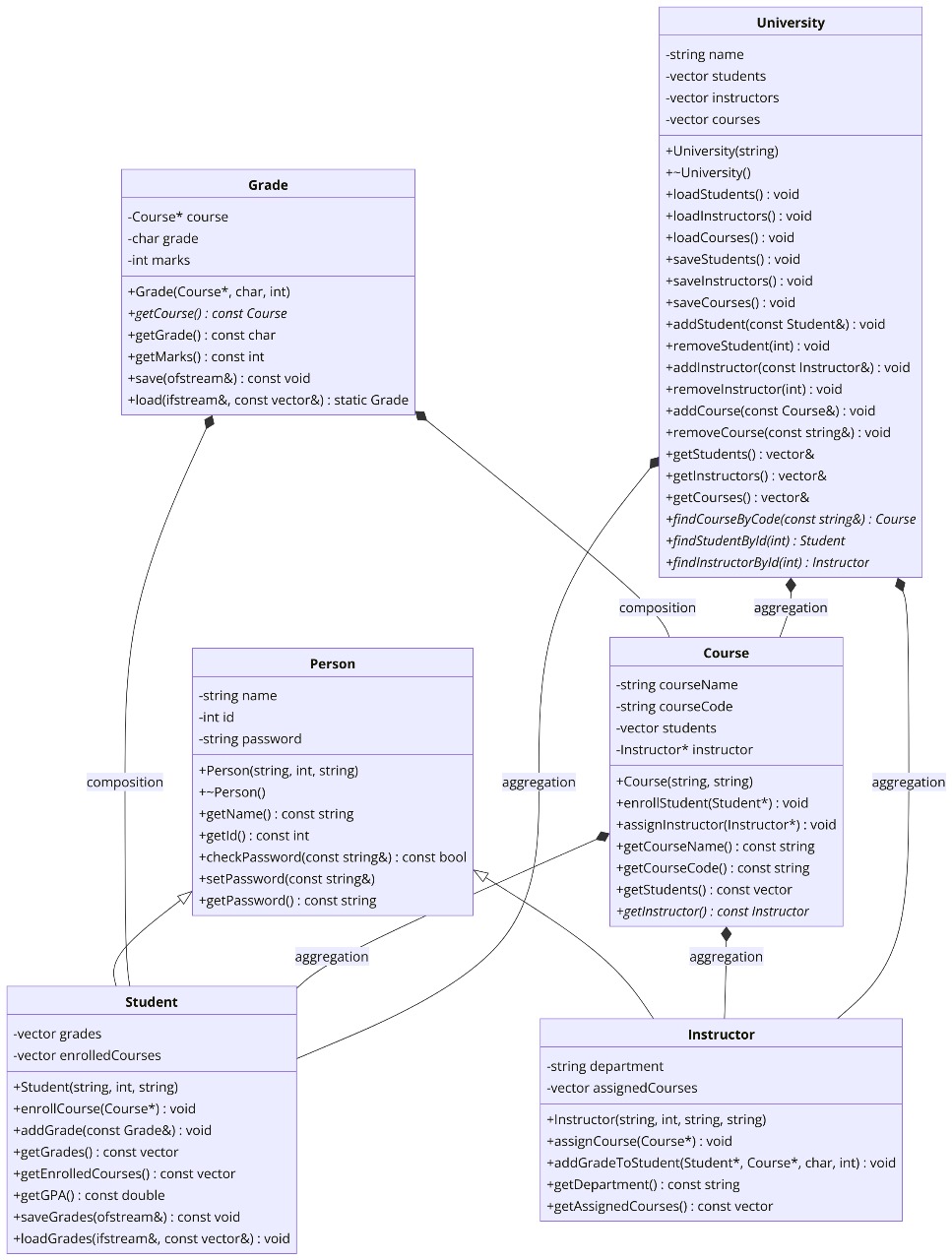
 **Data Members**: name, students, instructors, courses

 **Methods**: Constructors, addStudent(), removeStudent(), addInstructor(), removeInstructor(), addCourse(), removeCourse(), findCourseByCode(), findStudentById(), findInstructorById(), loadStudents(), loadInstructors(), loadCourses(), saveStudents(), saveInstructors(), saveCourses()

 **Features**: This class manages the entire university system including students, instructors, and courses. It includes methods to add, remove, and find these entities, as well as to load and save data to files.

**Functions:**

* String getPassword()
* Void adminMenu()
* Void studentMenu()
* Void instructorMenu()
* Int main()

**UML DIAGRAM:** 

**DESIGN PATTERN USED:**

1. **Singleton Pattern** (Implicit): The University class acts somewhat like a singleton within the context of this program, as there is a single instance of University that holds all the data for students, instructors, and courses. However, it's not strictly implemented as a singleton pattern because multiple instances could theoretically be created.
2. **Observer Pattern** (Implicit): When a Course enrolls a Student, or when an Instructor assigns a course, there is a form of an observer pattern where the state of Student, Course, and Instructor objects gets updated. However, this is done manually and not through a formal observer pattern implementation.

**Output**:

