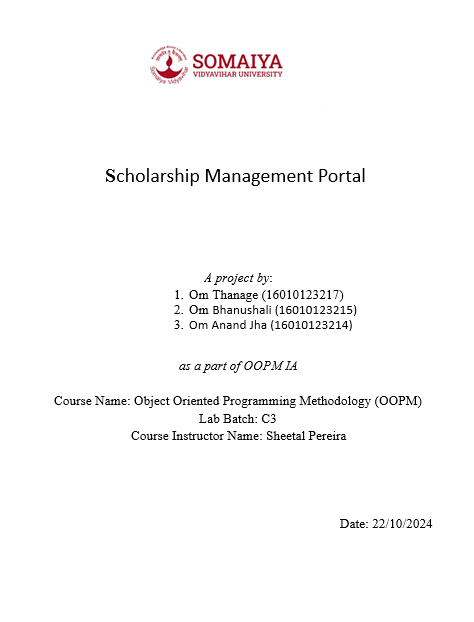
****

**Table of Contents:**

|  |  |
| --- | --- |
| **Topics** | **Pg No.** |
| 1. Introduction, Objective and Tech Stack | 3 |
| 1. System Analysis: Problem Definition, Class Diagram | 4 |
| 1. Implementation: Oops concepts and Code | 6 |
| 1. Appendices | 9 |

**Introduction**

1. The Scholarship Portal is a Java-based application with a Swing UI designed to manage scholarships efficiently.
2. It allows students to register, apply for scholarships, and view their applications.
3. Administrators can create scholarships and access the list of applicants.
4. The system is organized using packages for Student, Admin, Scholarship details, and ScholarshipManager, and uses ArrayList and Map for data management with getter and setter methods for efficiency.

**Objectives of Our Project**

1. **Streamline Scholarship Management**: Provide an efficient platform for students to register, apply for scholarships, and track their application status.
2. **Simplify Administrative Tasks**: Enable administrators to create new scholarships and manage student applications with ease.
3. **Enhance Data Organization**: Use structured data storage through ArrayList and Map for quick access, retrieval, and management of student and scholarship information.
4. **Improve User Experience**: Offer a user-friendly interface using Swing UI for smooth interaction between students and administrators.

**Tech Stack Used:**

* Programming Language: Java
* Programming paradigm: Object Oriented
* Libraries: Swing (GUI), Collection Framework(ArrayList, Hashmap)
* Tools: JDK

**System Analysis**

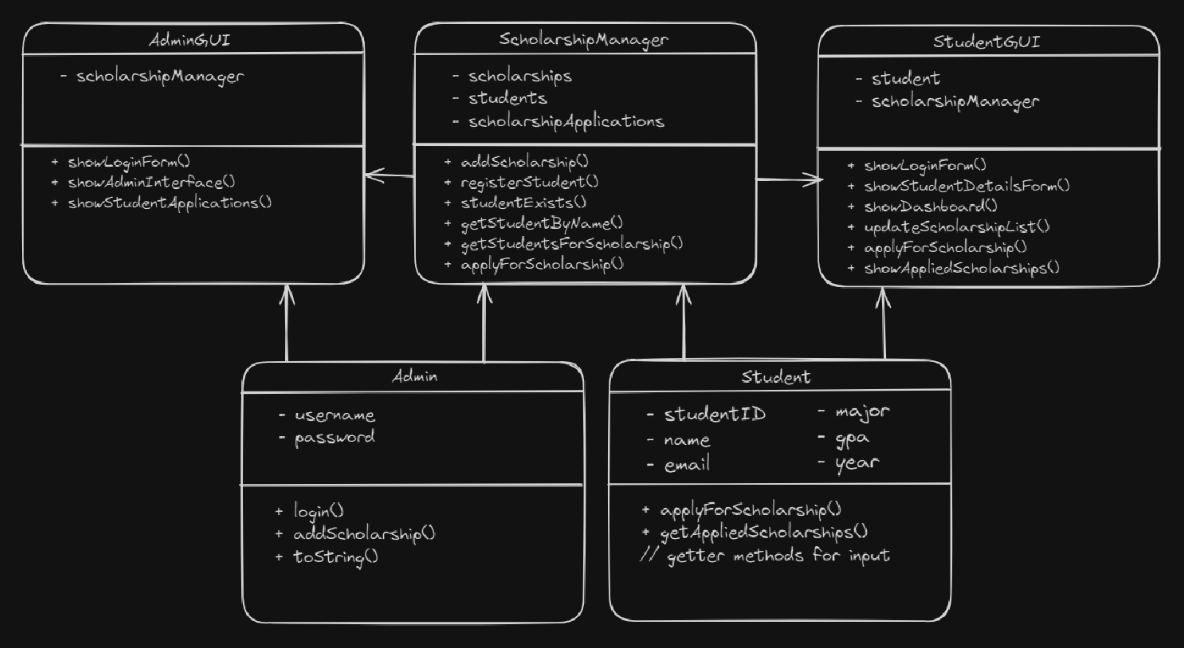
**Problem Definition:**

Many students face challenges in finding and applying for scholarships. This application simplifies the process by enabling students to register, view available scholarships based on their GPA, and apply online. It also allows them to view applied scholarships for easy reference.

The following Problems can be solved using our application:

* Manual Scholarship Processing: It eliminates the need for paper-based or manual scholarship applications, reducing errors and streamlining the application process.
* Lack of Centralized Data: Provides a centralized platform for storing and managing all student and scholarship data, making it easy to retrieve and update information.
* Time-Consuming Administration: Simplifies administrative tasks like scholarship creation and managing applications, saving time and effort for administrators.
* Student Accessibility: Offers students easy access to available scholarships and allows them to apply and track their applications from one portal.
* Miscommunication and Tracking Issues: Improves communication and transparency by allowing students to view their applied scholarships and application statuses in real-time.

**Class Diagram:**



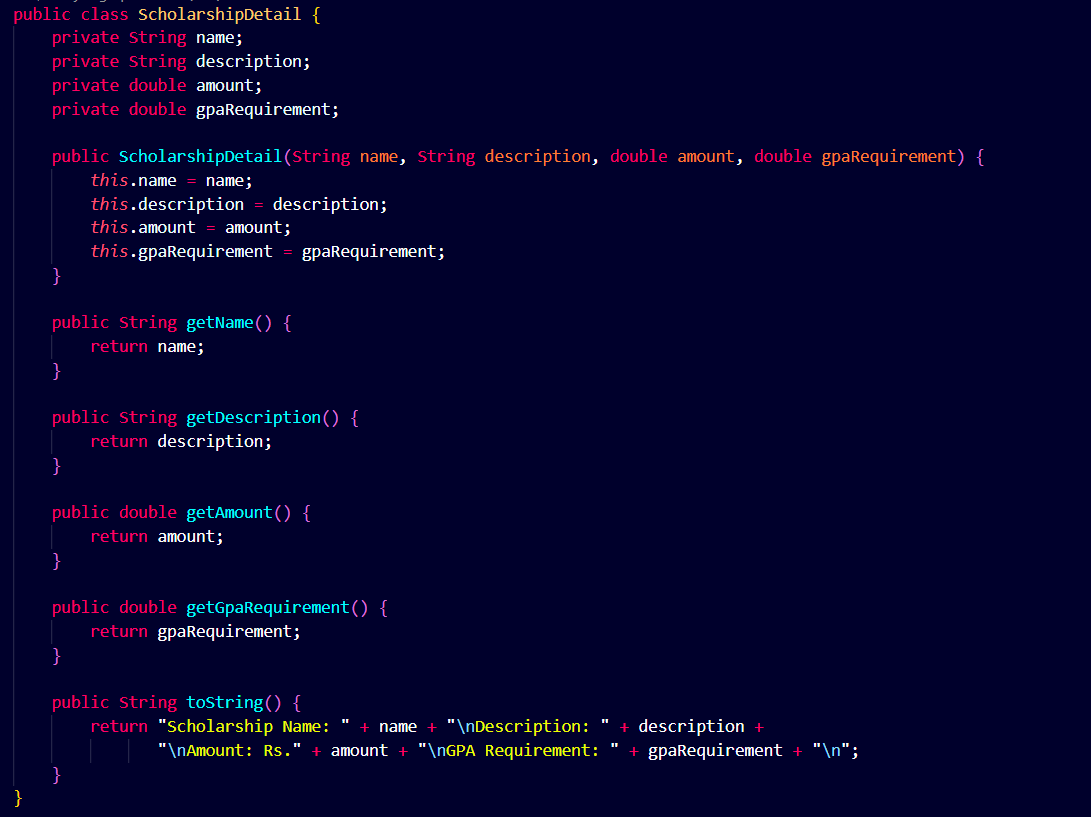
**Implementation:**

**Encapsulation**

Encapsulation is achieved by hiding the internal state of objects and exposing methods to interact with that state. In your project, the classes use private fields and provide public getter methods to access these fields.

In code usage:

* In ScholarshipDetail, the fields name, description, amount, and gpaRequirement are private, and public getter methods like getName(), getDescription(), getAmount(), and getGpaRequirement() are provided to access these fields.

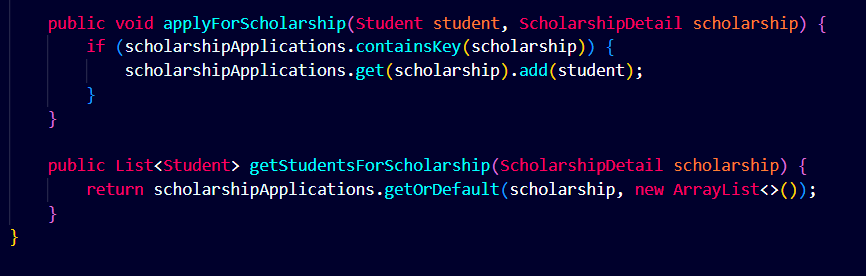


**Abstraction**

Abstraction is about hiding complex implementation details and showing only the necessary features of an object. While your project does not explicitly use abstract classes or interfaces, it abstracts the functionality of managing scholarships and students through classes like ScholarshipManager.

In code usage:

* The ScholarshipManager class abstracts the details of managing scholarships and student applications. Methods like addScholarship(), getScholarships(), and applyForScholarship() provide a simplified interface for these operations.

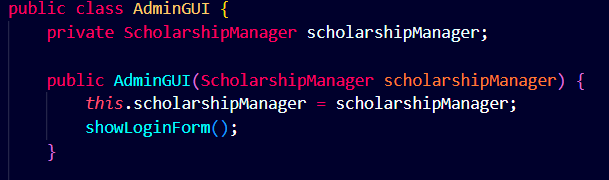


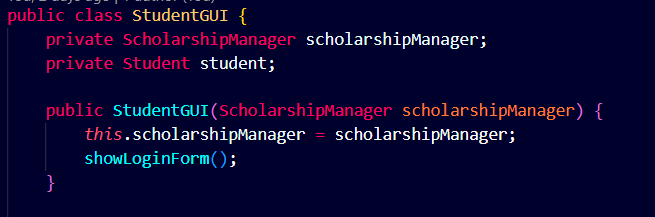
**Inheritance**

Inheritance allows a class to inherit properties and methods from another class. Your project does not show explicit inheritance hierarchies, but it uses composition to achieve similar functionality.

In code usage:

* The AdminGUI and StudentGUI classes use instances of ScholarshipManager to manage scholarships, demonstrating composition rather than inheritance.



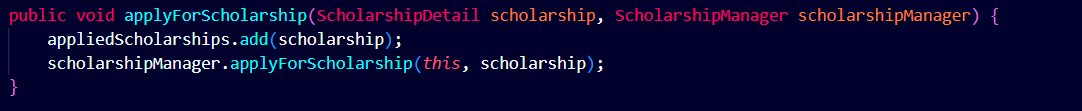


**Polymorphism**

Polymorphism allows methods to be used in different ways, typically through method overloading or overriding. Your project does not show explicit examples of method overloading or overriding, but it uses polymorphism through interface implementations and method calls.

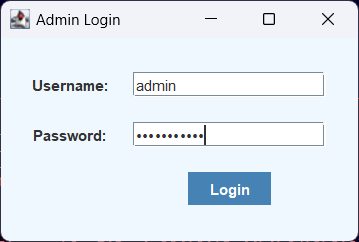
In code usage:

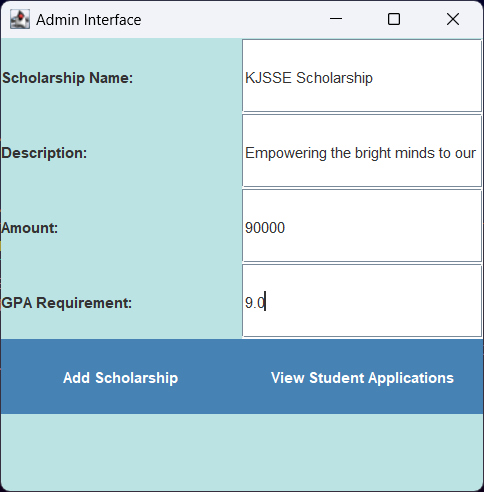
The applyForScholarship() method in the Student class calls the applyForScholarship() method in the ScholarshipManager class, demonstrating polymorphism through method calls.

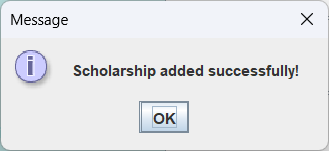


**Appendices**

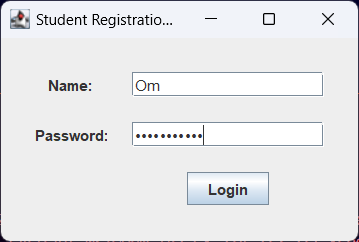
**Screenshots of the GUI**

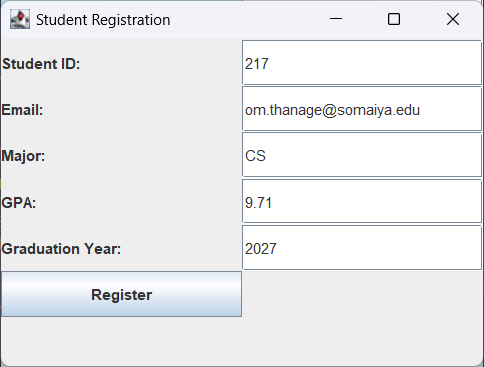
**Admin:**

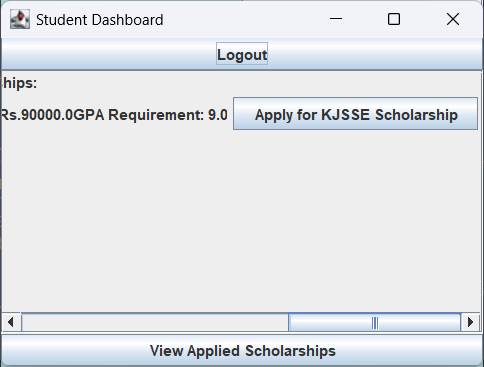


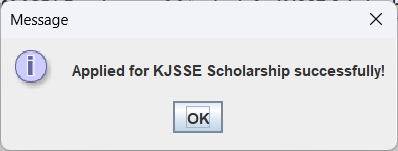


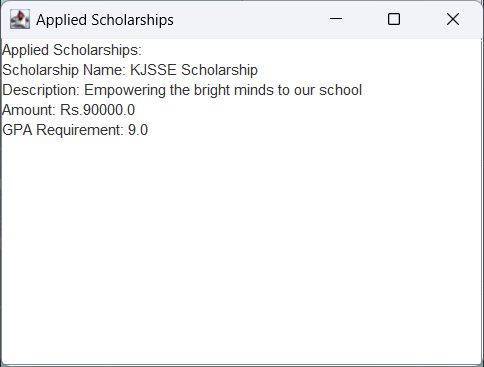
**Now for student:**



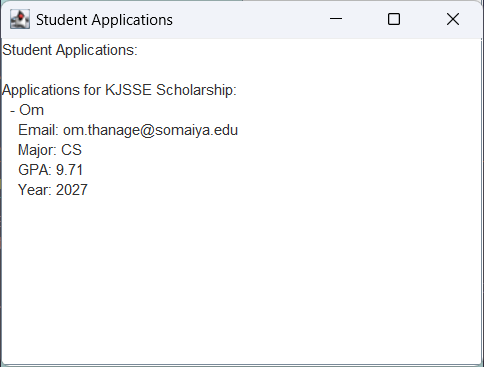








**Admin we will see when he clicks on View Student Applications**



Also, we added try and catch methods for incorrect inputs and Exceptions