Student Management System - Documentation

1. Project Overview

The **Student Management System** is a software application designed to manage student data, including course registrations, grade assignments, and other student-related activities. It uses multiple design patterns to structure the system efficiently, ensuring flexibility, scalability, and maintainability. The system includes features such as:

- Registering students for courses
- Assigning grades to students
- Logging course registration activities
- Using a graphical user interface (GUI) for interaction

2. Design Patterns Used

This project utilizes several design patterns to achieve a well-structured and maintainable system. The key patterns used are:

1. **Singleton Pattern**
2. **Factory Pattern**
3. **Builder Pattern**
4. **Adapter Pattern**
5. **Observer Pattern**

2.1. Singleton Pattern

The **Singleton Pattern** ensures that only one instance of a class exists throughout the application. This pattern is used in the **Course Registration System** and **Grade Processing System** classes to maintain a single instance for managing course registrations and grades.

.

2.2. *Factory Pattern*

The **Factory Pattern** is used to create different types of objects based on specific input parameters. In this system, it is used to create various types of courses and students**.**

Course Factory creates different course types (Core, Elective, Lab) based on the provided type**.**

Student Factory creates different types of students (Undergraduate, Graduate, Part-time) based on the provided type**.**

2.3. *Builder Pattern*

The **Builder Pattern** is used to construct complex **Student Profile** objects. It allows the creation of a student profile by setting various attributes like name, course, and student type in a step-by-step manner.

2.4. *Adapter Pattern*

The **Adapter Pattern** is used to integrate external courses with the system. The **External Course** class represents an external course, and the **External Course Adapter** adapts it to the system's course structure.

2.5. Observer Pattern

The **Observer Pattern** is implemented to notify observers when a student registers for a course. The **Course Registration System** acts as the subject, and **Course Registration Logger** is the observer that logs the registration details.

- The **Course Registration System** notifies the observer whenever a new course registration happens.
- **Course Registration Logger** stores and displays the registration records.

3. Core Components of the System

1. **Course Registration System**:
    - Manages the registration of students in courses.
    - Uses the **Singleton Pattern** to ensure only one instance exists.
    - Notifies observers (e.g., **Course Registration Logger**) when a new registration occurs.
2. **Grade Processing System**:
    - Manages the assignment of grades to students.
    - Uses the **Singleton Pattern** to ensure a single instance is responsible for grade management.
3. **Student Profile**:
    - Represents a student's profile, including name, course, and student type.
    - Uses the **Builder Pattern** to construct a student profile with multiple attributes.
4. **External Course and External Course Adapter**:
    - The **External Course** class represents an external course.
    - The **External Course Adapter** adapts external courses to fit into the existing course structure of the system using the **Adapter Pattern**.
5. **Course Factory**:
    - Creates different types of courses (Core, Elective, Lab) based on user input.
6. **Course Registration Logger**:
    - Logs the registration details for each student.
    - Implements the **Observer Pattern** to listen for course registration events.

---

4. Graphical User Interface (GUI)

The **GUI** is built using **Java Swing** components and provides an interface for users to interact with the system. The key components of the GUI include:

- **Text Fields** for entering student name and course details.
- **Combo Boxes** for selecting course type and student type.
- **Buttons** for registering students and assigning grades.
- **Text Area** for displaying logs of registered courses and grade

5.Workflow

- **Course Registration**:
  - The user enters the student's name, course name, and selects the course type and student type.
  - The system creates the appropriate course and student objects using the **Factory Pattern**.
  - The **Course Registration System** registers the student for the course and notifies the **Course Registration Logger**.
  - The system displays the student's profile and registration details on the GUI.
- **Grade Assignment:**
  - The user selects a student and assigns a grade.
  - The **Grade Processing System** updates the grade for the student.
  - The system displays the updated grade on the GUI.

---

6. Conclusion

The **Student Management System** is a robust and flexible application that leverages several design patterns to manage student data effectively. The use of patterns such as **Factory**, **Observer**, **Singleton**, **Builder**, and **Adapter** ensures that the system is scalable, maintainable, and easy to extend with new features.

This documentation provides an overview of the system's design and key components, enabling easy understanding of the code.