**Documentation for Student Management System**

**Overview**

The **Student Management System** is a Java-based desktop application designed to manage student records efficiently. It includes features for adding, updating, deleting, and viewing students and courses, along with generating reports. The project follows established design patterns to ensure scalability, maintainability, and clear separation of concerns.

**Key Features**

1. **Student Management**
   * Add, update, and delete student details.
   * View student profiles.
2. **Course Management**
   * Add, update, and delete courses.
   * Enroll students in courses.
3. **Report** 
   * Generate student performance reports.
   * View enrolled students for specific courses.
4. **Graphical User Interface (GUI)**
   * User-friendly interface using Java Swing.
   * Tables and forms for data display and interaction.

**Technologies Used**

* **NetBeans IDE:** Development environment.
* **Java:** Core programming language.
* **Microsoft SQL Server:** Database for storing application data.
* **JDBC:** Connectivity between Java application and database.
* **Java Swing:** Framework for building GUI components.

**Design Patterns**

**Singleton Pattern**

* **Class:** DatabaseConnection
* **Purpose:** Ensures a single instance of the database connection throughout the application.
* **Implementation:**
  + Private constructor to restrict instantiation.
  + Static method getInstance to provide global access to the single instance.

**Factory pattern**

* **Usage:** Creates objects dynamically for students, courses, and reports.
* **Example:** Simplifies object creation based on user input or configuration.

**Builder Pattern**

* **Class:** Student.Builder
* **Purpose:** Constructs Student objects with optional attributes in a step-by-step manner.

**Adapter Pattern**

* **Class:** StudentAdapter
* **Purpose:** Converts a list of Student objects into a format suitable for GUI display.
* **Example:** Converts a List<Student> to a DefaultTableModel for Swing tables.

**Observer Pattern**

* **Class:** NotificationSystem
* **Purpose:** Implements real-time updates to notify observers (e.g., GUI) of data changes.
* **Example:** Updates the view when a new student is added.
* **util**: Includes utility classes like DatabaseConnection and StudentAdapter.

**Class Descriptions**

**DatabaseConnection**

* **Purpose:** Manages database connectivity.
* **Details:**
  + Singleton pattern to ensure one connection instance.
  + Connects using JDBC to Microsoft SQL Server.

**Student**

* **Purpose:** Represents student entities.
* **Details:**
  + Attributes: id, name, email.
  + Builder pattern for creating Student objects with optional attributes.

**MainView**

* **Purpose:** Initializes and displays the main GUI.
* **Details:**
  + JFrame with forms and tables for interacting with student and course data.

**MainController**

* **Purpose:** Coordinates between the GUI and backend logic.
* **Details:**
  + Singleton pattern to ensure consistent application behavior.

**StudentAdapter**

* **Purpose:** Converts backend data into a GUI-compatible format.
* **Details:**
  + Converts List<Student> to DefaultTableModel for displaying in tables.

**Database Schema**

**Tables**

1. **Students**
   * id (Primary Key)
   * name
   * email
2. **Courses**
   * id (Primary Key)
   * name
   * description
3. **Enrollments**
   * student\_id (Foreign Key)
   * course\_id (Foreign Key)