This project is a simple desktop-based Student Management System built using Java and the Swing library. The purpose of the application is to provide a basic interface where users can add and remove student information, specifically each student’s name and grade. It demonstrates fundamental object-oriented programming concepts and how graphical user interfaces can be built and managed in Java.

The program is built around a graphical window created by extending the JFrame class. Inside the frame, a JPanel called MainPanel is used to hold all the visible components. Swing components such as JLabel, JTextField, JButton, and JTable are used to create the interface. The labels display the field names (“Name” and “Grade”), the text fields allow the user to enter data, and the buttons handle adding and removing students. The table displays all the students that have been entered. The layout of the interface is manually controlled with setBounds() since a null layout is used, which gives direct control over positioning each component.

Behind the graphical interface, the application maintains a list of students using an ArrayList<Student>. The Student class is a simple inner class that holds two variables: name and grade. Every time a new student is added, a new Student object is created and stored in this list. In addition to this internal list, the program uses a DefaultTableModel, which manages how data appears inside the JTable. This table model is connected to the table so that any changes made to it are immediately reflected visually.

When the user clicks the “Add” button, the program retrieves the values from the name and grade text fields. It then creates a new Student object and adds it to the ArrayList. A new row containing the student’s information is also added to the DefaultTableModel, which updates the table on the screen. After the data is added, the input fields are cleared to make it easy to enter another student. This provides a smooth, interactive way to add information without needing to restart the program or refresh the screen.

The “Remove” button performs the opposite function. When clicked, it checks which row is selected in the table. If a row is selected, it removes that row from both the DefaultTableModel and the ArrayList. This keeps the visual display and the internal data consistent. If no row is selected, the program does nothing, preventing errors that might occur from trying to remove a non-existent entry.

The main method of the program uses SwingUtilities.invokeLater() to ensure that the graphical interface is launched on the correct thread. This is standard practice in Java Swing development, as it ensures the GUI operates smoothly and avoids potential threading issues.

Overall, this project demonstrates the core principles of GUI programming and event-driven logic in Java. It shows how user interactions, such as button clicks, can trigger specific functions that modify both data and the interface. It also shows how an internal data structure (the ArrayList) can be synchronized with a visual component (the JTable) through the use of a table model. The program is simple, but it provides a solid foundation for understanding how desktop applications in Java handle input, display, and interaction.

Possible improvements for future versions could include adding input validation to prevent empty or invalid entries, saving and loading data from a file or database, or improving the layout by replacing manual positioning with a layout manager. Features such as sorting or searching the table could also make the program more practical. Despite its simplicity, this version of the Student Management System provides a clear demonstration of how Swing components and Java logic can work together to create a functional and interactive application.

