Project Documentation
STUDENT MANAGEMENT SYSTEM (SMS)

Introduction

The 'Student Management System' is a Java application that allows users to manage student records. It provides functionalities such as adding new students, viewing existing students, searching for students, updating student information, deleting students, and generating reports. By digitizing processes and centralizing information, it improves efficiency, communication, and data security. The system benefits administrators, teachers, parents, and students by simplifying tasks, enhancing collaboration, and providing valuable insights through reporting and analytics.

Class Structure

- **Student:** This class represents a student entity and contains attributes such as name, roll number, course, and marks. It provides getter and setter methods for accessing and modifying the student data.
- **StudentManagementSystem1:** This class extends the '`JFrame`' class to create a graphical user interface (GUI) for the student management system. It includes components such as text fields, buttons, and a table to display student records. It also handles user actions through event listeners and performs the corresponding operations.

Functionality

The 'Student Management System' provides the following functionalities:

1. **Fill Students:** This method reads student data from a file named "student.txt" and populates the `students` ArrayList with Student objects.
2. **Create GUI:** This method creates the graphical user interface for the application. It sets up the mainframe, input fields, buttons, and a table to display student records.
3. **Register Listeners:** This method registers event listeners for the buttons in the GUI. Each button is associated with a specific action, such as adding a student, viewing student records, searching for a student, updating student information, deleting a student, or generating a student report.
4. **Add Student:** This method validates the input fields and adds a new student to the `students` ArrayList and the table. It also writes the new student's data to the "student.txt" file.
5. **View Students:** This method displays all the student records in the table.
6. **Search Student:** This method allows users to search for a student by roll number or name. It displays the matching student's record in the table.
7. **Update Student:** This method enables users to update a student's information. It prompts the user to enter the roll number or name of the student to be updated

and provides options to modify the student's name, roll number, course, or marks. The updated information is reflected in the table and saved to the "student.txt" file.

8. **Delete Student:** This method allows users to delete a student from the records. It prompts the user to enter the roll number or name of the student to be deleted and removes the student's record from the `students` ArrayList and the table. The updated data is saved to the "student.txt" file.
9. **Generate Report:** This method generates a report for a specific student. It prompts the user to enter a roll number and calculates statistics such as average marks, percentage, highest marks, lowest marks, total subjects, and passed subjects. The report is displayed in a dialog box.
10. **Clear Fields and Table:** These methods clear the input fields and table, respectively.
11. **Main Method:** The main method creates an instance of the StudentManagementSystem1 class, making the application visible and starting its execution.

Usage

To use the 'Student Management System' application:

1. Compile and run the Java program.
2. The application window will open, displaying input fields, buttons, and a table.
3. Use the "Add" button to add a new student by entering the required information in the input fields. The student record will be displayed in the table and saved to the "student.txt" file.
4. Use the "View" button to display all student records in the table.
5. Use the "Search" button to search for a student by roll number or name.
6. Use the "Update" button to update a student's information. Enter the roll number or name of the student you want to update, select the desired field to modify (name, roll number, course, or marks), and enter the new value. The updated information will be reflected in the table and saved to the "student.txt" file.

7. Use the "Delete" button to delete a student from the records. Enter the roll number or name of the student you want to delete, and the student's record will be removed from the table and the "student.txt" file.
8. Use the "Report" button to generate a report for a specific student. Enter the roll number of the student, and a dialog box will display the student's statistics, such as average marks, percentage, highest marks, lowest marks, total subjects, and passed subjects.
9. Use the "Clear" button to clear all input fields.

File Handling

The 'Student Management System' application uses file handling to read student data from and write new student data to a file named **"student.txt"**. The file is located in the same directory as the Java program. The data in the file is stored in a comma-separated format, where each line represents a student's record with the format: **"name,rollNumber,course,marks"**. When the application starts, it reads the data from the file and populates the student records.

Dependencies

The 'Student Management System' application is built using Java and utilizes the Swing library for creating the graphical user interface. There are no external dependencies required to run the application.

Limitations and Future Enhancements

The provided code for the 'Student Management System' is a basic implementation and may have a few limitations, such as:

1. **Lack of input validation:** The code does not include extensive input validation, so it assumes valid input from the user. It would be beneficial to add input validation to ensure the correctness of the entered data.
2. **Limited error handling:** The code does not handle various error scenarios, such as incorrect file format or non-existent file. Implementing proper error handling mechanisms would enhance the application's robustness.
3. **User authentication and security:** The application currently does not include user authentication or secure data storage. Adding authentication mechanisms and securing the student records would be crucial for real-world usage.
4. **Database integration:** The application uses a file for storing data. However, for a more scalable and efficient solution, integrating a database would be a recommended enhancement.

These limitations and potential enhancements should be considered if the application is to be further developed for production use.

Conclusion

The 'Student Management System' is a Java application that provides functionalities for managing student records. It allows users to add, view, search, update, and delete student records. It also provides the ability to generate reports for individual students. The application utilizes file handling to store and retrieve student data. With further enhancements and refinements, this system can be expanded to meet the requirements of a real-world student management system.

Project and Documentation prepared by -

E.No. 12 - Zeeshan Sharif
E.No. 43 - Aryan Patel