SCHOOL SYSTEM MANAGEMENT

Done by

M.PAVITHRA {AD1-34| 231191101128}

M.MUBEENA {AD1-39 | 231191101112}

R.KEERTHANA{AD1-05|231191101077}

V.MONICA {AD1-06 | 231191101110}

S.POOJA{AD1-43 | 231191101130}

Key Concepts of the School Management System Project

The School Management System project employs several key concepts from software development and object-oriented programming (OOP). Here are the primary concepts utilized:

1. **Object-Oriented Programming (OOP):**
    - **Encapsulation:** This concept is used to bundle the data (attributes) and methods (functions) that operate on the data into a single unit called a class. For instance, `Student`, `Teacher`, and `Course` classes encapsulate their respective data and methods.
    - **Abstraction:** Abstraction is applied to hide complex implementation details and expose only the necessary parts. Each class provides a clear interface (methods) to interact with the objects without revealing internal details.
    - **Inheritance (Potential):** Although not explicitly implemented in the provided code, inheritance can be used to create a hierarchical relationship between classes. For example, a common superclass `Person` could be created for `Student` and `Teacher` classes.
    - **Polymorphism (Potential):** Polymorphism allows objects of different classes to be treated as objects of a common superclass. This can be useful for creating flexible and reusable code. For instance, methods can be designed to handle objects of the `Person` class, which could be either `Student` or `Teacher` objects.
2. **Classes and Objects:**
    - **Classes:** The project defines several classes (`Student`, `Teacher`, `Course`) to represent the main entities. Each class encapsulates data and methods relevant to that entity.
    - **Objects:** Instances of these classes are created and manipulated to perform the desired operations. For example, objects of the `Student` class represent individual students.
3. **Collections Framework:**
    - The Java Collections Framework is used to manage groups of objects. For instance, `ArrayList` is used to store lists of students, teachers, and courses. This allows dynamic resizing and provides useful methods for manipulating the lists.
4. **User Input Handling:**
    - The `Scanner` class is used to read user input from the console. This enables interaction with the user, allowing them to add students, teachers, and courses, and perform other operations via a menu-driven interface.
5. **Menu-Driven Interface:**
    - A simple console-based menu-driven interface guides the user through various options. This approach simplifies user interactions and makes the application easy to use.

6. **Modularity:**
    - The project is structured in a modular fashion, where each class has a specific responsibility. This separation of concerns makes the code easier to understand, maintain, and extend.

Summary of Key Concepts:

- **Object-Oriented Programming:** Encapsulation, Abstraction, (potential) Inheritance, (potential) Polymorphism.
- **Classes and Objects:** Representing real-world entities and their interactions.
- **Collections Framework:** Managing dynamic groups of objects.
- **User Input Handling:** Interacting with the user through the console.
- **Menu-Driven Interface:** Providing an intuitive way for users to interact with the system.
- **Modularity:** Structuring code for clarity and maintainability.

These key concepts collectively contribute to the design and implementation of the School Management System project, showcasing fundamental principles of software development and OOp

Packages Used:

1. **java.util:**
    - **ArrayList:**
        - A resizable array implementation of the `List` interface. It is used in the project to store lists of students, teachers, and courses. The `ArrayList` class allows dynamic resizing and provides methods for adding, removing, and accessing elements.
    - **List:**
        - An interface representing an ordered collection that can contain duplicate elements. The `ArrayList` class implements this interface, allowing for a flexible and dynamic collection of objects.
    - **Scanner:**
        - A class used for parsing primitive types and strings using regular expressions. In this project, `Scanner` is used to read user input from the console, enabling interaction with the user through a menu-driven interface.

Methods Used:

Student Class:

- **Constructor:**
    - Initializes a new `Student` object with the given attributes (ID, name, and grade). This method sets up the initial state of the student.
- **Getters and Setters:**
    - Getters are methods used to retrieve the values of private attributes (e.g., `getId()`, `getName()`, `getGrade()`).
    - Setters are methods used to modify the values of private attributes (e.g., `setGrade(int grade)`).

Teacher Class:

- **Constructor:**
  - Initializes a new `Teacher` object with the given attributes (ID, name, and subject). This method sets up the initial state of the teacher.
- **Getters:**
  - Methods used to retrieve the values of private attributes (e.g., `getId()`, `getName()`, `getSubject()`).

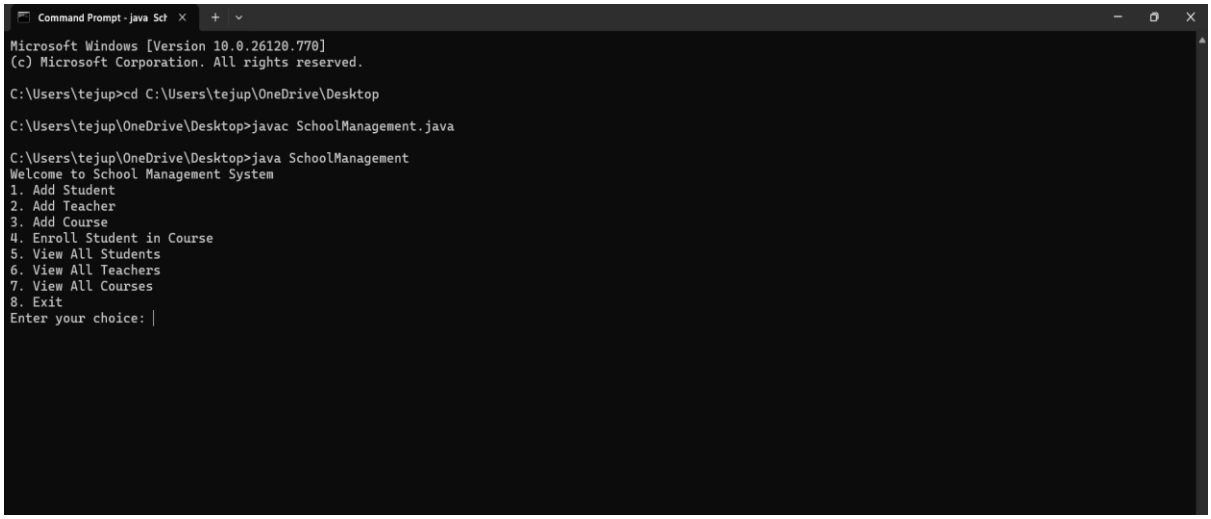
Course Class:

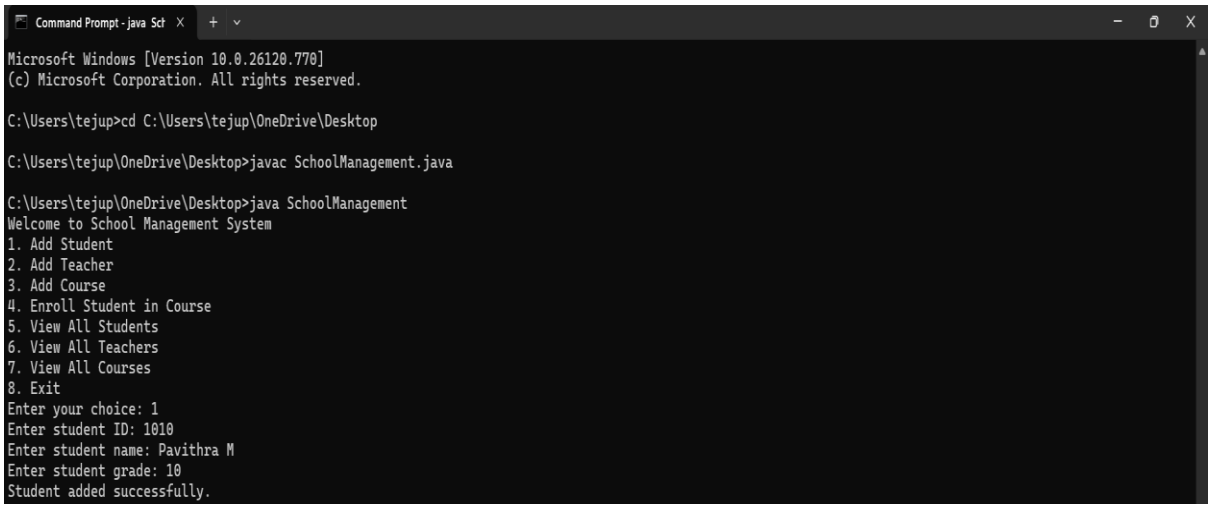
- **Constructor:**
  - Initializes a new `Course` object with the given attributes (course ID, name, and assigned teacher). This method sets up the initial state of the course.
- **Methods to Enroll and Remove Students:**
  - `enrollStudent(Student student)`: Adds a student to the list of enrolled students for the course.
  - `removeStudent(Student student)`: Removes a student from the list of enrolled students.
- **Getters:**
  - Methods used to retrieve the values of private attributes (e.g., `getCourseId()`, `getName()`, `getTeacher()`, `getStudentsEnrolled()`).

SchoolManagementApp Class:

- **Main Method:**
  - The entry point of the application (`public static void main(String[] args)`). This method contains the main logic and provides a menu-driven interface for user interaction.
- **Menu and User Interaction Methods:**
  - `addStudent()`: Prompts the user for student details and adds a new student to the list.
  - `addTeacher()`: Prompts the user for teacher details and adds a new teacher to the list.
  - `addCourse()`: Prompts the user for course details and adds a new course to the list.
  - `enrollStudentInCourse()`: Prompts the user to select a course and a student to enroll the student in the selected course.
  - `viewAllStudents()`: Displays a list of all students.
  - `viewAllTeachers()`: Displays a list of all teachers.
  - `viewAllCourses()`: Displays a list of all courses along with the students enrolled in each course.

OUTPUT


Command Prompt - java Sch
Microsoft Windows [Version 10.0.26120.770]
(c) Microsoft Corporation. All rights reserved.
C:\Users\tejup>cd C:\Users\tejup\OneDrive\Desktop
C:\Users\tejup\OneDrive\Desktop>javac SchoolManagement.java
C:\Users\tejup\OneDrive\Desktop>java SchoolManagement
Welcome to School Management System
1. Add Student
2. Add Teacher
3. Add Course
4. Enroll Student in Course
5. View All Students
6. View All Teachers
7. View All Courses
8. Exit
Enter your choice:


Command Prompt - java Sch
Microsoft Windows [Version 10.0.26120.770]
(c) Microsoft Corporation. All rights reserved.
C:\Users\tejup>cd C:\Users\tejup\OneDrive\Desktop
C:\Users\tejup\OneDrive\Desktop>javac SchoolManagement.java
C:\Users\tejup\OneDrive\Desktop>java SchoolManagement
Welcome to School Management System
1. Add Student
2. Add Teacher
3. Add Course
4. Enroll Student in Course
5. View All Students
6. View All Teachers
7. View All Courses
8. Exit
Enter your choice: 1
Enter student ID: 1010
Enter student name: Pavithra M
Enter student grade: 10
Student added successfully.

```
Welcome to School Management System
1. Add Student
2. Add Teacher
3. Add Course
4. Enroll Student in Course
5. View All Students
6. View All Teachers
7. View All Courses
8. Exit
Enter your choice: 1
Enter student ID: 1011
Enter student name: Shalini
Enter student grade: 9
Student added successfully.
Welcome to School Management System
1. Add Student
2. Add Teacher
3. Add Course
4. Enroll Student in Course
5. View All Students
6. View All Teachers
7. View All Courses
8. Exit
Enter your choice: 1
Enter student ID: 1012
Enter student name: Harini
Enter student grade: 9
Student added successfully.
Welcome to School Management System
1. Add Student
2. Add Teacher
3. Add Course
4. Enroll Student in Course
5. View All Students
6. View All Teachers
7. View All Courses
8. Exit
Enter your choice: 5
All Students:
ID: 1010, Name: Pavithra M, Grade: 9
ID: 1011, Name: Shalini, Grade: 9
ID: 1012, Name: Harini, Grade: 9
```

```
Enter your choice: 2
Enter teacher ID: 093
Enter teacher name: Gayathri
Enter subject taught: Maths
Teacher added successfully.
Welcome to School Management System
1. Add Student
2. Add Teacher
3. Add Course
4. Enroll Student in Course
5. View All Students
6. View All Teachers
7. View All Courses
8. Exit
Enter your choice: 2
Enter teacher ID: 094
Enter teacher name: Anbu
Enter subject taught: Science
Teacher added successfully.
Welcome to School Management System
1. Add Student
2. Add Teacher
3. Add Course
4. Enroll Student in Course
5. View All Students
6. View All Teachers
7. View All Courses
8. Exit
Enter your choice: 2
Enter teacher ID: 095
Enter teacher name: Divya
Enter subject taught: English
Teacher added successfully.
Welcome to School Management System
1. Add Student
2. Add Teacher
3. Add Course
4. Enroll Student in Course
5. View All Students
6. View All Teachers
7. View All Courses
8. Exit
Enter your choice: 6
All Teachers:
ID: 93, Name: Gayathri, Subject: Maths
ID: 94, Name: Anbu, Subject: Science
ID: 95, Name: Divya, Subject: English
```