SparkINN Novice Solution Pvt. Ltd

1st Weekend Task: Building a Command-Line Student Management System

Objective: Create a simple command-line student management system that applies the core Java concepts you've learned throughout the course. This project will help you practice Java basics, data types, control structures, and object-oriented programming.

Problem Statement: Develop a "Student Management System" that allows users to manage a list of students and their grades. The application should provide functionalities like adding a new student, viewing student details, updating student information, and calculating the average grade of all students.

Detailed Requirements:

1. **Introduction to Java Programming:**
   - Create a Java class named `StudentManagementSystem`.
   - Print a welcome message when the application starts.
2. **Anatomy of a Java Program:**
   - Ensure your `StudentManagementSystem` class includes the `main` method as the entry point of the application.
3. **Displaying Messages and Numbers in Java:**
   - Display user prompts and results (like student details, grades, etc.) in the console.
   - Ensure correct usage of `System.out.println` and formatting techniques.
4. **Configuring the Java Development Environment:**
   - Document the steps to set up the Java Development Environment in your README file.
5. **Creating, Compiling, and Executing a Java Program:**
   - Ensure that your `StudentManagementSystem` program can be compiled and executed from the command line.
6. **Java Packages, Classes, and Methods:**
   - Organize your code into packages. Create at least two classes: `Student` and `StudentManagementSystem`.
   - Implement methods like `addStudent`, `viewStudent`, `updateStudent`, and `calculateAverageGrade`.
7. **Public, Private, and Static in Java:**
   - Use `private` for member variables in the `Student` class.
   - Use `public` for methods that need to be accessed from the `StudentManagementSystem` class.
   - Use `static` methods where appropriate, such as for utility functions.
8. **The void Return Type in Java:**
   - Use `void` for methods like `addStudent` and `viewStudent`, which perform actions but do not return a value.
9. **Introduction to Variables and Constants in Java:**
   - Declare variables for student attributes like `name`, `id`, and `grades`.
   - Use `final` to declare constants, such as maximum or minimum grades.

10. **Identifiers in Java:**
    - Follow proper naming conventions for variables, methods, and classes.
11. **Introduction to Data Types in Java:**
    - Use appropriate data types for different variables (`String` for names, `int` for IDs, `double` for grades, etc.).
12. **The byte, short, and long Data Types in Java:**
    - Use these data types where necessary, such as in counting operations or specific scenarios.
13. **The int Data Type in Java - Practice:**
    - Implement features that use integer data types, such as counting the number of students or calculating sums.
14. **Bytes and Values in Java:**
    - Handle scenarios where smaller data types could be used, or demonstrate the concept through example calculations.
15. **The double and float Data Types in Java:**
    - Use `double` for student grades to allow decimal points.
16. **The char Data Type in Java:**
    - Optionally include a feature where a student's grade letter (A, B, C, etc.) is determined based on their numerical grade.
17. **The boolean Data Type in Java:**
    - Use `boolean` for control flags, such as determining whether a student is passing or failing.
18. **The String Data Type in Java:**
    - Use `String` for names, and other textual data.
19. **Concatenating Strings in Java:**
    - Combine strings to display full sentences or messages, such as "Student John Doe has an average grade of 85.5".
20. **Creating a String Object:**
    - Demonstrate different ways to create strings, such as using the `new` keyword or string literals.
21. **Strings are Immutable in Java:**
    - Explain in your code comments why strings are immutable and show how this affects string manipulation.
22. **The Scanner Class in Java:**
    - Use `Scanner` to take user input for adding or updating student details.

Deliverables:

- A fully functional Java program that runs in the command line.
- A GitHub repository link with the project files.
- A README file explaining how to compile, run the program, and a description of each feature.

Evaluation Criteria:

- Proper use of Java syntax and concepts.
- Correct application of object-oriented principles.
- Code clarity and organization.
- Handling user inputs and displaying outputs correctly.
- Comments explaining key parts of the code.

Submission Deadline: Sunday night, 11:59 PM.