CERTIFICATE

This is to certify that [Your Name], a student of Class XII-[Section] at [Your School Name], has successfully completed the project titled **“Bank Management System”** under the supervision of [Project Supervisor Name] during the academic year [2025-26].

This project has been completed in fulfillment of the requirements for the Computer Science course and adheres to the guidelines issued by CBSE, New Delhi.

The project work meets the expected standards and has been evaluated by the undersigned.

Signature of Principal: __________________________

Signature of Supervisor: _________________________

Signature of External Examiner: ________________

Date: ______________

Place: ______________

Student Details:

Student Names:

Class : XII

Subject: Information Technology

School Name: Torch Bearers Convent School

Affiliation By.: CBSE Board

ABSTRACT

The Student Grade Calculator is a software application developed to simplify the process of calculating student grades based on the marks obtained in various subjects. This project uses fundamental programming concepts such as arrays, loops, and conditionals to accept multiple marks as input, compute the total percentage, and assign grades according to predefined criteria. The tool aims to provide an accurate, efficient, and automated solution for educators and students to quickly evaluate academic performance. By automating the grade calculation process, this project reduces errors associated with manual calculations, offers transparency in grading, and enhances the overall management of student results.

Table of Contents

Introduction

In educational institutions, evaluating student performance accurately and efficiently is crucial for academic progress monitoring. The Student Grade Calculator project aims to automate the calculation of students' grades based on their marks in various subjects. Using fundamental programming concepts such as arrays, loops, and conditional statements, this project takes input marks, computes the percentage, and assigns corresponding grades according to predefined criteria.

This automated approach reduces manual errors, saves time for educators, and provides immediate feedback to students about their academic standings. The system is designed to be simple, user-friendly, and flexible enough to handle marks for multiple subjects and generate grades following standard grading scales.

By implementing this project, the team enhances understanding of essential programming constructs while creating a practical tool that supports academic evaluations effectively.

Objectives

- Automate student marks handling
- Calculate percentage scores
- Assign grades based on percentage thresholds
- Implement with programming constructs like arrays, loops, and conditionals.

System Design and Classes

The Student Grade Calculator is designed as a modular program leveraging arrays, loops, and conditionals to manage student marks and calculate grades efficiently. The core design includes:

1. Student Class:

- Attributes: name (String), marks (int array), percentage (float), grade (char).
- Methods: calculatePercentage() to compute the percentage based on marks, calculateGrade() to assign a letter grade based on percentage ranges.
- Encapsulation ensures data is handled securely with appropriate getter and setter methods.

2. Main Application:

- Uses an array of Student objects to store multiple student records.
- Implements input methods to accept marks for each student across subjects.
- Utilizes loops to iterate through the student array for processing calculations and displaying results.

3. Control Flow:

- Conditionals execute grading logic by comparing percentage to predefined grade boundaries (e.g., 90%+ = 'A', 80-89% = 'B', etc.).
- Validations are included to handle invalid inputs and ensure robust data handling.

Core Components and Code Snippets

- Student.java - Represents a student with name, marks, percentage, and grade calculation logic.

```java
/**
 * Represents a student with name, marks per subject, and grade.
 */
public class Student {
    private String name;
    private int[] marks;
    private double percentage;
    private char grade;

    public Student(String name, int[] marks) {
        this.name = name;
        this.marks = marks;
        calculatePercentage();
        calculateGrade();
    }

    private void calculatePercentage() {
        int total = 0;
        for (int mark : marks) {
            total += mark;
        }
        this.percentage = (double) total / marks.length;
    }

    private void calculateGrade() {
        if (percentage >= 90)
            grade = 'A';
        else if (percentage >= 80)
            grade = 'B';
        else if (percentage >= 70)
            grade = 'C';
        else if (percentage >= 60)
            grade = 'D';
        else
            grade = 'F';
    }

    public String getName() { return name; }
    public int[] getMarks() { return marks; }
    public double getPercentage() { return percentage; }
    public char getGrade() { return grade; }
}
```

- StudentManager.java - Manages a collection of Student objects.

```java
import java.util.ArrayList;
import java.util.List;

/**
 * Manages collection of Student objects.
 */
public class StudentManager {
    private List<Student> students;

    public StudentManager() {
        students = new ArrayList<>();
    }
    public void addStudent(Student student) {
        students.add(student);
    }

    public List<Student> getStudents() {
        return students;
    }
}
```

-

- GradeCalculator.java (optional) - Contains static methods for grade determination.

```
/**
 * Provides static methods to calculate grades.
 * (Optional extra class if grading logic separated)
 */
public class GradeCalculator {
    public static char calculateGrade(double percentage) {
        if (percentage >= 90)
            return 'A';
        else if (percentage >= 80)
            return 'B';
        else if (percentage >= 70)
            return 'C';
        else if (percentage >= 60)
            return 'D';
        else
            return 'F';
    }
}
```

-

- StudentInput.java - Handles user input for students and their marks.

```
import java.util.Scanner;

/**
 * Handles user input for student details and marks.
 */
public class StudentInput {
    private Scanner scanner;

    public StudentInput() {
        scanner = new Scanner(System.in);
    }

    public Student getStudentFromInput() {
        System.out.print("Enter student name: ");
        String name = scanner.nextLine();

        System.out.print("Enter number of subjects: ");
        int numSubjects = scanner.nextInt();
        scanner.nextLine(); // consume newline

        int[] marks = new int[numSubjects];
        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter marks for subject " + (i+1) + ": ");
            marks[i] = scanner.nextInt();
        }
        scanner.nextLine(); // consume newline after last input

        return new Student(name, marks);
    }
}
```

-

- StudentGradeApp.java - Main application class coordinating input, calculation, and output

```java
/**
 * Main class to run the Student Grade Calculator application.
 */
public class StudentGradeApp {
    public static void main(String[] args) {
        StudentManager manager = new StudentManager();
        StudentInput input = new StudentInput();

        Student student = input.getStudentFromInput();
        manager.addStudent(student);

        System.out.println("\nResults:");
        System.out.println("Name: " + student.getName());
        System.out.printf("Percentage: %.2f%%\n", student.getPercentage());
        System.out.println("Grade: " + student.getGrade());
    }
}
```

Transaction Handling

The core of the Student Grade Calculator lies in its ability to accurately handle the input and processing of student marks, calculate the total marks, compute the percentage, and assign a grade based on predefined criteria using arrays, loops, and conditionals.

Steps Involved in Transaction Handling

1. **Input Handling:**
    - The program accepts marks for each subject as input from the user, validating to ensure that no entry fields are left empty or contain invalid data.
    - Array data structures store the marks for ease of iteration and processing.
2. **Calculation of Total Marks and Percentage:**
    - Using a loop, the program sums all marks stored in the array to compute the total marks obtained by the student.
    - The percentage is then calculated by dividing the total marks by the maximum possible marks and multiplying by 100.
3. **Grade Assignment:**
    - Conditional statements evaluate the percentage and assign the corresponding grade as per defined thresholds.
    - For example:
        - Percentage ≥ 80% → Grade A
        - 60% ≤ Percentage < 80% → Grade B
        - 40% ≤ Percentage < 60% → Grade C
        - Percentage < 40% → Grade F
4. **Error Checking and Validation:**
    - The system checks for missing or invalid inputs and prompts the user accordingly to enter correct values.
5. **Output Generation:**
    - The results including total marks, percentage, grade, and pass/fail status are displayed clearly to the user.

Sample Code Logic (Pseudocode)

text

```
Initialize an array to hold marks for subjects
totalMarks = 0
maxMarks = totalSubjects * marksPerSubject

For each mark in the array:
    totalMarks += mark

percentage = (totalMarks / maxMarks) * 100

If percentage >= 80:
    grade = 'A'
Else if percentage >= 60:
    grade = 'B'
Else if percentage >= 40:
    grade = 'C'
Else:
    grade = 'F'

Display totalMarks, percentage, grade, and pass/fail message
```

This transaction handling ensures efficient processing of student data with high accuracy and user-friendly feedback, making the grade calculation seamless and reliable.

Conclusion and Team Contribution

The Student Grade Calculator project successfully meets its objective of automating the calculation of student grades from marks entered using arrays, loops, and conditional statements. This application efficiently handles multiple student marks, calculates percentages accurately, and assigns grades based on predefined criteria. The project not only simplifies the process of grade calculation but also provides an interactive and user-friendly interface for users to input marks and view results instantly. Throughout the development, the team gained valuable experience in implementing fundamental programming concepts and practical problem-solving skills.

Team Contribution

- **Kashish (Adhana):** Took the lead in designing the system architecture and implementing the core calculation logic using arrays and loops.
- **Bhavna:** Developed the user interface for inputting marks and displaying calculated results with proper validation.
- **Anjali (Gill):** Managed code testing, debugging, and documentation to ensure a smooth and error-free application.
- **Mayank (Adhana):** Assisted with integrating conditional grading logic and contributed to the project report and presentation.

Together, the team collaborated effectively to accomplish the project goals within the timeline, ensuring high code quality and documentation standards. This project has enhanced our programming skills and teamwork experience.