CS1102-01 | PROGRAMMING ASSIGNMENT UNIT 5 BONAVENTURE OKOYE

COURSE ENROLLMENT AND GRADE MANAGEMENT SYSTEM

This project is a simple command-line based course management system that allows administrators to manage courses and students. The system provides functionality to enroll students in courses, assign grades to students, and calculate overall course grades for each student. It consists of four main classes: Student, Course, CourseManagement, and AdministratorInterface.

Student: This class represents a student.

- name: A String variable that stores the name of the student.
- id: An integer variable that stores the ID of the student.
- enrolledCourses: A Map that stores the courses the student is enrolled in, mapped to the grade the student has in that course.

Course: This class represents a course.

- name: A String variable that stores the name of the course.
- code: An integer variable that stores the course code.
- maxCapacity: An integer variable that stores the maximum capacity of the course.
- enrolledStudents: An ArrayList that stores the students enrolled in the course.

CourseManagement: This class is the main entry point of the application. It contains the main method and several static methods and variables to manage the course and student data.

- courseList: A static ArrayList that stores all the Course objects.
- studentList: A static ArrayList that stores all the Student objects.
- scanner: A static Scanner object used to get user input from the command line.
- main: The main method that initializes the Scanner and starts the program.
- start: A static method that displays the administrator menu and handles user choices.
- addCourse: A static method that adds a new course to the course list.
- enrollStudent: A static method that enrolls a student in a course.
- assignGrade: A static method that assigns a grade to a student for a course.
- displayCourses: A static method that displays information about all courses.
- displayStudents: A static method that displays information about all students.

Static methods and variables are used in the CourseManagement class to track enrollment and grade-related information across multiple instances of the Student and Course classes. This allows the data to be accessed and modified from anywhere in the program without having to create an instance of the class.

Administrator: The Administrator class is designed to manage the course and student data. It has several methods to handle various tasks, such as adding a course, enrolling a student, assigning a grade, displaying courses, and displaying students.

- addCourse: A static method that takes parameters for the course name, course code, and maximum capacity. It creates a new Course object with the provided data and adds it to the courseList.
- enrollStudent: A static method that takes parameters for the student's name, ID, and the course code for the course you want to enroll them in. It creates a new Student object with the provided data, adds it to the studentList, and updates the enrolledStudents list in the respective Course object.
- assignGrade: A static method that takes parameters for the student's ID and the course code for the course you want to assign a grade for, along with the grade itself. It finds the respective Student and Course objects and updates the grade in the enrolledCourses map of the Student object.
- displayCourses: A static method that iterates through the courseList and prints out information about each course, including the course name, code, maximum capacity, and the students enrolled in the course.
- displayStudents: A static method that iterates through the studentList and prints out information about each student, including their name, ID, and the courses they are enrolled in along with their grades.

Student.java | Course.java | CourseManagement.java | AdministratorInterface.java

Analyzing

```java
import java.util.Arrays;
import java.util.Optional;
import java.util.Scanner;

public class AdministratorInterface {
    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("Welcome to the Administrator Interface");
        start();
    }

    private static void start() {
        System.out.println("\n(A:Add_New_Course B:Enroll_Student C:Assign_Grades D:Calculate_Overall_Grades E:Exit)");

        String input = scanner.nextLine().toUpperCase();

        switch (input) {
            case "A":
                addCourse();
                break;

            case "B":
                System.out.println("New Student? Y/N");
                String newStudent = scanner.nextLine().toUpperCase();

                if (newStudent.equals("Y")) {
                    enrollNewStudent();

                } else if (newStudent.equals("N")) {
                    enrollExistingStudent();

                } else {
                    System.out.println("Error: Unexpected Input");
                    start();
                }
                break;

            case "C":
                assignGrades();
                break;
            case "D":
                calculateOverallGrades();
                break;
            case "E":
                System.out.println("Goodbye!");
                System.exit(0);
            default:
                System.out.println("Error: Unexpected input");
                start();
                break;
        }
    }

    public static void addCourse() {
        System.out.println("Enter Course Name:");
        String name = scanner.nextLine();

        System.out.println("Enter Course Code:");
        int code;
        try {
            code = Integer.parseInt(scanner.nextLine());
        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid input for Course Code. Please enter a valid integer.");
            start();
            return;
        }
```

```
68
69          System.out.println("Enter Maximum Capacity:");
70          int maxCapacity;
71          try {
72              maxCapacity = Integer.parseInt(scanner.nextLine());
73          } catch (NumberFormatException e) {
74              System.out.println("Error: Invalid input for Maximum Capacity. Please enter a valid integer.");
75              start();
76              return;
77          }
78
79          CourseManagement.addCourse(name, code, maxCapacity);
80
81          System.out.println("Successfully added " + name + "\nCourse code: " + code);
82          start();
83      }
84
85      public static void enrollNewStudent() {
86          System.out.println("Enter Student Name:");
87          String name = scanner.nextLine();
88
89          System.out.println("Enter Student ID:");
90          int ID = scanner.nextInt();
91          scanner.nextLine();
92
93          boolean studentExists = false;
94          for (Student existingStudent : CourseManagement.getStudentList()) {
95              if (ID == existingStudent.getID()) {
96                  System.out.println("Error: ID already exists");
97                  studentExists = true;
98                  break;
99              }
100         }
101         if (studentExists) return;
102
103         Student student = new Student(name, ID);
104
105         while (true) {
106             System.out.println("Enter Course Code:");
107             int code = scanner.nextInt();
108             scanner.nextLine();
109
110             boolean courseExists = Arrays.stream(CourseManagement.getCourseList()).anyMatch(course -> course.getCourseCo
111
112             if (courseExists) {
113                 Optional<Course> courseOpt = Arrays.stream(CourseManagement.getCourseList()).filter(course -> course.get
114
115                 if (courseOpt.isPresent()) {
116                     Course course = courseOpt.get();
117                     CourseManagement.enrollStudent(student, course);
118                     System.out.println("Successfully enrolled " + student.getName() + " to " + course.getName());
119                     break;
120                 }
121             } else {
122                 System.out.println("Error: Course Code " + code + " doesn't exist");
123             }
124         }
125         start();
126     }
127
```

```java
    private static void enrollExistingStudent() {
        System.out.println("Enter Student ID:");
        int ID = scanner.nextInt();
        scanner.nextLine();
        Student student = getStudent(ID);

        if (student == null) {
            System.out.println("Error: ID doesn't exist");
            enrollExistingStudent();

        } else {
            System.out.println("Enter Course Code:");
            int code = scanner.nextInt();
            scanner.nextLine();
            Course course = getCourse(code);

            if (course != null) {
                System.out.println("Error: Course Code " + code + " doesn't exist");
                enrollExistingStudent();

            } else {
                CourseManagement.enrollStudent(student, course);
                System.out.println("Successfully enrolled " + student.getName() + " to " + course.getName());
                System.out.println("Student ID:" + ID);
                start();
            }
        }
    }

    private static void assignGrades() {
        System.out.println("Enter Student ID:");
        int ID = scanner.nextInt();
        scanner.nextLine();
        Student student = getStudent(ID);

        if (student == null) {
            System.out.println("Error: ID doesn't exist");
            assignGrades();

        } else {
            System.out.println("Enter Course Code:");
            int code = scanner.nextInt();
            scanner.nextLine();
            Course course = getCourse(code);

            if (course == null) {
                System.out.println("Error: Course code" + code + " doesn't exist");
                assignGrades();

            } else {
                System.out.println("Enter Grade (1-100):");
                int grade = scanner.nextInt();
                scanner.nextLine();

                CourseManagement.assignGrade(student, course, grade);
                System.out.println(
                        "Successfully added " + grade + " to " + student.getName() + " in " + course.getName());
                start();
            }
        }
    }
```

```
189
190     private static void calculateOverallGrades() {
191         System.out.println("Enter Student ID:");
192         int ID = scanner.nextInt();
193         scanner.nextLine();
194         Student student = getStudent(ID);
195
196         if (student == null) {
197             System.out.println("Error: ID doesn't exist");
198             assignGrades();
199
200         } else {
201             CourseManagement.calculateOverallGrade(student);
202             System.out.println("Successfully calculated Overall Grade");
203             System.out.println(student.getName() +
204                     "'s overall grade is " + CourseManagement.getStudentOverallGrade(student));
205             start();
206         }
207     }
208
209 @   private static Student getStudent(int ID) {
210         for (Student student : CourseManagement.getStudentList()) {
211             if (ID == student.getID()) {
212                 return student;
213             }
214         }
215         return null;
216     }
217
218 @   private static Course getCourse(int code) {
219         for (Course course : CourseManagement.getCourseList()) {
220             if (code == course.getCourseCode()) {
221                 return course;
222             }
223         }
224         return null;
225     }
226 }
```

How to run the program

To run the program, you need to have Java installed on your computer. You can then compile the program using the command javac CourseManagement.java and run it using the command java CourseManagement.

Interacting with the administrator interface

The program will display an administrator menu, and you can interact with the program by entering the corresponding numbers for each option and providing the necessary information when prompted.

Add Course: You will be prompted to enter the name, code, and maximum capacity of the course. The course will then be added to the course list.

Enroll Student: You will first be prompted to choose whether you want to enroll a new student or an existing student. If you choose to enroll a new student, you will be prompted to enter the student's name and ID, and then the course code for the course you want to enroll them in. If you choose to enroll an existing student, you will be prompted to enter the student's ID and the course code.

Assign Grade: You will be prompted to enter the student's ID and the course code for the course you want to assign a grade for. Then, you will be prompted to enter the grade.

Display Courses: This will display information about all courses, including the course name, code, maximum capacity, and the students enrolled in the course.

Display Students: This will display information about all students, including their name, ID, and the courses they are enrolled in along with their grades.

Exit: This will terminate the program.