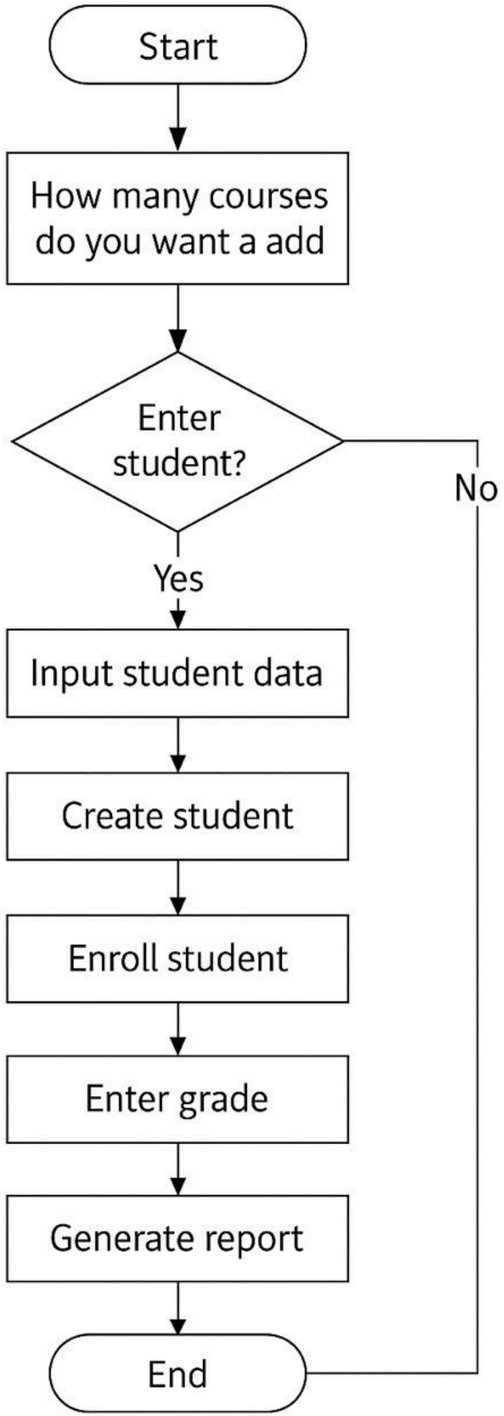
AIII108

FİVECODERS
STUDENT GRADİNG SYSTEM

- İLKE KUDRET
- ESMA KURT
- ENSAR ARIL
- MARİAMAWIT GUGSSA
- MOHAMED ABDELWAHID


Start
How many courses do you want a add
Enter student?
No
Yes
Input student data
Create student
Enroll student
Enter grade
Generate report
End

```python
class Student:
    def __init__(self, name , age , student_id ,student_email , department):
        self.name = name
        self.age = age
        self.student_id = student_id
        self.email = student_email
        self.department = department
        self.grades = {}
        self.attendance = []


    def add_grade(self, subject, score):
        self.grades[subject] = score

    def get_total_score(self):
        return sum(self.grades.values())

    def get_average_grade(self):
        if not self.grades:
            return 0
        return self.get_total_score() / len(self.grades)

    def mark_attendance(self, status):
        self.attendance.append(status)

    def get_attendance_percentage(self):
        if not self.attendance:
            return 0
        present_count = self.attendance.count("present")
        return (present_count / len(self.attendance)) * 100

    def get_status(self):
        status_dict = {}
        for subject, score in self.grades.items():
           status_dict[subject] = "PASS" if score >= 50  else "FAIL"
        return status_dict
```

We wrote this code using the class method. As seen at the beginning, it collects information about the student's identity and in-class performance.

The add_grade method records a course grade, while get_total_score calculates the total points and get_average_grade returns the student's average. The mark_attendance method logs attendance status, and get_attendance_percentage computes the student's participation rate. The get_status method evaluates each course based on a 50-point passing threshold and labels it as PASS or FAIL. Overall, this class provides an organized way to manage and analyze a student's academic performance and attendance.

```python
class Course:
    def __init__(self, course_name):
        self.course_name = course_name
        self.students = []

    def add_student(self, student):
        self.students.append(student)

    def get_course_average(self):
        if not self.students:
            return 0
        total = sum([self.get_average_grade() for s in self.students])
        return total / len(self.students)
```

- The Course class represents a single course in the system. It stores the course name and keeps a list of all students enrolled in that course. The constructor (__init__) initializes two attributes: course_name, which holds the name of the course, and students, which starts as an empty list that will later store Student objects.
- The add_student() method simply adds a Student object to the course's list, allowing the system to track which students belong to this course.
- The get_course_average() method calculates the overall average grade for the course. It first checks whether the course has any students; if not, it returns 0.
- If students exist, it sums each student's average grade using a list comprehension and then divides that total by the number of students.
- This method provides a meaningful overall performance score for the course based on all enrolled students.

```python
class SchoolManagementSystem:
    def _init_(self):
        self.students = {}
        self.courses = {}

    def add_student(self, student):
        self.students[student.student_id] = student

    def add_course(self, course):
        self.courses[course.course_name] = course

    def enroll_student(self, student_id, course_name):
        student = self.students.get(student_id)
        course = self.courses.get(course_name)
        if student and course:
            course.add_student(student)

    def generate_report(self, student_id):
        student = self.students.get(student_id)
        if not student:
            return "Student not found."
        return {
            "Name": student.name,
            "Age": student.age,
            "Department": student.department,
            "Email": student.email,
            "Grades": student.grades,
            "Total Score": student.get_total_score(),
            "Average Grade": student.get_average_grade(),
            "Attendance %": student.get_attendance_percentage(),
            "Status": student.get_status()  }
```

This piece of code forms the central management and reporting brain of the School Management System.

self.students and self.courses.

These dictionaries hold all our student and course objects. They provide quick and unique access, keyed by student IDs and course names.

add_student / add_course:

These methods allow new student and course objects from outside to be added to these central dictionaries and ensure their storage and retention by the system.

enroll_student

This method first retrieves both the student and the course from the central repository (dictionaries) using the given Student ID and Course Name. Then it performs a check: 'If both the student and the course exist in our system',We add the student object to the student list within the course object.

The most important user-facing output of this class is the generate_report method.

This method takes a single student ID and creates a 'report dictionary' that summarizes all of that student's performance.
Note: this method not only retrieves the student's basic data (Name, Department, Age), but also calls the calculation methods in the Student class.

```python
def save_reports_to_excel(self, filename="student_reports.xlsx"):
    import openpyxl
    wb = openpyxl.Workbook()
    ws = wb.active
    ws.title = "Reports"
    ws.append(["Student ID","Name","Age","Department","Email","Grades",
        "Total Score","Average Grade","Attendance %","Status"])

    for student_id, student in self.students.items():
        report = self.generate_report(student_id)

        ws.append([
            student_id,
            report["Name"],
            report["Age"],
            report["Department"],
            report["Email"],
            str(report["Grades"]),
            report["Total Score"],
            report["Average Grade"],
            report["Attendance %"],
            str(report["Status"]) ])

    wb.save(filename)
    print(f"Report saved to {filename}")
```

- This function save_reports_to_excel() is used to create excel file with header in row

- openpyxl is a Python library used to create, edit and save Excel (.xlsx) files.

- This code initializes a new Excel workbook in memory.
- Each new workbook comes with one default sheet.
- Using `wb.active`, you can access that sheet to begin entering data.
- We use' ws.title' to rename the sheet to "Reports" for greater clarity.
- Excel sheets have designated names (default: "Sheet").
- The method `ws.append()` adds a complete row to the sheet.
  - Each item in the list corresponds to a column, such as Student ID, Name, Age, Department, Email, Grades, Total Score, Average Grade, Attendance %, and Status.
- The `w.append()` method writes one row into Excel.
- Iterates over each student stored in the system dictionary.
- The command `wb.save(filename)` saves the Excel workbook to your computer.
  - If `filename = "student_reports.xlsx"`, it creates (or overwrites) that file in the same directory as your program.
- The str we see while reporting the grade and status is necessary because we cannot save data in a dictionary as Excel data; thus, we need to convert it into string format. Finally, we confirm whether the file is saved with the intended name or not.

```python
if __name__ == "_main_":
    system = SchoolManagementSystem()
    num_courses = int(input("How many courses do you want to add? "))
    for i in range(num_courses):
        course_name = input("Course name: ")
        course = Course(course_name)
        system.add_course(course)
    print("Enter student (type 'Q' for name to quit) : \n")
    while True:
        name = input("Name : ")
        if name == "Q":
            break
        age = int(input("Age : "))
        student_id = int(input("Student_id : "))
        email = f"{student_id}@std.neu.edu.tr"
        print("Email:" , {email})
        department = input("department : ")
        s = Student(name , age , student_id , email , department)
        system.add_student(s)
        for student in system.students:
            system.enroll_student(student_id, course_name)
        print("\n    Mark Attendance   ")
        for student in system.students.values():
            while True:
                status = input(f"Attendance for {student.name}  (present/absent): ")
                student.mark_attendance(status)
                more_attendance =input("Add another attendance for this student? (yes / no): ")
                if more_attendance == "no":
                    break
        print("\n    Enter Grade   ")
        for course_name in system.courses:
            grade = int(input(f"{course_name} grade for {student.name}: "))
            student.add_grade(course_name, grade)
        print("\n   Student Report   ")
        for student_id in system.students:
            print(system.generate_report(student_id))
        system.save_reports_to_excel("student_reports.xlsx")
```

This part of the program begins with the statement if __name__ == "__main__":, which ensures the code runs only when the file is executed directly. The system first creates a SchoolManagementSystem object and asks the user how many courses they want to add. After entering the course names, the program moves on to collecting student information such as name, age, ID, email, and department. Each student is added to the system and enrolled in the available courses. The program then allows the user to record attendance for every student and enter their grades for each course. Once all the information is entered, the system generates a detailed report for each student, showing their grades, total score, average grade, attendance percentage, and pass/fail status. Finally, all reports are saved into an Excel file named student_reports.xlsx.

THANK YOU FOR LİSTENİNG TO US