CS4520/5520: Mobile Application Development

Weekly Assignment 1.2. Swift Basics

In this assignment, we will be building a rudimentary Student Management System in Swift. The concepts it will cover are: Protocols, Structs, Arrays, Loops, Closures, and Sorting.

Requirements:

Part 1: Protocol and Struct

a. Define a protocol called StudentInfoProtocol with the following components:
    i. A computed property `displayInfo` of type `String`
    ii. A method `updateGrade(subject: String, grade: Double)`
b. Create a struct called `Student` that adopts the `StudentInfoProtocol` protocol with these properties:
    i. `name:String`
    ii. `studentID:Int`
    iii. `grades:` Dictionary where keys are String (subject names) and values are Double (grades)
    iv. `year:Int` (1-4 representing freshman to senior)
c. Implement the protocol requirements:
    i. `displayInfo:` should return a formatted string with student details
    ii. `updateGrade:` should add or update a grade for the given subject

Part 2: Student Management Class

a. Create a class called `StudentManager` with:
    i. An array, `students` of type `[Student]`
    ii. An initializer that takes an array of students
b. Add the following methods to `StudentManager:`
    i. `addStudent(_ student: Student)`
        1. Add a new student to the array

ii. `displayAllStudents()`
    1. Use a loop to iterate through all students
    2. Print each student's `displayInfo`
iii. `findStudentsByYear(_ year: Int) -> [Student]`
    1. Use a loop to find all students in a specific year
    2. Return an array of matching students
iv. `sortStudentsByName() -> [Student]`
    1. Use the `sorted(by:)` method with a **closure** to sort students alphabetically by name
    2. Return the sorted array
v. `sortStudentsByAverage() -> [Student]`
    1. Create a helper method to calculate the average grade for a student
    2. Use sorted(by:) with a closure to sort students by their average grade (highest to lowest)
    3. Return the sorted array
vi. `getTopPerformers(count: Int) -> [Student]`
    1. Use your sorting method and array slicing to get the top N performers
    2. Return an array of the top students

Part 3: Your Tasks

a. A list of 5 sample students is provided with the skeleton code. Add as many more as you want to create the initial data set. Each student will have their:
    i. Name
    ii. Student ID
    iii. Year (mix of 1-4)
    iv. Grades in various subjects (at least 3 subjects per student)
b. Initialize a `StudentManager` object with the sample data.
c. Demonstrate the following operations:
    i. Display all students
    ii. Add a new student
    iii. Find students by year (pick any year)

iv. Show students sorted by name
v. Show students sorted by average grade
vi. Display top 3 performers

d. You have to write a (main) function called `runStudentManager()` to run and complete the tasks. You will write the control code here. This function must include the following function calls from the student manager. A few outputs with examples would be:
   i. `manager.displayAllStudents()` which outputs something like:

      === All Students ===
      Student: Alice Johnson (ID: 1001, Year: 2)
      Subjects: ["Science: 88.0", "English: 92.0", "Math: 95.0"]
      Average: 91.67

      Student: Bob Smith (ID: 1002, Year: 1)
      Subjects: ["English: 80.0", "Science: 85.0", "Math: 78.0"]
      Average: 81.00

      Student: Charlie Brown (ID: 1003, Year: 3)
      Subjects: ["English: 89.0", "History: 94.0", "Science: 96.0", "Math: 92.0"]
      Average: 92.75

      Student: Diana Prince (ID: 1004, Year: 2)
      Subjects: ["Science: 91.0", "Math: 87.0", "English: 95.0"]
      Average: 91.00

      Student: Edward Wilson (ID: 1005, Year: 4)
      Subjects: ["English: 78.0", "Math: 76.0", "History: 80.0", "Science: 82.0"]
      Average: 79.00

ii. `manager.addStudent()` which outputs something like:

=== Adding New Student ===

Added student: Frank Miller

iii. `manager.findStudentsByYear(2)` which outputs something like:

=== Students in Year 2 ===

Student: Alice Johnson (ID: 1001, Year: 2)

Subjects: Science: 88.0, English: 92.0, Math: 95.0

Average: 91.67

Student: Diana Prince (ID: 1004, Year: 2)

Subjects: Math: 87.0, Science: 91.0, English: 95.0

Average: 91.00

iv. `manager.sortStudentsByName()` which outputs something like:

=== Students Sorted by Name ===

Alice Johnson - Average: 91.67

Bob Smith - Average: 81.00

Charlie Brown - Average: 92.75

Diana Prince - Average: 91.00

Edward Wilson - Average: 79.00

Frank Miller - Average: 90.33

v. `manager.sortStudentsByAverage()` which outputs something like:

=== Students Sorted by Average Grade ===

Charlie Brown - Average: 92.75

Alice Johnson - Average: 91.67

Diana Prince - Average: 91.00

Frank Miller - Average: 90.33

Bob Smith - Average: 81.00

Edward Wilson - Average: 79.00

vi. `manager.getTopPerformers(count: 3)` which outputs something like:

=== Top 3 Performers ===

1. Charlie Brown - Average: 92.75
2. Alice Johnson - Average: 91.67
3. Diana Prince - Average: 91.00

e. **Bonus:**
   i. **Can you give the students letter grades?**
   ii. **Can you list the students by year (1-4)?**
   iii. **Can you find a student by their ID and update their grade?**