Max Morella

Assignment 1A (Pseudocode)

See Java implementation and live demo at repl.it/@MaxMorella/Assignment-1-Tests-and-TestManager

TestsPseudocode.txt

```
    // A data type to store a student's test scores
CLASS Tests
    BEGIN
    CREATE firstName // string: The student's first name
    CREATE lastName // string:  The student's last name
    CREATE scores // int[]: The student's test scores

    CONSTRUCTOR Tests(param: newFirstName, newLastName, newScores)
        firstName = newFirstName
        lastName = newLastName
        scores = newScores.clone()
    END CONSTRUCTOR

    // Return the student's first name
    METHOD getFirstName()
        RETURN firstName
    END METHOD

    // Return the student's last name
    METHOD getLastName()
        RETURN lastName
    END METHOD

    // Combine the student's first and last name
    METHOD getFullName()
        RETURN firstName + " " + lastName
    END METHOD

    // Return the score at a given index
    METHOD getScore(param: index)
        IF (0 < index < scores.length)
            RETURN scores[index]

        ELSE
            // Index out of bounds
            RETURN -1
    END METHOD

    // Set the score at a given index to a value (0, 150)
    METHOD setScore(param: index, newScore)
        IF (0 < index < scores.length)
            IF (0 < newScore)
                scores[index] = newScore
    END METHOD
```

```
    END METHOD

    // Returns the student's test scores as a simple list (ex. "98 47 85 79 82")
    METHOD getScoreList()
        CREATE list = ""
        FOREACH score IN scores
            list += score
        END FOREACH
        RETURN list
    END METHOD

    // Averages the student's test scores
    METHOD getAverageScore()
        CREATE sum = 0
        CREATE length = scores.length
        FOREACH score IN scores
            sum += score
        END FOREACH
        RETURN sum / length
    END METHOD

    // Return the letter grade corresponding to the student's average
    METHOD getLetterGrade()
        CREATE average = CALL getAverageScore()
        IF (average < 60)
            RETURN "F"
        IF (60 <= average < 70)
            RETURN "D"
        IF (70 <= average < 80)
            RETURN "C"
        IF (80 <= average < 90)
            RETURN "B"
        IF (90 <= average)
            RETURN "A"
    END METHOD

    // Return a simple string with all the relevant data
    // ex. "NAME: Stephen Strange - SCORES: [98 47 85 79 82] - AVG: 78.0 - GRADE: C"
    METHOD toString()
        RETURN "NAME: " + getFullName() + ", "
               + "SCORES: " + getScoreList() + ", "
               + "AVG: " + getAverage() + ", "
               + "GRADE: " + getLetterGrade()
    END METHOD
END CLASS
```

TestManagerPseudocode.txt

```
    // A program to handle and display a class of students and their test scores
CLASS TestManager
BEGIN
    CREATE students // Tests[]: All the students in the class

    // Generate with pre-defined students array
    CONSTRUCTOR TestManager(param newStudents)
        students = newStudents.clone()
    END CONSTRUCTOR

    // Generate students array via user input
    CONSTRUCTOR TestManager(param: length) {
        CREATE students = NEW Tests[length]
        FOR each index between 0 and length
            PRINT "Student #" + (i + 1)
            students[i] = CALL inputTests()
            PRINT students[i].toString()
        END FOR
    END CONSTRUCTOR

    // Helper method: Generate Tests object via user input
    METHOD inputTests()
        PRINT "First name: " // "First name: John⏎"
        CREATE firstName = READ string
        PRINT "Last name: " // "Last name: Smith⏎"
        CREATE lastName = READ string
        CREATE scores = NEW integer array with length 5
        FOR indexes between 0 and 5
            // "Enter score #1: 99⏎"
            PRINT "Enter score #" + (i + 1)
            scores[i] = READ integer
        END FOR

        RETURN NEW Tests(firstName, lastName, scores)
    END METHOD

    // Average all the students' scores
    METHOD getClassAverage()
        CREATE sum = 0
        CREATE length = students.length
        FOREACH student IN students
            sum += student.getAverageScore()
        END FOR
        RETURN sum / length
    END METHOD

   // Print a table of all the relevant class data
    METHOD displayTable() {
        PRINTLN tableHelper("Name", "Scores", "AVG", "Grade")
        PRINTLN "-------------------------" // Table divider
        FOREACH student IN students) {
            PRINT tableHelper(
                student.getFullName(), // Name
                student.getScoreList(), // Scores
                student.getAverage(), // Average
```

```
                student.getLetterGrade() // Grade
            )
        END FOREACH
        PRINT "Class Average: " + getClassAverage()
    END METHOD

    // Helper method: Create a formatted table row with fixed width
    METHOD tableHelper(param: name, scores, avg, grade)
        RETURN String.format("%-25s %-20s %-5s %-5s", name, scores, avg, grade)
    END METHOD

    // Demonstrate functionality using example data
    MAIN
    BEGIN
        CREATE exampleData = NEW Tests[]{
                NEW Tests("Jack", "Johnson", {85, 83, 77, 91, 76}),
                NEW Tests("Lisa", "Aniston", {80, 90, 95, 93, 48}),
                NEW Tests("Andy", "Cooper", {78, 81, 11, 90, 73}),
                NEW Tests("Ravi", "Gupta", {92, 83, 30, 69, 87}),
                NEW Tests("Bonny", "Blair", {23, 45, 96, 38, 59}),
                NEW Tests("Danny", "Clark", {60, 85, 45, 39, 67}),
                NEW Tests("Samantha", "Kennedy",{77, 31, 52, 74, 83}),
                NEW Tests("Robin", "Bronson", NEW int[]{93, 94, 89, 77, 97}),
                NEW Tests("Sun", "Xie", NEW int[]{79, 85, 28, 93, 82}),
                NEW Tests("Kiran", "Patel", NEW int[]{85, 72, 49, 75, 63}),
            }
        CREATE exampleManager = NEW TestManager(exampleData)
        CALL example.displayTable()
    END MAIN
END CLASS
```