```
MAIN
BEGIN
    CREATE students ← NEW Tests[10]
    PRINT "Enter data for students..."
    FOR i ← 0 to students.Length
        students[i] ← NEW Tests()
        PRINTLINE " Student # " + (i + 1)
        PRINT "  first name: "
        firstName ← READ
        PRINT "  last name: "
        lastName ← READ
        testScores ← NEW float[students[i].GetTestScores().Length]
        FOR a ← 0 to students[i].GetTestScores().Length
            PRINT "  test #" + (a + 1) + ": "
            testScores[a] ← READ
        ENDFOR
        students[i] ← NEW Tests(firstName, lastName, testScores)
    ENDFOR
    PRINT "First name: Last Name: Test 1: Test 2: Test 3: Test 4: Test 5:
Average: Letter Grade: "
    total ← 0
    FOREACH student in students
        student.CalculateAverage()
        student.DetermineLetter()
        PRINTLINE student.GetData()
        total ← total + student.GetAverage()
    END FOREACH
    PRINTLINE
    PRINTLINE "The Class Average = " + (total / students.Length)
END MAIN

CLASS Tests
BEGIN
    firstName ← "John"
    lastName ← "Smith"
    testScores ← NEW float[5]
    average ← 0
    letterGrade ← 'F'

    CONSTRUCTOR Tests ()
    END CONSTRUCTOR

    CONSTRUCTOR Tests (parameter: newFirstName, newLastName,
newTestScores)
        firstName ← newFirstName
        lastName ← newLastName
        testScores ← newTestScores
    END CONSTRUCTOR

    METHOD SetFirstName(parameter: newFirstName) => firstName ←
newFirstName

    METHOD SetLastName(parameter: newLastName) => lastName ← newLastName
    METHOD SetTestScores(parameter: newTestScores) => testScores ←
newTestScores

    METHOD SetAverage(parameter: newAverage) => average ← newAverage

    METHOD SetLetterGrade(parameter: newLetterGrade) => letterGrade ←
newLetterGrade

    METHOD SetIndividualTest(parameter: elementNumber, testScore) =>
testScores[elementNumber] ← testScore

    METHOD GetFirstName() => RETURN firstName

    METHOD GetLastName() => RETURN lastName

    METHOD GetTestScores() => RETURN testScores

    METHOD GetAverage() => RETURN average

    METHOD GetLetterGrade() => RETURN letterGrade

    METHOD GetIndividualTest(parameter: elementNumber) => RETURN
testScores[elementNumber]

    METHOD CalculateAverage()
    BEGIN
        sum ← 0
        FOREACH score in testScores
            sum ← sum + score
        END FOREACH
        average ← sum / testScores.Length
    END METHOD

    METHOD DetermineLetter()
    BEGIN
        IF (average >= 90) THEN
            letterGrade ← 'A'
        ELSE IF (average >= 80) THEN
            letterGrade ← 'B'
        ELSE IF (average >= 70) THEN
            letterGrade ← 'C'
        ELSE IF (average >= 60) THEN
            letterGrade ← 'D'
        ELSE
            letterGrade ← 'F'
        ENDIF
    END METHOD

    METHOD GetData()
    BEGIN
        formattedScores ← ""
        FOREACH score in testScores
            formattedScores ← formattedScores + " " + score
        END FOREACH
        RETURN firstName + " " + lastName + formattedScores + " " +
average + " " + letterGrade
    END METHOD
END CLASS
```