Rochester Institute of Technology
National Technical Institute for the Deaf
Information and Computing Studies Department

Name: Edward Riley

NACA.161 Programming Fundamentals II
In-class Exercise #10 – Arrays

Overview

This exercise is designed to give you some more experience using arrays.
All input must be done using the Scanner class.

Grades Class

1) Create a class called **Grades.**

2) Declare an array of doubles named **arrGrades** as an attribute of the class. Do **NOT** create the array [do not use the new operator] since the size of the array will be specified in the constructor. Do **NOT** create an attribute to keep the size of the array as you will use the read-only attribute **length** of an array.

3) Create a one-parameter constructor that creates the array to the size specified by the value of the integer parameter. After the array is created, the constructor needs to set all the elements in the array to the value 10.5.

4) Create a method called **getSize** that returns the number of elements in the array.

5) Create a method called **getGrade** that accepts one parameter. Use this parameter as the index of an element of the array and return the value of that array element. For example, if the value of the parameter is 2, this method returns the value of the element 2.

TestGrades Class

6) Write a class called **TestGrades** that contains a main method.

7) Prompt the user to enter the number elements in the array. Use this value in the constructor when you create the object.

8) Write a loop that displays each value of the array in the **Grades** class. Call the **getGrade** method to get the value of each element of the array. If the size of the array is 5, what are the valid index values for the **getGrade** method?

Answer 6

9) Compile and run the **TestGrades** class. Use a value of `5` when prompted for the size of the array. The 5 printed values should all be 10.5. If not, fix your code before continuing.

10) Run the **TestGrades** class again, but enter a value of `-5` for the size of the array. What exception occurs when the size of the array is `-5`. Be precise.

    Out of Range

11) Modify the code in the main method to have a loop to require that the value for the size of the array be greater than or equal to 1 as the constructor can not return any value to indicate if the size is valid or invalid. If the input value is not greater than or equal to 1, print an error message and prompt the user to enter the value again.

Back to the Grades Class

12) Create a method called **setGrade** that sets a grade. This method has two parameters. The first parameter is the index of an element of the array and the second parameter is the value of the array element.

    For example `setGrade(2, 63.45)` would set element 2 to the value 63.45. If the index is a valid, set the grade and return a value of **true**; otherwise do not change any grade and return a value of **false**.

13) Compile the **Grades** class. If the class does not compile, fix the errors.

Back to the TestGrades Class

14) Add code that will prompt the user for the array index and the grade to set in that element of the array. Use these 2 values to call the **setGrade** method. If the **setGrade** method indicates that the index is invalid, display a message.

15) Copy the loop from an earlier step to display the contents of the array after the one value has been changed.

16) Test your program until it works properly.

Back to the Grades Class

17) We will be using the Grades class to hold the grades that a class of student earned on a test. Write a method called **avgGrade** to compute the average grade for the students in a class. To compute the average, sum all of the grades and divide by the number of students [size of array].

18) Write a method called **maxGrade**, which returns the largest value in the grade array.

19) Write a method called **minGrade**, which returns the lowest value in the grade array.

20) Write a method called **numBGrades** which returns the number of "B" grades that is, grades greater than or equal to 80 and less than 90.

Back to the TestGrades Class

21) Remove the code to request one grade and add code to request the user to enter a grade for each student in a class. For example, if the class has 3 students [size of the array], the code would prompt the user as follows:

```
Enter grade for student 1: 77
Enter grade for student 2: 88
Enter grade for student 3: 99
```

In the above example, the user has entered the grades 77, 88, and 99. With a properly designed loop, you should never have an invalid index.

Note that the student numbers are 1, 2, 3 but the element numbers in the array are 0, 1, 2.

22) Add code to call the method to count the number of 'B' grades and print it; call the method to find the maximum grade and print it; call the method to find the minimum grade and print it; and call the method to compute the average grade and print it.

23) Test your program with the following 3 grades: 70, 80, and 90.

Test your program with the following 4 grades: 60, 70, 90 and 100.

Test your program again with the following 10 grades:
100, 76, 94, 89, 100, 68, 80, 83, 91, and 65.

When you complete all of the steps successfully and answer all of the questions, contact your instructor to check if your program(s) executes correctly and to review your code. I will initial the line below.

________ Successful execution of code

If you do not finish the program during the class period, contact your instructor or teaching assistant to initial below so that you can complete it before the next class period.

________ Code not completed during lab time

You may then have a tutor or your instructor verify your work at the <u>start</u> of work period in the next class. If you do not have a signature, then you can not receive any points for this assignment.