**CMSC203**

**Lab 6 - JUnit Lab**

You will be creating a JUnit Test Class for Gradebook.java, (listing 1.1) and you can download it from blackboard.

Gradebook has two attributes: an array of int called scores to hold scores and scoreSize that indicates how many scores are currently held in the array. This field is initially set to 0.

**Task #1:**

1. Add a getScoreSize() method to the Gradebook class which returns scoresSize;
2. Add a toString() method to the Gradebook class that returns a string with each score in scores field of the Gradebook separated by a space.

.

**Task #2:** Create the Test Class GradebookTester by right clicking on the GradeBook.java, select New, Junit Test Case.

Using the wizard:

1. Select the setUp and tearDown method check boxes and click Next.
2. Select all of the methods of Gradebook, except for the constructor to create tests for. Then click finish.

.

**Task #3:**

1. In the setUp method of GradebookTester, create at least two objects of Gradebook to hold 5 scores. Then call the addScore method for each of the Gradebook objects at least twice (but no more than 5 times) to add some random scores of your choice to the GradeBook objects
2. In the teardown method of GradebookTester, set the two objects of Gradebook to null.

**Task #4:** Create test for the methods of Gradebook:

* 1. addScore
     + Use the toString method to compare the contents of what is in the scores array vs. what is expected to be in the scores array assertTrue( . . .)
     + Compare the scoreSize to the expected number of scores entered, using assertEquals(. . .)
  2. sum
* Compare what is returned by sum() to the expected sum of the scores entered.
  1. minimum
* Compare what is returned by minimum() to the expected minimum of the scores entered.
  1. finalScore
* Compare what is returned by finalScore() to the expected finalScore of the scores entered. The finalScore is the sum of the scores, with the lowest score dropped if there are at least two scores, or 0 if there are no scores.

**Task #5:** Upload both files to GitHub in a directory named CMSC203\_Lab6.

Example:

As a private attribute (member) of GradeBookTest:

GradeBook g1;

In setup:

g1 = new GradeBook(5);

g1.addScore(50);

g1.addScore(75);

In teardown:

g1 = null;

in testSum():

assertEquals(125, g1.sum(), .0001);

in testMinimum():

assertEquals(50, g1.minimum(), .001);

in addScoreTest();

assertTrue(g1.toString().equals(“50.0 75.0 ”);

**Listing 1.1 – GradeBook.java**

import java.util.ArrayList;

public class GradeBook

{

private double[] scores;

private int scoresSize;

/\*\*

Constructs a gradebook with no scores and a given capacity.

@capacity the maximum number of scores in this gradebook

\*/

public GradeBook(int capacity)

{

scores = new double[capacity];

scoresSize = 0;

}

/\*\*

Adds a score to this gradebook.

@param score the score to add

@return true if the score was added, false if the gradebook is full

\*/

public boolean addScore(double score)

{

if (scoresSize < scores.length)

{

scores[scoresSize] = score;

scoresSize++;

return true;

}

else

return false;

}

/\*\*

Computes the sum of the scores in this gradebook.

@return the sum of the scores

\*/

public double sum()

{

double total = 0;

for (int i = 0; i < scoresSize; i++)

{

total = total + scores[i];

}

return total;

}

/\*\*

Gets the minimum score in this gradebook.

@return the minimum score, or 0 if there are no scores.

\*/

public double minimum()

{

if (scoresSize == 0) return 0;

double smallest = scores[0];

for (int i = 1; i < scoresSize; i++)

{

if (scores[i] < smallest)

{

smallest = scores[i];

}

}

return smallest;

}

/\*\*

Gets the final score for this gradebook.

@return the sum of the scores, with the lowest score dropped if

there are at least two scores, or 0 if there are no scores.

\*/

public double finalScore()

{

if (scoresSize == 0)

return 0;

else if (scoresSize == 1)

return scores[0];

else

return sum() - minimum();

}

}