**Calculator Subsystem**

**Group 2:**

John Abueg, William Rios Crespo, Joshua Kerley, Michael Lancaster

CMSC 495 63802

**Author Note:**

Document Version GS003

Date: 2018-10-04

**Version Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document** | **Date** | **Action** | **Name** | **Email** |
| GS001 | 2018-09-14 | Created | John Abueg | jkillakerlz@gmail.com |
| GS002 | 2018-09-26 | Modified | John Abueg | william.rioscrespo19@gmail.com |
| GS003 | 2018-10-04 | Modified | William Rios Crespo | william.rioscrespo19@gmail.com |

import java.util.Collections;

import java.util.ArrayList;

class Calculator {

//find and return smallest grade in array

public static double minimum(ArrayList<Double> grades){

double min = 100;

for (int i = 0; i < grades.size() - 1; i++){

if (grades.get(i) < min) {min = grades.get(i);}

}

return min;

}

//find and return largest grade in array

public static double maximum(ArrayList<Double> grades){

double max = 0;

for (int i = 0; i < grades.size() - 1; i++){

if (grades.get(i) > max) {max = grades.get(i);}

}

return max;

}

//find unweighted median average

public static double median(ArrayList<Double> grades){

ArrayList<Double> medianList = new ArrayList<>(grades);

Collections.sort(medianList); //sort medianList

//determine if medianList size is even or odd, then find median

if (medianList.size() % 2 == 0){ //even

int y = medianList.size() - 1;

int z = medianList.size();

return ((medianList.get(y) + medianList.get(z)) / 2);

}

else { //odd

int y = (medianList.size() - 1) / 2;

return medianList.get(y);

}

}

//find weighted mean average

public static double mean(ArrayList<Double> grades, ArrayList<Double> weight){

double total = 0;

double weightTotal = 0;

for (int i = 0; i < grades.size(); i++){

total += (grades.get(i) \* weight.get(i));

weightTotal += weight.get(i);

}

return (total / weightTotal);

}

//find standard deviation

public static double stdDev(ArrayList<Double> grades, ArrayList<Double> weight){

double stDevTotal = 0;

double meanAverage;

//holds all grades for standard deviation calculations

ArrayList<Double> devList = new ArrayList<>(grades);

meanAverage = mean(grades, weight);

//for each value in devList, subtract mean and square result

//then replace value with result

for (int i = 0; i < devList.size(); i++){

double tempResult = Math.pow(devList.get(i) - meanAverage, 2);

devList.set(i, tempResult);

}

//find mean of squared differences, then return square root of mean

double stDevMean = mean(devList, weight);

return Math.sqrt(stDevMean);

}

//find "what-if" grade and weight

public static double[] whatIf(ArrayList<Double> grades,

ArrayList<Double> weight, int whatIfValue) {

ArrayList<Double> gradeList = new ArrayList<>(grades);

ArrayList<Double> weightList = new ArrayList<>(weight);

double meanAverage;

double newMeanAverage;

double targetAverage;

double tempGrade;

int numAddedValues = 0; //counts number (weight) of added grades

meanAverage = mean(gradeList, weightList); //get actual mean avg

//if/else in case target score is lower, e.g. "What can I score on the

//final and still pass?"

if (meanAverage < whatIfValue) {

do { //repeats until "what if" is met/exceeded

gradeList.add(100.0);

weightList.add(1.0);

numAddedValues += 1;

newMeanAverage = mean(gradeList, weightList); //get new mean avg

} while (newMeanAverage < whatIfValue);

//loop to "fine tune" last grade submitted

tempGrade = 100;

double tempMeanAverage; //holds mean value for do-while loop

do { //repeats while "what if" is lower than mean avg

tempGrade -= 1;

gradeList.set(gradeList.size() - 1, tempGrade);

tempMeanAverage = mean(gradeList, weightList);

} while (tempMeanAverage > whatIfValue);

}

else {

do { //repeats until "what if" is met/exceeded

gradeList.add(0.0);

weightList.add(1.0);

numAddedValues += 1;

newMeanAverage = mean(gradeList, weightList); //get new mean avg

} while (newMeanAverage > whatIfValue);

//loop to "fine tune" last grade submitted

tempGrade = 0;

double tempMeanAverage; //holds mean value for do-while loop

do { //repeats while "what if" is higher than mean avg

tempGrade += 1;

gradeList.set(gradeList.size() - 1, tempGrade);

tempMeanAverage = mean(gradeList, weightList);

} while (tempMeanAverage < whatIfValue);

}

//Here we take the total of all added values to gradeList

//and add them together. If more than one value was added all

//values EXCEPT for the last value are either 0 or 100 depending on

//if the target was higher or lower. The last value is the

//tempGrade value. Total is then divided by numAddedValues to get

//the score and weight needed to achieve "what if" grade.

if (meanAverage < whatIfValue) {

tempGrade = tempGrade + ((numAddedValues - 1) \* 100);

}

//if target was lower than mean, only the tempGrade is needed

targetAverage = tempGrade / numAddedValues;

//place values in array

double[] whatIfNumbers = new double[2];

whatIfNumbers[0] = targetAverage; //first array value set to score

whatIfNumbers[1] = numAddedValues; //second array value set to weight

return whatIfNumbers; //use array to get "what-if" numbers

}

}

public class Driver {

public static void main(String[] args) {

//using test case 1 to check, adding what-if at the end

ArrayList<Double> gList = new ArrayList<>();

ArrayList<Double> wList = new ArrayList<>();

gList.add(100.0);

gList.add(80.0);

gList.add(80.0);

gList.add(70.0);

gList.add(95.0);

wList.add(20.0);

wList.add(20.0);

wList.add(20.0);

wList.add(20.0);

wList.add(20.0);

System.out.println("Min: " + Calculator.minimum(gList));

System.out.println("Max: " + Calculator.maximum(gList));

System.out.println("Median: " + Calculator.median(gList));

System.out.println("Mean: " + Calculator.mean(gList,wList));

System.out.println("Standard Deviation: " + Calculator.stdDev(gList, wList));

//values should be 90 entered as test “what-if” score, expected results are 100, 50

double[] results = Calculator.whatIf(gList, wList, 90);

System.out.println("What-if target score: " + results[0]);

System.out.println("What-if target weight: " + results[1]);

}

}