Name: _____

First, you must implement programming project P3.4 on page 127 as Version1 of this project, due by the date/time listed on Canvas.

You must then meet additional requirements for Version2 due one week after Version 1. Details for Version 2 are in these instructions.

For both versions, **you must**:

- **follow all style guidelines in Appendix I** and do **NOT** create a java package or project (do it like P1)
- provide essentially the same set of **both electronic and physical deliverables** as required for the previous project (See below).

Version1:

P3.4 Implement a **class Student**. For the purpose of this exercise, a student has a **name** and a **total quiz score**. Supply an appropriate **constructor** and methods **getName(), addQuiz(int score), getTotalScore(),** and **getAverageScore().** To compute the average, you also need to store the **number of quizzes** that the student took. Supply a **StudentTester class** that **tests all methods**. Your Student class must have at least the listed:

1. three instance variables,
2. constructor and
3. five methods
4. **StudentTester class must thoroughly test all methods**

Version2 (enhances version1 whith more data and methods):

It is often a good idea for teachers to analyze student grades for trends, including Standard Deviation(*a measure of how much data values in a collection deviate from the average value*).

StudentTester:

This class' *main* method should instantiate an object of the Student class and an object of the Scanner class. It must do at least the following:

1. instantiate an object of the Student class name **firstStudent** (this student will demonstrate "good input")
2. instantiate an object of the Scanner class and use it to get **all** input
3. Collect 5 quiz scores
4. Print all possible output/statistics for data collected so far (with expected values) for **firstStudent**
5. Collect 5 more quiz scores
6. Print all possible output/statistics for data collected so far (with expected values) for **firstStudent**
7. instantiate an object of the Student class name **secondStudent** (this student will demonstrate "bad input")
8. demonstrate that your code properly manages good and a wide variety of bad input data
9. Print all possible output/statistics for data collected so far (with expected values) for **secondStudent**
10. instantiate an object of the Student class name **thirdStudent** (this student will demonstrate "bad input")
11. prove that your code properly manages a wide variety of valid but extreme input to show that **all** your Student class methods behave properly, (for example with all zero quiz scores or only one score).
12. Print all possible output/statistics for data collected as needed (with expected values) for **thirdStudent**. When statistics/output is needed, calculate and produce output showing the statistics for **all of the data** read in for the object at the current point in the program.
13. Since the inputData method can throw an exception, (non-numeric input), it should be encased in a try block, and have an accompanying catch block for the possible IllegalArgumentException. **See section 11.4**. It must print out a message noting that the data is out of range / being ignored, and let the program continue

Student class:

This class must have at least the following instance variables:

1. ***numberOfElements,***
2. ***highValue,***
3. ***lowValue,***
4. ***scoresEntered*** - a String holding all valid scores entered so far (See printReceipt method of E3.6 CashRegister class)
5. ***sumOfData,***
6. ***mean*** - *Create a method to calculate this average(You could recalculate it after each score is or when the calculate method is called. Which would be better? Why? Does the class need this instance variable? Why?)*
7. separate variables to count each of ***As, Bs, Cs, Ds and Fs. scored (see ranges below)***

This class should have at least the following methods:

1. **a default constructor**, you may create other constructors with parameters (such as name, for example)
2. ***public boolean inputData***
   - *2.1. takes in a Scanner object passed in from main()*
   - *2.2. Reads from the user a number in the range of 0 through 100.*
   - *2.3. It must read a number intoan integer variable unless a non-number value is entered, in which case the method will return false.*
   - *2.4. If the datum/score is accepted, the addGrade method must be called.*
   - *2.5. If addGrade returns false, the inputData method should throw an IllegalArgumentException. Otherwise, inputData should then return true. See section 11.4*
3. ***public boolean addGrade***
   - ***3.1. takes in an integer grade. If grade is not in the range of 0 through 100 the method should return false. Otherwise, it should appropriately change the values of numberOfElements, highValue, lowValue,***
   - ***3.2. sumOfData, sumOfSquares, and add one to the count of the proper variable for A's, B's, C's,***
   - ***3.3. D's or F's. The method should then call calculate and return true***
4. ***private void calculate***
   calculates *mean, variance*
   - *4.1. mean = sumOfData / numberOfElements*
5. ***public String toString (see*** @see String.format() p.A-27, fomat specifiers p.524-5 and p.148-150)
   must return a multi-line String containing the results in tabular form showing at least the following:
   - ***5.1. the total number of elements***
   - ***5.2. the summation of values***
   - ***5.3. the range of values (the lowest and highest)***
   - ***5.4. the mean value (to 3 decimal places)***
   - ***5.5. followed by the number of A's (90 - 100), B's (80 - 89), C's (70 - 79), D's (60 - 69) and F's (0 59).***
   - ***5.6. The table should have fixed column widths***

note: if the total number of data is 0, all other values should be printed as "-"

1. You must submit all these Deliverables. See Canvas for due date.

Physical: All Pages must be stapled in order (print on both sides if possible to save paper).

1. All pages of these instructions (*printed from the web*), with your **name clearly printed** on it, as a cover sheet.
2. Printed Source Code with Comments (*including all javadoc style comments, you must **fix line wrapping***)
3. Sample Input and Output screen captures (*printed*)
4. a simple **test plan** including explanations of any discrepancies and reasons for each test.
   - 4.1. Show actual input and include all of the outputs (expected & actual) in the test plan
   - 4.2. **ALL** values output as well as **ALL** expected output.
   - 4.3. Remember to test an <Enter> for expected inputs, a non-numeric character and just **one** numeric input.
   - 4.4. test multiple ***sets*** of data in one run.

Electronic:

5. All .java, .class, .html (javadocs), input and output and image files, zipped **together in one file**. Do not use rar or any format other than zip. Name the file: "<YourName>_P2.zip".
6. Submit this single ***zip*** file on **Canvas**

Extra Credit **(for a maximum of 5% of your initial earned score):**

Demonstrate and note in your class javadoc heading that you successfully enhanced version2 and fully tested the following instance variables, new statistics and private methods for your student class. You must produce all new statistics in your calculations and your toString printout.:

1. *sumOfSquares,*
2. *variance (to 3 decimal places)*
3. *average not including the lowest score*
4. *standardDeviation (to 3 decimal places)*
5. *the variance (to 3 decimal places), and*
   - 5.1. *variance and standard deviation* can be calculated by these formulae:
   - 5.2. *sumOfTheSquares = sum of each quiz squared (SumOfSquare = SumOfSquare + score * score)*
   - 5.3. *variance = (sumOfTheSquares - (sumOfData2 / numberOfElements)) / (numberOfElements - 1)*
   - 5.4. *standardDeviation = the square root of the variance.*
   - 5.5. *if the number of data is 1, the variance and standard deviation should be set to 0 (do not divide by 0)*
6. ***You must also submit expected results calculated with excel to earn any extra credit***