ECE 325 Lab Assignment 5: Computing the salaries of band members

(10 points total) Your band is finally earning some money! Not a lot, but you feel it is time to automate the computation of the salaries of the band members. You started to implement a static method to compute the salary (see Salary.java), but the other band members complained that it is very buggy. One even implied that it was not tested at all (how dare they). In this assignment, you need to improve the implementation of the pay() method, and, given the importance of the correctness of this method, provide a test suite for the method to make sure that you are not incorrectly paying the band members.

The method has three parameters:

- salary: The base salary earned by a band member during a salary period (in dollars).
- snacksAmount: The total amount spent by a band member on snacks during tours and shows during a salary period (in dollars).
- bonus: The bonus percentage that the member earned this salary period (in percentage).

The pay is computed by deducting the amount spent on snacks from the base salary, and then computing and adding the bonus (if any) over that amount. So, if a band member earned 100 dollars, spent 50 on snacks, and earned a bonus of 10 (%), their pay will be 55 dollars.

The pay parameters have the following properties (aside from 'common sense' properties, which you should figure out yourself):

- The base salary can be maximum $1,000.
- The bonus can be between 0 and 10 (%).
- A band member cannot spend more on snacks than their base salary.
- Because of a strange compatibility issue with your accountant's software, the salary and snacksAmount are both Double objects, and the bonus is specified as an Integer object. The pay method must return a Double object. You cannot change this as your accountant will get mad.

For this assignment, you need to:
1) Improve the pay() method given the above requirements.
2) Create a test plan to test the pay() method. The test plan consists of a table with the parameter values and the expected output of the method. Also, please indicate in which test method you cover the case.
3) Implement your test plan as JUnit test class TestSalary. Make sure to give your test cases descriptive names, to make it easier to debug your code if a test fails.

Please do not modify any of the names/methods we've defined in the provided *.java files. You will get 0 points for this assignment if you do.

Hints:

- Because you're doing computations with Doubles (i.e., you're doing floating point arithmetic), there will be some precision errors. JUnit allows you to specify a delta to verify whether the result is within a certain error range (https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html#assertEquals-double-double-double-). For this assignment, you can set delta to 0.001.

- For invalid results, your pay() method should throw an IllegalArgumentException. Make sure to include a message that explains why the exception is thrown (you can decide exactly which message).
- You are not allowed to change the signature of the pay() method. If you do, you will get 0 points for this assignment.

Rubric

(20 points total)

(16 points) Correct implementation of pay() method. Your pay() method will be tested using our extensive JUnit test suite.

(4 points) Test plan

If you do not submit a test plan and/or test suite, or they do not make sense or do not match your implementation, we can deduct more points.

Please submit:

1) A zip file containing your improved Salary class, your test suite and a PDF with your test plan.
Name the file 'FirstName_ID_asg5.zip' and keep the exact same file structure as the zip that was provided for the assignment. You can store the test file in the same location as the Salary class. For example,
Filename: Cor-Paul_1234567_asg5.zip

```
|------- testplan.pdf
|------- src
|        |------- ece325
|        |        |------- lab
|        |        |        |------- assignment5
|        |        |        |        |------- Salary.java
|        |        |        |        |------- TestSalary.java
```

2) A screencast/movie that shows the following steps:

- Open your Canvas with your name shown
- Open your IDE
- Show your code briefly
- Execute your test suite and show that all tests pass.

Please do not modify any of the names/methods we've defined in the provided *.java files.