CPSC 233 – Team Assignment 7 – Requirements

As a team, decide which tests to use from individual team members as a starting point for a JUnit test for CreditHistory. Then add tests for one additional method in CreditHistory named getCreditRating which does not take any arguments and returns a double.

A credit rating for a credit history is computed by taking a weighted average of all the ratings in the history. The most recent rating should have the highest weight and the oldest rating should have the lowest weight. When moving from oldest rating to newest rating, the increment in weight should be the same throughout and the average weight should be 1. The following table shows an example. The ratings at the left of the table are the oldest ratings.

Rating	4	3	0	-1	-4	5	5	3	3
Weight of rating	.2	.4	.6	.8	1	1.2	1.4	1.6	1.8

Note that the weight increments by .2 (starting from 0) for each rating. The weighted average that should be returned by getRatings in this case is (.2*4 + .4*3 + .6*0 + .8*-1 + 1*-4 + 1.2*5 + 1.4*5 + 1.6*3 + 1.8*3)/9 = 20.4/9 = 2.267.

Another example with just three ratings 4, -1 and 3 would result in weight .5, 1, and 1.5 with resulting weighted average: (.5*4 + 1*-1 + 1.5*3)/3 = 1.833.

Remember: when comparing floating point numbers we generally don't require that the numbers are exactly the same and we allow some rounding error. The amount of rounding error allowed should be the last argument in the assert statement in your test. (Eg: assertEquals("message", expectedValue, actualValue, 0.0001) would allow a different less than or equal to 0.0001.)

Submission

Choose one team member that will submit the team solution(s) to WebCAT. Note: that there is no D2L dropbox for this assignment! Only one team member should submit. If multiple team members submit, the first team member encountered when grading will be the one used for grading.

Place your team number at the top of your test class (in the documentation).

Rubric

	1 mark	.5 mark	0 marks
Coverage	How thorough are your tests? WebCAT determines the number of marks (2.3 maximum)		
Legibility	Each test method is clearly named and represents a single test case. Code in test is clearly organized into: test setup, test execution, verification of results.	Names of test methods are clear but may test multiple cases or code is not clearly organized.	Name of test methods don't give any information about tests.
Documentation	Documentation contains team number. Test sections are clearly documented (test setup, execution, verification)	Some documentation provided that helps understand organization of code.	Team number not included in documentation.