CSE542 - Software Engineering Concepts
Testing Submission Requirements

Each team's submission must include:

- **Java source code** for the GUI program the team wrote. This program should start by **asking the user** for the **number of team members** and **if there were scores previously** entered by this student. The number of team members will be between 2 and 7; your group should makeup names and previously entered scores. The program will then **display a screen** in which the user can **enter/update their peer- and self-evaluation** scores. Once **all the scores are filled in**, the user should be **allowed to "submit"** their scores. The system should **normalize these scores** and, in a final screen, **display the normalized score** for each teammate.

- **JUnit test cases** verifying that your normalization code works correctly. Each distinct test should be defined in its own method. A test should be written so it could detect a specific bug and so may contain 1 or more `assert___` statements. The name of the method should clearly state the situation it is testing. Since it is impossible to prove that the code is correct using unit tests, each group should define a sufficient number of tests to either convince others their code works or make clear where the bug exists.

CSE542 - Software Engineering Concepts
Grade Sheet – Testing
GUI Application

Initial Functionality

Excellent (20-18)	Acceptable (17-10)		Bad (0)
Asks user for number of students on team **-AND-** Asks users if there were scores entered previously **-AND-** Correctly initializes peer- and self-evaluation window using user's answers	Asks user for number of students on team **-AND-** Asks users if there were scores entered previously		Does not ask user for number of students on team **-OR-** Does not ark user if there were scores entered previously **-OR-** Code to do this contains 1 or more bugs

Peer- and Self-Evaluation

Excellent (20-18)	Acceptable (17-10)		Bad (0)
Displays names and scores for all members of the team **-AND-** Allow users to update scores **-AND-** Prevents the user from entering invalid scores or displays an error when an invalid score is entered **-AND-** Allows the user to "submit" scores and displays the normalized results in another window once the scores are submitted	Displays names and scores for all members of the team **-AND-** Allow users to update scores **-AND-** Allows the user to "submit" scores and displays the normalized results when this is done		Does not displays names and scores for all members of the team **-OR-** Does not allow users to update scores **-OR-** Does not include an option to "submit" scores **-OR-** Does not display the normalized results when this is done

Comments

Excellent (5-4)	Acceptable (3-2)	Poor (1)	Bad (0)
All classes and methods documented **-AND-** Comments describe *what* the class/method and not *how* it does it **-AND-** Methods document parameters and return values	All classes and methods documented **-AND-** Methods document parameters and return values	All classes and methods documented	1 or more classes or methods lack comments

JUnit Test Cases

Test Cases

Excellent (20-18)	Acceptable (17-10)	Poor (9-6)	Bad (5-0)
Each method defines a test that would uncover a bug **-AND-** Each method's name specifies the bug it is trying to find **-AND-** There are not multiple tests looking for the same bug **-AND-** Fully tests the method(s) normalizing scores	Each method's name specifies the bug it is trying to find **-AND-** There are not multiple tests looking for the same bug **-AND-** Fully tests the method(s) normalizing scores	Each method's name specifies the bug it is trying to find **-AND-** Fully tests the method(s) normalizing scores	Does not fully test the method(s) normalizing scores

Testing

Excellent (25)			Bad (0)
All test cases pass			1 or more test cases fail

Professionalism

Excellent (10)	Acceptable (9 - 8)	Poor (7 - 6)	Bad (5 - 0)
GUI application is easy to use **-AND-** Code is organized in a way that groups classes by functionality	GUI application is easy to use **-AND-** Most of the code is organized in a way that groups classes by functionality	GUI application requires minimal explanation before using it **-AND-** Most of the code is organized in a way that groups classes by functionality	GUI application requires substantial explanation in order to use it **-OR-** Most of the code is not organized by functionality

Total Score (sum of individual scores) _____