

# Assignment T5: Second Iteration

✓ Published

 Edit

⋮

This is a team assignment. This assignment will be graded as 0 to max points. Scroll down for submission instructions at the end.

Complete the implementation and release your MVP. This should expand and harden your MVP sufficiently to be usable. Also take into consideration any feedback from your IA mentor based on your first iteration.

part 1:

Revise your user stories, from the first iteration and/or project proposal, to reflect what is now fully implemented, tested and working. **Do not include anything that is not actually evident in your github repository and/or you will not be able to show in your final demo.**

part 2

Write a test plan that explains the **equivalence partitions** and **boundary conditions** necessary to unit-test each of the major subroutines in your system (methods or functions, excluding constructors, getters/setters, helpers, etc.) and then implement your plan. Associate the names of your specific test case(s) with the corresponding equivalence partitions and boundaries (if applicable). Your test suite should include test cases from both **valid** and **invalid** equivalence partitions, and just below, at, and just above each equivalence class boundary (or inside vs. outside the equivalence class when boundary analysis does not apply). Note the same test case might apply to multiple equivalence classes. Say there is a method whose input should be an integer between 1 and 12. Then there is an equivalence class 1-12, an equivalence class  $<1$  (or  $\leq 0$ ), and an equivalence class  $>12$  (or  $\geq 13$ ). A test case with input 0 would be just below the lower boundary of the equivalence class 1-12 and also at the high boundary of the equivalence class  $<1$ .

Include the link to the folder in your github repository that contains your automated test suite.

part 3:

Measure the **branch coverage** achieved by your automated test suite. This requires using a coverage tool appropriate for your programming language and platform. Branch coverage should strive to achieve 100%, but may not reach 100%. Add more test cases until you reach at least 90%. Each additional test case should try to force a particular branch that was not previously covered. Tell us what branch coverage you finally did achieve. If the coverage tool reports less than 100% (that is, between 90% and 99%), discuss one example of a branch that your test cases did not cover and explain why it is difficult to test this branch. If you were unable to reach 90%, explain why not.

Include the link to the folder in your repository that contains means you need to configure your coverage tool to produce your repository.

part 4:

As part of the release effort, you need to institute *continuous* idea is to integrate automated build and test with your version similar tool, so that build and test is automatically initiated with main branch of your github repository. Make sure to ask for deadline if you have trouble getting CI to work.

Include links to the files in your repository that configure the that includes the CI reports. Note this means you need to that can be saved as files in your repository.

You can discuss this assignment in this course's team\_project (<https://courseworks2.columbia.edu/courses/104335/external> forums, on stackoverflow, etc. However, do not ask for anyone developing your project; this is a violation of the university's

**Submission instructions:** One member of your team should submit a single file (preferably pdf) with the required information for parts 1-4. Your file should contain your team name and the names/unis of all team members. You may submit repeatedly until the deadline.


choose which version of Java in Eclipse

SC

**Sang Jun Chun** 9/15/20  
10:44 AM  
Eclipse -> preferences -> java -> installed JREs

MG

**Minxuan Gao** 9/15/20  
10:45 AM  
Maven is a build automation tool



Send

**Points** 40

**Submitting** a file upload

**File Types** pdf, doc, docx, txt, xls, and xlsx

Due	For	Available from	Until
Dec 6, 2020	Everyone	Nov 17, 2020 at 12:01am	Dec 13, 2020 at 11:59pm

## Second Iteration Rubric

Criteria	Ratings		Pts
<p>Part 1. User stories</p> <p>Compare the user stories given in the second iteration report to those given in the first iteration report (part 2). If there are any differences you did not expect, you should discuss the discrepancies with the team. There is no grade for this part, except do not grade the rest of the report if the user stories are missing.</p>	<p><b>0 pts</b> <b>Full</b> <b>Marks</b></p>	<p><b>0 pts</b> <b>No</b> <b>Marks</b></p>	<p>0 pts</p>
<p>Part 2. Equivalence Partions &amp; Boundary Conditions (20 points max)</p> <p>8 points max for reporting equivalence classes. Check whether the written report defines at least one valid equivalence class and at least one invalid equivalence class for each major method (not counting constructors, etc.). Take off 1/2 point for each major method that does not have at least one valid equivalence class, up to 4 points (8 methods), and take off 1/2 point for each major method that does not have at least one invalid equivalence class, up to 4 points (8 methods). If there are 9 or more major methods missing defined equivalence classes, the score for this part should be 0, not negative.</p> <p>8 points max for the test cases (in the codebase) corresponding to valid vs. invalid equivalence classes. Take off 1/2 point for each major method that does not have at least one test case that provides inputs for a valid equivalence class, up to 4 points (8 methods), and take off 1/2 point for each major method that does not have at least one test case that provides input for an invalid equivalence class, up to 4 points (8 methods). If there are 9 or more major methods missing test cases corresponding to equivalence classes, the score for this part should be 0, not negative.</p> <p>If the project has less than 8 major methods, ask the instructor to look at it.</p> <p>4 points max for boundary analysis. 2 points max for explaining, in the written report, appropriate boundaries for at least one equivalence class for some major method. 2 points max for the test cases that check exactly at the boundary and immediately outside the boundary for at least one equivalence class for some major method. If there are no major methods with equivalence classes that actually have boundaries (as opposed to just inside vs. outside the class), give up to 4 points for a meaningful discussion about not having any equivalence classes with boundaries.</p>	<p><b>20 to &gt;0.0 pts</b> <b>Full Marks</b></p>	<p><b>0 pts</b> <b>No</b> <b>Marks</b></p>	<p>20 pts</p>
<p>Part 3. Branch Coverage (10 points max)</p> <p>10 points for the branch percentage covered, computed by dividing the percentage by 10. That is, 100% is 10 points, 90% is 9 points, 82.5% is 8.25 points, 10% is 1 point. If the coverage was between 90-99%, add back up to 1 point for a good technical explanation of why it was hard to test the given example branch (do not go over 10 max).</p> <p>If the coverage was less than 90%, add back up to 1 point for a good technical</p>	<p><b>10 to &gt;0.0 pts</b> <b>Full Marks</b></p>	<p><b>0 pts</b> <b>No</b> <b>Marks</b></p>	<p>10 pts</p>

Criteria	Ratings		Pts
<p>explanation of why it was hard to achieve 90%. Running out of time is not a good technical explanation.</p> <p>If it seems truly impossible (to the grader) to reach at least 80%, ask the instructor to look at it.</p>			
<p>Part 4. Continuous Integration (10 points max)</p> <p>5 points for CI configuration (e.g., .travis.yml file), 0 points if not found.</p> <p>5 points for a folder in the github report containing CI reports, or any other mechanism allowing you (the grader) to access the CI reports, that demonstrates the CI indeed works and is being used. This should be 0 if CI does not appear to be working.</p>	<p><b>10 to &gt;0.0 pts</b></p> <p><b>Full Marks</b></p>	<p><b>0 pts</b></p> <p><b>No Marks</b></p>	<p>10 pts</p>
Total Points: 40			