Assignment 1 - Fall 2021
CSCI 5408 - Advance Topics in Software Development

Assignment by:

Full Name: Adesh Nalpet Adimurthy

Banner Number: B00886154

Dal FCS CSID: adimurthy

Gitlab Profile: https://git.cs.dal.ca/adimurthy

Assignment Gitlab Repository (A1):

https://git.cs.dal.ca/courses/2021-fall/csci-5308/assignments/adesh_n/-/tree/master/A1

Steps followed:

Note: The header of every step is linked to the respective commit on Gitlab.

1. **Setup:** I started with a Java - Maven project, using the JUnit and Surefire plugin for test results (uploaded to gitlab). Since the repository adesh_n is a multi-project repository, the CI pipeline is specific to a project, and the global .gitlab-ci.yml has the "include" keyword to point the sub-project's path.
2. **Step 1:** Created "Calculator.java" and a "CalculatorTest.java" classes, with a test case to verify that the class "Calculator" exists, achieved using a ClassLoader.
3. **Step 2:** Added a new method, "add" - takes two Integer arguments and returns an Integer result, which is supposed to be the sum of both the arguments. However, there was no logical implementation of the add method at this stage; instead, it was just a contract. I followed test-driven development approach first to write the failing test cases for adding two positive and negative Integers, followed by the correct implementation of add method, and verified the test cases.

   Note: I have used the wrapper class "Integer" instead of primitive type "int" to abide by the principles of OOPs.
4. **Step 3:** Similarly, I repeated Step 2 for other mathematical operations: subtract, multiply and divide.
5. **Step 4:** Added a new method, fizzbuzz, that takes an Integer and returns a List of Strings. For a given integer N, 1 to N is considered as the input and returns "fizz" if divisible by 3, "buzz" if divisible by 5, and "fizzbuzz" if divisible by 3 and 5 (by 15) for every integer from 1 to N. Once

again, TDD approach was followed first to write test cases, followed by the implementation. In this case, I had to fix the implementation to have divisibility by 15 as the first condition, determined from the failing test cases.

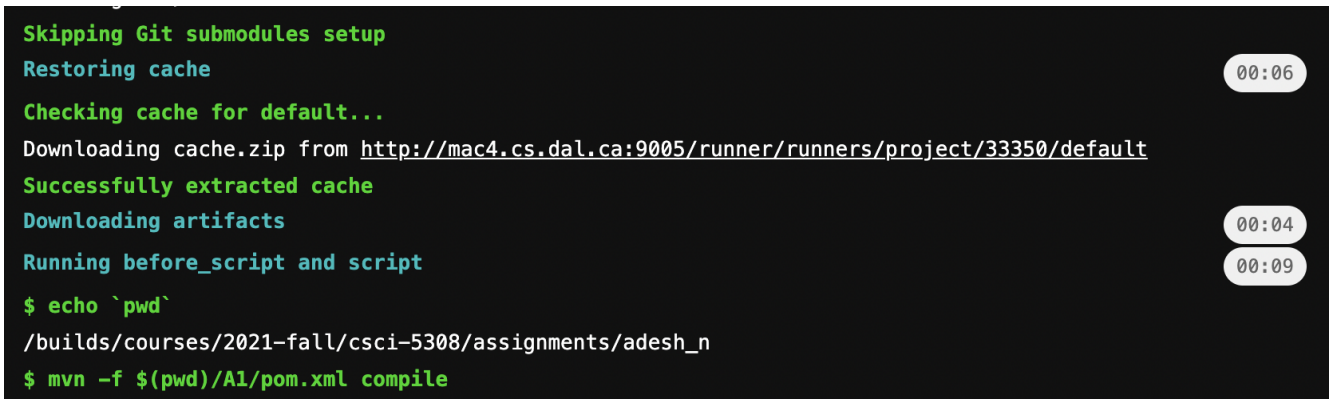
6. **Steps 5 and 6:** Similar to the "add" method previously written, a new method takes two Strings and returns a String sum result and further checks if the given string does not have more than two decimal places. With the TDD approach, the test cases covered several other scenarios for validations and threw an IllegalArgumentException when there were more than two decimal places. To ensure reusability, during the implementation, I created a private method to specify the scale and decimal format of the result for rounding and verified the test scenarios; furthermore, the methods are add overloaded (Static Polymorphism).
7. **Steps 7 and 8:** Repeated step 6 for subtract, multiply, and divide. However, the divide method has more test cases to check if the second argument is zero and throw an exception for zero. Test-Driven-Development was followed for all the methods, including the various test cases (both arguments positive, negative, one positive, and other negative and validation tests), then verified post-logical implementation. Since the validation for the number of decimals is common across operations, I added a new private method within the Calculator class to prevent code-duplication.
8. **Step 9:** Similar to step 1, a new "Validation" class was created. A test case to verify the class exists was added before and verified again after class creation.
9. **Steps 10 and 11:** Added a new method, "validate," in the Validation class that takes an array of strings (vargs) and returns a list of String, which is a list of validation messages. Following TDD, the first set of test cases was to check the size of the result list when the input array has all elements more than two decimals, none have more than two decimals, and a combination of both. After that, the failing tests were re-verified to check if they passed after logical implementation of the validate method.
10. **Steps 12 and 13:** Now that validation for the scale (number of digits after the decimal point) is in the validation class, a new set of methods with the same contract as the previously overloaded methods add, subtract, multiply and divide are re-written (addRevised, subtractRevised, multiplyRevised and divideRevised), but this time without using the validator private method in the calculator class, instead an instance of Validation class is passed to the Calculator constructor to use the validate method, thereby improving reusability and reducing code-duplication. As usual, the approach here is TDD, with the same tests as before but for the new methods. This way, we can affirm that the functionality of the methods is not affected, and the code quality is improved.

11. **Improvements:** Interfaces are added to both Calculator and Validation classes to better readability and ensure the implementation is decoupled.
12. **Additional Test Cases:** Additional test cases have been added for inputs such as alphanumeric strings, numbers without prefix and postfix from the decimal point, numbers with the improper format, different precision/scale for both Validation and Calculator classes. Example Commits: Commit 1 and Commit 2

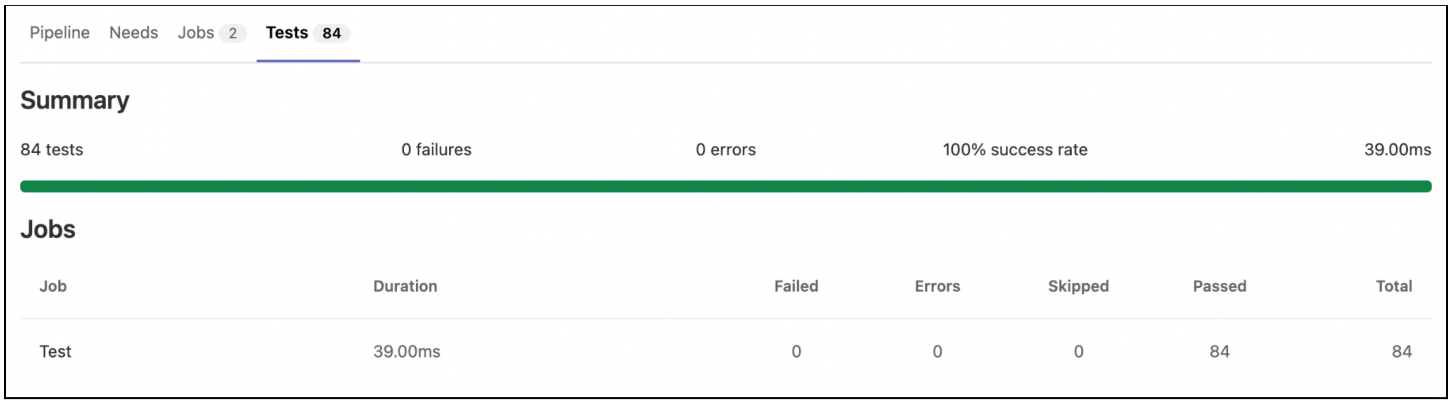
Note:

1. The runner: "dalfcs_gitlab_docker_ci" stopped working from October 5th afternoon. Hence the .gitlab-cli.yml is updated to use "ugrad", which is the reason for a lot of canceled and failed jobs here.
2. The "master" branch is protected to ensure direct mergers are not done and mandatory to raise a merge request.

Faster Build: Downloading all the dependencies every time increases the build time. Hence, the m2 directory is cached.



Test Report: Sample test report for a total of 84 test cases.



Quicks Links:

1. Calculator.java, CalculatorImpl.java
2. Validation.java, ValidationImpl.java
3. CalculatorTest.java
4. ValidationTest.java

References:

1. FizzBuzz Wiki.
2. Maven JUnit Documentation.
3. Surefire Plugin Documentation.
4. Gitlab CI Maven project example by Gitlab.