UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

GROUP MEMBERS:

KATUKUNDA ROCHELLE - A94169

MABIRA CONRAD - A94170

MUKISA ISAIAH - A94160

COURSE:

SOFRWARE CONSTRUCTION

LECTURER:

MR. LUBAMBO SIMON

<u>Test-Driven Development Report</u>

<u>Red Phase:</u>

During the Red Phase, our primary focus was on defining the behaviour of the Calculator class through unit tests without implementing any code. We identified the functionalities required for the Calculator class: addition, subtraction, multiplication, and division. Each test case was written to cover these functionalities based on the specified requirements. Challenges faced during this phase included ensuring that the test cases were comprehensive and covered edge cases, such as division by zero. The rationale behind this phase was to establish a clear understanding of the expected behaviour of the Calculator class before writing any implementation code.

<u>Green Phase:</u>

In the Green Phase, we implemented the Calculator class methods to satisfy the failing test cases written during the Red Phase. The focus was on writing the simplest code necessary to make the tests pass. Challenges encountered during this phase were primarily related to handling edge cases, such as division by zero, and ensuring that the implemented methods produced correct results. The rationale behind this phase was to iteratively build and verify the functionality of the Calculator class while adhering to the principles of TDD.

<u>Refactor Phase:</u>

During the Refactor Phase, we reviewed the implemented code to identify areas for improvement and optimization while ensuring that all test cases continued to pass. Although the initial implementation was straightforward, we paid attention to code readability, efficiency, and adherence to best practices. Refactoring decisions were made to enhance code clarity, remove redundancy, and improve maintainability. Challenges in this phase included balancing the need for refactoring with the risk of introducing new bugs. The rationale behind refactoring was to ensure that the codebase remained clean, understandable, and scalable, facilitating future enhancements or modifications.

Overall, the Test-Driven Development approach allowed us to incrementally develop the Calculator class while maintaining a focus on code quality and correctness. It provided a structured framework for software development, enabling us to iteratively refine the codebase and deliver a robust solution that met the specified requirements.