NTD Coding Challenge

Author: Caio Costa

1. Intro

This test plan guides through the quality perspective to ensure that the new Calculator application meets the high and expected quality standards before its public release. Although initial unit tests suggest the calculator functions correctly, comprehensive testing is necessary to uncover any hidden bugs and ensure robustness and scalability.

2. Scope

The plan covers the functional and usability testing for the Calculator. It includes tests for all arithmetic operations available in the application and examines the application behavior under different scenarios, like edge cases and invalid inputs.

3. Objectives

As stated previously, the main goal is to assure the app quality and stability before its public release. Steps taken to achieve this goal are:

- ➔ Identify and properly document any bugs or issues in the app;
- ➔ For bugs found, provide additional information that might be helpful for the team to identify the root cause as soon as possible;
- ➔ Ensure that the default arithmetic operations returns correct values for specified parameters;
- ➔ Check for edge cases that might break the application;

4. Test Strategy

Both manual and automated testing will be applied, that way in future releases we can assure that the main functionalities are still working.

- **Manual Testing**
    - Perform exploratory testing to identify possible failures and bugs, even on edge cases;
    - Execute E2E testing to validate that known functionalities are behaving correctly;
- **Automated Testing**
    - Development of automated test cases with Ruby, RSpec and Aruba;
    - Use of automated testing to ensure consistent results, avoiding manual efforts to validate that the main functionalities are performing as expected;

5. Test Environment

The testing will be conducted in an environment with Docker installed, as the application is packaged as a Docker image. The supported OS for testing are:

- macOS (Apple Silicon and Intel)
- Linux (x86_64 and arm64)

6. Testing Tools

Following tools are going be used during the testing phase:

- **Docker**
  - For running the application image.
- **Ruby**
  - The programming language used for writing automated tests.
- **RSpec**
  - A testing framework for Ruby, used for writing and executing tests.
- **Aruba**
  - A library used to test command-line applications.
- **Google Docs**
  - In order to document test cases and identified issues, **Docs** will take place.

Ideally, a task monitoring software like Jira, ClickUp or Trello is used to track the status of a ticket in large scale companies.

7. Resources

Following resources are required in order to assure a consistent behavior:

1. A Unix-like operating system (macOS or Linux recommended)
2. Docker installed in order to pull the app image
3. The application image (*public.ecr.aws/l4q9w4c5/loanpro-calculator-cli:latest*)

8. Test Cases

1. Add

Description	Input	Expected Result	Status
Verify addition of two positive numbers	add 3 5	8	Pass
Verify addition of a positive and a negative number	add 1 -3	-2	Pass
Verify addition of two floating-point numbers	add 1.1 -3.7	-2.6	Pass
Verify addition of two negative numbers	add -2 -1	-3	Pass

2. Subtract

Description	Input	Expected Result	Status
Verify subtraction of two positive numbers	subtract 10 5	5	Pass
Verify subtraction resulting in a negative number	subtract 10 15	-5	Pass
Verify subtraction of two floating-point numbers	subtract 2.5 1.1	1.4	Pass
Verify subtraction of two negative numbers	subtract -2 -1	-1	Pass
Verify subtraction of a positive and a negative number	subtract -2 5	7	Pass

3. Multiplication

Description	Input	Expected Result	Status
Verify multiplication of two positive numbers	multiply 5 2	10	Pass
Verify multiplication of a positive and a negative number	multiply 5 -2	-10	Pass
Verify multiplication of two floating-point numbers	multiply 0.5 0.5	0.25	Pass
Verify multiplication of two negative numbers	multiply -2 -5	10	Pass
Verify multiplication with zero	multiply 0 5	0	Pass

4. **Division**

Description	Input	Expected Result	Status
Verify division of two positive numbers	divide 10 2	5	Pass
Verify division of a positive and a negative number	divide 5 -2	-2.5	Pass
Verify division of two floating-point numbers	divide 4.4 2.2	2	Pass
Verify division by zero	divide 2 0	Error: Cannot divide by zero	Pass

5. **Edge cases**

Description	Input	Expected Result	Status
Verify addition with large numbers	add 999999999 999999999	1999999998	Pass
Verify addition with small numbers	add .00000001 2	2.00000001	Pass
Verify addition with NaN	add a 5	Invalid argument. Must be a numeric value.	Pass
Verify addition with scientific notation	add 2e2 5	205	Pass
Verify invalid operation	test 1 1	Unknown operation: test	Pass

9. Conclusion and findings

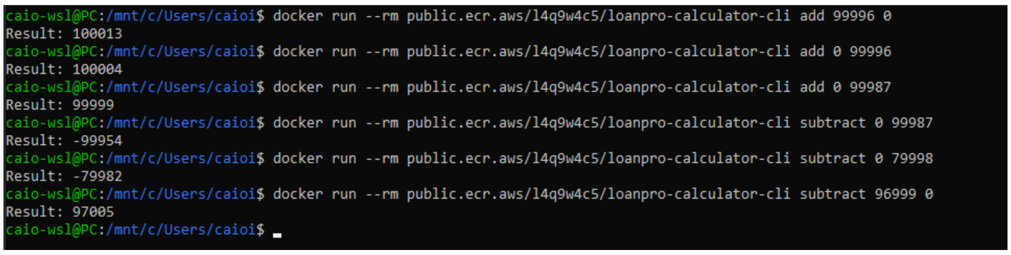
After using some reverse-engineering techniques, like getting the encoded *.jar* file from the Docker image, looking at the source code and translating it to Java (since I can't read

Clojure scripts 😅), I was able to find one bug that was implemented on purpose in the application. Feel free to ask if you want to know the step-by-step on how I got into the code.

Bug: Random values when the sum of one of the parameters is 42

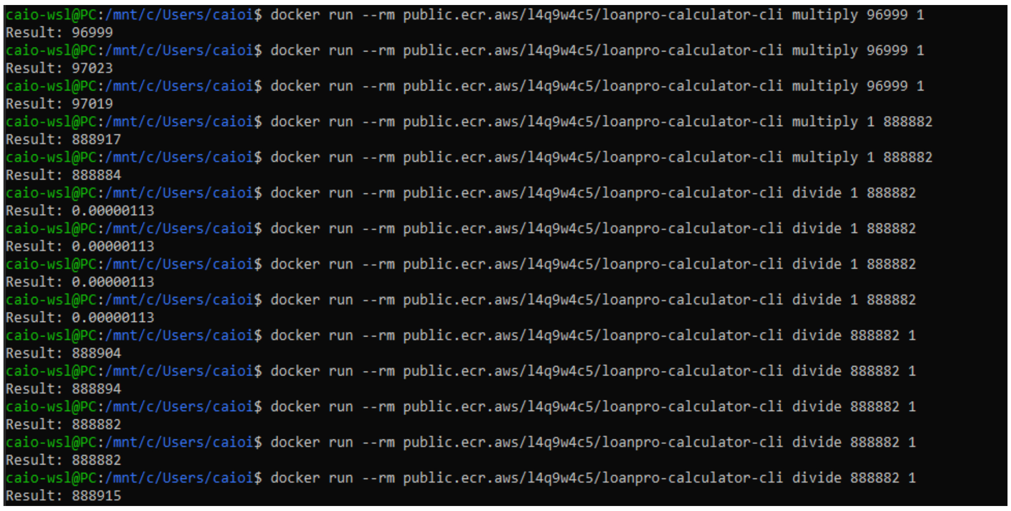
Steps to reproduce:

1. Addiction and Subtraction
    a. Total sum of one of the numbers parameters must be 42 and the other parameter must be equal to 0 (zero)
    b. Result is incorrectly displayed, returning different values when executed multiple times:



```
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli add 99996 0
Result: 100013
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli add 0 99996
Result: 100004
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli add 0 99987
Result: 99999
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli subtract 0 99987
Result: -99954
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli subtract 0 79998
Result: -79982
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli subtract 96999 0
Result: 97005
caio-wsl@PC:/mnt/c/Users/caioi$ 
```

2. Multiply and Divide
    a. Total sum of one of the numbers parameters must be 42 and the other parameter must be equal to 1
    b. Result is incorrectly displayed, returning different values when executed multiple times:



```
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 96999 1
Result: 96999
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 96999 1
Result: 97023
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 96999 1
Result: 97019
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 1 888882
Result: 888917
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli multiply 1 888882
Result: 888884
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 1 888882
Result: 0.00000113
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 1 888882
Result: 0.00000113
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 1 888882
Result: 0.00000113
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 1 888882
Result: 0.00000113
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 888882 1
Result: 888904
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 888882 1
Result: 888894
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 888882 1
Result: 888882
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 888882 1
Result: 888882
caio-wsl@PC:/mnt/c/Users/caioi$ docker run --rm public.ecr.aws/l4q9w4c5/loanpro-calculator-cli divide 888882 1
Result: 888915
```