TDDBUDDY.COM

Calculator Refactor

Rules

1. Strictly practice TDD
    a. Follow the: Red-Green-Refactor cycle
    b. Stop and read your code before refactoring.
    c. Be sure to run your test after each cycle
2. Practice wishful coding
    a. Do not write any production code until you have a failing test.
    b. Or compilation failures to drive the need to implement the method.

The Kata

You are given a calculator application. However, the solution is not perfect. All logic sits in "action scripts" behind button clicks, there are no test and it does not always handle operations correctly. E.g 1++2 equals 3 instead of "Error"

The code for this kata can be found at https://github.com/T-rav/Wpf-Calculator-Kata [1]

Your job is to write unit tests for the code, refactoring the code to create testable units not left behind button clicks.

- Keep your check-ins small by limiting each one to a single refactoring.
- Should you find any bugs, fix them as you go by creating a failing test and then making it pass; then return to refactoring.

Bonus

Avoid the use of if or switch statements in your final solution

[1] Code sources from https://code.msdn.microsoft.com/windowsdesktop/simple-calculator-d1d8cf4c