Feature #3: CPU/Assembly VM

Coders: Justin

Testers: Ci'an, Mary

Feature scope: This feature acts as a go between for the game loop, user written assembly and "car" class. This class does **not** interact with the canvas or graphics directly, instead all graphical interactions should be done with "car" class methods.

Summary: Set of classes and API interface that requires a set of methods to interact with the main game loop and render engine. This feature will provide public methods that will be called from the game loop. After being called from the main game loop, the "CPU" will "parse" a provided assembly file interrupting commands within it. These commands will then be executed by calling public methods of the "car" class.

Testing:

1. Spelling Error (Not Command)
    - Status: Tested
    - Expected Result: An error of some sort to be thrown because what's there is unrecognized
    - Actual Result: The code continues to run as if that command just wasn't there
        - How the test was ran: Manually
        - How was it tested: Spelling mistakes were put in the direction for the cars movement
2. Random Characters (Not Command)
    - Status: Tested
    - Expected Result: Because there is no resemblance to the command, with random characters, I would expect for the code to throw some sort of error
    - Actual Result: The code continues to run as if that command just wasn't there
        - How the test was ran: Manually
        - How was it tested: Random characters were inserted within the cars directional commands
3. All Caps (Not Command)
    - Status: Tested
    - Expected Result: I expect the code would work normally
    - Actual Result: The code does run normally
        - How the test was ran: Manually
        - How was it tested: Capital characters were inserted within the cars directional commands
4. Spelt Wrong (Command)
    - Status: Tested
    - Expected Result: Should not run, some sort of error should be thrown
    - Actual Result: Code run likes command wasn't there

        - How the test was ran: Manually
        - How was it tested: Commands were intentionally misspelled to see if the VM would still read them
5. All Caps (Command)
    - Status: Tested
    - Expected Result: Should not run, some sort of error should be thrown
    - Actual Result: Code run likes command wasn't there
        - How the test was ran: Manually
        - How was it tested: All commands were made fully capital, the VM still read in the commands accurately
6. no_op Input (int plus character)
    - Status: Tested
    - Expected Result: Should not run, takes an integer, if it's not an integer the code should stop working and throw an error
    - Actual Result: NumberFormatException is thrown and the code stops
        - How the test was ran: Automatically
        - How was it tested: An integer plus a character (i.e. 50a) inputted to see if the VM takes the number from the two together
7. no_op Input (large integer)
    - Status: Tested
    - Expected Result: Some sort of error to be thrown and for the code to stop
    - Actual Result: NumberFormatException is thrown and the code stops working
        - How the test was ran: Automatically
        - How was it tested: An excessively long input (i.e. 5 followed by about 40 0's) was ran to test the limit of the integers size
8. no_op Input (Non-integer)
    - Status: Tested
    - Expected Result: Some sort of error to be thrown and for the code to stop
    - Actual Result: NumberFormatException is thrown and the code stops working
        - How the test was ran: Automatically
        - How was it tested: A Non-integer (i.e. 50.0) was used to see if the VM would be able to read it in.

Design Decisions:

- After the tests were run, some parts of the VM were redesigned to better accommodate testing and fix errors. These changes will then be more robustly tested in Cycle 3. The way command parameters were loaded was changed to support an unknown length list of parameters with string data types. This list is then converted to the correct data type before the command is run. A future change will be to reimplement how the VM handles errors and make it so the default will be a CPUExpection that the code throws. This will result in error handling and checking being dealt with much easier and prevent user code from crashing the program.