CS 1632 - DELIVERABLE 6- Testing Strategy for RPN++

Siming Zheng

Github URL: https://github.com/Ningyou134679/CS1632_Deliverable6
Download URL: https://github.com/Ningyou134679/CS1632_Deliverable6.git

Review quality

Subsystem1 : repl mode
Status1 : Green
Notes1 : execution of program without an argument enters the repl mode, and exits only when use enters “quit”, case-insensitive. Line properly updates. Every prompt starts with a “> ”. All expected output generated.

Subsystem1 : read file mode
Status1 : Green
Notes1 : execution of program with more than one argument enters the read file mode, line that starts with print, case-insensitive prints out proper message, multiple files execute in order, and terminate when quit, or error, or end of file line increments throughout all files
proper exit code is executed for each error encountered

Subsystem3 : Indentation and case
Status3 : Green
Notes3 : All empty lines in the file or white spaces in any input are ignored
All inputs are considered case-insensitive

Subsystem4 : Exception and speed
Status4 : Green
Notes4 : No java exception or message ever displayed
No tellable delay from program

Areas of Concern

No defect was found. No java exception or message was found.

Some behavior of the program that may not be present in the requirement (but which should not contradict any of the requirements):

1) Encountering a file that does not exist will exit with code 5, without printing any error message.
2) Assumption that file is executed line by line, meaning that “LET” in first line, and “x 1” in second line will result in error in the first line, and program exits.
3) Assumption that the total lines of file do not exceed the capacity of int
4) readFile mode can accept any file, even if it does not end in “.rpn”
5) When “quit” is encountered in either mode, the program exits with code 0
6) Any white space, is ignored, so something like “ lEt B a 1 - ” works.
7) A blank line in the file is ignored and continue onto the next line
8) Assumption that run.setup() method is called before any other non-main methods or constructor
9) Any non-alphabetical letter, which is also not a number, like “Δ” or “%”, will result in error
10) All number in the program will be stored as BigDecimal to fulfill requirement 3 and floating point calculations
11) Floating point calculation will print out with 2 decimal places
12) any arbitrary number of digits and decimal places within the bound of BigDecimal can be store into a variable
13) Assumption that any output on the program will either be a single number, or an error message
14) read file mode only prints out a result if print is the first token of a line, or an error is encountered, where as any behavior except user entering “quit” in repl mode will print out a result. (ex. “1 1 +” in repl mode prints out “2”, but is ignored in read file mode)

Testing Strategy

Static Test: using FindBugs to analyze program, found out some useless code

Unit Testing: Created a universal method, that attempts to take in any form of single-line input from either a file or user input. The method is used in the junit test file, input all sort of strings according to the requirement, and assert to check if a proper output or error message is returned. It cannot test things like if the program terminates at appropriate time; what exit code follows; if anything is actually printed on the screen; line number; any user interaction in repl mode; if error occurs in one file and the following code and files are ignored; if an empty line in the testing file is ignored. Therefore, those are left for manual testing.

Manual Testing: Anything that I listed in unit testing, which cannot be tested with just a single number or error message as result. Mainly everything in the repl mode is manually testing, like if everyone prompting for user input starts with a “> ”, line number displayed when an error occurs but not when a proper result is produced. All exit code when in the read file mode is tested manually. Any line or file ignored after quit or error in read file are manually tested, and so on.

Performance Testing: Opens up visualVM in repl mode. All methods take up like 0% CPU time. Actually manually testing the program or input file has no signs of delay.