CISP312: Undo

In this assignment you will implement the undo feature for a simple calculator. The undo feature must be implemented using a generic or template-based Stack data structure. You may utilize language supplied libraries (e.g. java.util.stack) or a stack implementation of your own design. No third-party downloads or software installs are allowed. The undo feature must allow for an unlimited number of undo operations. If the user attempts to perform an undo operation and the stack is empty, then a message must be displayed indicating that there are no commands to undo.

The calculator portion of the assignment must function as follows.

The calculator must be implemented in a stand-alone class called 'Calculator'.

The calculator must support only the following operations: addition (+), subtraction (-), multiplication (*), and division (/).

The calculator must support whole and real numbers.

The calculator must support the following commands: UNDO, CLEAR, EXIT

UNDO: Ignores the prior calculation and restores state as if the calculation never occurred. If there are no prior calculations to ignore, then display the message "UNDO IS NOT AVAILABLE" and request the user to enter a command or calculation request.

CLEAR: Resets the calculator to the initial state, returns the Running Total to 0, and clears the undo stack.

EXIT: Terminates the program without producing any results.

If a command does not start with one of the preceding three commands, then the input is expected to be a calculation request in the following format:

The user will supply input into the calculator in the following format:

decimal-number operation decimal-number

decimal-number:
  decimal-integer-literal
  real-literal

real-literal:
  decimal-digits **.** *decimal-digits*

decimal-integer-literal:
  decimal-digits

decimal-digits:
  decimal-digit
  decimal-digits decimal-digit

decimal-digit: one of
```
0   1   2   3   4   5   6   7   8   9
```
operation: one of
```
+   -   *   /
```

Any input that does no conform to this input format should be rejected with an error message indicating that the input is invalid.

The calculator must maintain a running total of all the operations. The running total is the result of the last operation added the running total.

For example, if the calculator is presented with the following input:

10 + 5

Sum: 15; Running Total : 15

20 – 5

Difference: 15; Running Total: 30

2 * 3

Product: 6; Running Total: 36

UNDO

Running Total: 30

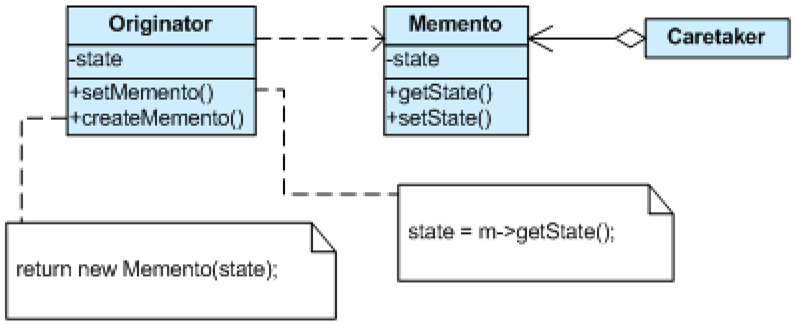
CLEAR

Running Total: 0

EXIT

The Memento Design Pattern

Memento design pattern is a behavioral design pattern and is used when the state of an object needs to be saved in case it needs to be acted upon in the future. The pattern consists of three objects: Originator, Memento, and Caretaker (defined below).



1. *Originator* - the object that knows how to save itself.
    1. Creates a memento containing a snapshot of its current internal state.
    2. Uses the memento to restore its internal state.
    3. In this program, the Calculator class is the Originator of the memento object.
2. *Caretaker* - the object that knows why and when the Originator needs to save and restore itself.
    1. Is responsible for the memento's safekeeping.
    2. Never operates on or examines the contents of a memento.
    3. In this assignment, the Caretaker is a stack data structure that will contain Memento objects provided by the Calculator class.
3. *Memento*: stores internal state of the Originator object. The memento may store as much or as little of the originator's internal state as necessary at its originator's discretion.
    1. Stores internal state of the Originator object. The memento may store as much or as little of the originator's internal state as necessary at its originator's discretion.
    2. Protect against access by objects of other than the originator.

Your solutions will be graded on three components: design, development (code), and testing. Design is weighted 40% of the total grade, development is weighted 40% of the total grade, and testing is weighted 20% of the total grade.

Upload your completed project to Blackboard as a single ZIP or 7z file. Your projects must be in one of the following formats: Visual Basic, C#, and C++ programs must be submitted as a Visual Studio .NET 2010 or 2012 project. Java projects must be submitted as an Eclipse project.