## Calculator Business Rules

## Internal State of the Calculator

The internal state of the calculator is determined by six variables:

* **display**:
  + string of characters representing what is displayed in the calculator's result field
  + initial value : '0'
* **memory**:
  + string of characters representing the content of the calculator's memory field
  + initial value : '0'

* **captureModeOn**:
  + boolean value used to define the nature of 'display'
    - *false* :
      * 'display' is a result rather than a number being entered (captured) by the user
        + eg: if the user wants to multiply 15 by 10, he presses these keys: '1', '5', '\*', '1', '0', '='
        + the display field now reads '150', and corresponds to the result of a calculation, as opposed to a number being entered by the user
    - *true* :
      * 'display' is a number currently being entered (captured) by the user
  + initial value : false
* **operator**:
  + the last binary operator ( '+', '-', '\*' or '/') used
  + reset to '+' after processing the '=' operator
  + initial value : '+'
* **firstOperand**:
  + copy of 'display' following the processing of binary operator ('+', '-', '\*' or '/')
  + reset to '0' after processing the '=' operator
  + initial value : '0'
* **lastKey**:
  + last key pressed by the user
  + initial value : '0'

## Input Processing

The input processing rules that must be applied when the user clicks on a key are the following:

* if key is a redundant operator, do nothing
* if key is not a redundant operator:
  + lastKey ← key
  + apply the rules as described in the rules table

### *Key redundancy detection*

* The objective is to determine whether the user has clicked on the same binary operator more than once, and to ignore subsequent clicks if this is the case
* Definition:
  + key is a redundant operator if key is the same as lastKey and key is a binary operator ('+', '-', '\*', '/', or '=')

### *Rules Table*

| key | Processing |
| --- | --- |
| '0' to '9' | * if captureModeOn is false: captureModeOn ← true & display ← key * if captureModeOn is true:   + if value of display is '0'     - if value of key is '0': do nothing     - if value of key is not '0': display ← key   + if value of display is not '0': display ← display + key (concatenation) |
| '.' | * if captureModeOn is true and display doesn't contain a '.' : display ← display + '.' |
| '+', '-', '\*', '/' | * if captureModeOn is true:   + captureModeOn ← false   + firstOperand ← firstOperand operator display (eg: firstOperand + display)   + display ← firstOperand   + operator ← key |
| '=' | * if captureModeOn is true:   + captureModeOn ← false   + firstOperand ← firstOperand operator display (eg: firstOperand + display)   + display ← firstOperand   + operator ← key   + firstOperand ← 0   + operator ← '+' |
| 'MS' | * memory ← display |
| 'M+' | * memory ← memory + display |
| 'M-' | * memory ← memory - display |
| 'MC' | * memory ← '0' |
| 'MR' | * captureModeOn ← false display ← memory |
| ← | * if captureModeOn is true:   + remove last character from display |
| √ | * display ← square root of display * captureModeOn ← false |
| 1/x | * display ← 1 / display * captureModeOn ← false |
| % | * display ← display / 100 * captureModeOn ← false |
| +/- | * if display is a positive number, display ← '-' + display, otherwise display ← display.substring(1) |
| C | * display ← '0', captureModeOn ← false, operator ← '+' , lastKey = '0', firstOperand = '0' |
| CE | * display ← '0', captureModeOn ← false |

### *Example*

This table presents the state of the calculator after the following clicks:

**3+22\*4=2+3=\*2=**

| key | display | captureModeOn | operator | firstOperand | memory | lastKey |
| --- | --- | --- | --- | --- | --- | --- |
|  | '0' | FALSE | '+' | '0' | '0' | '0' |
| '3' | '3' | TRUE | '+' | '0' | '0' | '3' |
| '+' | '3' | FALSE | '+' | '3' | '0' | '+' |
| '2' | '2' | TRUE | '+' | '3' | '0' | '2' |
| '2' | '22' | TRUE | '+' | '3' | '0' | '2' |
| '\*' | '25' | FALSE | '\*' | '25' | '0' | '\*' |
| '4' | '4' | TRUE | '\*' | '25' | '0' | '4' |
| '=' | '100' | FALSE | '+' | '0' | '0' | '=' |
| '2' | '2' | TRUE | '+' | '0' | '0' | '2' |
| '+' | '2' | FALSE | '+' | '2' | '0' | '+' |
| '3' | '3' | TRUE | '+' | '2' | '0' | '3' |
| '=' | '5' | FALSE | '+' | '0' | '0' | '=' |
| '\*' | '5' | FALSE | '\*' | '5' | '0' | '\*' |
| '2' | '2' | TRUE | '\*' | '5' | '0' | '2' |
| '=' | '10' | FALSE | '+' | '0' | '0' | '=' |