Simple Calculator

The goal of this application was to create a simple calculator application with some additional useful tools that a desktop calculator might not have.

Included Features

- Binary and Hex Conversion
- Calculator style formula input using regex
- Save and open formulas by object serialization

Usage

Calculator

Basic calculator usage can be acomplished either by using the on screen buttons or the correlating keys to activate the button. Any syntax or calculation error will be notified to the user.

Binary Conversion

Binary conversion will convert only the integer portion of a number to it's binary equivalent. It uses the current number in the display, which can either be typed or the result of a math operation.

```
123.45 = 1111011
32 = 100000
```

Hexadecimal Conversion

It functions similarly to the binary conversion, but will include a decimal value.

```
123.45 = 0x1.edccccccccccdp6
32 = 0x1.0p5
```

Formula Input

Formula input works much like a handheld calculator as far as input. It supports the operations addition, subtraction, multiplication and division, as well as priority based on parenthesis. It supports the correlating characters: `+ - * / ( )` as well as numbers `0-9`. White space characters will automatically be ignored. Decimal Calculations are not currently supported.

```
2+2 = 4.0
(1 + 1) * 5 - 7 = 3.0
32 / 12 + 5 = 5.5
```

Technical Information

Enum: Operation

Members

- `ADD`
- `SUB`
- `MUL`
- `DIV`

Functions

```
public abstract double math(double _one, double _two)
```

Class: FXMLDocumentController

Functions

`private void pressX(ActionEvent event)` appends number X to current number in display

`public void initialize(URL url, ResourceBundle rb)` initalizes controller and creates a keystroke listener

`private void clearPress(ActionEvent event)` clears display and resets member variables `CurrentNum`, `temp`, and `result`

`private void signPress(ActionEvent event)` changes sign of number of display by multiplying it by `-1`

`private void percentPress(ActionEvent event)` creates percentage of number in display by multiplying by `.01`

`private void decPress(ActionEvent event)` appends `.` to display

`private void multPress(ActionEvent event)` sets current operation to `Operation.MUL` and calls `opCheck()`

`private void divPress(ActionEvent event)` sets current operation to `Operation.DIV` and calls `opCheck()`

`private void addPress(ActionEvent event)` sets current operation to `Operation.ADD` and calls `opCheck()`

`private void subPress(ActionEvent event)` sets current operation to `Operation.SUB` and calls `opCheck()`

`private void equalPress(ActionEvent event)` attempts to set `result` by completing the current operation. Will catch `NumberFormatException`

`private void opCheck()` checks if there is a current operation waiting to be completed

`private void completeOp(Operation _op)` completes math operation of current operation and sets display. Will catch `NumberFormatException`

`private double getNum()` returns current number from display by parsing it double from string

`private void updateDisplay(String _num)` sets display text from string

`private void updateDisplay(Double _num)` sets display text from double

`private void numberError()` called if a calculation/parsing error occurs. Displays Error dialog and resets calculator and member values

`private void binPressed(ActionEvent event)` Converts integer portion of number in current display to binary. Displays result in a pop-up window

`private void hexPressed(ActionEvent event)`Converts number in current display and hexadecimal. Displays result in a pop-up window.

`private void formulaPressed(ActionEvent event)` loads and displays formula input window. Creates a consumer to store and calculate the formula inputted.

`private String calcFormula(String formula)` calculates formula entered by using a regex pattern (`(\(\-\d*\))|-|\*|\)|\/|\(|\+|\d*?\S`) to match, seperate, and place them in a `List<String>`. Elements of this list are then converted to Reverse Polish notation by a Shunting-yard algorithm and Placed in another `List<String>`. The list is then calculated by pushing items off and onto a stack. The final result is then stored in result and returned as a string.

`private static boolean isNumber(String str)` checks if a string is a number or not.

`private void savePressed(ActionEvent event)` opens a `JFileChooser` window to save a file of most recent formula.

`private void openPressed(ActionEvent event)` opens a file of a formula using a `JFileChooser` window and displays the result of the formula on calculator display.

Class: FormulaInputFXMLController

Functions

`void initData(Consumer<String> _onComplete)` initalizes `onComplete` member variable.

`private void calcuatePressed(ActionEvent event)` has `onComplete` accept the formula inputted and closes the popup window