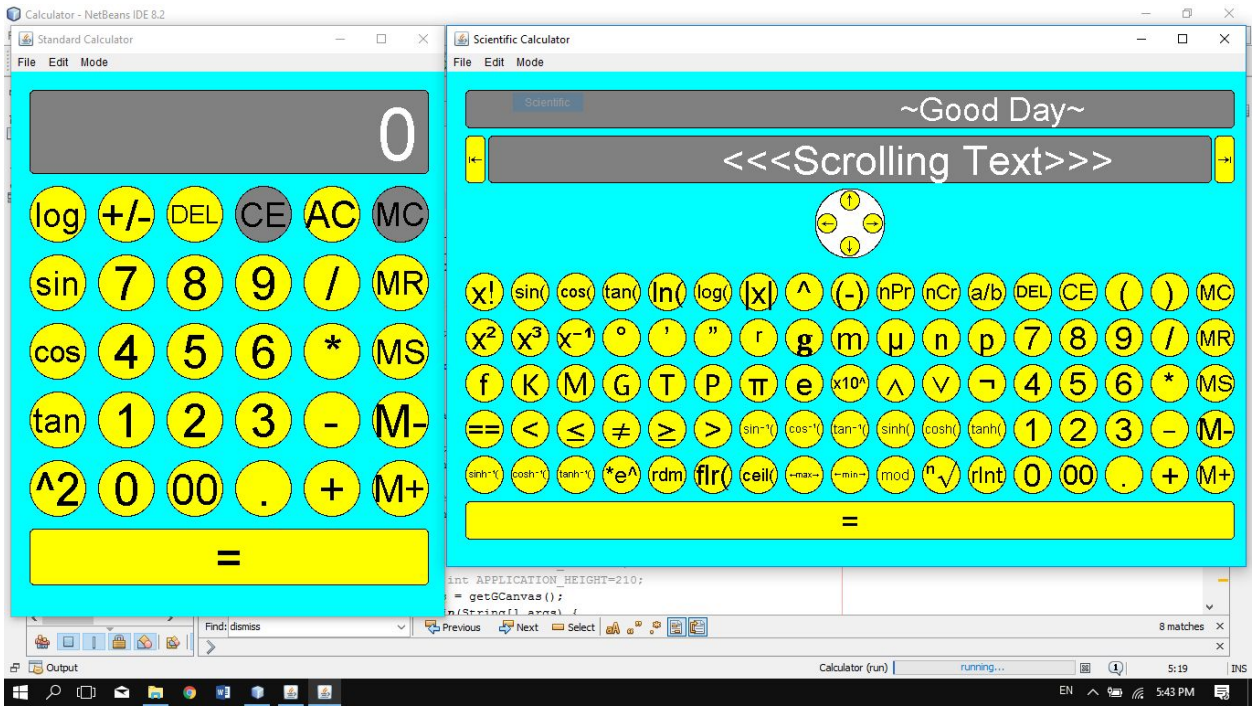
LBYCP12

Data Structures and Algorithm Analysis Laboratory



Laboratory Activity 5

A Simple Arithmetic Calculator with Stack Data Structure

By

Patrick Matthew J. Chan, LBYCP12-EQ1

INTRODUCTION

In this activity, we will develop a simple arithmetic calculator application. The application evaluates postfix arithmetic expressions containing integer numbers and the operators +, -, *, and / utilizing the Stack data structure.

In normal algebra we use the infix notation like a+b*c. The corresponding postfix notation is abc*+. The algorithm for the evaluating postfix is as follows :

Scan the Postfix string from left to right. Initialise an empty stack. If the scanned character is an operand, add it to the stack. If the scanned character is an operator, there will be at least two operands in the stack. If the scanned character is an Operator, then we store the top most element of the stack(topStack) in a variable temp. Pop the stack. Now evaluate topStack(Operator)temp. Let the result of this operation be retVal. Pop the stack and Push retVal into the stack. Repeat this step till all the characters are scanned. After all characters are scanned, we will have only one element in the stack. Return topStack.

The point of using postfix notation instead of infix, is efficiency. Unlike infix notations, postfix notations eliminate the need for parentheses, and thus the expression can simply be evaluated from left to right, instead of having the computer search the whole argument for parentheses and higher operator precedences every time an operation is done. As shown in the algorithm above, postfix simplifies the evaluation process on the expression, but then how about the conversion from the input infix notation to the postfix notation? This is the essential stage that the students must learn the basic algorithm, its possible expansions to more complex operations, and its implementation in the making of a Java calculator.

OBJECTIVES

- To create a simple Arithmetic calculator application in Java using `acm.graphics` package for the graphical user interface (GUI).
- To be able to implement a stack data structure in Java using a separate interface and implementation.
- To evaluate postfix arithmetic expressions containing integer numbers and the operators '+', '-', '*', and '/' utilizing the Stack data structure.

MATERIALS

1. Java SE Development Kit (JDK) 8
2. NetBeans integrated development environment (IDE)
3. `acm.jar` by the ACM Java Task Force

PROCEDURE

1. Utilizing `acm.jar` objects, design a simple arithmetic calculator GUI, as shown in Fig. 1:

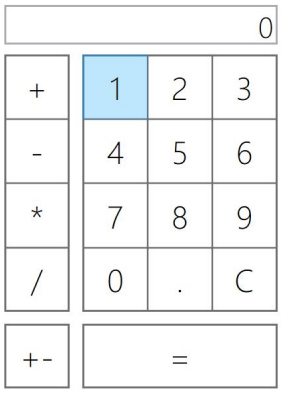


Figure 1: Arithmetic Calculator GUI

2. In general, the input must be read in as characters (because the operators are read in as characters), and converted to numbers. You can assume that the numbers are one digit numbers, i.e. (0-9). The end of the expression is marked by the expression terminator character, '='. (ADDITIONAL CREDIT: Allow real number with multiple digits and decimals, e.x. 12.44)

3. Implement a Stack data structure as shown in the following Unified Modeling Language (UML) diagram (Fig. 2). Note: separate the interface from its implementation as exemplified by previous exercises.

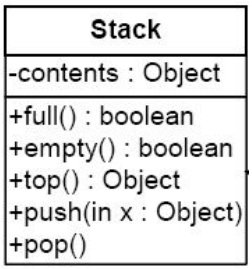


Figure 2: Stack Data Structure UML

4. Utilize a Stack to store and evaluate the input in Postfix notation. When an operator is (the characters '+', '-', '*', and '/') is encountered, we apply it to the previous two operands (which have been read in, or are results of previous computation).

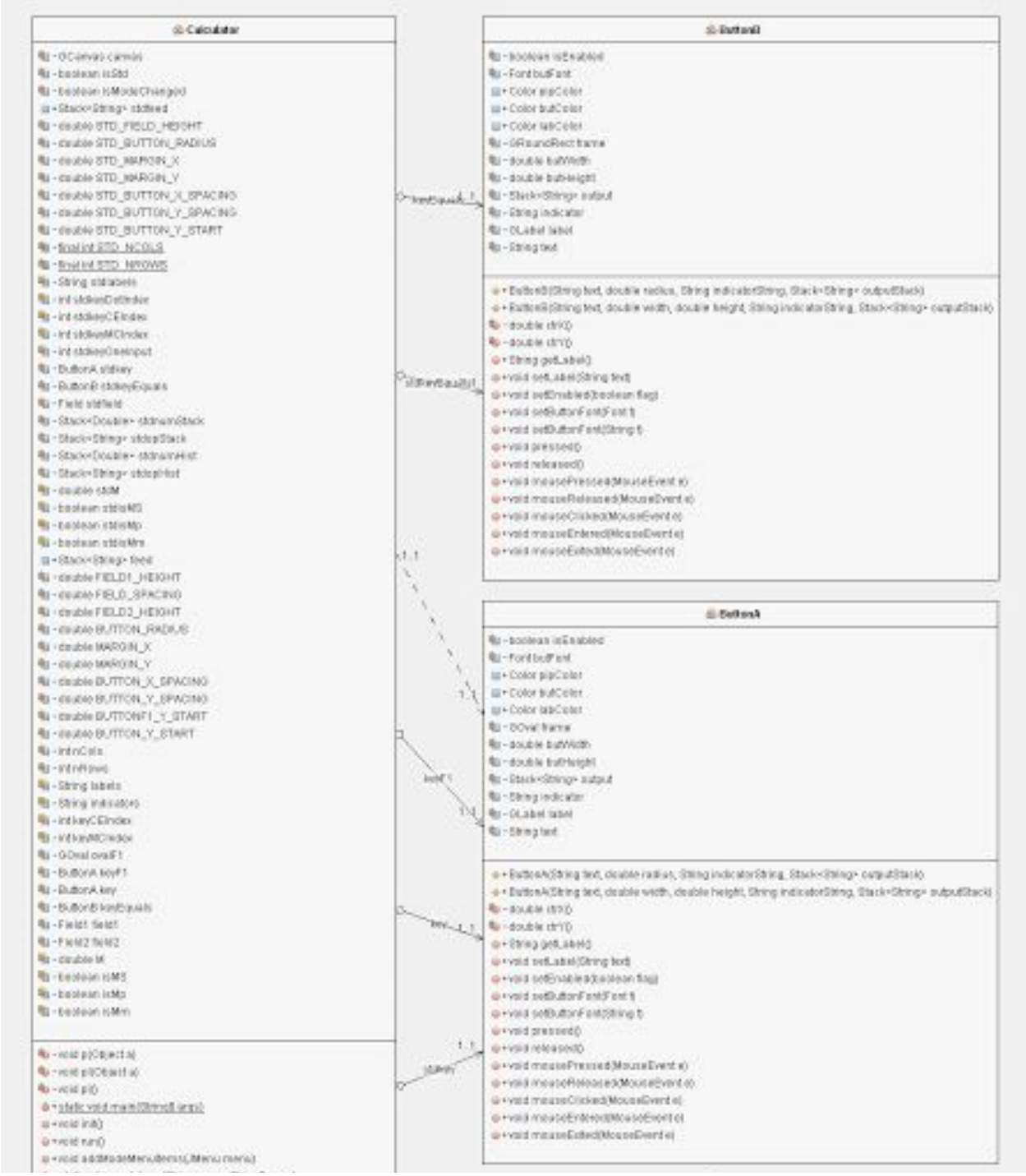
   Ex. Postfix Expression

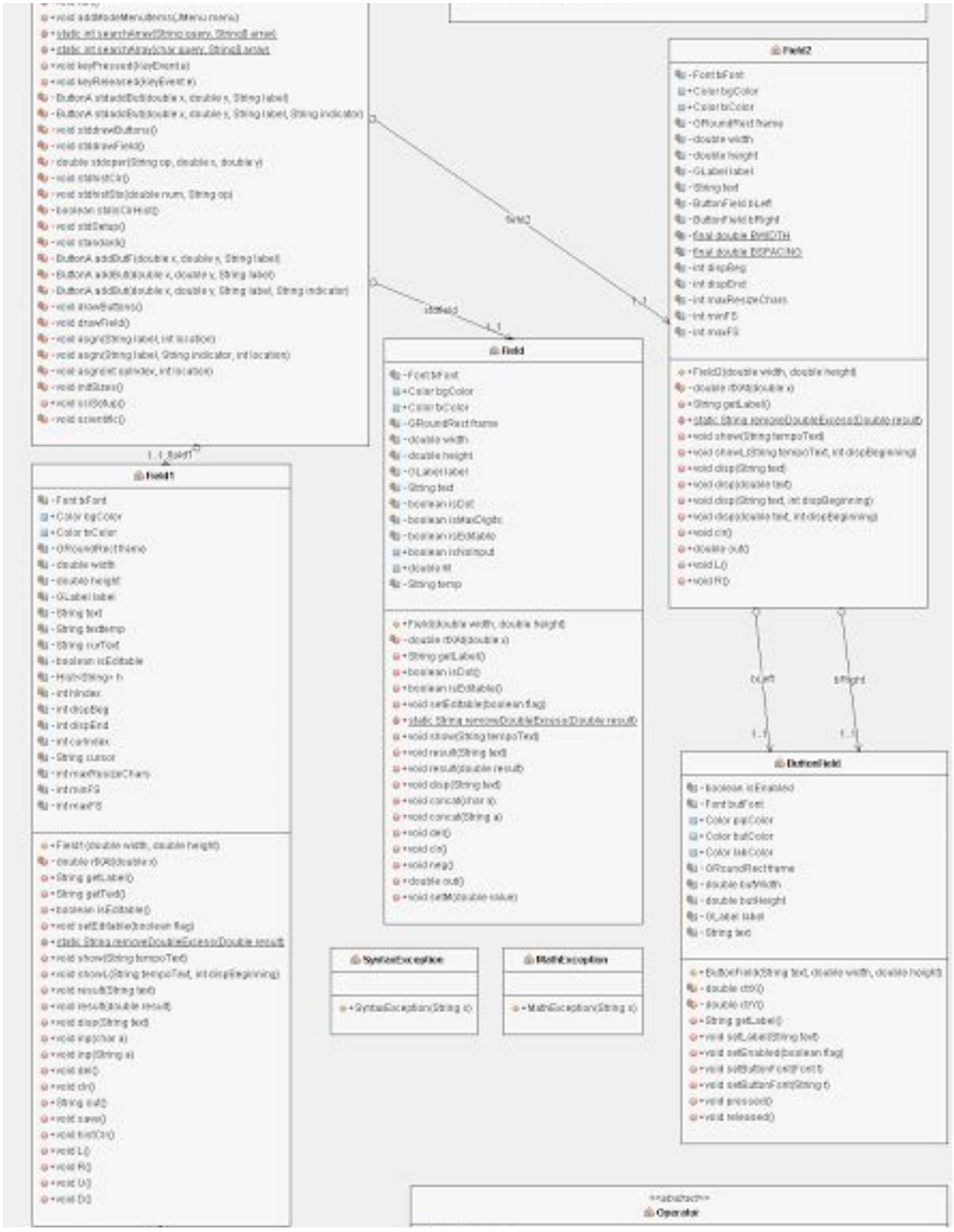
   8 2 / 6 +

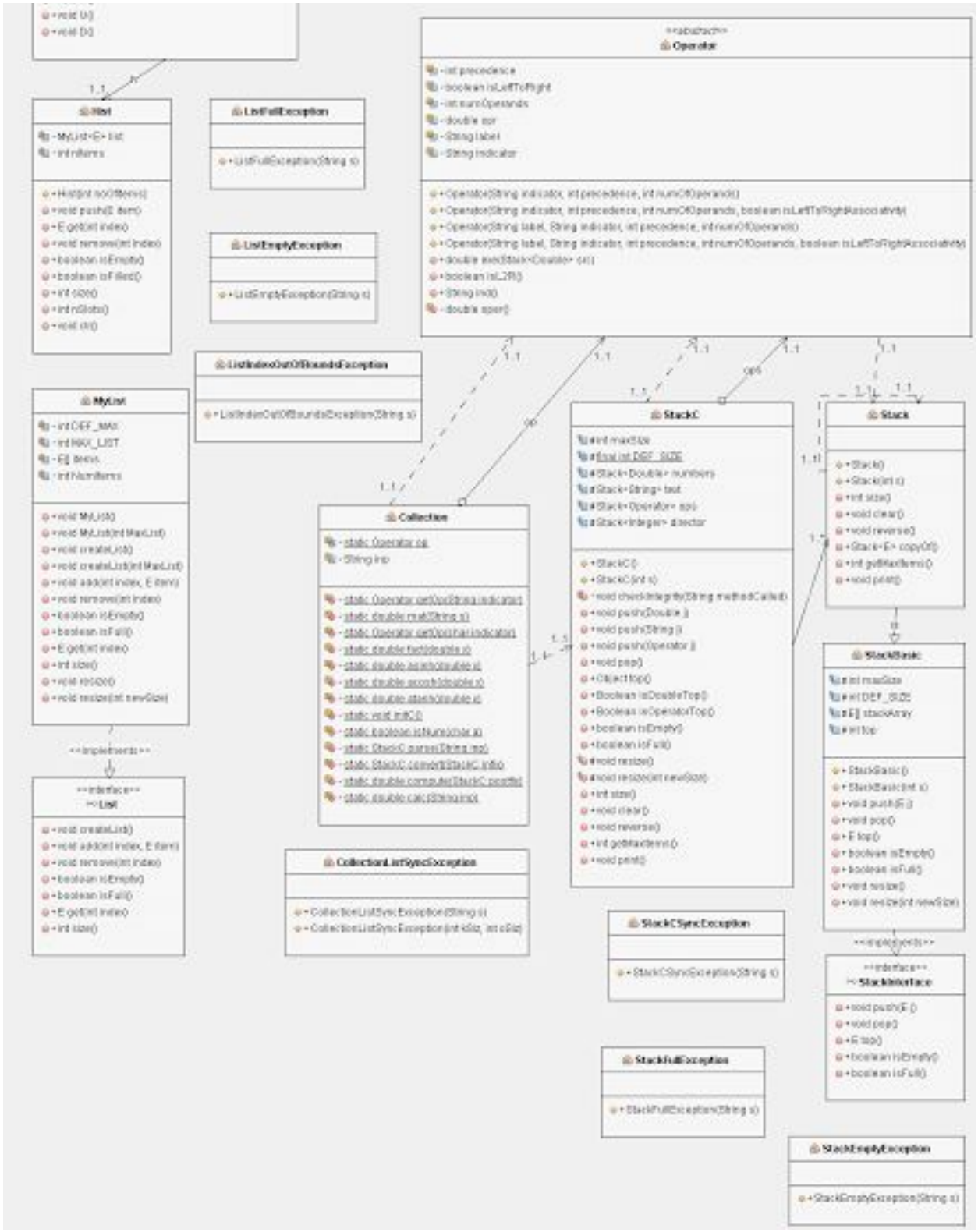
   we read the 8 and the 2 and then read the operand. The operand is applied to the two numbers. If we place numbers, or results of equations on a stack, then we can apply operations to the two top elements of the stack. In the expression above, we read a 8, and push it onto the stack. Then we read a 2 and push it onto the stack. We read the character '/', and apply it to the two top elements of the stack. 8/2 is 4 so we push 4 onto the stack. We read a 6 and push it onto the stack. When we read the final '+', we apply it to the two top elements of the stack (4 and 6), giving the result of 10.

RESULTS AND DISCUSSION

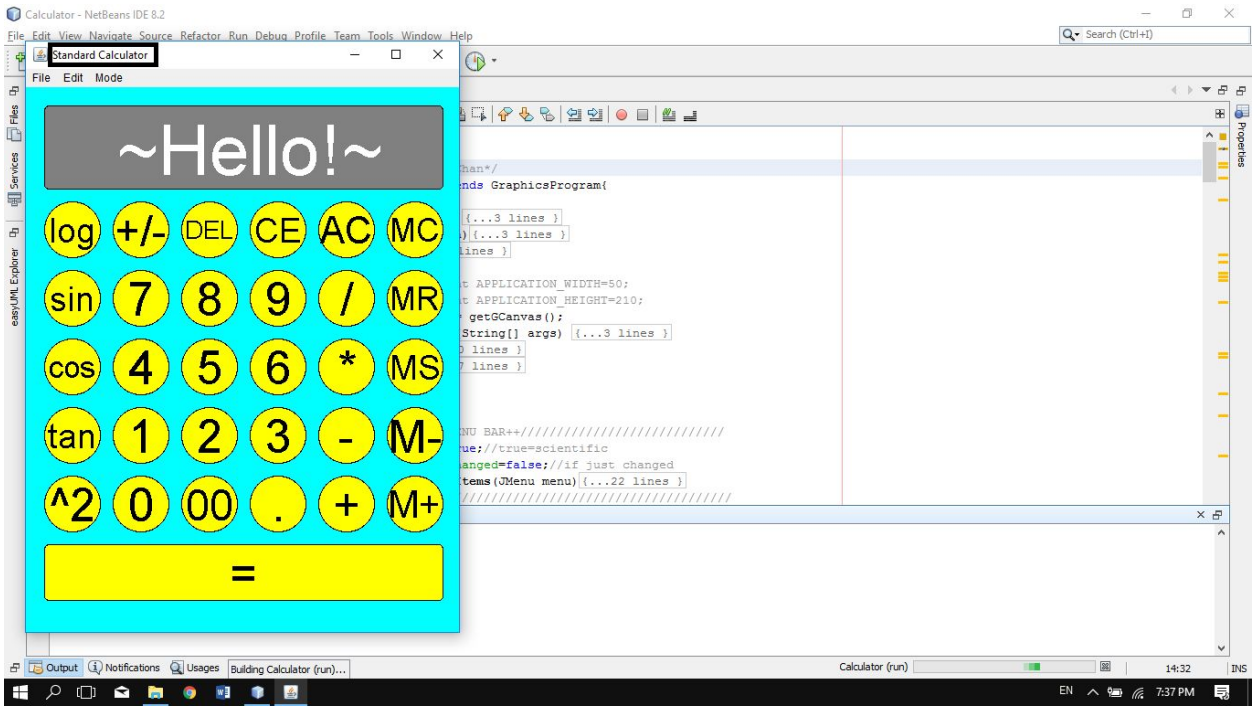
The UML:



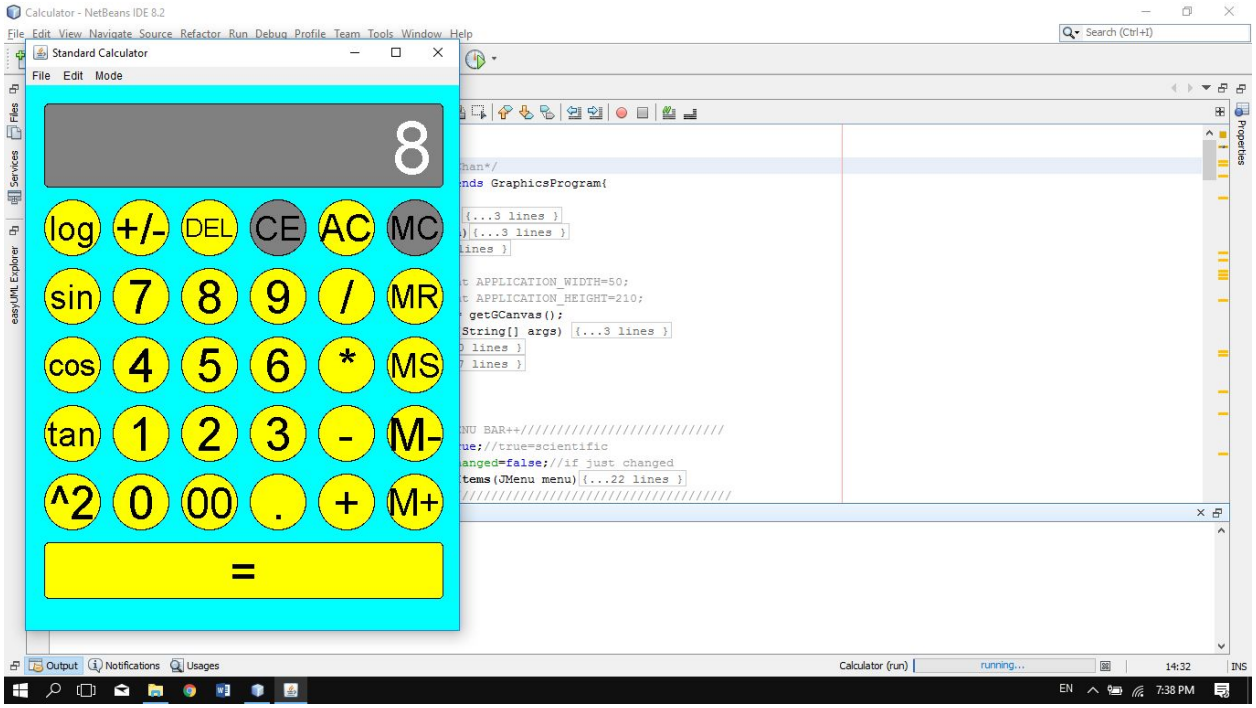

Field2
Field
Field1
SyntaxException
MathException
ButtonField
Operator


Operator
Hist
ListFullException
ListEmptyException
ListIndexOutOfBoundsException
MyList
Collection
StackC
Stack
StackBasic
List
StackInterface
CollectionListSyncException
StackCSyncException
StackFullException
StackEmptyException

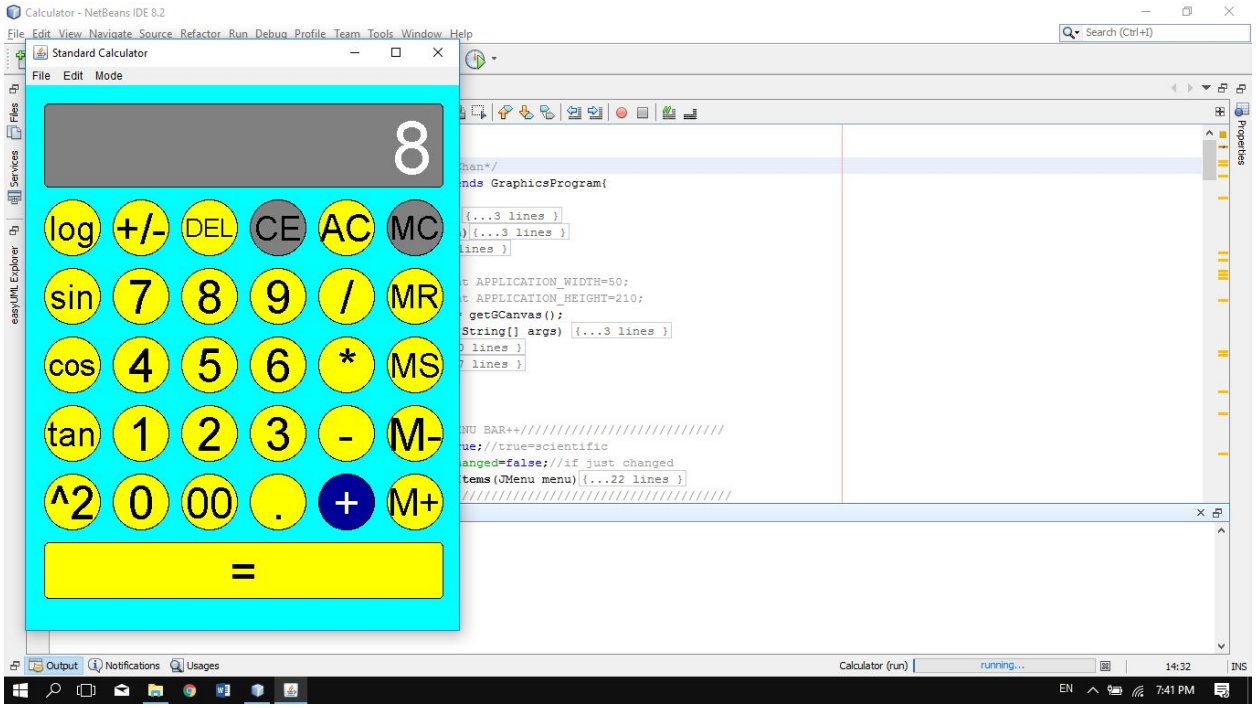
Screenshots:



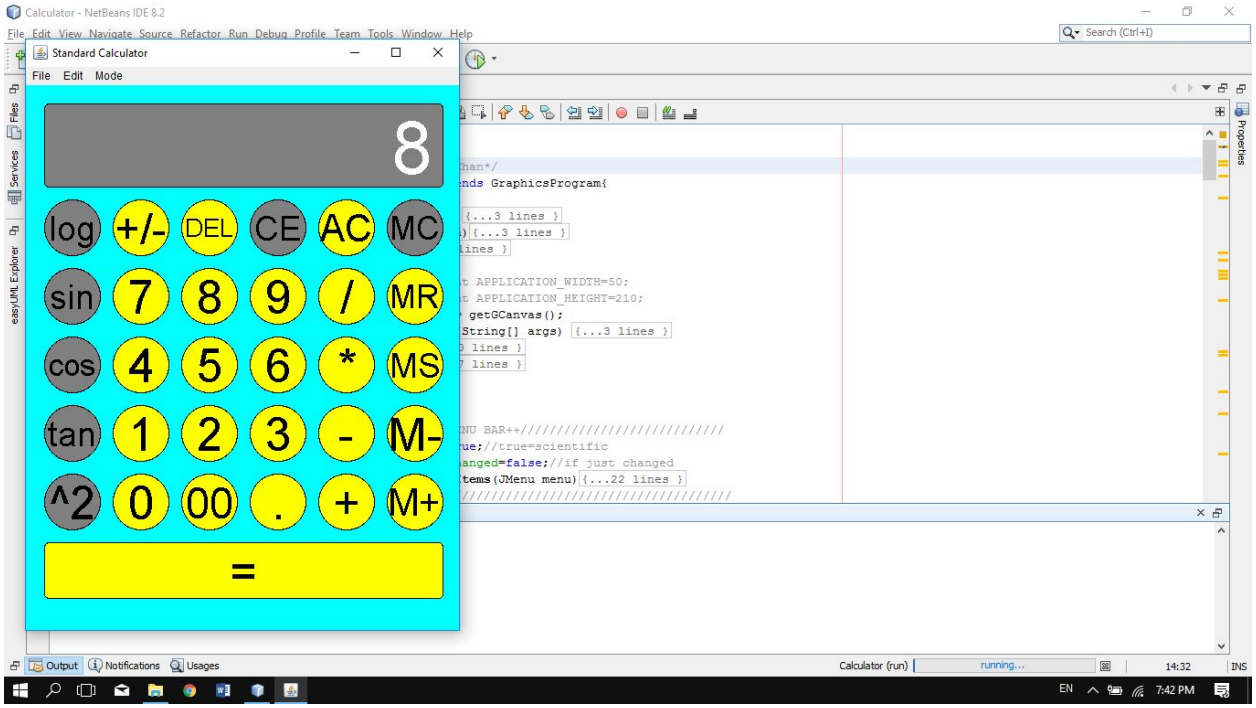
The program greets the user, and loads the standard calculator by default



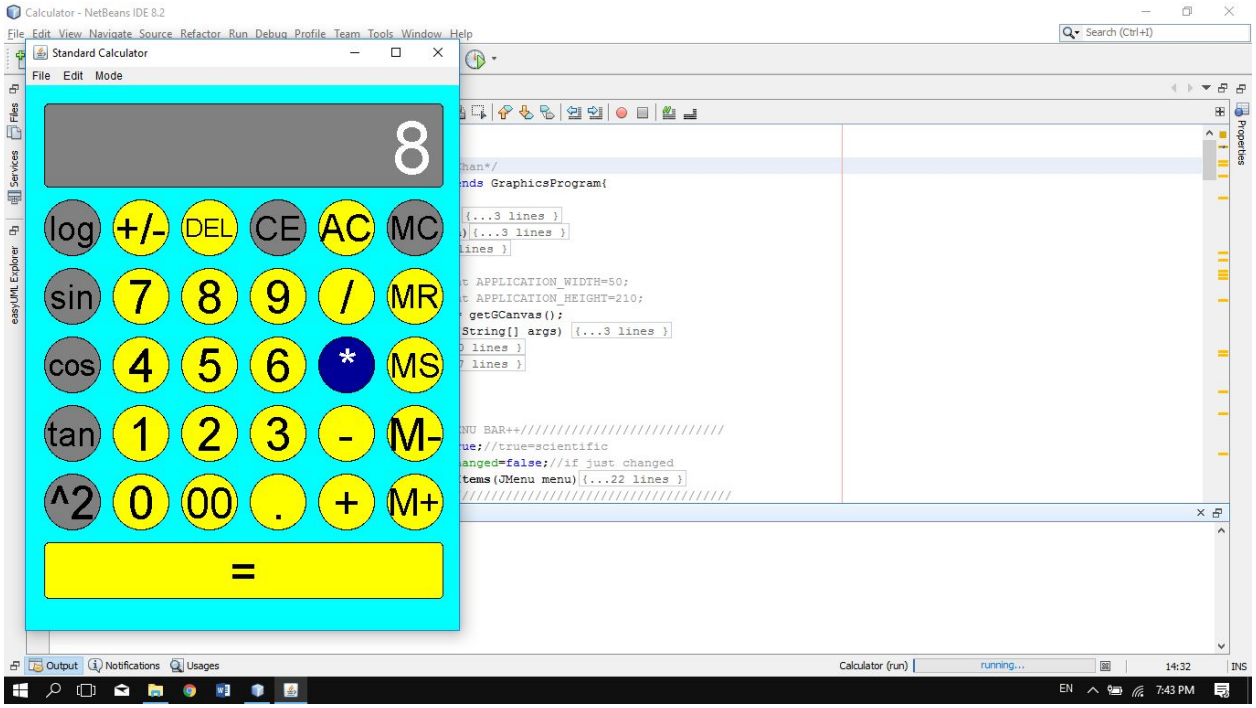
Standard calculator mimics a usual restaurant/store use calculator.



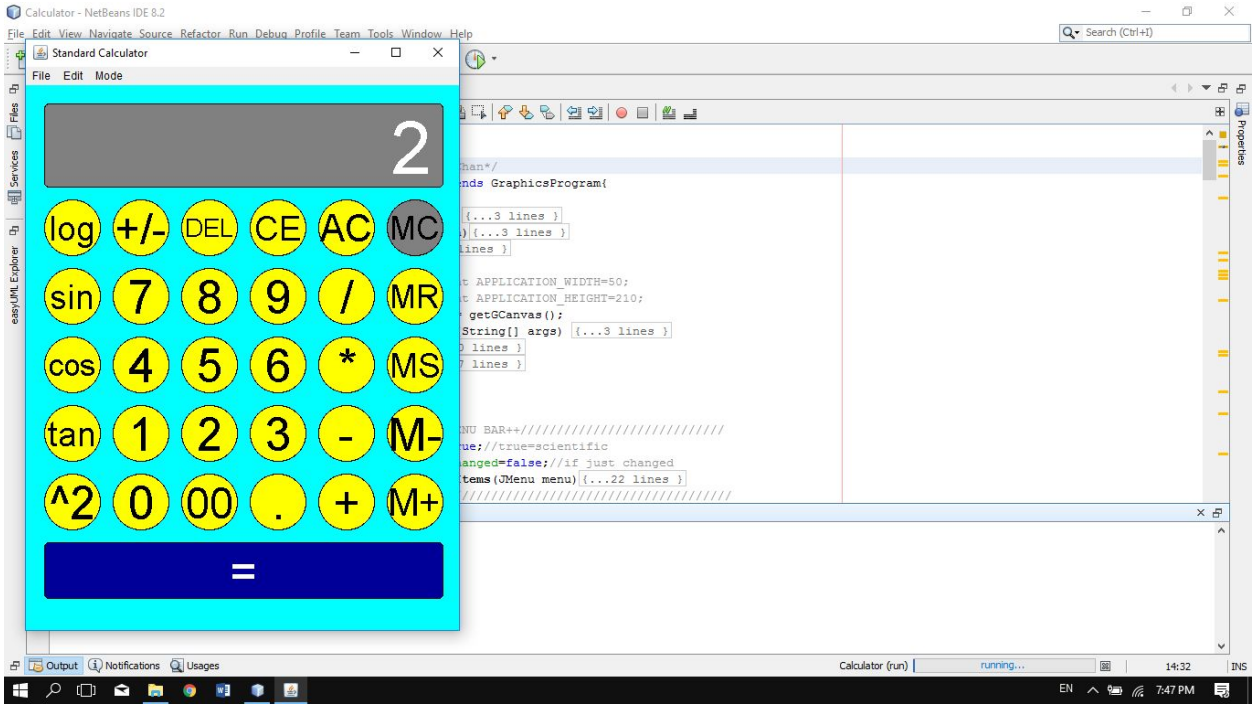
Buttons “can be pressed”, and operations can be “chained”, like a physical calculator.



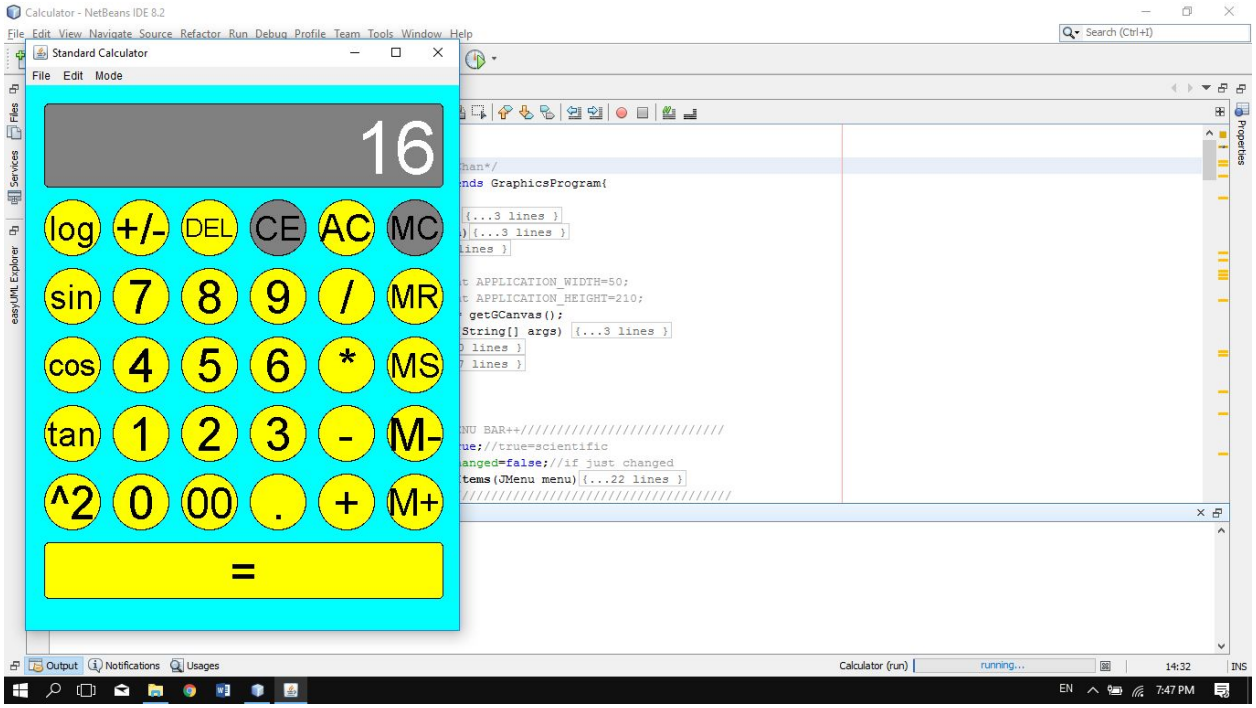
When an operator is pressed, operation selection mode is activated, which means that the operator to use can be changed, as long as no numerical input has been detected.



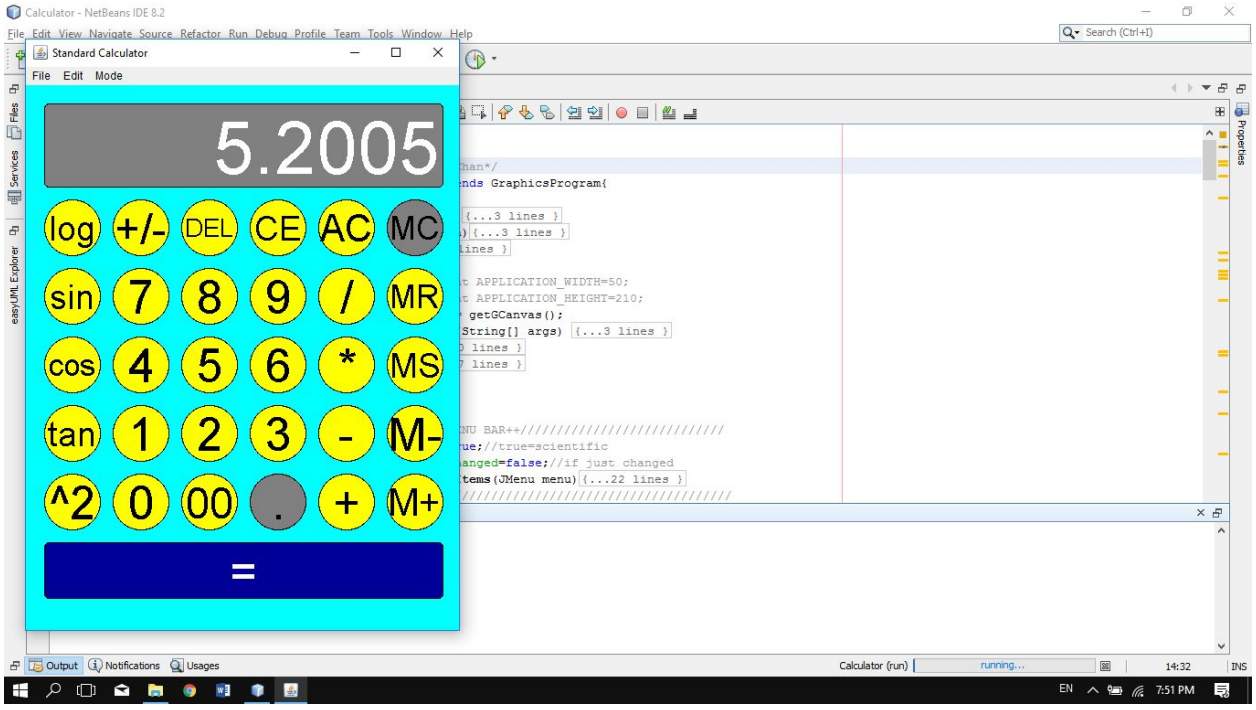
Here, the operator '+' earlier is changed to '*'. Also, the extra operators on the left side are disabled during operation selection mode by default. This is because as standard calculators immediately perform the given operation, the extra operators cannot be used at this point, since they operate on one number only. (More on this later)



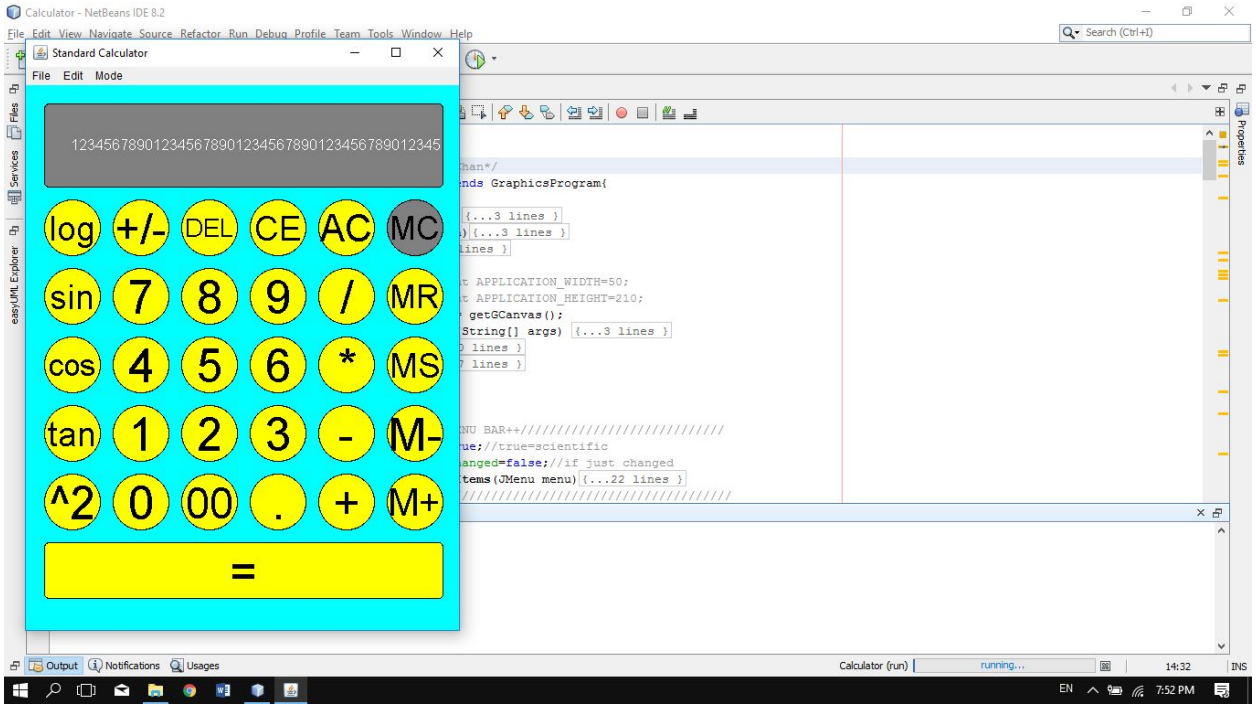
When a number is inputted, the calculator exits operation selection mode. Note that the CE button means "Clear Entry", and is used during user input of numbers.



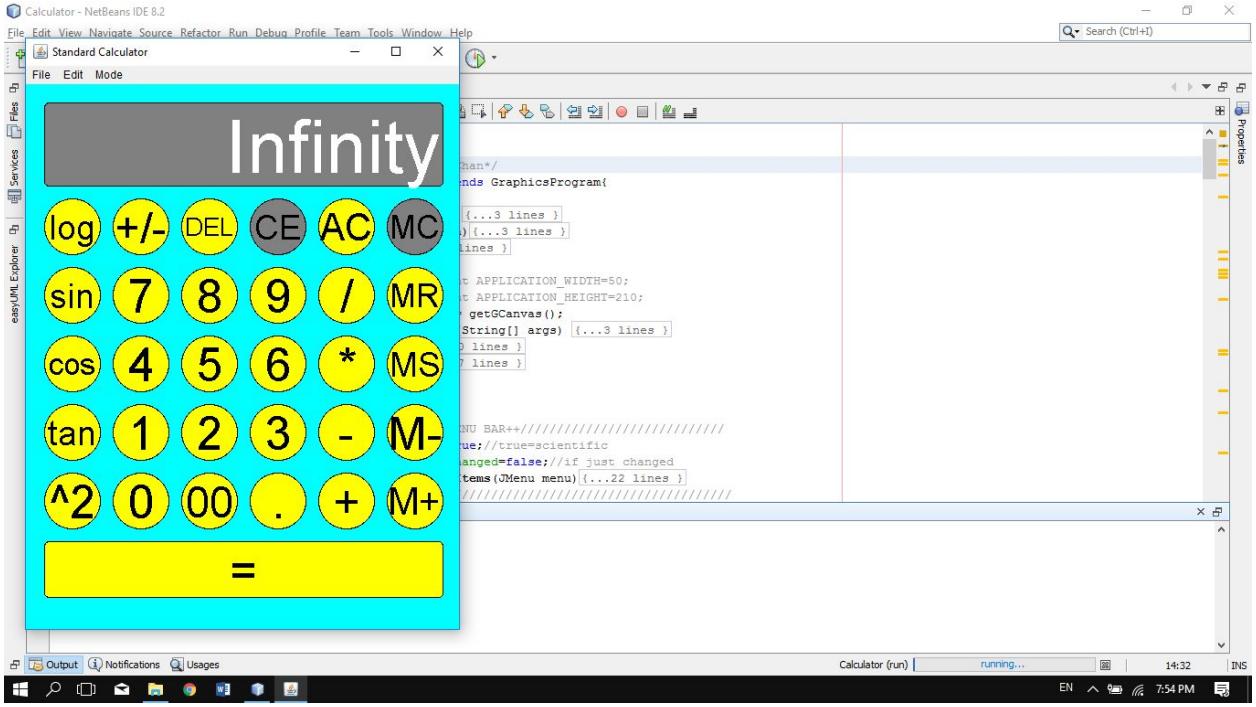
The result is evaluated. CE button is disabled when the calculator is out of editing mode.



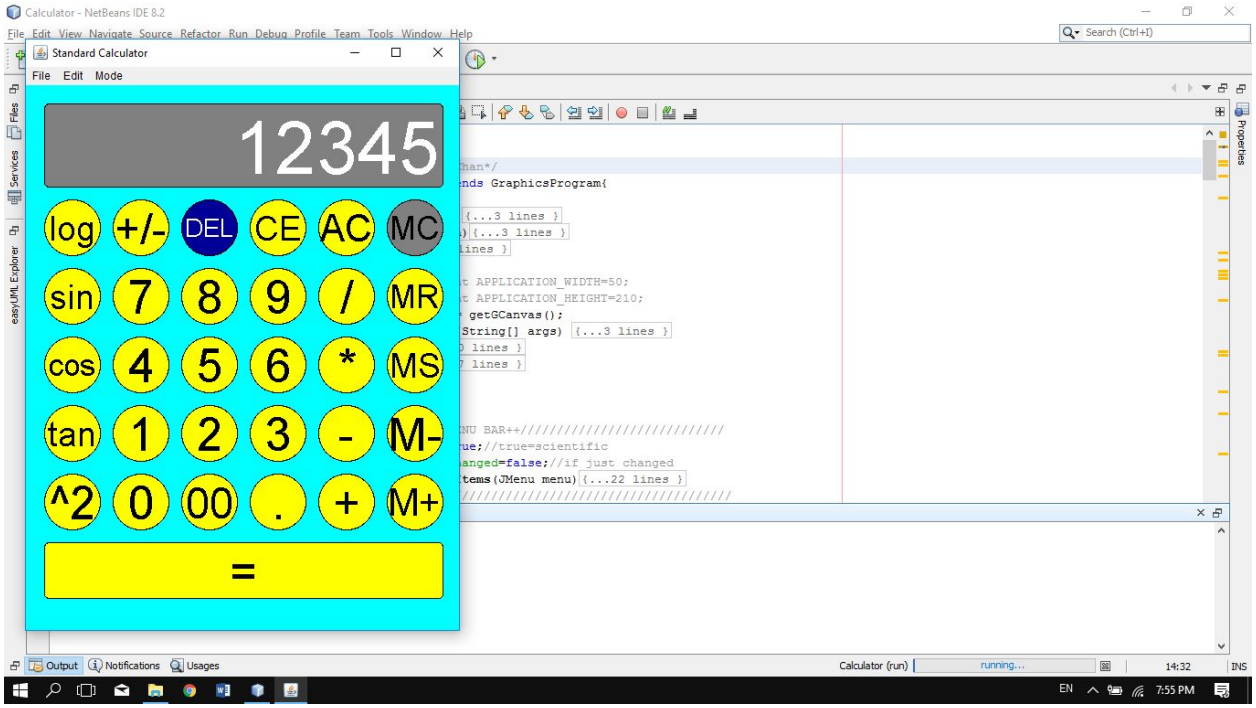
Calculator supports decimal operations.



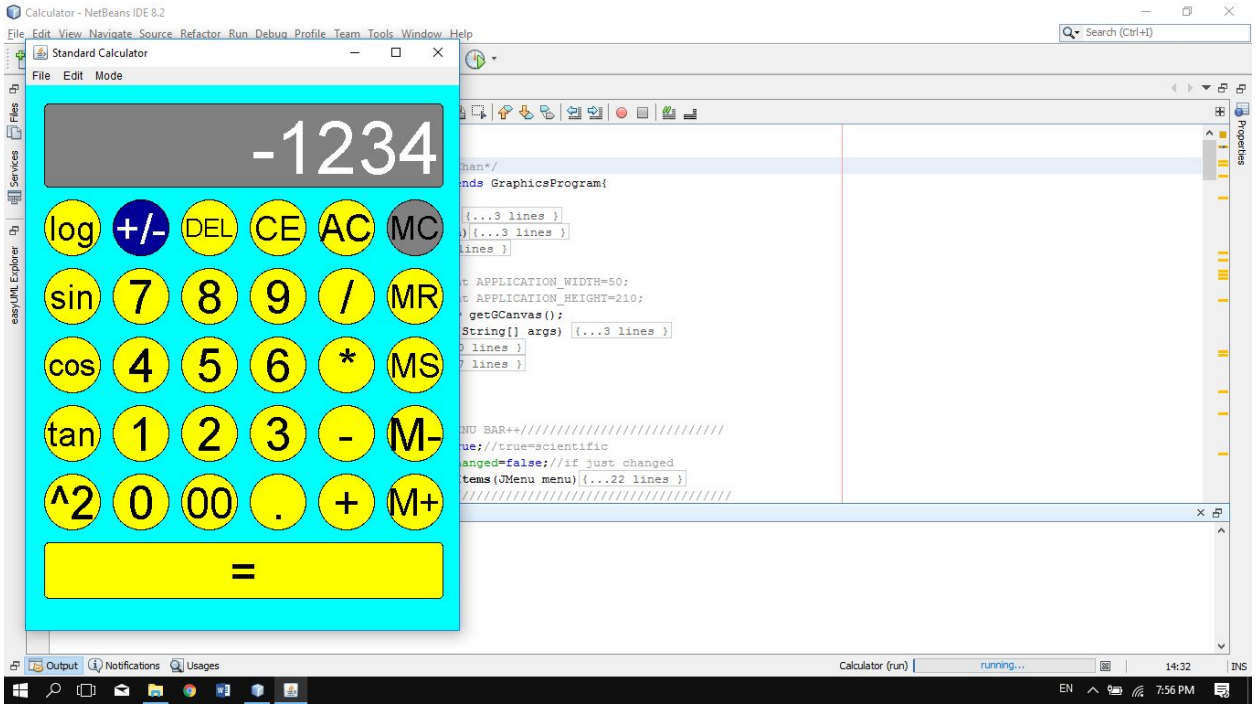
Calculator supports multi-digit input, and resizes the display accordingly to the length of the display. The calculator limits the input when any further input makes the font too small to be read.



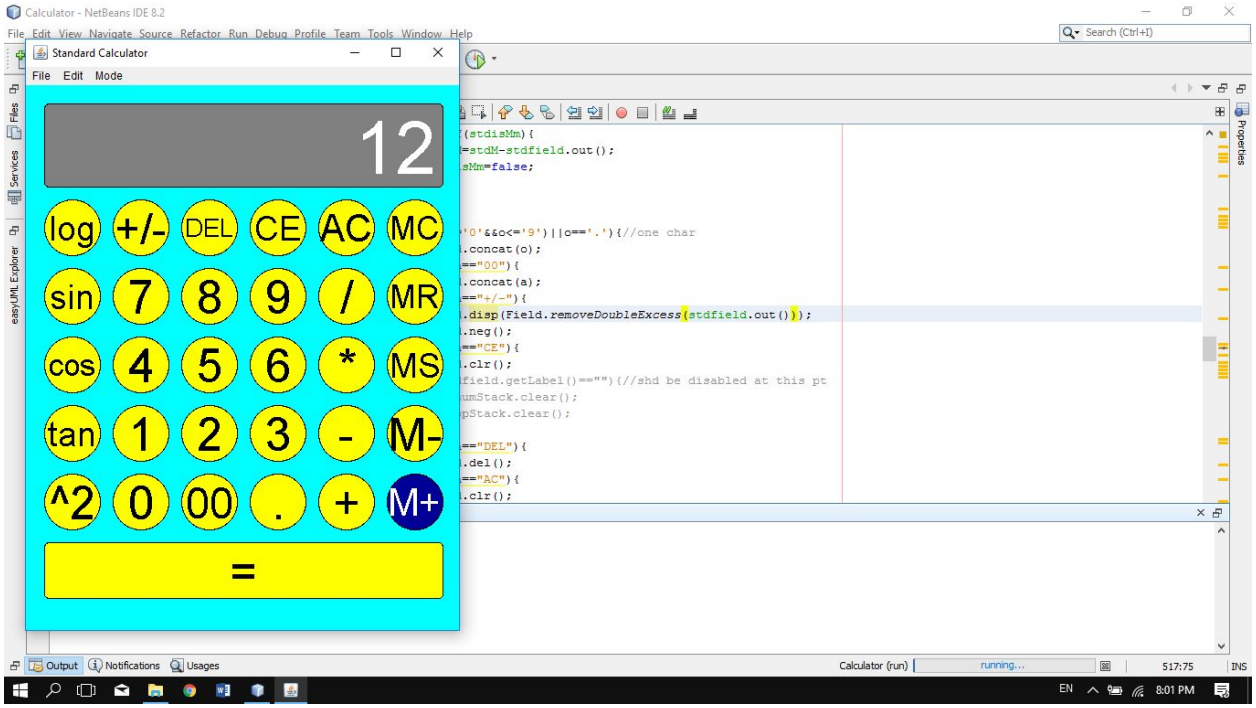
When the result is too large, the calculator displays “Infinity”



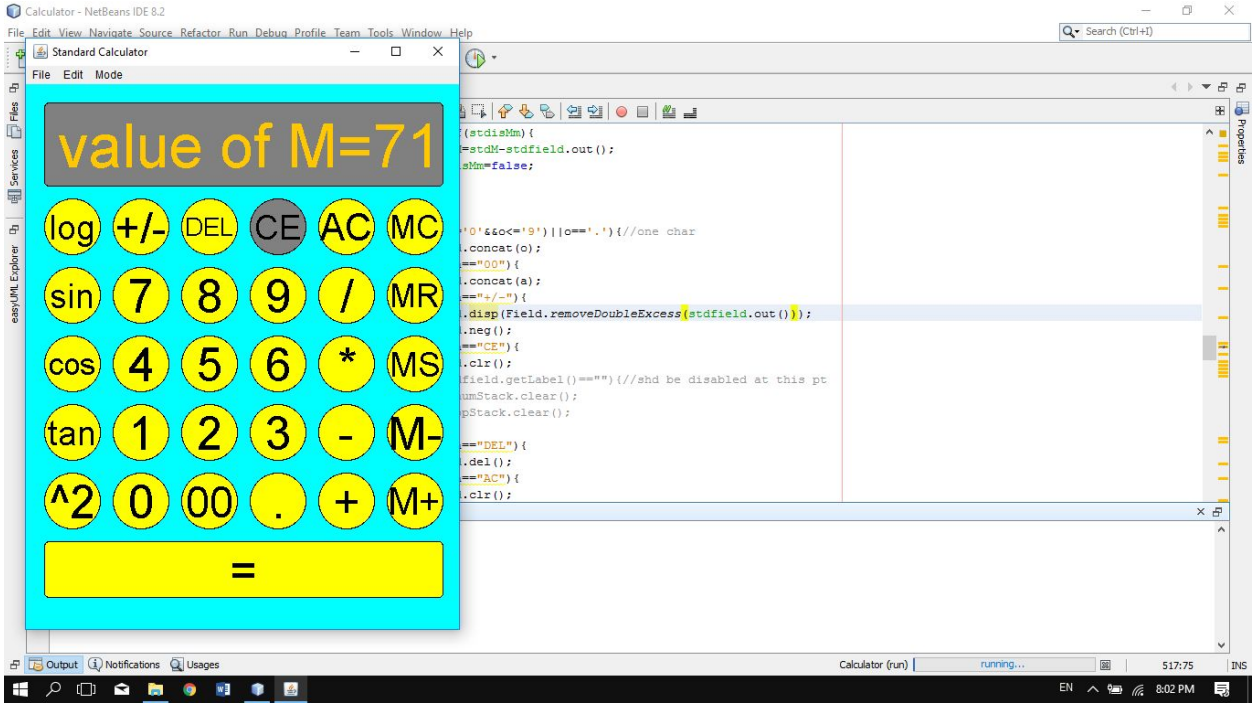
"DEL" button deletes most recent input, while "AC" or "All Clear" resets the input (it also removes operands from memory, unlike "CE")



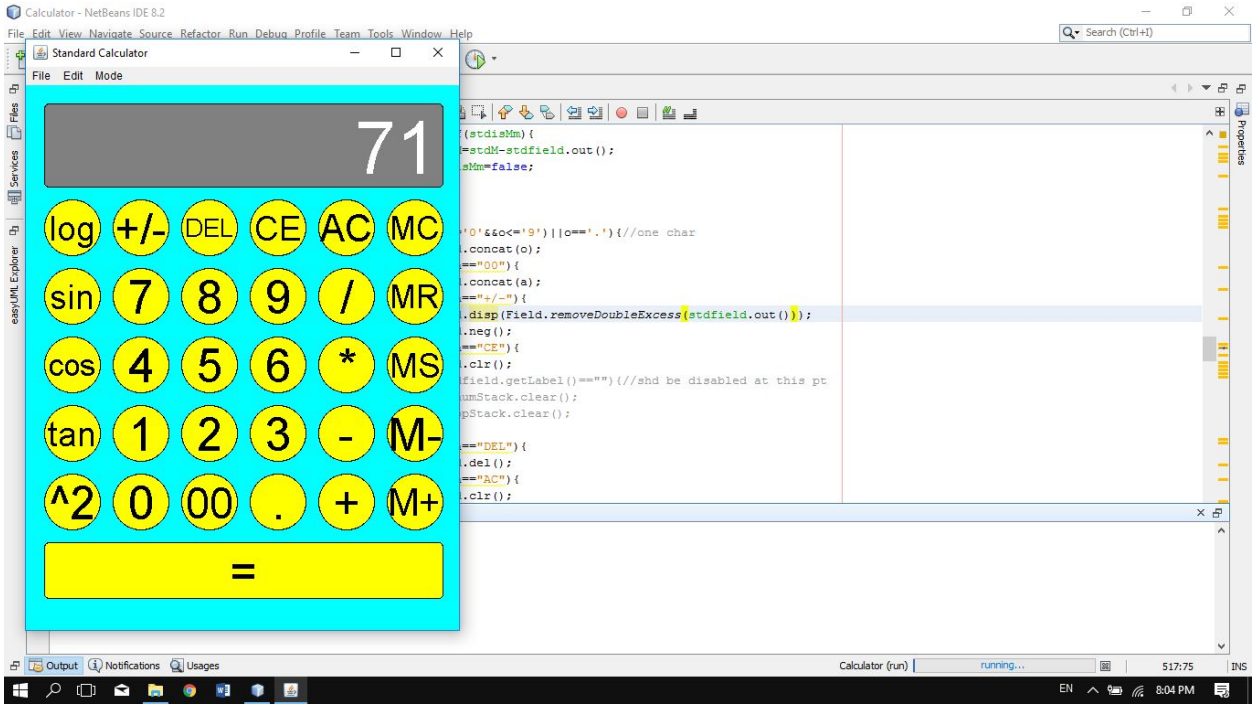
"+/-" button negates the input. It acts as a toggle for the sign of the input.



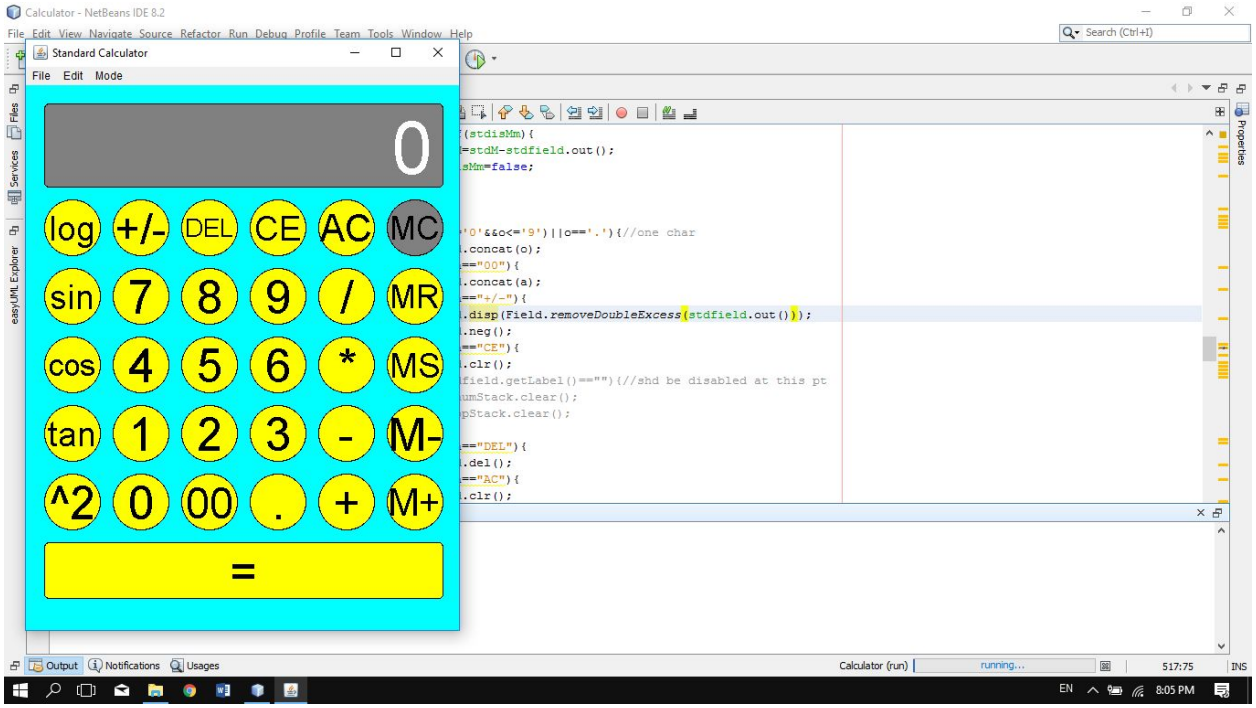
M+ adds the result to variable M, and M- subtracts the result from M. On the other hand, MS (M Store) simply stores the result to M.



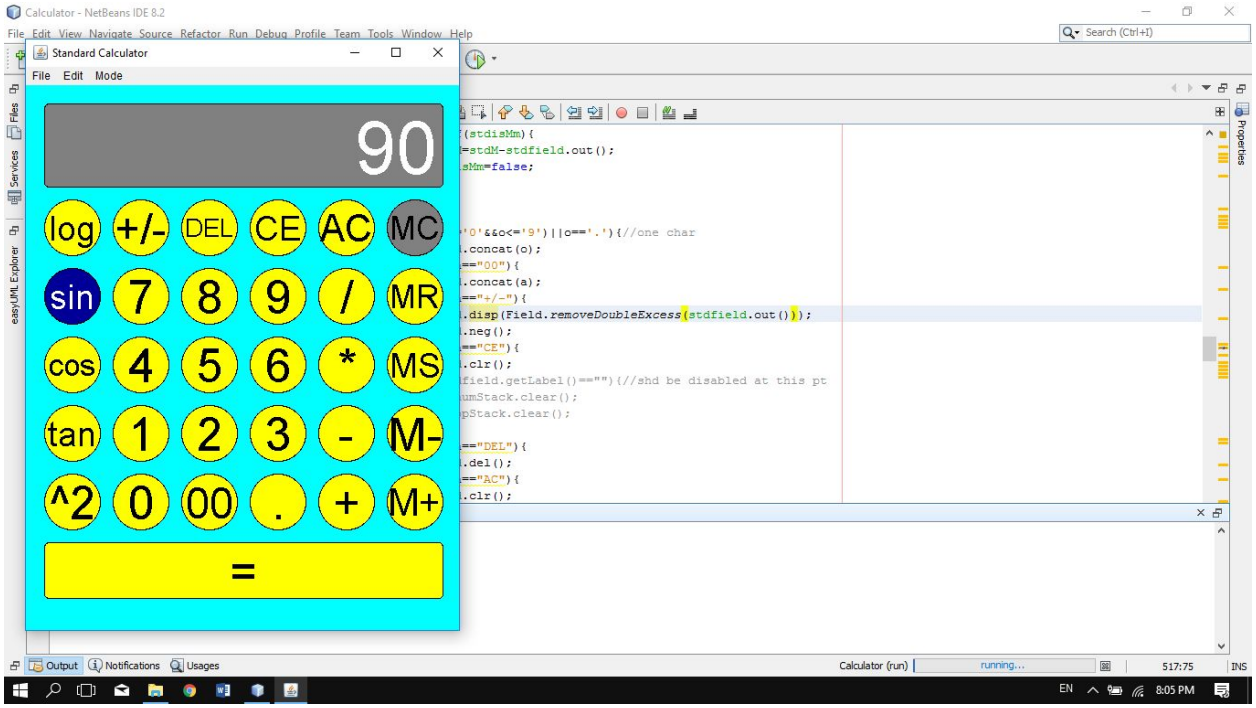
Hovering mouse over display shows the value currently stored in M.



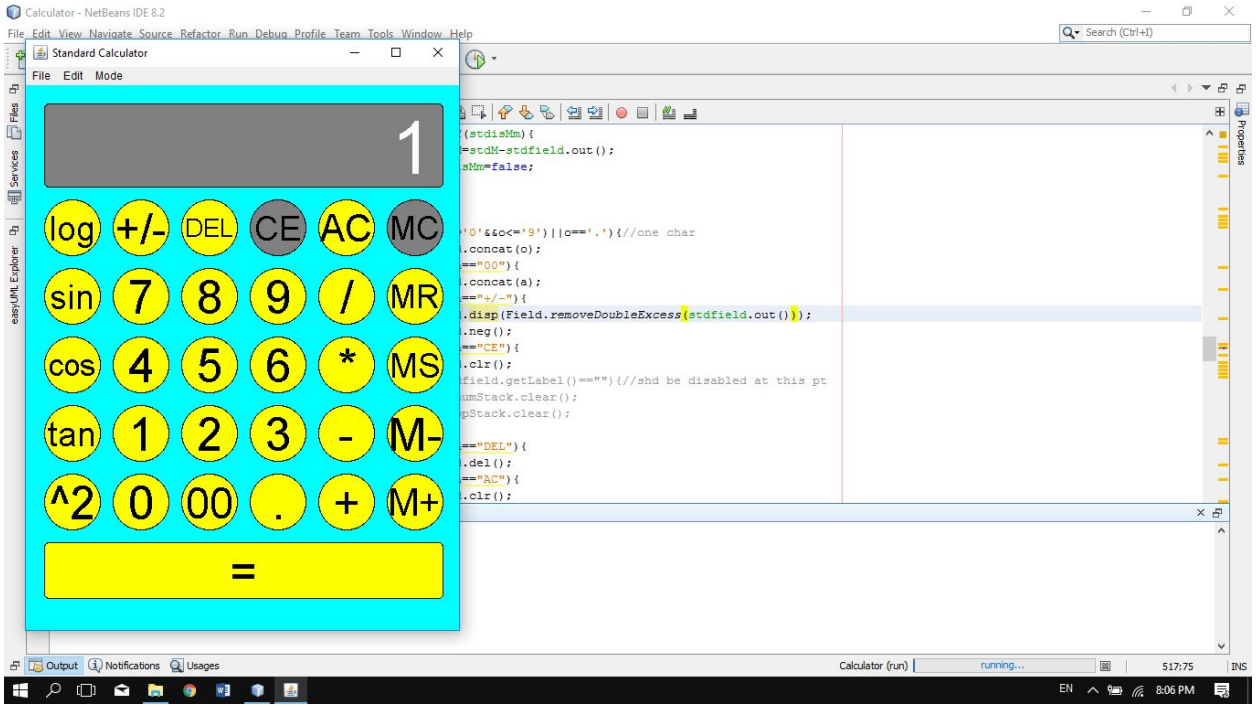
MR (M Recall) gets the value of M, which can be used as input, while MC (M Clear) resets the value of M to zero.



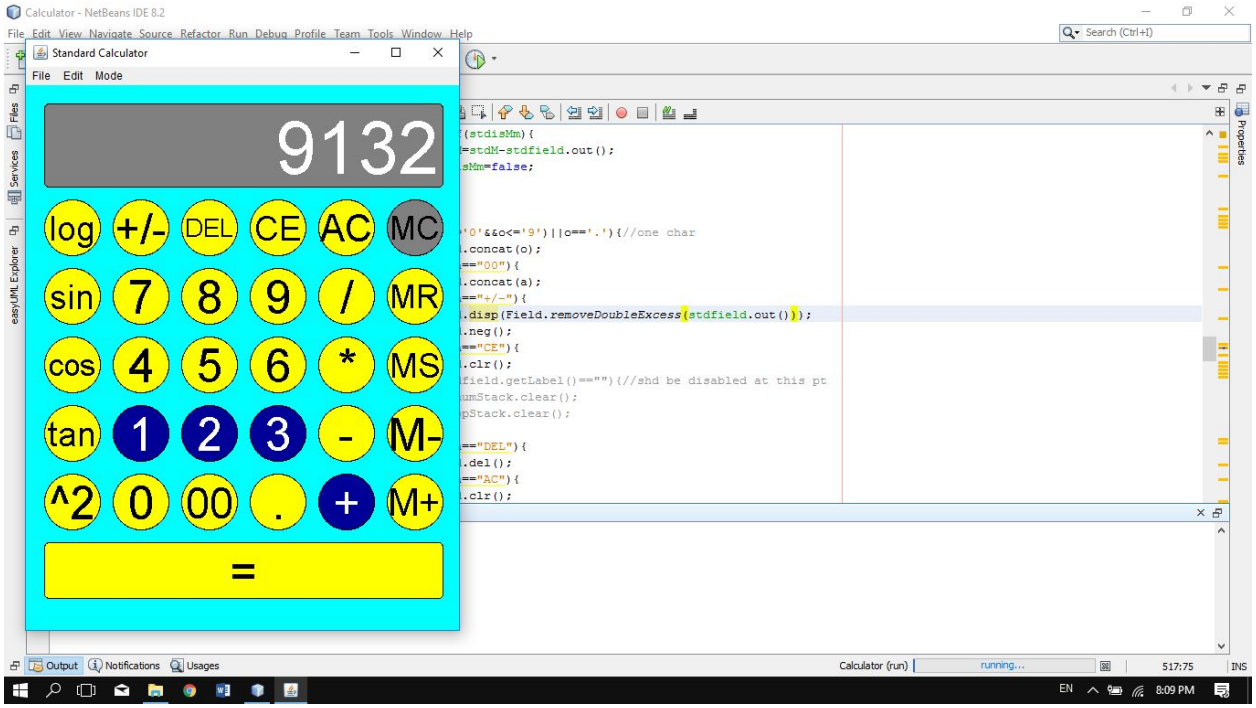
MC button is disabled when M=0.



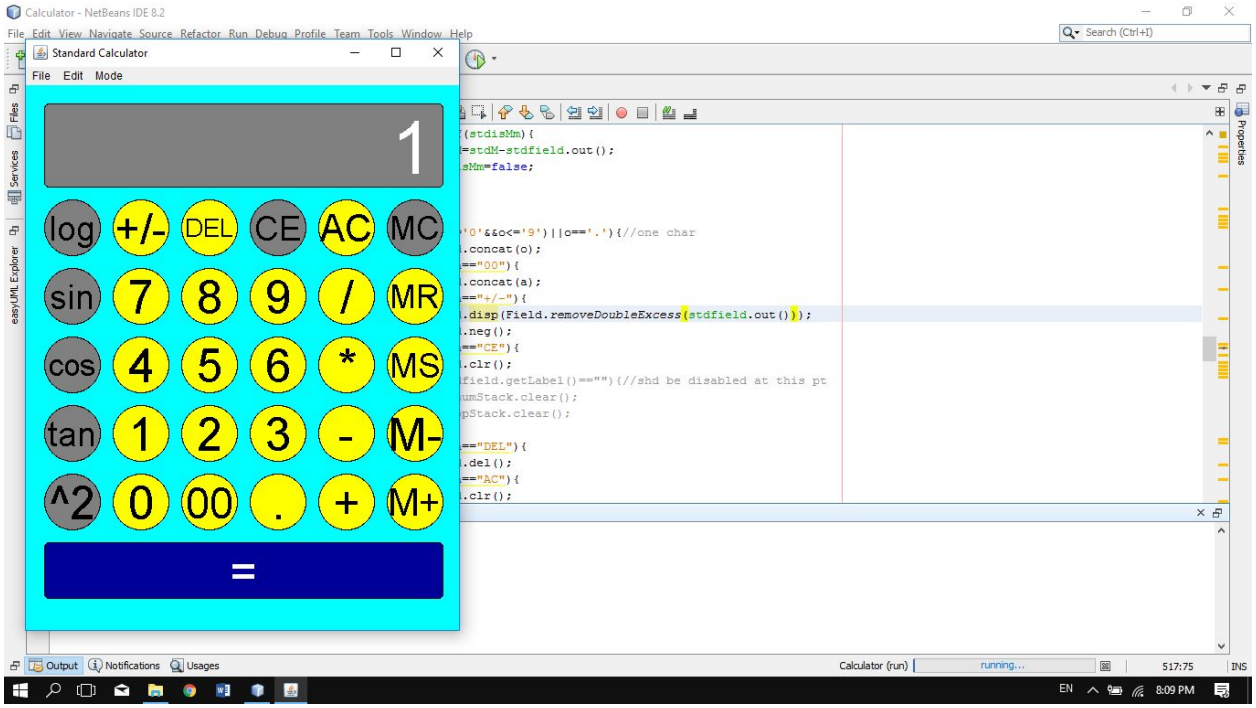
The extra operations immediately operate on the input number.



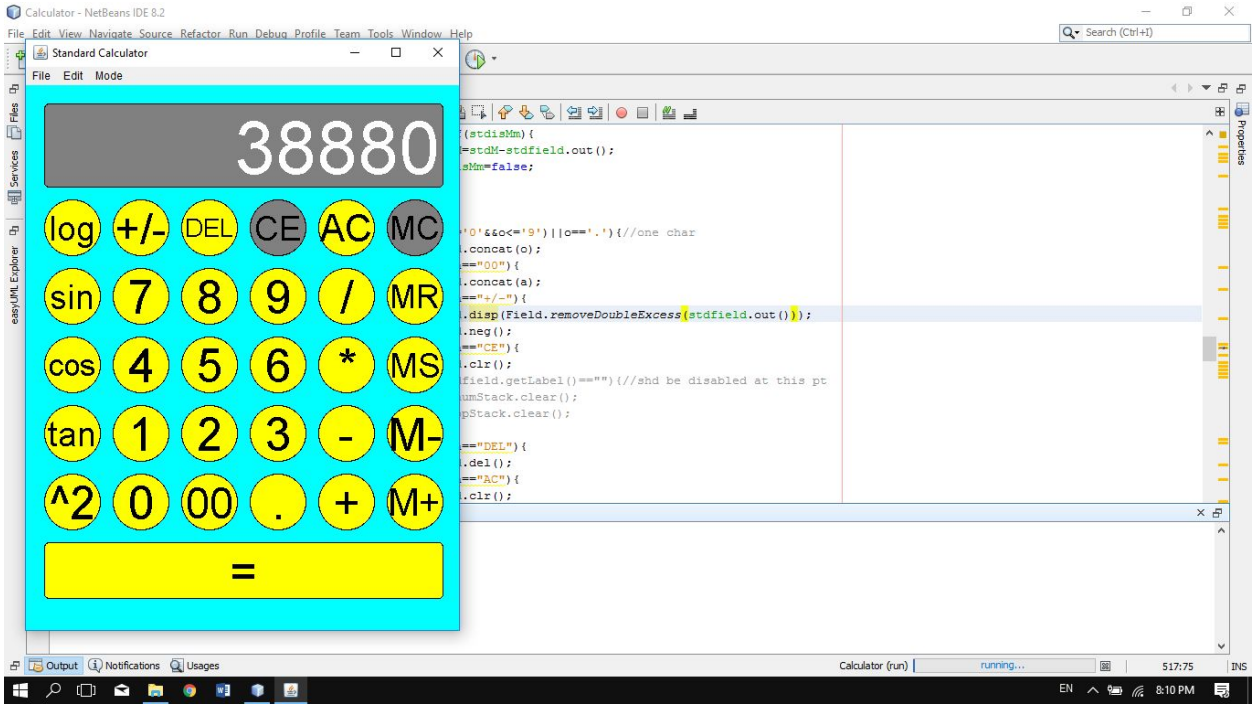
sin, cos, tan in this case are defaulted to degrees, so sin 90 = 1.



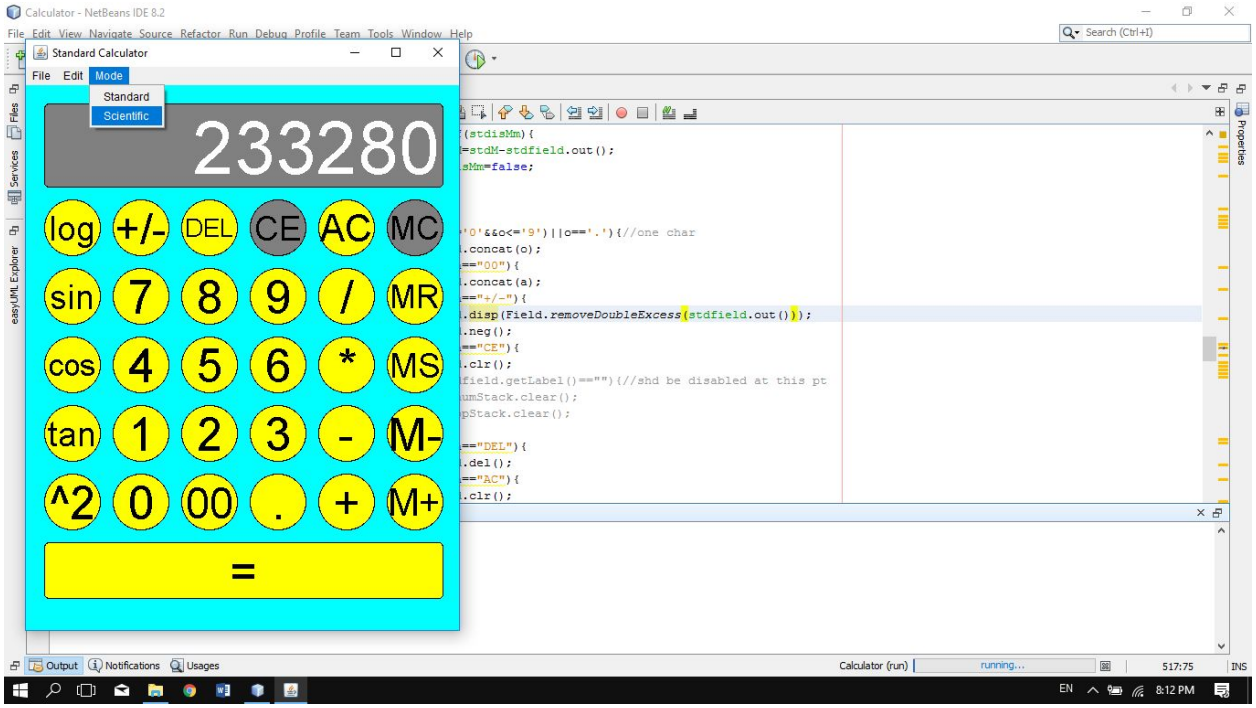
The keyboard can be used to press the keys, and are read when the keys are released.



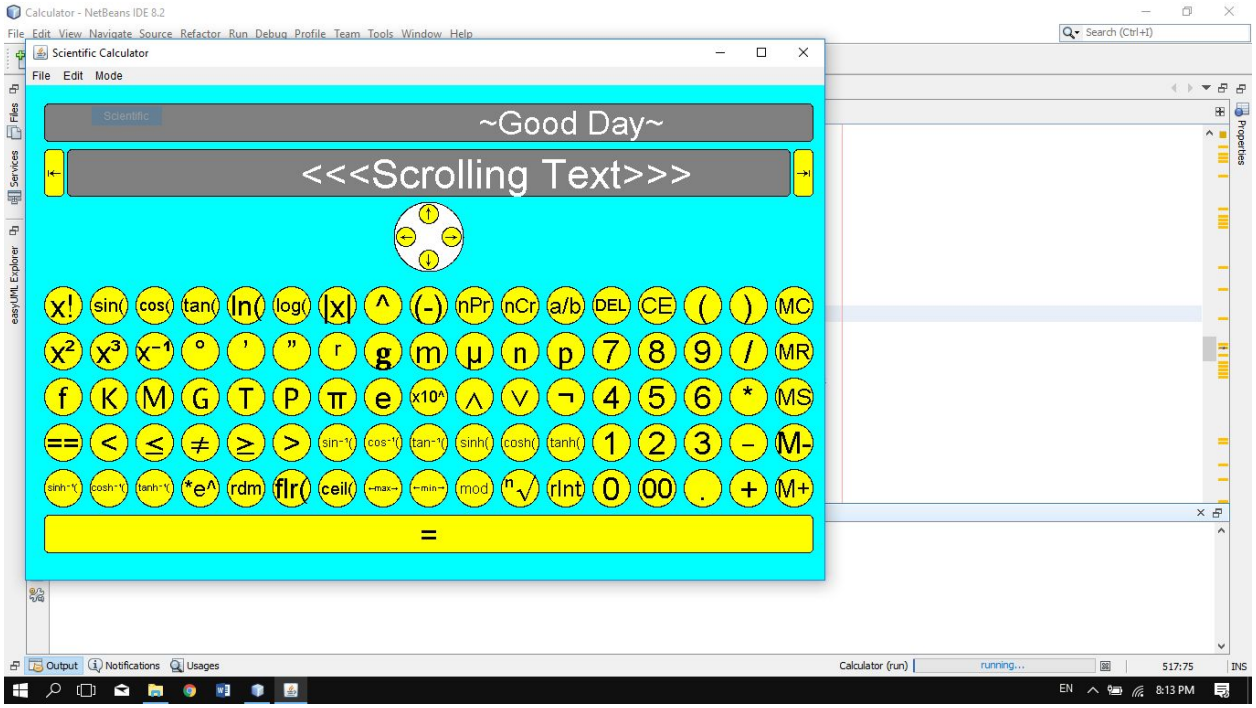
Doing an operation without a second operand results in the calculator copying the first operand as the second operand, like some physical calculators do.



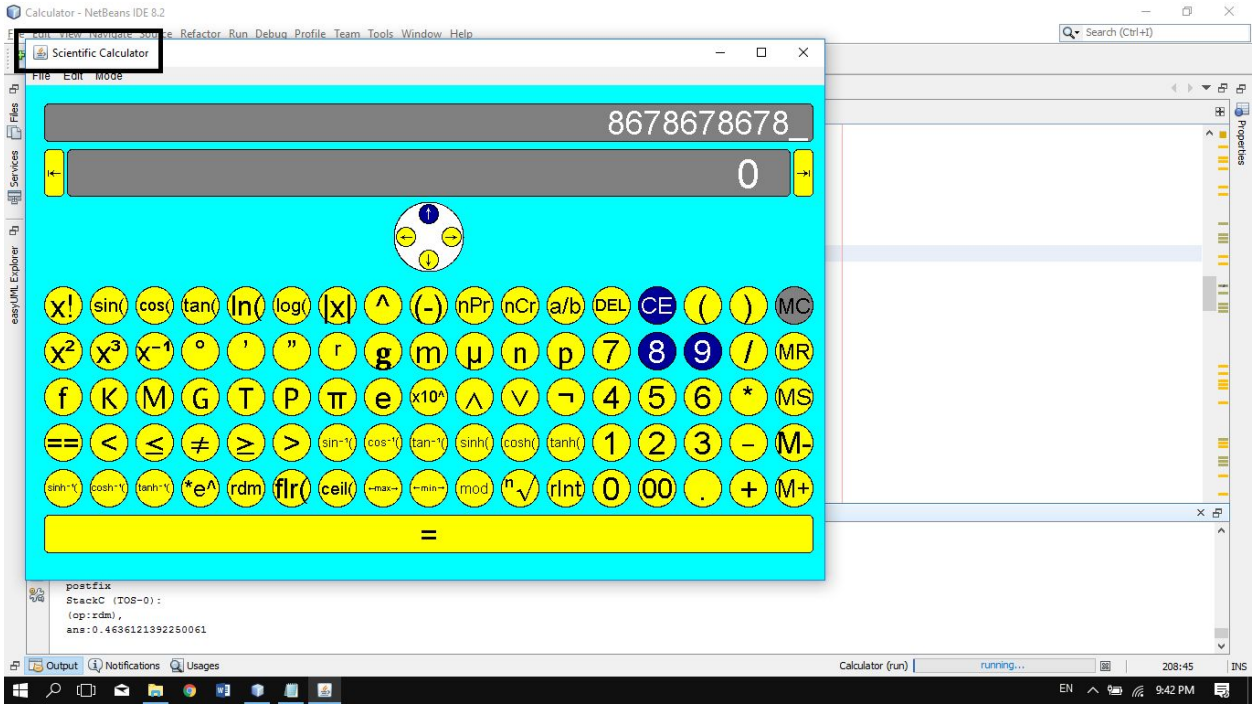
Previous operations are remembered by the calculator, and continuous “=” presses signal the calculator to keep repeating this operation, like physical calculators do.



That’s not all! The menu bar has an additional menu called “Mode” which lets the user pick which kind of calculator is needed.

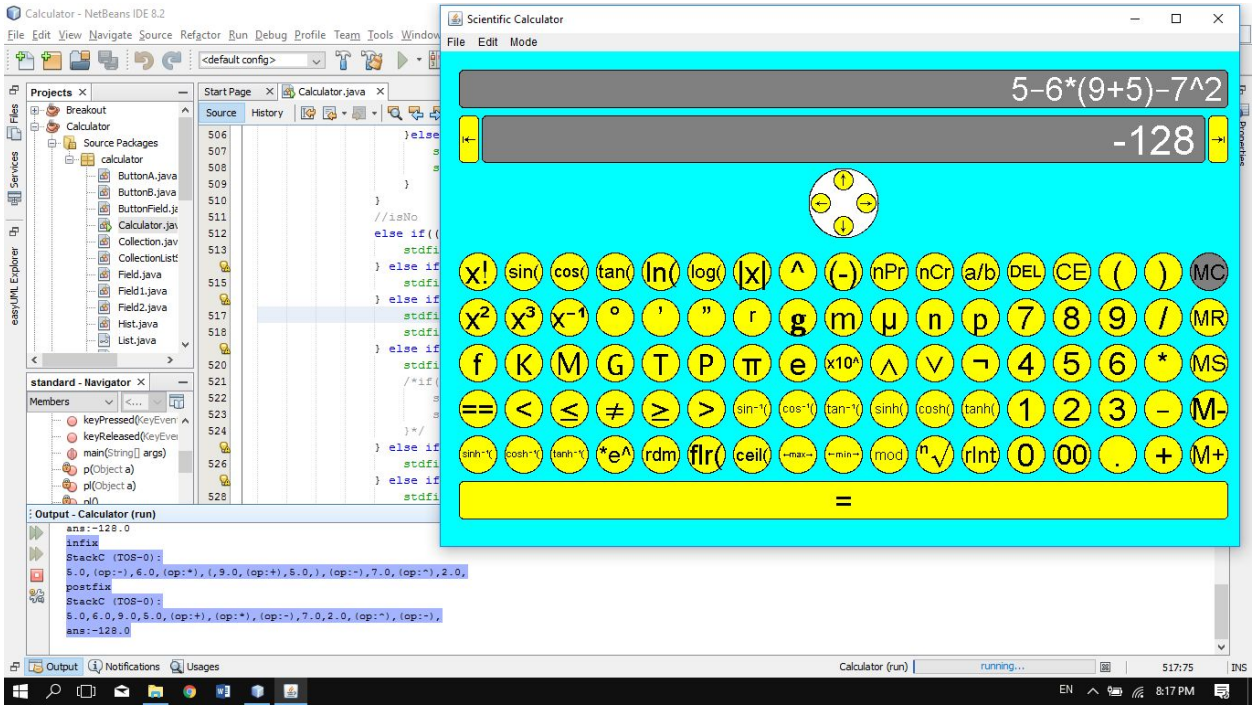


Choosing scientific calculator changes the display, and initializes the calculator.

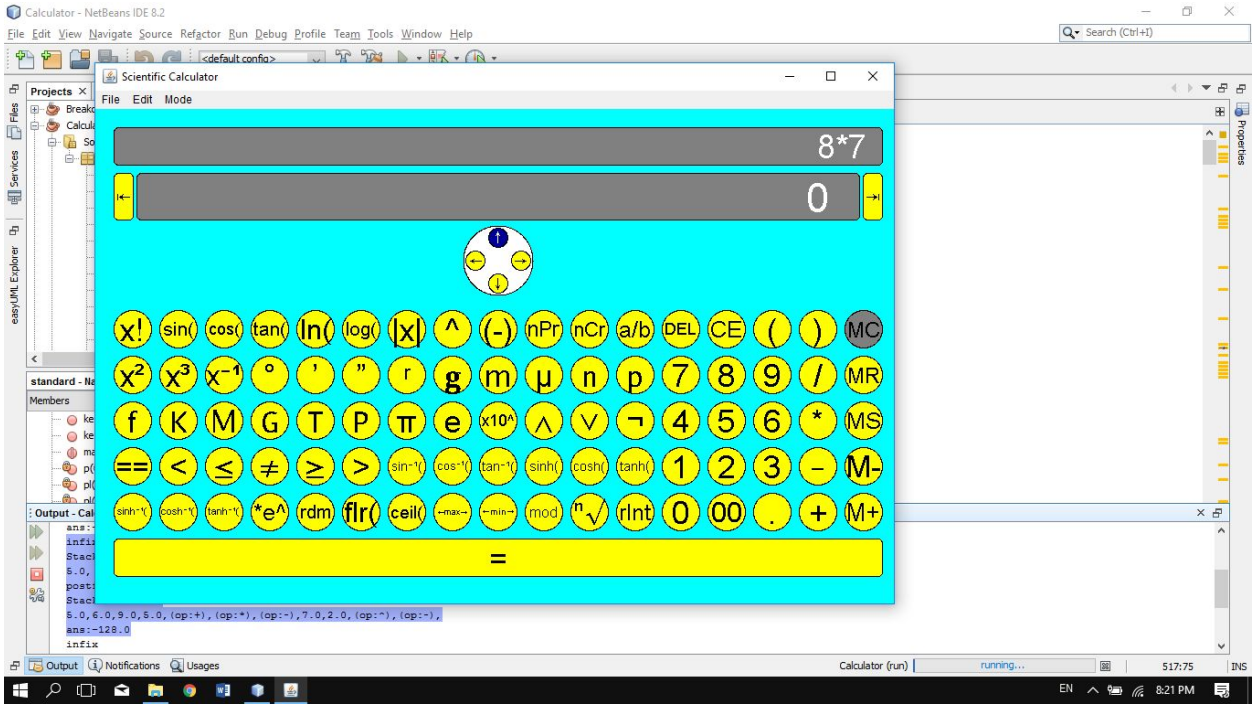


It has some similar features to the Standard calculator (e.g. can be pressed with keyboard, Title Bar, M functions)

One main difference in this mode, is that operations are now done by precedence (PEMDAS) like a physical scientific calculator.

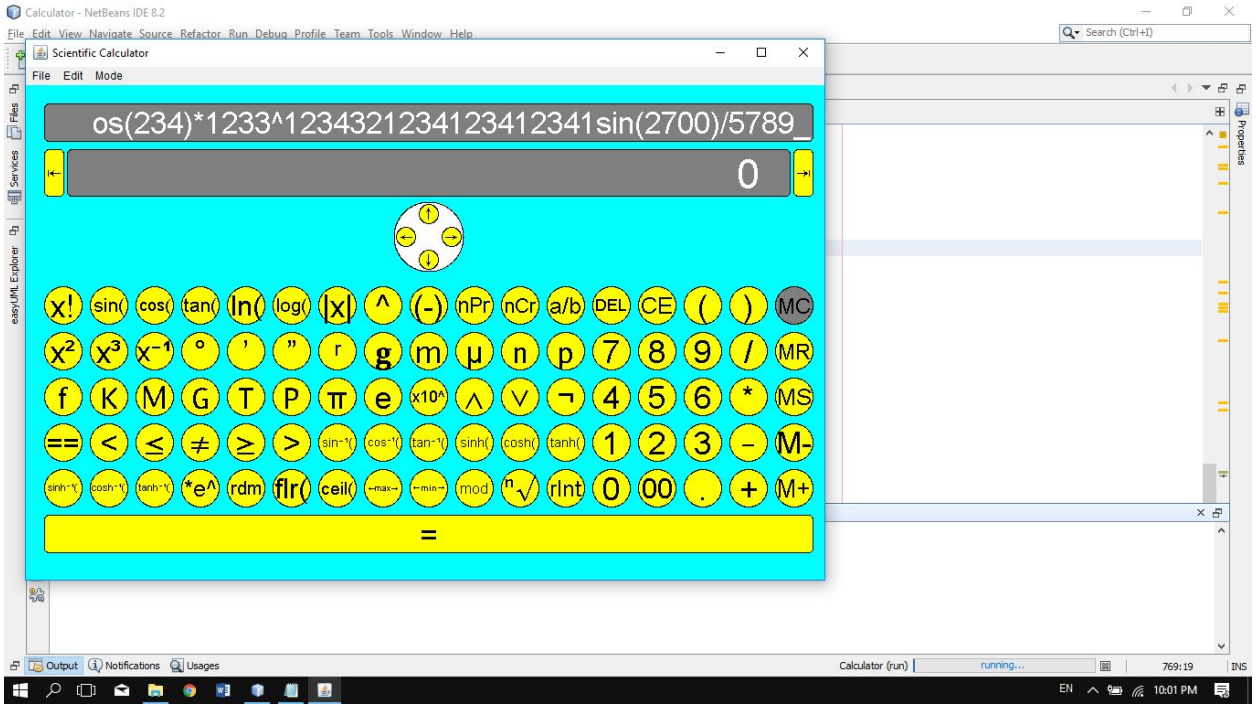


Highlighted text is proof the the calculator uses postfix to evaluate the expression, after converting the input infix to postfix.

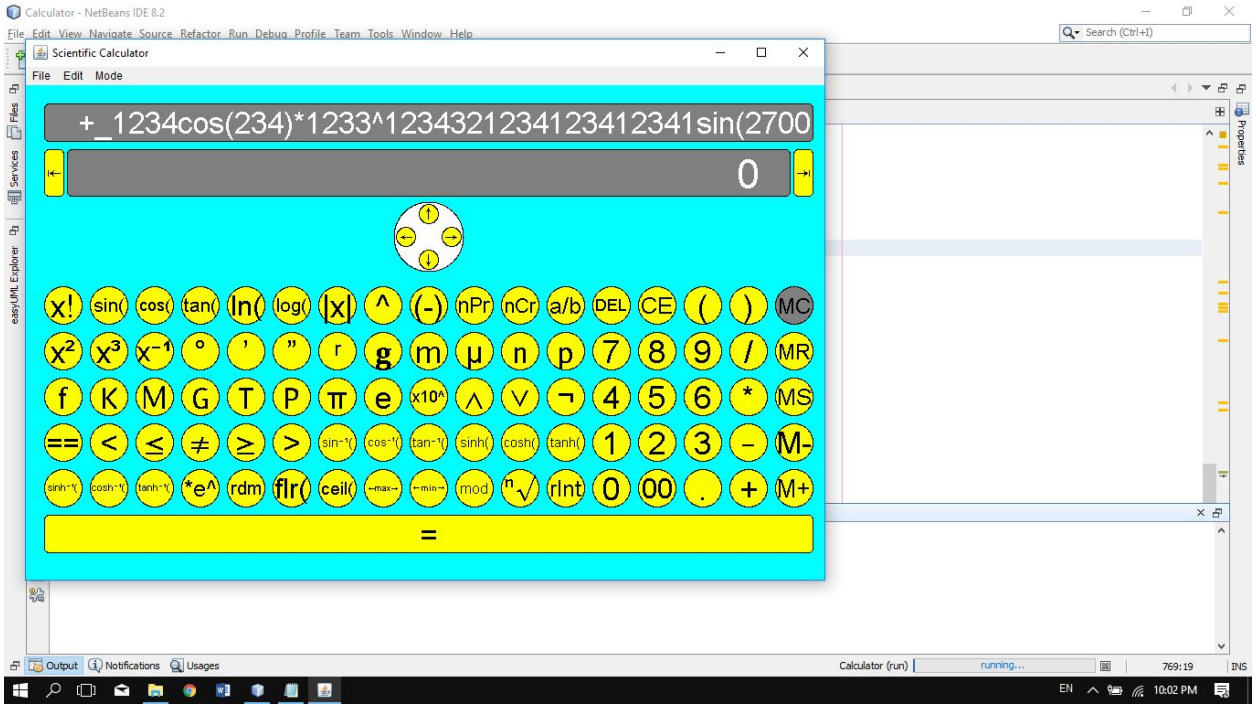


Calculator also features a history replay button that, by default, is set to remember the last 20 input strings. Note the output is set to 0, since it is still waiting for which input to evaluate.

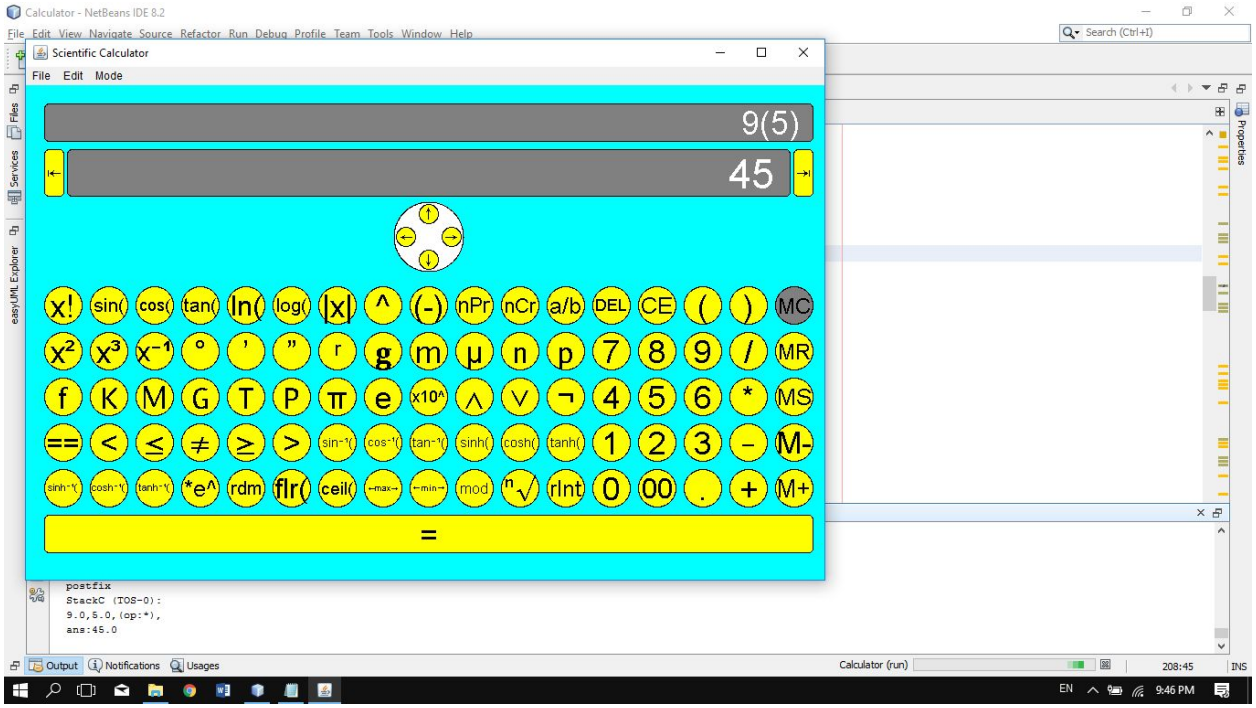
Calculator also features a cursor, which controls where the input is placed, or which character is deleted.



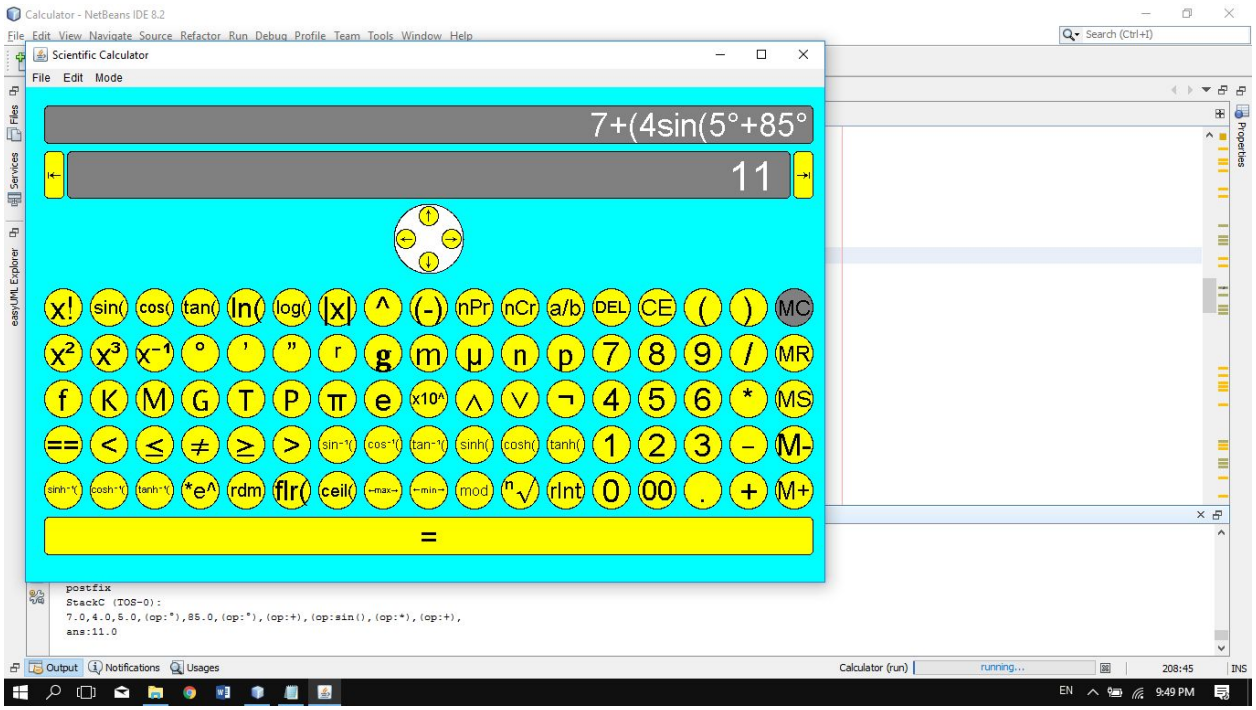
If the text is long, the text is resized a bit. If it is still too long to fit, the display then shows only a part of the input.



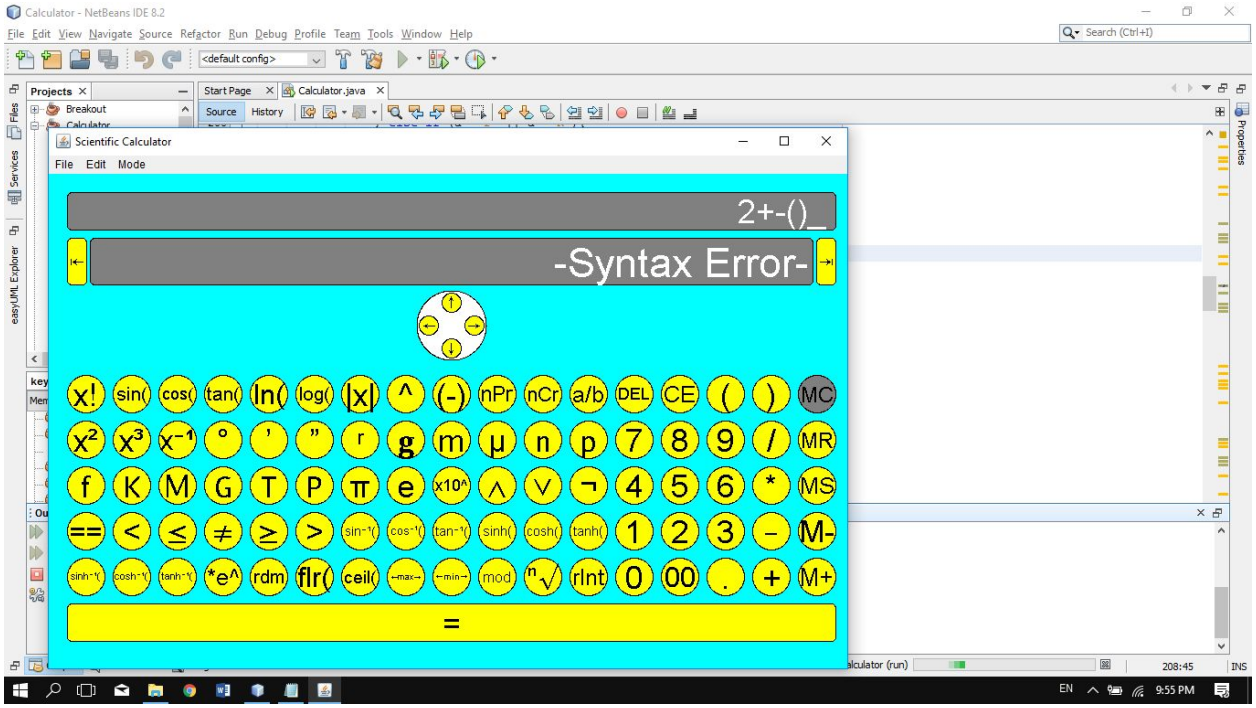
This part of the input shown is easily navigable via cursor position, like a physical scientific calculator.



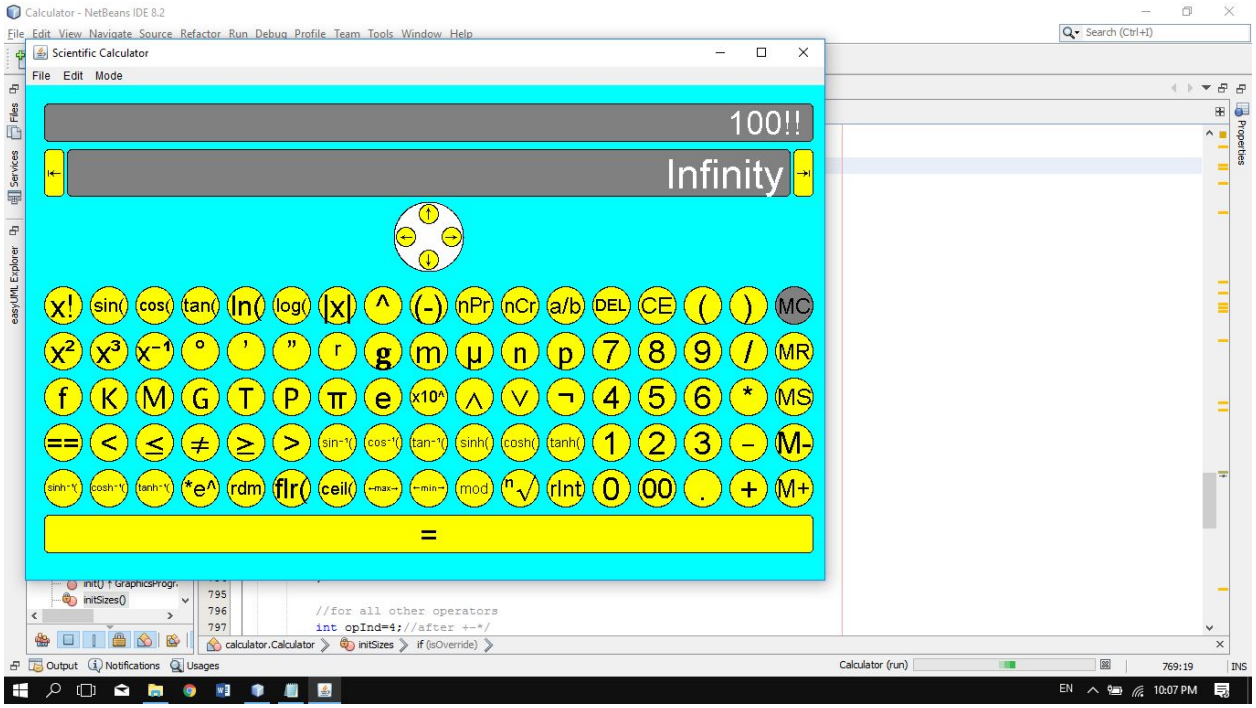
If no operation is placed, the calculator automatically infers it as multiplication.



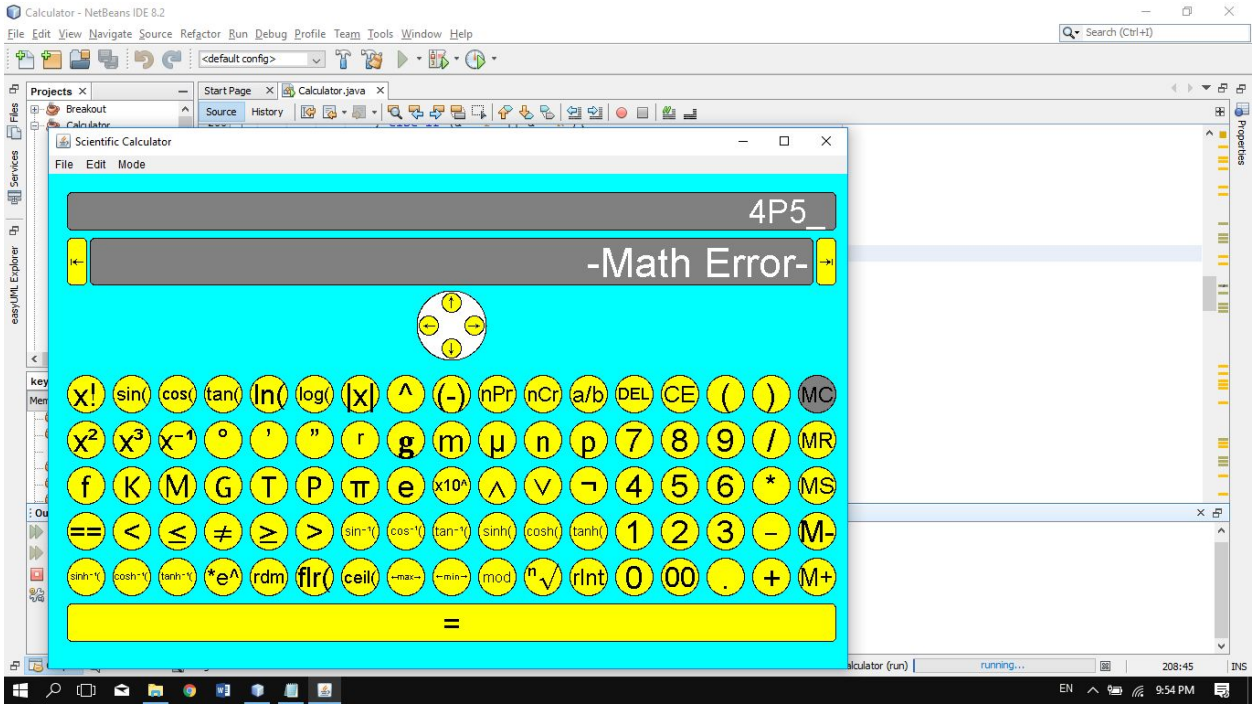
Ending closing parentheses may be not placed, and calculator can still understand the expression.



Incorrect syntax leads to Syntax Error

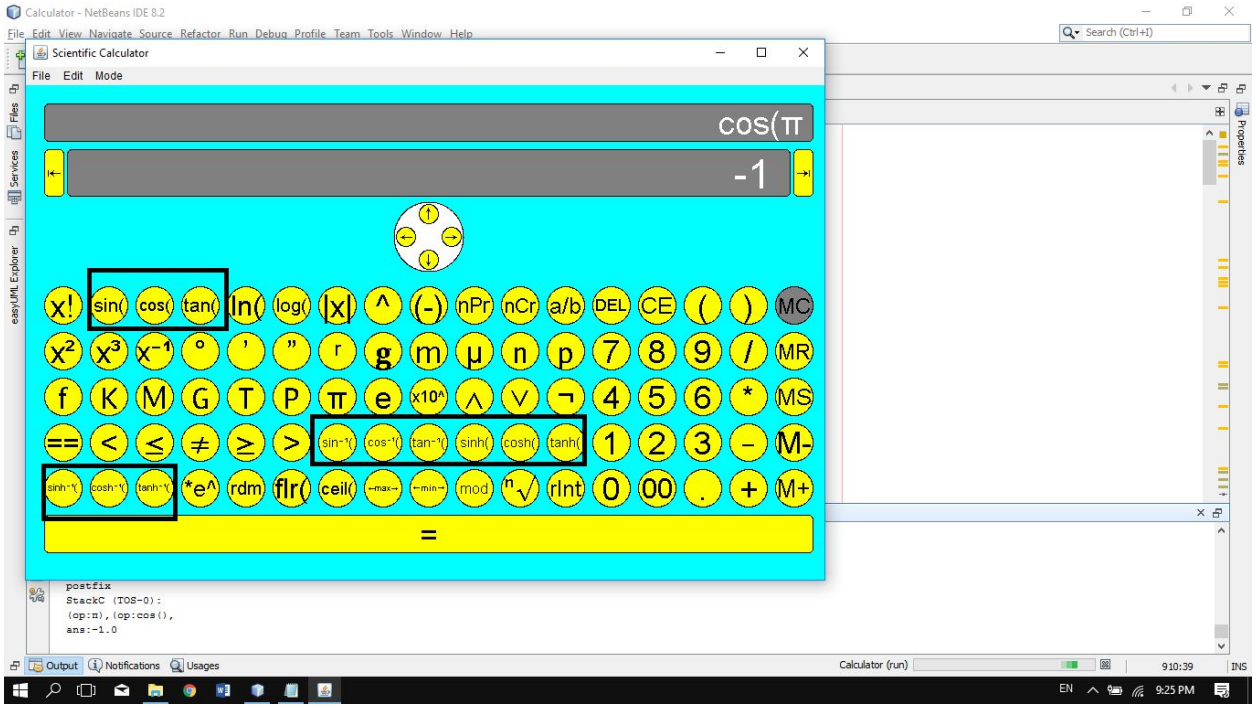


If the value is too high, Infinity is shown

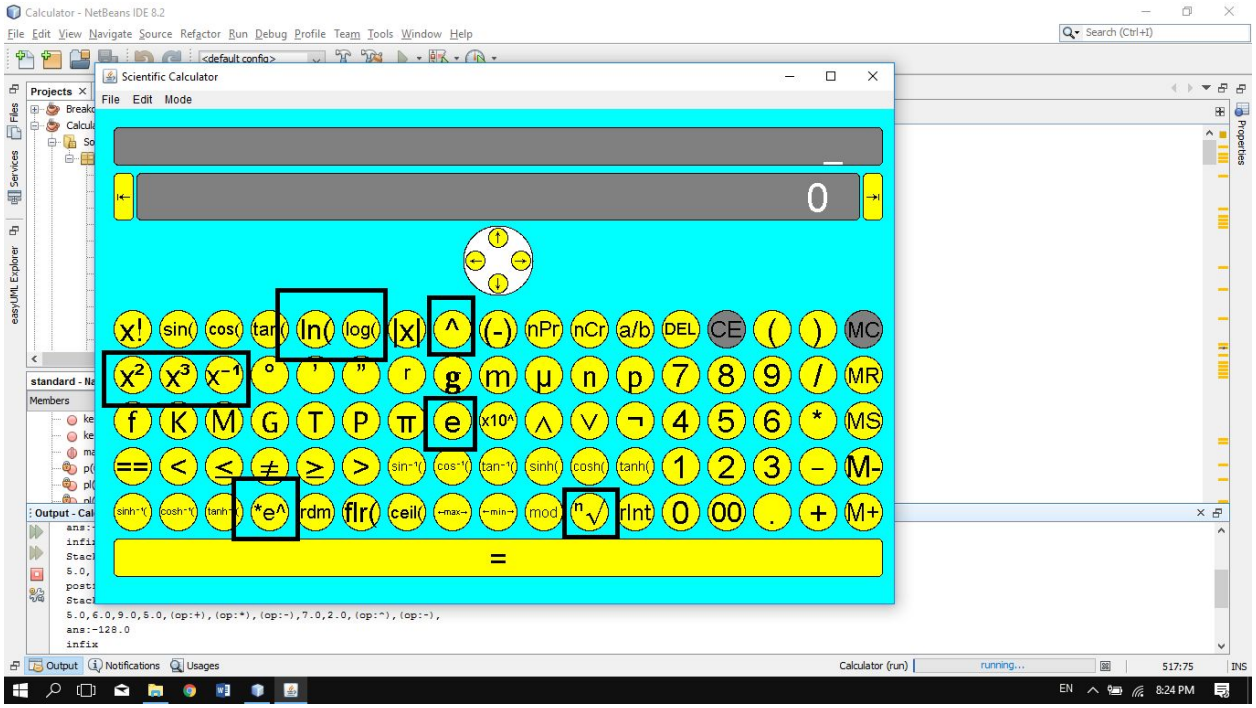


Incorrect parameters with correct syntax also result in a Math Error

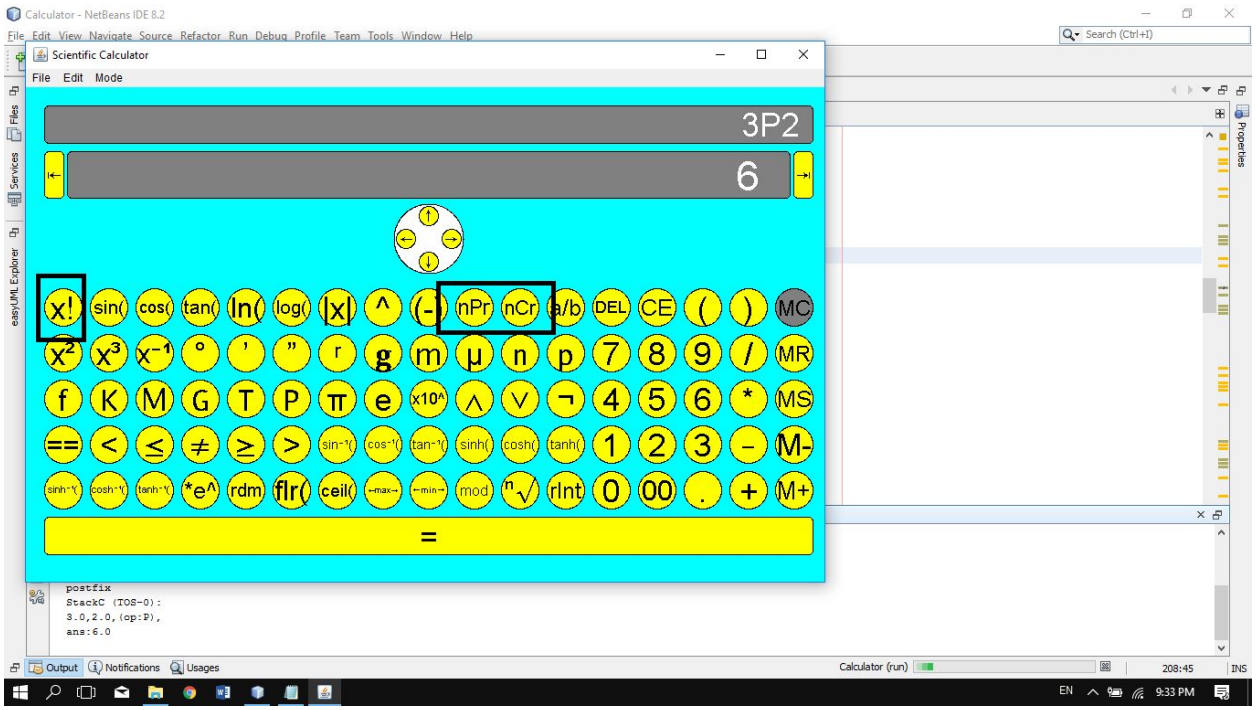
Calculator features the following operators:



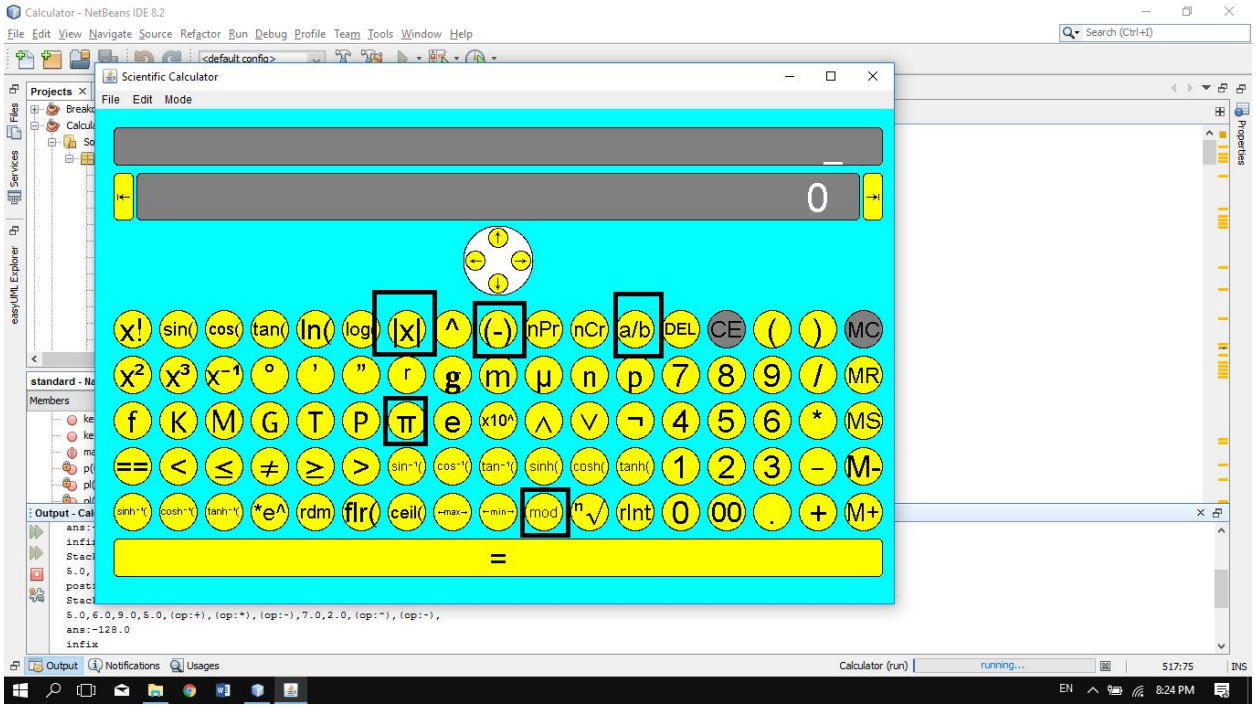
Trigonometric (including hyperbolic and inverses)
[Note: here, the default for trigonometric functions is radians]



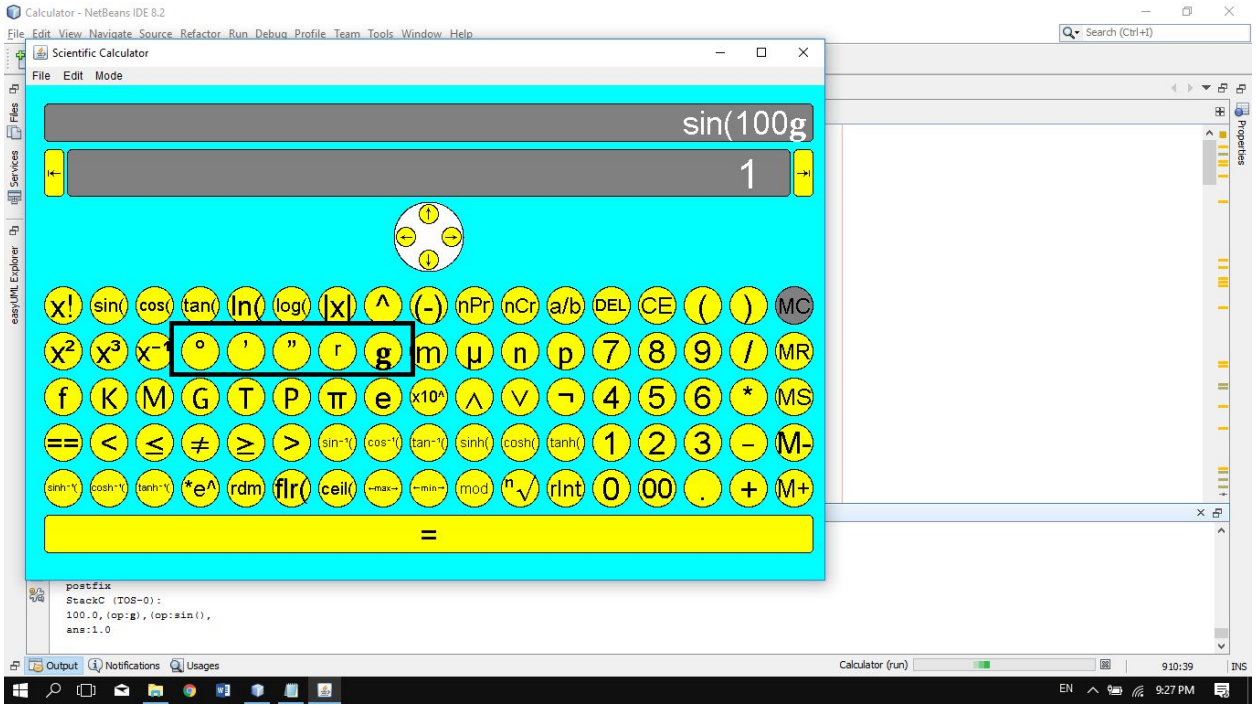
Exponential, logarithmic, root, functions related to e (Euler's number)



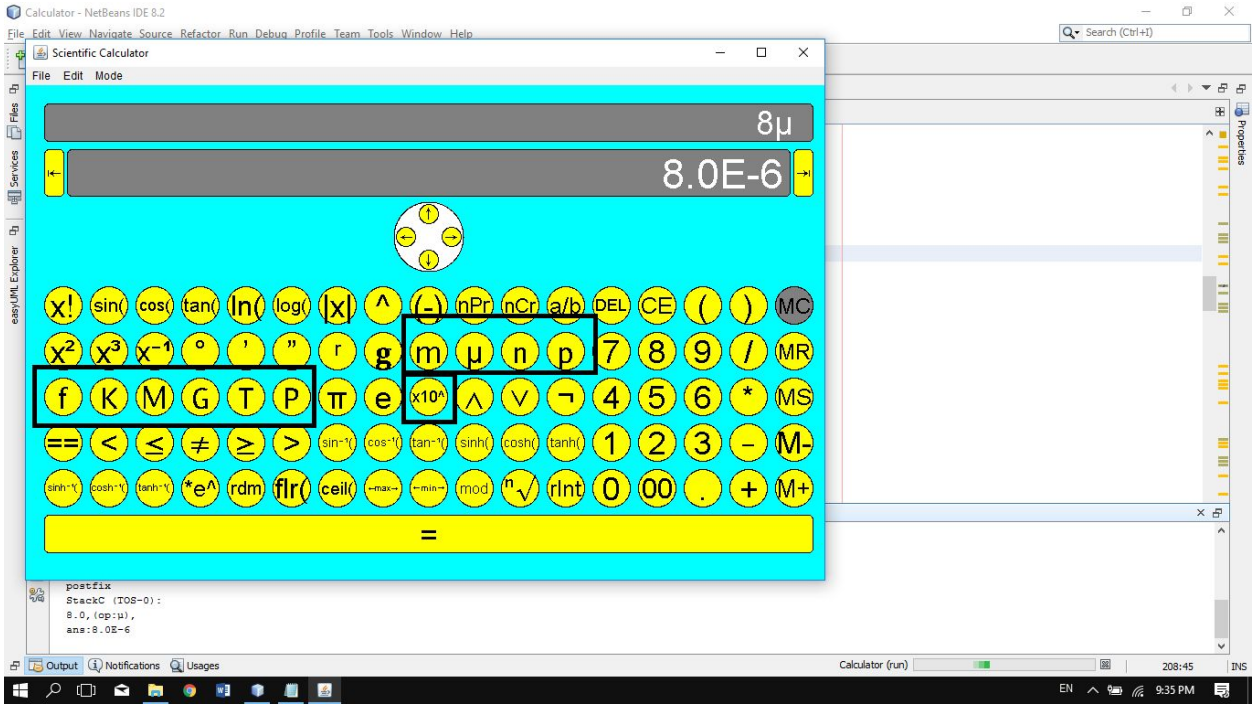
Factorial, Permutation, Combination



Miscellaneous Operators [(-) represents negative sign]



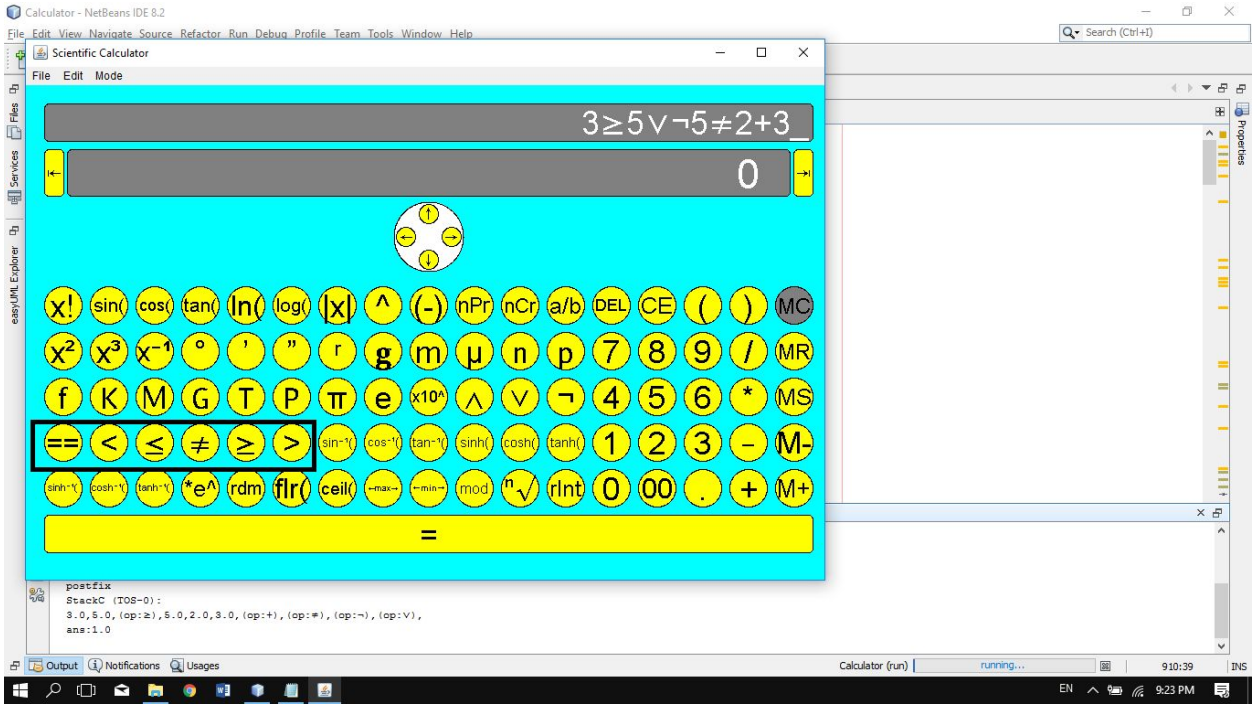
Unit suffixes for angle operations [g--gradients]



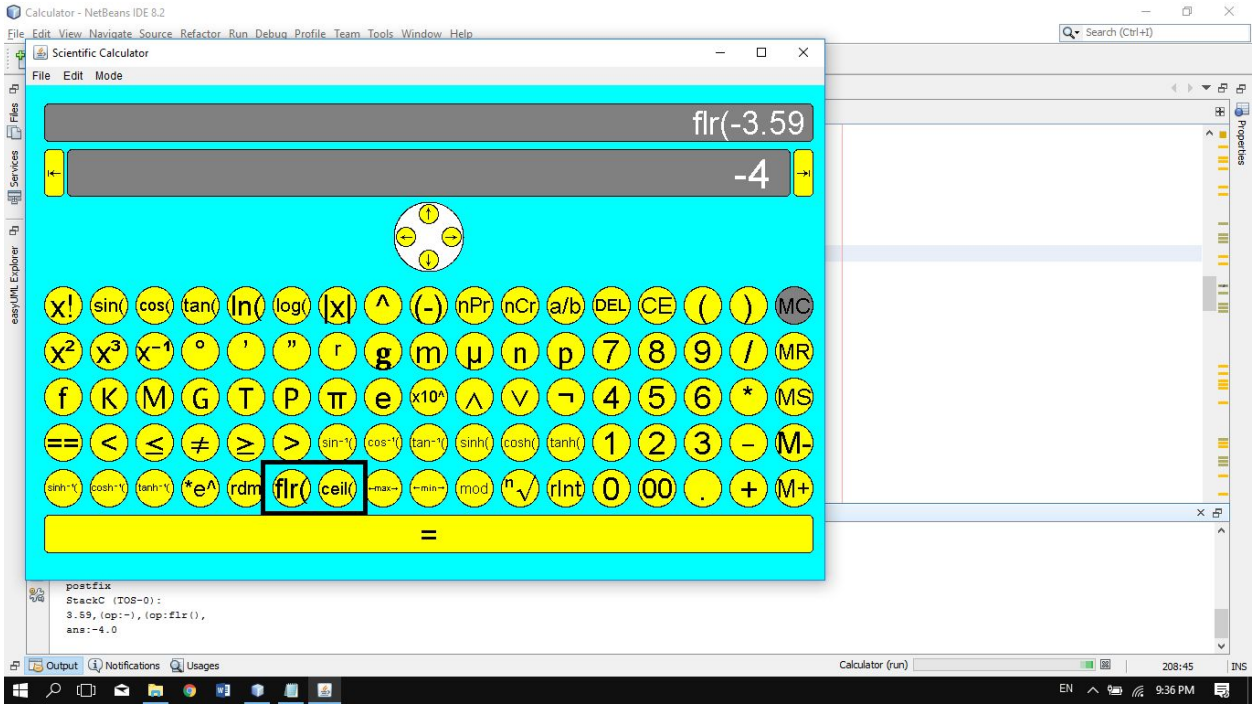
Engineering symbols, and scientific notation

Logical Operators

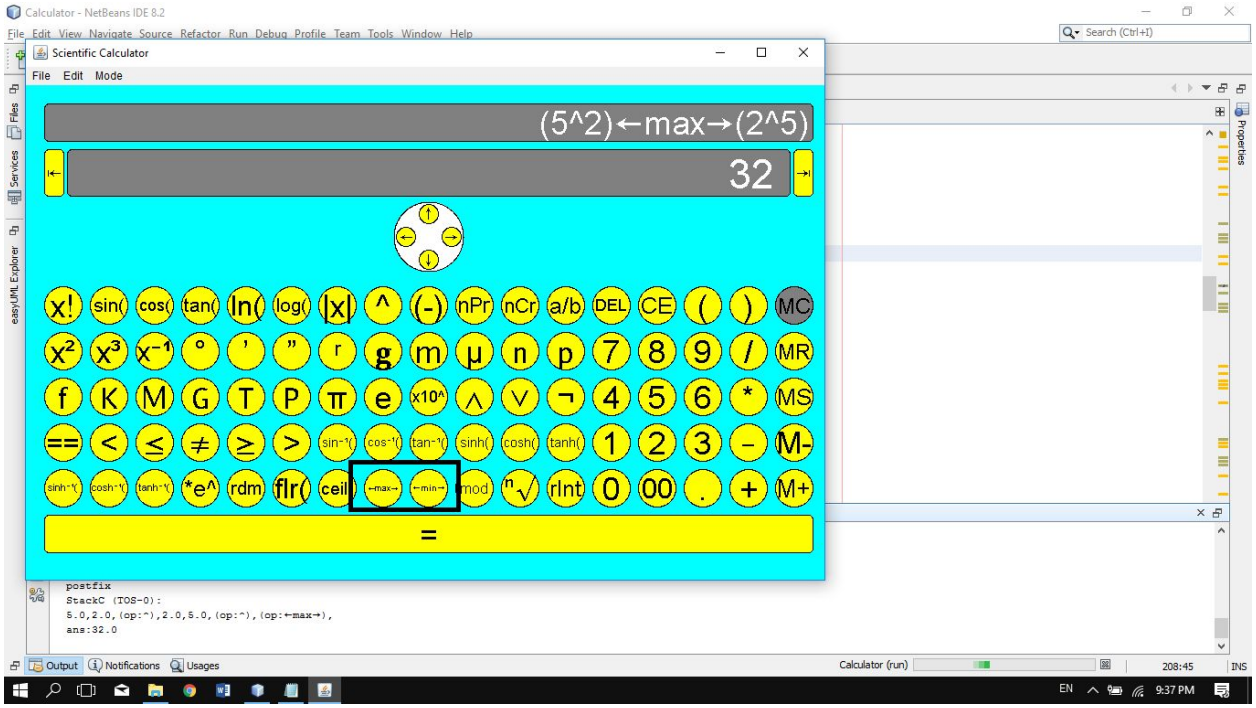
[anything between -1 to 1 exclusive is OFF (false), and anything else is ON(true), like in logic circuits] [The result is shown as 1(ON/true) or 0(OFF/false)]



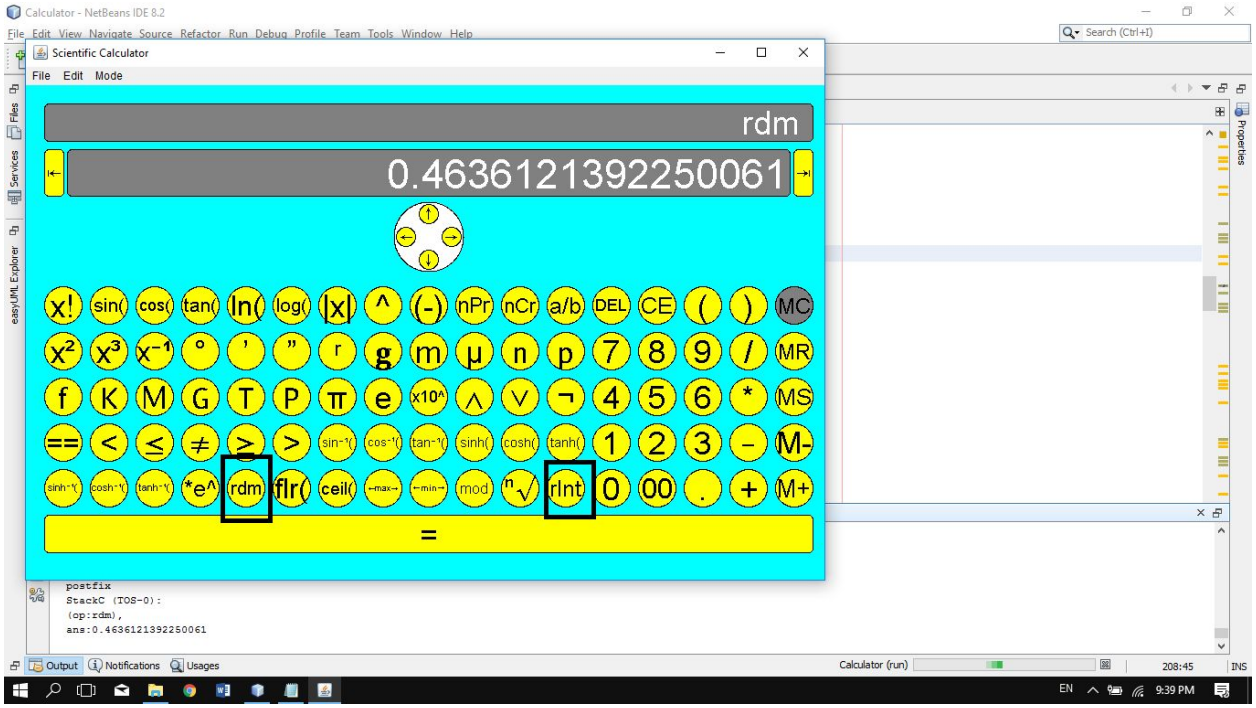
Relational Operators
[These give out 1 if true, 0 if false]



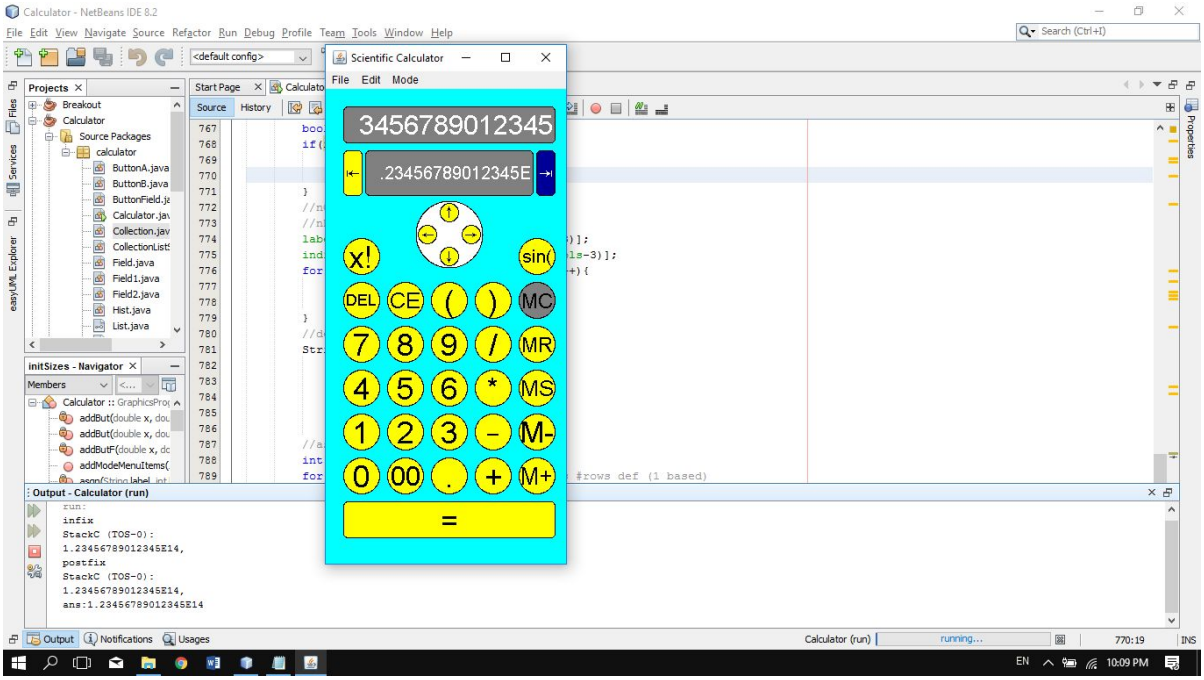
Floor and Ceiling functions



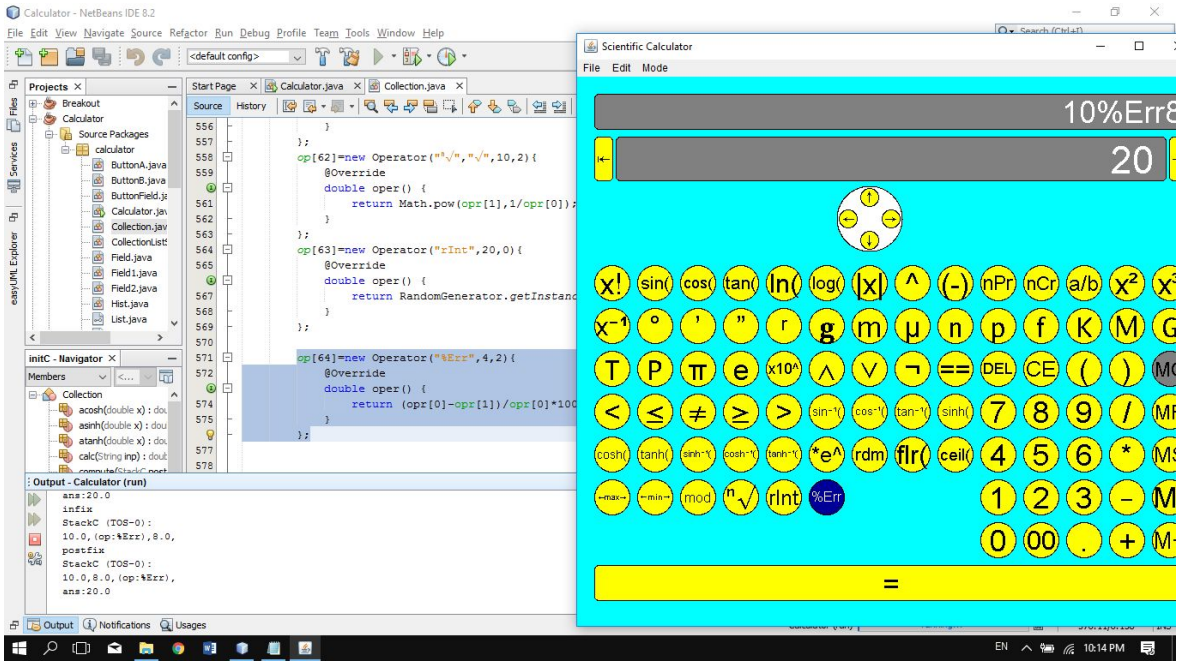
MAX and MIN operations



Random Number Generators



Here, the program was temporarily sized smaller to illustrate the scrolling text functionality of the bottom display field.



EXTRA: The code for the program itself features an application auto resize based on the number of added operators (which can be overridden), and quick addition of new operators, for quick expansion.

For example, if I wanted to add a %Err operation, by simply adding the highlighted text to the code(which specifies its precedence and required number of operands), and removing the sizing override, my calculator now has a functional additional key to it.

SUMMARY

This activity is a tough and exciting learning experience that integrates both our lesson in Stack ADT and in infix and postfix notations. It was quite fitting, since a calculator is one of the major applications of a Stack Data Type, and is used in everyday life. In this activity, even though a calculator seemed to be a very simple device, as it turns out, it actually involves a whole lot of coding/algorithms in its operations, depending on the type of calculator to be made.

In doing the activity, the first problem I encountered, was how to let the program know which buttons the user clicked. Unlike past activities, which only needed to track mouse motion, or used a library that laid out the graphics and a channel for communication for me, this time, I was on my own. In order to get past this, my workaround was to create a new button class, so that each button can sense whether it is clicked, on its own. I did this instead of adding mouse listeners to the whole canvas and getting the location of the mouse click, so that if I move my buttons in the display, I don't have to worry about my mouse listener code stop functioning. Now, all I needed was a common "stream" that I could pass from my main class to each instance of the button object and monitor, so that the program would know whenever a button is pressed.

Actually, creating the button class was my first step towards learning a bit more about java classes, since, as I progressed through the activity, I found it easier to track and more organized to create a separate class to handle the calculator display field, and simply add it to the canvas as one single object. Now, as I thought that postfix evaluating calculators are hard for a user to appreciate with a single display, I wanted to create a scientific mode that featured two displays, so that the user can review and edit his/her entries as a whole argument, then view it alongside the result.

In converting infix to postfix, I made use of what I have learned in my DATASAL class, and it uses an output stack, a "temporary" operator stack, and the input, parsed into a stack. This parsing was what made it possible for multiple digit and floating inputs in the calculator. In looking at the input stack, aside from simply looking at the precedence of the incoming non-numerical input, I also found out that I needed to watch out for parentheses and implied multiplications. As I passed the handling of implied multiplication to the parsing of the input string, which simply added a "*" operator to the input stack whenever appropriate, the parentheses needed an additional conditional statement in the conversion, in which open parentheses "(" were to be pushed into the

operator stack, which would separate each succeeding operator in the stack from the rest, until a closing parenthesis ")" would be encountered, which would signal the popping of all operators from the stack until an opening parenthesis "(" was encountered.

REFERENCES

1. E Roberts. *Art and Science of Java*. Pearson; 2013.

APPENDIX

The code for the program consists of multiple classes, and the package is included in the same Google Drive folder as this document. The link to the package is as follows: https://drive.google.com/drive/folders/0B_ngwdgBk09tWklJRzRHaUZpS0E?usp=sharing.

Here is a copy of the code in all classes:

A) Calculator.java

```
package ph.edu.dlsu.chan.calculator;
import acm.graphics.*;
import acm.io.*;
import acm.program.*;
import acm.util.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javafx.scene.input.MouseEvent;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
/**@author Patrick Matthew Chan*/
public class Calculator extends GraphicsProgram{
   //for debugging only
   public static void p(Object a){//debug
      System.out.print(a+"");
   }
   public static void pl(Object a){//debug
```

```
    System.out.println(a+"");
}
public static void pl(){//debug
    System.out.println();
}
//necessary
//public static final int APPLICATION_WIDTH=50;
//public static final int APPLICATION_HEIGHT=210;
private GCanvas canvas = getGCanvas();
public static void main(String[] args) {
        new Calculator().start(args);
}
public void init(){
    //menu bar--add mode
    JMenu modeMenu = new JMenu("Mode");
    modeMenu.setMnemonic('M');
    addModeMenuItems(modeMenu);
    this.getMenuBar().add(modeMenu);
    //default
    isStd=true;
    isModeChanged=true;
}
public void run() {
    while(true){
        if(isStd){//std
            if(isModeChanged){
                stdSetup();
                isModeChanged=false;
            }
            standard();
        } else {//sci
            if(isModeChanged){
                sciSetup();
                isModeChanged=false;
            }
            scientific();
        }
    }
}



//////////////////////MENU BAR++//////////////////////////
private boolean isStd=true;//true=scientific
private boolean isModeChanged=false;//if just changed
public void addModeMenuItems(JMenu menu){
    JMenuItem m1 = new JMenuItem("Standard");
```

```
    m1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if(!isStd){
                isStd=true;
                isModeChanged=true;
            }
        }
    });
    JMenuItem m2 = new JMenuItem("Scientific");
    m2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if(isStd){
                isStd=false;
                isModeChanged=true;
            }
        }
    });
    menu.add(m1);
    menu.add(m2);
}
////////////////////////////////////////////////////////////




////////////--METHODS FOR GENERAL USAGE--//////////////////////
//++ for search
public static int searchArray(String query,String[] array){//returns index
    for(int i=0;i!=array.length;i++){
        if(array[i].equals(query)){
            return i;
        }
    }
    return -1;//not found
}
public static int searchArray(char query,String[] array){//returns index
    for(int i=0;i!=array.length;i++){
        if(array[i].length()==1 && array[i].charAt(0)==query){
            return i;
        }
    }
    return -1;//not found
}
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
```

```
//////////////~~~~KEY PRESS: SHARED USAGE~~~~//////////////////
//++ stdkey Press
public void keyPressed(KeyEvent e){
   if(isStd){
      Character a=e.getKeyChar();
      int i=searchArray(a,stdlabels);
      if(i!=-1){
         stdkey[i].pressed();
      } else {
         if(a=='='||a=='\n'||a=='\r'){
            stdkeyEquals.pressed();
         } else if (a=='\b' || a==(char)127){
            i=searchArray("DEL",stdlabels);
            stdkey[i].pressed();
         } else if (a=='c' || a=='C'){
            stdkey[stdkeyCEIndex].pressed();
         } else if (a=='a' || a=='A'){
            i=searchArray("AC",stdlabels);
            stdkey[i].pressed();
         } else if (a=='m' || a=='M'){
            i=searchArray("M+",stdlabels);
            stdkey[i].pressed();
         } else if (a=='r' || a=='R'){
            i=searchArray("MR",stdlabels);
            stdkey[i].pressed();
         }
      }
   } else {//sci
      int c=e.getKeyCode();
      Character a=e.getKeyChar();
      int i=searchArray(a,indicators);
      if(i!=-1 && a!='-'){
         key[i].pressed();
      } else {
         if(a=='='||a=='\n'||a=='\r'){
            keyEquals.pressed();
         } else if (a=='\b' || a==(char)127){
            i=searchArray("DEL",labels);
            key[i].pressed();
         } else if (a=='c' || a=='C'){
            key[keyCEIndex].pressed();
         }/* else if (a=='a' || a=='A'){
            i=searchArray("AC",labels);
            key[i].pressed();
         }*/ else if (a=='m' || a=='M'){
            i=searchArray("M+",labels);
            key[i].pressed();
```

```
            } else if (a=='r' || a=='R'){
                i=searchArray("MR",labels);
                key[i].pressed();
            } else if (a=='-'){
                i=searchArray("–",labels);
                key[i].pressed();
            } else if (c==38){
                keyF1[0].pressed();
            } else if (c==40){
                keyF1[1].pressed();
            } else if (c==37){
                keyF1[2].pressed();
            } else if (c==39){
                keyF1[3].pressed();
            }
        }
    }
}
public void keyReleased(KeyEvent e){
    if(isStd){
        Character a=e.getKeyChar();
        int i=searchArray(a,stdlabels);
        if(i!=-1){
            stdkey[i].released();
        } else {
            if(a=='='||a=='\n'||a=='\r'){
                stdkeyEquals.released();
            } else if (a=='\b' || a==(char)127){//127 is delete
                i=searchArray("DEL",stdlabels);
                stdkey[i].released();
            } else if (a=='c' || a=='C'){
                stdkey[stdkeyCEIndex].released();
            } else if (a=='a' || a=='A'){
                i=searchArray("AC",stdlabels);
                stdkey[i].released();
            } else if (a=='m' || a=='M'){
                i=searchArray("M+",stdlabels);
                stdkey[i].released();
            } else if (a=='r' || a=='R'){
                i=searchArray("MR",stdlabels);
                stdkey[i].released();
            }
        }
    } else {//sci
        int c=e.getKeyCode();
        Character a=e.getKeyChar();
        int i=searchArray(a,indicators);
```

```
        if(i!=-1 && a!='-'){
            key[i].released();
        } else {
            if(a=='='||a=='\n'||a=='\r'){
                keyEquals.released();
            } else if (a=='\b' || a==(char)127){//127 is delete
                i=searchArray("DEL",labels);
                key[i].released();
            } else if (a=='c' || a=='C'){
                key[keyCEIndex].released();
            }/* else if (a=='a' || a=='A'){
                i=searchArray("AC",labels);
                key[i].released();
            }*/ else if (a=='m' || a=='M'){
                i=searchArray("M+",labels);
                key[i].released();
            } else if (a=='r' || a=='R'){
                i=searchArray("MR",labels);
                key[i].released();
            } else if (a=='-'){
                i=searchArray("–",labels);
                key[i].released();
            } else if (c==38){
                keyF1[0].released();
            } else if (c==40){
                keyF1[1].released();
            } else if (c==37){
                keyF1[2].released();
            } else if (c==39){
                keyF1[3].released();
            }
        }
    }
  }
  ////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////
//##########################################################################
##//
//#================== C A L C U L A T O R S ==================#//
//##########################################################################
##//
////////////////////////////////////////////////////////////////////////////


  ////////////////////////////////////////////////////////////////
```

```
   //----------------------STANDARD CALCULATOR-------------------------//
   ////////////////////////////////////////////////////////////////////
   //-------------------VARS
   //global vars--std
   public Stack<String> stdfeed=new Stack<>(5);//need 1, 5 to be safe
   //buttons
   private final double STD_FIELD_HEIGHT = 90;
   private final double STD_BUTTON_RADIUS = 30;
   private final double STD_MARGIN_X =20;
   private final double STD_MARGIN_Y =20;
   private final double STD_BUTTON_X_SPACING =15;
   private final double STD_BUTTON_Y_SPACING
=15;//(getWidth()-2*STD_NCOLS*STD_BUTTON_RADIUS)/(STD_NCOLS+1);
   private final double STD_BUTTON_Y_START=
STD_MARGIN_Y+STD_FIELD_HEIGHT+STD_BUTTON_Y_SPACING;
   private static final int STD_NCOLS = 6;
   private static final int STD_NROWS = 5;
      //button stdlabels: (row by row)
      String stdlabels[]=new String[] {
         "log","+/-","DEL","CE","AC","MC",
         "sin","7","8","9","/","MR",
         "cos","4","5","6","*","MS",
         "tan","1","2","3","-","M-",
         "^2","0","00",".","+","M+"};
   int stdkeyDotIndex=0;//special purpose
   int stdkeyCEIndex=0;//special purpose
   int stdkeyMCIndex=0;//special purpose
   int stdkeyOneInput[]=new int[5];//sp purpose
   private ButtonA stdkey[]=new ButtonA[STD_NCOLS*STD_NROWS];
   private ButtonB stdkeyEquals;
   //stdButtons
   private ButtonA stdaddBut(double x,double y,String label){
      ButtonA temp=new ButtonA(label,STD_BUTTON_RADIUS,label,stdfeed);
      add(temp,x,y);
      return temp;
   }
   private ButtonA stdaddBut(double x,double y,String label,String indicator){
      ButtonA temp=new ButtonA(label,STD_BUTTON_RADIUS,indicator,stdfeed);
      add(temp,x,y);
      return temp;
   }
   //stdfield
   private Field stdfield=new
Field((STD_BUTTON_X_SPACING+2*STD_BUTTON_RADIUS)
            *STD_NCOLS-STD_BUTTON_X_SPACING,STD_FIELD_HEIGHT);
   //methods
   private void stddrawButtons(){
```

```
      //NOT EQUALS
      int labelsIndex=0;//for the loop
      for(int i=1;i<=STD_NROWS;i++){
         for(int j=1;j<=STD_NCOLS;j++){
            /*if(stdlabels[labelsIndex]=="."){
               stdkeyDotIndex=labelsIndex;
            }*/
            stdkey[labelsIndex]=stdaddBut(STD_MARGIN_X+(STD_BUTTON_X_SPACING

+2*STD_BUTTON_RADIUS)*(j-1),STD_BUTTON_Y_START+(STD_BUTTON_Y_SPACING
                  +2*STD_BUTTON_RADIUS)*(i-1),stdlabels[labelsIndex]);
            labelsIndex++;
         }
      }
      stdkeyDotIndex=searchArray(".",stdlabels);
      stdkeyCEIndex=searchArray("CE",stdlabels);
      stdkeyMCIndex=searchArray("MC",stdlabels);
      stdkeyOneInput[0]=searchArray("log",stdlabels);
      stdkeyOneInput[1]=searchArray("sin",stdlabels);
      stdkeyOneInput[2]=searchArray("cos",stdlabels);
      stdkeyOneInput[3]=searchArray("tan",stdlabels);
      stdkeyOneInput[4]=searchArray("^2",stdlabels);
      //EQUALS
      //this.stdaddBut(100, 0, "Hi", stdkeyEquals);
      stdkeyEquals=new
ButtonB("=",(STD_BUTTON_X_SPACING+2*STD_BUTTON_RADIUS)
            *STD_NCOLS-STD_BUTTON_X_SPACING,2*STD_BUTTON_RADIUS,"=",
            stdfeed);

add(stdkeyEquals,STD_MARGIN_X,STD_BUTTON_Y_START+(STD_BUTTON_Y_SPACING
            +2*STD_BUTTON_RADIUS)*(STD_NROWS));

   }
   private void stddrawField(){
      add(stdfield,STD_MARGIN_X,STD_MARGIN_Y);
   }

   //variables--calculator
   private Stack<Double> stdnumStack=new Stack<>(5);//need only 2, 5 to be safe
   private Stack<String> stdopStack=new Stack<>(5);//need only 2, 5 to be safe
   private double stdoper(String op,double x,double y){//op is stdoperator
      double ans;
      if(op=="+"){
         ans=x+y;
      } else if(op=="-"){
         ans=x-y;
      } else if(op=="*"){
```

```
            ans=x*y;
        } else if(op=="/"){
            ans=x/y;
        } else{
            throw new RuntimeException("Unrecognized Operator:"+op);
        }
        return ans;
    }
    //for continuous stdoperations "1+,=,=,=,=,=,..."
    private Stack<Double> stdnumHist=new Stack<>(5);//need only 2, 5 to be safe
    private Stack<String> stdopHist=new Stack<>(5);//need only 2, 5 to be safe
    private void stdhistClr(){//clears history
        stdnumHist.clear();
        stdopHist.clear();
    }
    private void stdhistSto(double num,String op){//updates history
        stdhistClr();
        stdnumHist.push(num);
        stdopHist.push(op);
    }
    private boolean stdisClrHist(){
        return (stdnumHist.isEmpty() && stdopHist.isEmpty());
    }




    //-------------------SETUP
    private void stdSetup(){
        canvas.removeKeyListener(this);
        removeAll();

setSize((int)(2*STD_MARGIN_X+(STD_BUTTON_X_SPACING+2*STD_BUTTON_RADIUS)*S
TD_NCOLS),

(int)(60+STD_BUTTON_Y_START+(STD_BUTTON_Y_SPACING+2*STD_BUTTON_RADIUS)*
                    (STD_NROWS+1)+STD_MARGIN_Y));//60 is from title & menu bar
        this.setBackground(Color.CYAN);
        stddrawButtons();
        stddrawField();
        stdfield.show("   ~Hello!~   ");
        canvas.addKeyListener(this);
        this.setTitle("Standard Calculator");
        pause(250);
        stdfield.clr();
    }
    //-------------------LOOP
```

```
double stdM=0;//calc M
   boolean stdisMS=false;
   boolean stdisMp=false;
   boolean stdisMm=false;
private void standard(){
   //while(true){
      //debug //pl("Label:"+stdfield.getLabel());
      stdkey[stdkeyDotIndex].setEnabled(!stdfield.isDot());
      stdkey[stdkeyCEIndex].setEnabled(!(stdfield.isNoInput));
      boolean isNotChoosingOperator=(stdopStack.isEmpty()||!(stdfield.isNoInput));
      for(int i: stdkeyOneInput){//enhanced for; java 5
         stdkey[i].setEnabled(isNotChoosingOperator);
      }
      stdkey[stdkeyMCIndex].setEnabled(stdM!=0);
      stdfield.setM(stdM);
      //when button is pressed
      pause(1);//stability
      if(stdfeed.size()>0){
         String a=stdfeed.top();
         stdfeed.pop();
         int len=a.length();
         char o='\0';//first char
         if(len==1){
            o=a.charAt(0);
         }    //pl("stdfeed:"+a+",o="+o);//debug

         /*
         Notes:
         >Store to stdnumStack whenever stdoperator is pressed,
            then store stdoperator to stdopStack,
            then store these to hist
         >"=" does not store to stdnumStack,stdopStack anymore
         (numstack is simply used to store nos used for stdoperations,
         for computations done later)
         */

         //M Operations
         if(a.charAt(0)=='M'){
            char bb=a.charAt(1);
            if(bb=='C'){
               stdM=0;
               stdfield.result(stdM);
               stdfield.isNoInput=false;
            }else if(bb=='R'){
               stdfield.result(stdM);
               stdfield.isNoInput=false;
            }else if(bb=='S'){
```

```
            a="=";
            o='=';
            stdisMS=true;
        }else if(bb=='-'){
            a="=";
            o='=';
            stdisMm=true;
        }else if(bb=='+'){
            a="=";
            o='=';
            stdisMp=true;
        }
    }



    //isEq
    if(a=="="){
        if(stdopStack.isEmpty()){//no stdoperation, num input only
            if(!(stdisClrHist()||stdisMS||stdisMp||stdisMm)){//chain of =,=,=,=,...
                String op=stdopHist.top();//stdoperator
                stdopHist.pop();
                double x=stdfield.out();//*this is x instead,y from hist
                double y;
                if(stdnumHist.isEmpty()){//"2+="means 2+2=
                    y=0;//should not occur (just for "safety")
                } else {
                    y=stdnumHist.top();
                    stdnumHist.pop();
                }
                double ans=stdoper(op,x,y);
                stdfield.result(ans);
                //store to history
                stdhistSto(y,op);
                //stack stdoperations
                stdnumStack.clear();
                stdopStack.clear();
            } else {
                stdfield.result(stdfield.out());
                stdhistClr();
            }
        } else {
            String op=stdopStack.top();//stdoperator
            stdopStack.pop();
            double y=stdfield.out();//equals uses this directly
            double x;
```

```
            double ans;
            if(stdisMS||stdisMp||stdisMm){//"2-M+"means"2M+"
               x=0;
               ans=y;
            }else{
               if(stdnumStack.isEmpty()){//"2+="means 2+2=
                  x=y;
               } else {
                  x=stdnumStack.top();
                  stdnumStack.pop();
               }
               ans=stdoper(op,x,y);
            }
            stdfield.result(ans);
            //store to history
            stdhistSto(y,op);
            //stack stdoperations
            stdnumStack.clear();
            stdopStack.clear();
         }
         if(stdisMS){
            stdM=stdfield.out();
            stdisMS=false;
         }else if(stdisMp){
            stdM=stdM+stdfield.out();
            stdisMp=false;
         }else if(stdisMm){
            stdM=stdM-stdfield.out();
            stdisMm=false;
         }
      }
      //isNo
      else if((o>='0'&&o<='9')||o=='.'){//one char
         stdfield.concat(o);
      } else if (a=="00"){
         stdfield.concat(a);
      } else if (a=="+/-"){
         stdfield.disp(Field.removeDoubleExcess(stdfield.out()));
         stdfield.neg();
      } else if (a=="CE"){
         stdfield.clr();
         /*if(stdfield.getLabel()==""){//shd be disabled at this pt
            stdnumStack.clear();
            stdopStack.clear();
         }*/
      } else if (a=="DEL"){
         stdfield.del();
```

```
} else if (a=="AC"){
   stdfield.clr();
   stdnumStack.clear();
   stdopStack.clear();
}
//++ single no. stdoperators
else if (a=="log"){
   Double ans=Math.log10(stdfield.out());
   stdfield.result(ans);
   stdfield.isNoInput=false;
}else if (a=="sin"){
   Double ans=Math.sin(Math.toRadians(stdfield.out()));
   stdfield.result(ans);
   stdfield.isNoInput=false;
}else if (a=="cos"){
   Double ans=Math.cos(Math.toRadians(stdfield.out()));
   stdfield.result(ans);
   stdfield.isNoInput=false;
}else if (a=="tan"){
   Double ans=Math.tan(Math.toRadians(stdfield.out()));
   stdfield.result(ans);
   stdfield.isNoInput=false;
}else if (a=="^2"){
   Double ans=Math.pow(stdfield.out(),2);
   stdfield.result(ans);
   stdfield.isNoInput=false;
}
//isOp
else if(a=="+"||a=="-"||a=="*"||a=="/"){
   //if just correcting ("+,*" is *)
   //(!(stdopStack.isEmpty())&&(stdfield.isNoInput))
   if(!isNotChoosingOperator){//is Choosing Operator
      stdopStack.clear();
      stdopStack.push(a);    //pl("correcting stdoperator:"+a);//debug
   } else {
      //first stdoperand
      if(stdopStack.isEmpty()){
         stdnumStack.push(stdfield.out());
         stdhistClr();
         stdopStack.push(a);
         stdfield.setEditable(false);
         //pl("1st stdoperand:"+stdnumStack.top()+a);//debug
      }else{//toph's "i-Chain mo, dali!"
         //(ans immediately pushed back in)
         String op=stdopStack.top();//stdoperator to use
         stdopStack.pop();
         double y=stdfield.out();
```

```
                    double x=stdnumStack.top();
                    stdnumStack.pop();
                    double ans=stdoper(op,x,y);
                    stdfield.result(ans);
                    //store to history
                    stdhistSto(y,op);
                    //stack stdoperations
                    stdnumStack.clear();
                    stdopStack.clear();
                    //ready for next stdoperation a
                    stdnumStack.push(ans);
                    stdopStack.push(a);
                    //pl("stdoperated:"+x+op+y+"="+ans);//debug
                }
            }
        }
    }
  //}
}




//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//




  ////////////////////////////////////////////////////////////////////
  //--------------------SCIENTIFIC CALCULATOR----------------------//
  ////////////////////////////////////////////////////////////////////
  //--------------------VARS
  public Stack<String> feed=new Stack<>(5);//need 1, 5 to be safe
  //buttons
  private final double FIELD1_HEIGHT = 40;
  private final double FIELD_SPACING = 10;
  private final double FIELD2_HEIGHT = 50;
  private final double BUTTON_RADIUS = 20;
  private final double MARGIN_X =20;
  private final double MARGIN_Y =20;
  private final double BUTTON_X_SPACING =10;
  private final double BUTTON_Y_SPACING
=10;//(getWidth()-2*nCols*BUTTON_RADIUS)/(nCols+1);
  private final double BUTTONF1_Y_START= MARGIN_Y+FIELD1_HEIGHT
        +FIELD_SPACING+FIELD2_HEIGHT+BUTTON_Y_SPACING;
```

```
   private final double BUTTON_Y_START=
BUTTONF1_Y_START+3*(BUTTON_RADIUS+BUTTON_Y_SPACING);
   private int nCols = 6;//just an ini. value
   private int nRows = 5;//just an ini. value
   String labels[];
   String indicators[];
   int keyCEIndex=0;//special purpose
   int keyMCIndex=0;//special purpose

   private GOval ovalF1=new GOval((BUTTON_RADIUS+BUTTON_X_SPACING/2)
         *3,(BUTTON_RADIUS+BUTTON_Y_SPACING/2)*3);
   //0-u,1-d,2-l,3-r
   private ButtonA keyF1[]=new ButtonA[4];
   private ButtonA key[];
   private ButtonB keyEquals;
   private ButtonA addButF(double x,double y,String label){
      ButtonA temp=new ButtonA(label,BUTTON_RADIUS/2,label,feed);
      add(temp,x,y);
      return temp;
   }
   private ButtonA addBut(double x,double y,String label){
      ButtonA temp=new ButtonA(label,BUTTON_RADIUS,label,feed);
      add(temp,x,y);
      return temp;
   }
   private ButtonA addBut(double x,double y,String label,String indicator){
      ButtonA temp=new ButtonA(label,BUTTON_RADIUS,indicator,feed);
      add(temp,x,y);
      return temp;
   }
   //field
   private Field1 field1;
   private Field2 field2;
   //methods
   private void drawButtons(){
      //FOR FIELD 1
      int ctrCol=(nCols+1)/2;
      ovalF1.setFilled(true);
      ovalF1.setFillColor(Color.white);
      //ovalF1.setFillColor(this.getBackground().brighter());
      ovalF1.setColor(Color.black);
      add(ovalF1,MARGIN_X+(BUTTON_X_SPACING+2*BUTTON_RADIUS)
            *(ctrCol-1)+BUTTON_RADIUS/2-(BUTTON_X_SPACING)/4
            -BUTTON_RADIUS-BUTTON_X_SPACING/2,BUTTONF1_Y_START
            +(BUTTON_Y_SPACING/2+BUTTON_RADIUS)*(1-1)
            -(BUTTON_Y_SPACING)/4);
```

```
keyF1[0]=addButF(MARGIN_X+(BUTTON_X_SPACING+2*BUTTON_RADIUS)
      *(ctrCol-1)+BUTTON_RADIUS/2,BUTTONF1_Y_START
      +(BUTTON_Y_SPACING/2+BUTTON_RADIUS)*(1-1)," ↑ ");
keyF1[1]=addButF(MARGIN_X+(BUTTON_X_SPACING+2*BUTTON_RADIUS)
      *(ctrCol-1)+BUTTON_RADIUS/2,BUTTONF1_Y_START
      +(BUTTON_Y_SPACING/2+BUTTON_RADIUS)*(3-1)," ↓ ");
keyF1[2]=addButF(MARGIN_X+(BUTTON_X_SPACING+2*BUTTON_RADIUS)
      *(ctrCol-1)+BUTTON_RADIUS/2-BUTTON_X_SPACING/2
      -BUTTON_RADIUS,BUTTONF1_Y_START
      +(BUTTON_Y_SPACING/2+BUTTON_RADIUS)*(2-1),"←");
keyF1[3]=addButF(MARGIN_X+(BUTTON_X_SPACING+2*BUTTON_RADIUS)
      *(ctrCol-1)+BUTTON_RADIUS/2+BUTTON_X_SPACING/2
      +BUTTON_RADIUS,BUTTONF1_Y_START
      +(BUTTON_Y_SPACING/2+BUTTON_RADIUS)*(2-1),"→");


//NOT EQUALS
int labelsIndex=0;//for the loop
for(int i=1;i<=nRows;i++){
   for(int j=1;j<=nCols;j++){
      if(labels[labelsIndex].length()>0){
         key[labelsIndex]=addBut(MARGIN_X+(BUTTON_X_SPACING
               +2*BUTTON_RADIUS)*(j-1),BUTTON_Y_START+(BUTTON_Y_SPACING
               +2*BUTTON_RADIUS)*(i-1),labels[labelsIndex],
               indicators[labelsIndex]);
      }
      labelsIndex++;
   }
}
//addtl buttons on top (1 row)
for(int j=1;j<=nCols;j++){
   if(j<ctrCol-1 || j>ctrCol+1){
      if(labels[labelsIndex].length()>0){
         key[labelsIndex]=addBut(MARGIN_X+(BUTTON_X_SPACING
               +2*BUTTON_RADIUS)*(j-1),BUTTON_Y_START+(BUTTON_Y_SPACING
               +2*BUTTON_RADIUS)*(0-1),labels[labelsIndex],
               indicators[labelsIndex]);
      }
      labelsIndex++;
   }
}
keyCEIndex=searchArray("CE",labels);
keyMCIndex=searchArray("MC",labels);
//EQUALS
//this.addBut(100, 0, "Hi", keyEquals);
keyEquals=new ButtonB("=",(BUTTON_X_SPACING+2*BUTTON_RADIUS)
      *nCols-BUTTON_X_SPACING,2*BUTTON_RADIUS,"=",
```

```
            feed);
    add(keyEquals,MARGIN_X,BUTTON_Y_START+(BUTTON_Y_SPACING
            +2*BUTTON_RADIUS)*(nRows));

}
private void drawField(){
    add(field1,MARGIN_X,MARGIN_Y);
    add(field2,MARGIN_X,MARGIN_Y+FIELD1_HEIGHT
            +FIELD_SPACING);
}
//+++(**asgn is 0 BASED!!**)
private void asgn(String label,int location){
    if(location<labels.length){
        labels[location]=label;
        indicators[location]=label;
    } else {
        pl("asgn-incorrect index:"+location+";;max="+(labels.length-1));
    }
}
private void asgn(String label,String indicator,int location){
    if(location<labels.length){
        labels[location]=label;
        indicators[location]=indicator;
    } else {
        pl("asgn-incorrect index:"+location+";;max="+(labels.length-1));
    }
}
private void asgn(int opIndex,int location){
    if(location<labels.length){
        Operator a=Collection.op[opIndex];
        if(a==null){
            pl("opindex "+opIndex+" is null");
            return;
        }
        labels[location]=a.label;
        indicators[location]=a.ind();
    } else {
        pl("asgn-incorrect index:"+location+";;max="+(labels.length-1));
    }
}
private void initSizes(){
    //opCount
    int opCount=0;//excluding +-*/
    while(Collection.op[opCount+4]!=null){
        opCount++;
    }       //opCount=105;//debug
    if(opCount<=2){
```

```
    nCols=5;
    nRows=5;
} else if(opCount<=12){
    nCols=5;
    nRows=5+(int)Math.ceil((double)(opCount-2)/nCols);
} else {
    nRows=7;
    nCols=5+(int)Math.ceil((double)(opCount-12)/(nRows+1));
}


//override:
boolean isOverride=false;
if(isOverride){
    nCols=17;
    nRows=5;
}
//nCols=5;//min
//nRows=7;//min
labels=new String[nCols*nRows+(nCols-3)];
indicators=new String[nCols*nRows+(nCols-3)];
for(int i=0;i<nCols*nRows+(nCols-3);i++){
    labels[i]="";
    indicators[i]="";
}
//default buttons box (row by row)
String def[]=new String[] {
    "DEL","CE","(",")","MC",
    "7","8","9","/","MR",
    "4","5","6","*","MS",
    "1","2","3","–","M-",
    "0","00",".","+","M+"};
//assign
int defIndex=0;
for(int i=nRows-4;i<=nRows;i++){//5 is #rows def (1 based)
    for(int j=nCols-4;j<=nCols;j++){//5 is #cols def (1 based)
        asgn(def[defIndex],(nCols)*(i-1)+(j-1));
        defIndex++;
    }
}

//for all other operators
int opInd=4;//after +-*/
//pl(Collection.op[opInd].ind());
for(int i=0;(Collection.op[opInd]!=null && i<nCols*nRows+(nCols-3));i++){
    //p("hi");pl(Collection.op[opInd].ind()+"label:"+labels[i]);//debug
    if(labels[i].length()==0){
```

```
            asgn(opInd,i);
            opInd++;
         }
      }
      /*labels=new String[] {
         "DEL","CE","(",")","MC",
         "7","8","9","/","MR",
         "4","5","6","*","MS",
         "1","2","3","−","M-",
         "0","00",".","+","M+","","","};*/









      key=new ButtonA[nCols*nRows+(nCols-3)];
      field1=new Field1((BUTTON_X_SPACING+2*BUTTON_RADIUS)
            *nCols-BUTTON_X_SPACING,FIELD1_HEIGHT);
      field2=new Field2((BUTTON_X_SPACING+2*BUTTON_RADIUS)
            *nCols-BUTTON_X_SPACING,FIELD2_HEIGHT);

   }
   //-------------------SETUP
   public void sciSetup(){
      canvas.removeKeyListener(this);
      removeAll();
      Collection.initC();
      initSizes();
      setSize((int)(2*MARGIN_X+(BUTTON_X_SPACING+2*BUTTON_RADIUS)*nCols),
            (int)(60+BUTTON_Y_START+(BUTTON_Y_SPACING+2*BUTTON_RADIUS)*
                  (nRows+1)+MARGIN_Y));//60 is from title & menu bar
      this.setBackground(Color.CYAN);
      drawButtons();
      drawField();
      field1.show("~Good Day~                ");
      field2.disp("<<<Scrolling Text>>>        ");
      canvas.addKeyListener(this);
      this.setTitle("Scientific Calculator");
      pause(250);
      field1.clr();
      field2.clr();
   }
   //-------------------LOOP
   double M=0;//calc M
      boolean isMS=false;
```

```
    boolean isMp=false;
    boolean isMm=false;
  private void scientific(){
    //while(true){//pl("Label:"+field.getLabel());    //debug
        key[keyCEIndex].setEnabled(!(field1.getText()==""));
        key[keyMCIndex].setEnabled(M!=0);
        pause(1);//stability
        if(feed.size()>0){
            String a=feed.top();
            feed.pop();
            int len=a.length();
            char o='\0';//first char
            if(len==1){
                o=a.charAt(0);
            }    //pl("feed:"+a+",o="+o);//debug


            //M Operations
            if(a.length()>1 && a.charAt(0)=='M'){
                char bb=a.charAt(1);
                if(bb=='C'){
                    M=0;
                    field2.disp(M);
                    a="";
                }else if(bb=='R'){
                    field2.disp(M);
                    field1.inp(Field1.removeDoubleExcess(M));
                    a="";
                }else if(bb=='S'){
                    a="=";
                    o='=';
                    isMS=true;
                }else if(bb=='-'){
                    a="=";
                    o='=';
                    isMm=true;
                }else if(bb=='+'){
                    a="=";
                    o='=';
                    isMp=true;
                }
            }




            //isEq
```

```
if(a=="="){
   boolean isErr=false;
   double ans=0;
   try{
      ans=Collection.calc(field1.out());
   } catch (SyntaxException e){
      field2.disp("-Syntax Error-");
      //p("SYNTAX ERROR:");
      pl(e.getMessage());
      isErr=true;
   } catch (MathException e){
      field2.disp("-Math Error-");
      pl(e.getMessage());
      isErr=true;
   }
   if(!isErr){
      field2.disp(ans);
      field1.save();
      if(isMS){
         M=ans;
         isMS=false;
      }else if(isMp){
         M=M+ans;
         isMp=false;
      }else if(isMm){
         M=M-ans;
         isMm=false;
      }
   }

}
//isF1(disp)buttons
else if(o==' ↑ '){
   field1.U();
   field2.clr();
} else if(o==' ↓ '){
   field1.D();
   field2.clr();
} else if(o=='←'){
   field1.L();
   field2.clr();
} else if(o=='→'){
   field1.R();
   field2.clr();
}
//isInput
else if((o>='0'&&o<='9')||o=='.'){//one char
```

```
                field1.inp(o);
            } else if (a=="00"){
                field1.inp(a);
            } else if (a=="CE"){
                field1.clr();
            } else if (a=="DEL"){
                field1.del();
            } /*else if (a=="AC"){
                field1.clr();
                field2.clr();
            } else if (a=="ON"){
                field1.clr();
                field1.histClr();
                field2.clr();
                field1.disp("");
                field2.disp("");
            }*/ else {//I assume all others are operators
                field1.inp(a);
            }

        }
      //}
   }


}
```

B) ButtonA.java

```
package ph.edu.dlsu.chan.calculator;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
/*@author Patrick*/
public class ButtonA extends GCompound implements MouseListener{
   private boolean isEnabled=true;
   private Font butFont=new Font("Dialog", 0, 12);
   public Color pipColor = new Color(153);
   public Color butColor = Color.yellow;
   public Color labColor = Color.BLACK;
   private GOval frame;
   private double butWidth;
   private double butHeight;
   private Stack<String> output;
```

```
    private String indicator;
    private GLabel label;
    private String text;//button text
    //constructor
    public ButtonA(String text,double radius,String indicatorString,Stack<String>
outputStack)
    {
      //init
      butWidth=radius*2;
      butHeight=radius*2;
      output=outputStack;
      indicator=indicatorString;
      //draw
      this.frame=new GOval(butWidth,butHeight);
      this.frame.setFilled(true);
      this.frame.setColor(Color.black);
      this.frame.setFillColor(butColor);
      this.label = new GLabel("");
      this.label.setFont(butFont);
      this.label.setColor(labColor);
      add(this.frame);
      add(this.label);
      setLabel(text);
      setEnabled(true);
      addMouseListener(this);
    }
    public ButtonA(String text,double width,double height,String
indicatorString,Stack<String> outputStack)
    {
      //init
      butWidth=width;
      butHeight=height;
      output=outputStack;
      indicator=indicatorString;
      //draw
      this.frame=new GOval(width,height);
      this.frame.setFilled(true);
      this.frame.setColor(Color.black);
      this.frame.setFillColor(butColor);
      this.label = new GLabel("");
      this.label.setFont(butFont);
      this.label.setColor(labColor);
      add(this.frame);
      add(this.label);
      setLabel(text);
      setEnabled(true);
      addMouseListener(this);
```

```
    }
    //methods--utility
    private double ctrX(){
        return (this.getWidth()-label.getWidth())/2;
    }
    private double ctrY(){
        return (this.getHeight()-label.getAscent())/2+1;
    }

    //methods--for usage
    public String getLabel(){
        return text;
    }
    public void setLabel(String text)
    {
        this.label.setLabel(text);
        this.text=text;//this to call global var text
        //sizing
        double size=this.label.getFont().getSize();
        double height=this.frame.getHeight();
        double width=this.frame.getWidth();
        double ratio1=size/height;
        double ratio2=this.label.getWidth()/width;
        //System.out.println(ratio1+","+ratio2+","+this.label.getBounds().getWidth());
        int fs;//font size
        if(ratio1<ratio2){
            fs=(int)((this.label.getFont().getSize())/ratio2*0.92);
        } else {
            fs=(int)((this.label.getFont().getSize())/ratio1*0.85);
        }
        this.label.setFont("*-*-"+fs);
        this.label.setLocation(ctrX(),butHeight*0.5+this.label.getFont().getSize()*0.4);//this
is TAE
    }
    public void setEnabled(boolean flag)
    {
        if(isEnabled!=flag){
            this.isEnabled = flag;
            this.frame.setFillColor(this.isEnabled ? butColor : Color.gray);
        }
    }
    //other "settings"
    public void setButtonFont(Font f){
      label.setFont(f);
      butFont=label.getFont();
      setLabel(text);
    }
```

```
    public void setButtonFont(String f){
      label.setFont(f);
      butFont=label.getFont();
      setLabel(text);
    }


    //++
    public void pressed(){
      if (this.isEnabled)
      {
          this.frame.setFillColor(pipColor);
          this.label.setColor(Color.white);
      }
    }
    public void released(){
      if (this.isEnabled)
      {
        this.frame.setFillColor(butColor);
        this.label.setColor(labColor);
        output.push(indicator);
        //System.out.println(indicator);
      }
    }

    //events
    public void mousePressed(MouseEvent e)
    {
      pressed();
    }

    public void mouseReleased(MouseEvent e)
    {
      released();
    }

    public void mouseClicked(MouseEvent e) {}

    public void mouseEntered(MouseEvent e) {}

    public void mouseExited(MouseEvent e) {}
}
```

C) ButtonB.java

```
package ph.edu.dlsu.chan.calculator;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
/*@author Patrick*/
public class ButtonB extends GCompound implements MouseListener{
    private boolean isEnabled=true;
    private Font butFont=new Font("Dialog", 0, 12);
    public Color pipColor = new Color(153);
    public Color butColor = Color.yellow;
    public Color labColor = Color.BLACK;
    private GRoundRect frame;
    private double butWidth;
    private double butHeight;
    private Stack<String> output;
    private String indicator;
    private GLabel label;
    private String text;//button text
    //constructor
    public ButtonB(String text,double radius,String indicatorString,Stack<String>
outputStack)
    {
      //init
      butWidth=radius*2;
      butHeight=radius*2;
      output=outputStack;
      indicator=indicatorString;
      //draw
      this.frame=new GRoundRect(butWidth,butHeight);
      this.frame.setFilled(true);
      this.frame.setColor(Color.black);
      this.frame.setFillColor(butColor);
      this.label = new GLabel("");
      this.label.setFont(butFont);
      this.label.setColor(labColor);
      add(this.frame);
      add(this.label);
      setLabel(text);
      setEnabled(true);
      addMouseListener(this);
    }
    public ButtonB(String text,double width,double height,String
indicatorString,Stack<String> outputStack)
    {
      //init
      butWidth=width;
```

```
  butHeight=height;
  output=outputStack;
  indicator=indicatorString;
  //draw
  this.frame=new GRoundRect(width,height);
  this.frame.setFilled(true);
  this.frame.setColor(Color.black);
  this.frame.setFillColor(butColor);
  this.label = new GLabel("");
  this.label.setFont(butFont);
  this.label.setColor(labColor);
  add(this.frame);
  add(this.label);
  setLabel(text);
  setEnabled(true);
  addMouseListener(this);
}
//methods--utility
private double ctrX(){
   return (this.getWidth()-label.getWidth())/2;
}
private double ctrY(){
   return (this.getHeight()-label.getAscent())/2+1;
}

//methods--for usage
public String getLabel(){
   return text;
}
public void setLabel(String text)
{
   this.label.setLabel(text);
   this.text=text;//this to call global var text
   //sizing
   double size=this.label.getFont().getSize();
   double height=this.frame.getHeight();
   double width=this.frame.getWidth();
   double ratio1=size/height;
   double ratio2=this.label.getWidth()/width;
   //System.out.println(ratio1+","+ratio2+","+this.label.getBounds().getWidth());
   int fs;//font size
   if(ratio1<ratio2){
      fs=(int)((this.label.getFont().getSize())/ratio2*0.92);
   } else {
      fs=(int)((this.label.getFont().getSize())/ratio1*0.85);
   }
   this.label.setFont("*-*-"+fs);
```

```
      this.label.setLocation(ctrX(),butHeight*0.5+this.label.getFont().getSize()*0.4);//this
is TAE
   }
   public void setEnabled(boolean flag)
   {
    this.isEnabled = flag;
    this.label.setColor(this.isEnabled ? labColor : Color.gray);
   }
   //other "settings"
   public void setButtonFont(Font f){
    label.setFont(f);
    butFont=label.getFont();
    setLabel(text);
   }
   public void setButtonFont(String f){
    label.setFont(f);
    butFont=label.getFont();
    setLabel(text);
   }



   //++
   public void pressed(){
    if (this.isEnabled)
    {
       this.frame.setFillColor(pipColor);
       this.label.setColor(Color.white);
    }
   }
   public void released(){
    if (this.isEnabled)
    {
     this.frame.setFillColor(butColor);
     this.label.setColor(labColor);
     output.push(indicator);
     //System.out.println(indicator);
    }
   }

   //events
   public void mousePressed(MouseEvent e)
   {
    pressed();
   }

   public void mouseReleased(MouseEvent e)
```

```
    {
      released();
    }

    public void mouseClicked(MouseEvent e) {}

    public void mouseEntered(MouseEvent e) {}

    public void mouseExited(MouseEvent e) {}
}
```

D) ButtonField.java

```
package ph.edu.dlsu.chan.calculator;
import acm.graphics.*;
import java.awt.*;
/*@author Patrick*/
public class ButtonField extends GCompound{
    private boolean isEnabled=true;
    private Font butFont=new Font("Dialog", 0, 12);
    public Color pipColor = new Color(153);
    public Color butColor = Color.yellow;
    public Color labColor = Color.BLACK;
    private GRoundRect frame;
    private double butWidth;
    private double butHeight;
    private GLabel label;
    private String text;//button text
    //constructor
    public ButtonField(String text,double width,double height)
    {
      //init
      butWidth=width;
      butHeight=height;
      //draw
      this.frame=new GRoundRect(width,height);
      this.frame.setFilled(true);
      this.frame.setColor(Color.black);
      this.frame.setFillColor(butColor);
      this.label = new GLabel("");
      this.label.setFont(butFont);
      this.label.setColor(labColor);
      add(this.frame);
      add(this.label);
      setLabel(text);
```

```
    setEnabled(true);
  }
  //methods--utility
  private double ctrX(){
     return (this.getWidth()-label.getWidth())/2;
  }
  private double ctrY(){
     return (this.getHeight()-label.getAscent())/2+1;
  }

  //methods--for usage
  public String getLabel(){
     return text;
  }
  public void setLabel(String text)
  {
     this.label.setLabel(text);
     this.text=text;//this to call global var text
     //sizing
     double size=this.label.getFont().getSize();
     double height=this.frame.getHeight();
     double width=this.frame.getWidth();
     double ratio1=size/height;
     double ratio2=this.label.getWidth()/width;
     //System.out.println(ratio1+","+ratio2+","+this.label.getBounds().getWidth());
     int fs;//font size
     if(ratio1<ratio2){
        fs=(int)((this.label.getFont().getSize())/ratio2*0.92);
     } else {
        fs=(int)((this.label.getFont().getSize())/ratio1*0.85);
     }
     this.label.setFont("*-*-"+fs);
     this.label.setLocation(ctrX(),butHeight*0.5+this.label.getFont().getSize()*0.4);//this
is TAE
  }
  public void setEnabled(boolean flag)
  {
    this.isEnabled = flag;
    this.label.setColor(this.isEnabled ? labColor : Color.gray);
  }
  //other "settings"
  public void setButtonFont(Font f){
    label.setFont(f);
    butFont=label.getFont();
    setLabel(text);
  }
  public void setButtonFont(String f){
```

```
    label.setFont(f);
    butFont=label.getFont();
    setLabel(text);
  }



  //++
  public void pressed(){
    if (this.isEnabled)
    {
       this.frame.setFillColor(pipColor);
       this.label.setColor(Color.white);
    }
  }
  public void released(){
    if (this.isEnabled)
    {
      this.frame.setFillColor(butColor);
      this.label.setColor(labColor);
    }
  }

}
```

E) Collection.java

```
package ph.edu.dlsu.chan.calculator;

//all static amap,.....nvm just instantiate

import acm.graphics.GMath;
import acm.util.RandomGenerator;
import java.util.Objects;

/*@author Patrick*/
public class Collection {
  static Operator op[]=new Operator[100];//just change size if not enough

  /*static boolean isEq(String a,String b){
     char ca[]=a.toCharArray();
     char cb[]=b.toCharArray();
     if(a.length()==b.length()){
        for(int i=0;i<a.length();i++){
           int za=(int)ca[i];
```

```
            int zb=(int)cb[i];
            boolean result=(Objects.deepEquals(za,zb));
            //System.out.println(""+za+"("+ca[i]+")&"+zb+"("+cb[i]+") is "+result);
            if(!result){
               return result;
            }

         }
         return true;
      }
      return false;
   }*/


   //operator search
   static Operator getOp(String indicator){
      for(int i=0;i<op.length;i++){
         if(op[i]!=null){
            if(op[i].ind().equals(indicator)){
            //if(isEq(op[i].ind(),indicator)){
               return op[i];
            }
         } else {
            return null;//not found
         }
      }
      return null;//not found
   }
   static double mat(String s) throws MathException{
      throw new MathException(s);
   }
   static Operator getOp(char indicator){
      for(Operator i:op){
         if(i!=null){
            if(i.ind().length()==1&&i.ind().charAt(0)==indicator){
               return i;
            }
         }
      }
      return null;//not found
   }
   static double fact(double x){
      if(x>=0 && Math.floor(x)==x){
         double ans=1;
         while(x>0){
            ans=ans*x;
            x--;
```

```
            if(ans>Double.MAX_VALUE){
               return Double.POSITIVE_INFINITY;
            }
         }
         return ans;
      } else {
         return mat(x+",factorial only permits positive integers");
      }
   }
   static double asinh(double x)
   {
   return Math.log(x + Math.sqrt(x*x + 1.0));
   }

   static double acosh(double x)
   {
   return Math.log(x + Math.sqrt(x*x - 1.0));
   }

   static double atanh(double x)
   {
   return 0.5*Math.log( (x + 1.0) / (x - 1.0) );
   }
   static void initC(){//you need to call this in your program init
      //operators
      /*PRECEDENCE
      if precedence=100, it is a suffix
      %,!,Mega,Giga,...
      20-constants

      0-logical or,bitwise or,xor,xnor
      1-logical and,bitwise and
      2-logical not
      3-relational
      4-+,-
      5-*,/,implied *
      6-nPr,nCr
      7-(statistical estimations--idk this)
      8-neg sign(-)
      9-fractions a/b
      10-exponents,suffixes* (+eng symbols)
      x 2, x–1, x!, °’ ”, °, r, g
      ^(, x'(
      't
      %
      m, u, n, p, f, k, M, G, T, P
      11-parenthetical (must imply the *)[input pa lang,lagyan na ng ( sa harap,
```

```
constants also]
        Pol(, Rec(
        ∫ (, d/dx(, d2/dx2(, Σ (, P(, Q(, R(
        sin(, cos(, tan(, sin –1(, cos –1(, tan –1(, sinh(, cosh(,
        tanh(, sinh –1(, cosh –1(, tanh –1(
        log(, ln(, e^(, 10^(,'(, 3'(
        Arg(, Abs(, ReP(, ImP(, Conjg(
        Not(, Neg(, Det(, Trn(, Rnd(
        Int(, Frac(, Intg(
        +++in parsing, add another ( before 11's and constants (π,e),
              then add * when encountered ( w/o operation))
        */
        //String indicator,int precedence,int numOfOperands,+lefttoright
        op[0]=new Operator("+",4,2){
           @Override
           double oper() {
              return opr[0]+opr[1];
           }
        };
        op[1]=new Operator("–",4,2){//**THIS IS N-DASH
           @Override
           double oper() {
              return opr[0]-opr[1];
           }
        };
        op[2]=new Operator("*",5,2){
           @Override
           double oper() {
              return opr[0]*opr[1];
           }
        };
        op[3]=new Operator("/",5,2){
           @Override
           double oper() {
              /*if(opr[1]==0){
                 if(opr[0]==0){
                    return mat("NaN");
                 } else{
                    return mat("Inf");
                 }
              }*/
              return opr[0]/opr[1];
           }
        };
        op[4]=new Operator("x!","!",100,1){
           @Override
           double oper() {
```

```
            return fact(opr[0]);
        }
    };
    op[5]=new Operator("sin(",11,1){//all radians
        @Override
        double oper() {
            return Math.sin(opr[0]);
        }
    };
    op[6]=new Operator("cos(",11,1){
        @Override
        double oper() {
            return Math.cos(opr[0]);
        }
    };
    op[7]=new Operator("tan(",11,1){
        @Override
        double oper() {
            return Math.tan(opr[0]);
        }
    };
    op[8]=new Operator("ln(",11,1){
        @Override
        double oper() {
            return Math.log(opr[0]);
        }
    };
    op[9]=new Operator("log(",11,1){
        @Override
        double oper() {
            return Math.log10(opr[0]);
        }
    };
    op[10]=new Operator("|x|","abs(",11,1){
        @Override
        double oper() {
            return Math.abs(opr[0]);
        }
    };
    op[11]=new Operator("^",10,2,false){
        @Override
        double oper() {
            if(opr[0]==0 && opr[1]==0){
                return Double.NaN;
            }
            return Math.pow(opr[0],opr[1]);
        }
```

```
        };
        op[12]=new Operator("(-)","-",8,1){
            @Override
            double oper() {
                return -1*(opr[0]);
            }
        };
        op[13]=new Operator("nPr","P",6,2){
            @Override
            double oper() {
                return fact(opr[0])/fact(opr[0]-opr[1]);
            }
        };
        op[14]=new Operator("nCr","C",6,2){
            @Override
            double oper() {
                return fact(opr[0])/(fact(opr[0]-opr[1])*fact(opr[1]));
            }
        };
        op[15]=new Operator("a/b","┘",9,2){
            @Override
            double oper() {
                return opr[0]/opr[1];
            }
        };
        op[16]=new Operator("x²","²",100,1,false){
            @Override
            double oper() {
                return Math.pow(opr[0],2);
            }
        };
        op[17]=new Operator("x³","³",100,1,false){
            @Override
            double oper() {
                return Math.pow(opr[0],3);
            }
        };
        op[18]=new Operator("x⁻¹","⁻¹",100,1,false){
            @Override
            double oper() {
                return Math.pow(opr[0],-1);
            }
        };
        op[19]=new Operator("°",100,1){
            @Override
            double oper() {
                return Math.toRadians(opr[0]);
```

```
        }
    };
    op[20]=new Operator("’",100,1){
        @Override
        double oper() {
            return Math.toRadians(opr[0]/60);
        }
    };
    op[21]=new Operator("”",100,1){
        @Override
        double oper() {
            return Math.toRadians(opr[0]/3600);
        }
    };
    op[22]=new Operator("ʳ",100,1){
        @Override
        double oper() {
            return opr[0];
        }
    };
    op[23]=new Operator("g",100,1){
        @Override
        double oper() {
            return Math.toRadians(opr[0]*9/10);
        }
    };
    op[24]=new Operator("ᵐ",100,1){
        @Override
        double oper() {
            return opr[0]*1e-3;
        }
    };
    op[25]=new Operator("μ",100,1){
        @Override
        double oper() {
            return opr[0]*1e-6;
        }
    };
    op[26]=new Operator("ⁿ",100,1){
        @Override
        double oper() {
            return opr[0]*1e-9;
        }
    };
    op[27]=new Operator("ᵖ",100,1){
        @Override
        double oper() {
```

```
            return opr[0]*1e-12;
        }
    };
    op[28]=new Operator("f",100,1){
        @Override
        double oper() {
            return opr[0]*1e-15;
        }
    };
    op[29]=new Operator("K",100,1){
        @Override
        double oper() {
            return opr[0]*1e3;
        }
    };
    op[30]=new Operator("M",100,1){
        @Override
        double oper() {
            return opr[0]*1e6;
        }
    };
    op[31]=new Operator("G",100,1){
        @Override
        double oper() {
            return opr[0]*1e9;
        }
    };
    op[32]=new Operator("T",100,1){
        @Override
        double oper() {
            return opr[0]*1e12;
        }
    };
    op[33]=new Operator("P",100,1){
        @Override
        double oper() {
            return opr[0]*1e15;
        }
    };
    op[34]=new Operator("π",20,0){
        @Override
        double oper() {
            return Math.PI;
        }
    };
    op[35]=new Operator("e",20,0){
        @Override
```

```
        double oper() {
            return Math.E;
        }
    };
    op[36]=new Operator("x10^","*10^",10,1){//actually just uses them separately
        @Override
        double oper() {
            return opr[0]*Math.pow(10,opr[1]);
        }
    };
    op[37]=new Operator("∧",1,2){
        @Override
        double oper() {
            boolean a=((Math.abs(opr[0])>=1)&&(Math.abs(opr[1])>=1));
            if(a){
                return 1;
            } else {
                return 0;
            }
        }
    };
    op[38]=new Operator("∨",0,2){
        @Override
        double oper() {
            boolean a=((Math.abs(opr[0])>=1)||(Math.abs(opr[1])>=1));
            if(a){
                return 1;
            } else {
                return 0;
            }
        }
    };
    op[39]=new Operator("¬",2,1){
        @Override
        double oper() {
            boolean a=(!(Math.abs(opr[0])>=1));
            if(a){
                return 1;
            } else {
                return 0;
            }
        }
    };
    op[40]=new Operator("==",3,2){
        @Override
        double oper() {
            boolean a=(opr[0]==opr[1]);
```

```
            if(a){
                return 1;
            } else {
                return 0;
            }
        }
    };
    op[41]=new Operator("<",3,2){
        @Override
        double oper() {
            boolean a=(opr[0]<opr[1]);
            if(a){
                return 1;
            } else {
                return 0;
            }
        }
    };
    op[42]=new Operator("≤",3,2){
        @Override
        double oper() {
            boolean a=(opr[0]<=opr[1]);
            if(a){
                return 1;
            } else {
                return 0;
            }
        }
    };
    op[43]=new Operator("≠",3,2){
        @Override
        double oper() {
            boolean a=(opr[0]!=opr[1]);
            if(a){
                return 1;
            } else {
                return 0;
            }
        }
    };
    op[44]=new Operator("≥",3,2){
        @Override
        double oper() {
            boolean a=(opr[0]>=opr[1]);
            if(a){
                return 1;
            } else {
```

```
            return 0;
        }
    }
};
op[45]=new Operator(">",3,2){
    @Override
    double oper() {
        boolean a=(opr[0]>opr[1]);
        if(a){
            return 1;
        } else {
            return 0;
        }
    }
};
op[46]=new Operator("sin⁻¹(",11,1){
    @Override
    double oper() {
        return Math.asin(opr[0]);
    }
};
op[47]=new Operator("cos⁻¹(",11,1){
    @Override
    double oper() {
        return Math.acos(opr[0]);
    }
};
op[48]=new Operator("tan⁻¹(",11,1){
    @Override
    double oper() {
        return Math.atan(opr[0]);
    }
};
op[49]=new Operator("sinh(",11,1){
    @Override
    double oper() {
        return Math.sinh(opr[0]);
    }
};
op[50]=new Operator("cosh(",11,1){
    @Override
    double oper() {
        return Math.cosh(opr[0]);
    }
};
op[51]=new Operator("tanh(",11,1){
    @Override
```

```
        double oper() {
            return Math.tanh(opr[0]);
        }
    };
    op[52]=new Operator("sinh⁻¹(",11,1){
        @Override
        double oper() {
            return asinh(opr[0]);
        }
    };
    op[53]=new Operator("cosh⁻¹(",11,1){
        @Override
        double oper() {
            return acosh(opr[0]);
        }
    };
    op[54]=new Operator("tanh⁻¹(",11,1){
        @Override
        double oper() {
            return atanh(opr[0]);
        }
    };
    op[55]=new Operator("*e^",11,2){
        @Override
        double oper() {
            return opr[0]*Math.pow(Math.E,opr[1]);
        }
    };
    op[56]=new Operator("rdm",20,0){
        @Override
        double oper() {
            return RandomGenerator.getInstance().nextFloat();
        }
    };
    op[57]=new Operator("flr(",11,1){
        @Override
        double oper() {
            return Math.floor(opr[0]);
        }
    };
    op[58]=new Operator("ceil(",11,1){
        @Override
        double oper() {
            return Math.ceil(opr[0]);
        }
    };
    op[59]=new Operator("←max→",10,2){
```

```
    @Override
    double oper() {
        return Math.max(opr[0],opr[1]);
    }
};
op[60]=new Operator("←min→",10,2){
    @Override
    double oper() {
        return Math.min(opr[0],opr[1]);
    }
};
op[61]=new Operator("mod",5,2){
    @Override
    double oper() {
        int x=(int)opr[0];
        int y=(int)opr[1];
        if(x==opr[0]&&y==opr[1]){
            return x%y;
        } else {
            return mat("mod is for integers only");
        }
    }
};
op[62]=new Operator("ⁿ√","√",10,2){
    @Override
    double oper() {
        return Math.pow(opr[1],1/opr[0]);
    }
};
op[63]=new Operator("rInt",20,0){
    @Override
    double oper() {
        return RandomGenerator.getInstance().nextInt();
    }
};
```

```
}
```

```
//methods
String inp;//input
//if char is still part of a number
static boolean isNum(char a){
    if(a>='0'&&a<='9'){
        return true;
    } else if (a=='.'){
        return true;
    }
    return false;
}
//[cannot pass empty input here!]
//1.inp-infix (operators are properly parsed)
static StackC parse(String inp){
    //init
    StackC infix=new StackC();
    char[] arr=inp.toCharArray();
    int startIndex=0;
    //parse
    for(int i=0;i<arr.length;i++){
        if(isNum(arr[startIndex])){//parsing numbers
            if(i+1==arr.length || !isNum(arr[i+1])){//next char is not num
                //i+1 is the first non-numerical input
                String num=inp.substring(startIndex,i+1);
                double result;
                try{
                    result=Double.parseDouble(num);
                } catch(NumberFormatException e){
                    throw new SyntaxException("Number Format:"+num);
```

```
            }
            infix.push(result);
            startIndex=i+1;
        }
    } else if(!isNum(arr[startIndex])){//parsing operators
        String test=inp.substring(startIndex,i+1);
        //parentheses
        if(test.equals("(")){
            //implied *
            if(!infix.isEmpty() && isNum(arr[startIndex-1])){//isNum more accurate
                infix.push(getOp("*"));
            }
            infix.push(test);
            startIndex=i+1;
        } else if(test.equals(")")){
            infix.push(test);
            startIndex=i+1;
        }
        //other op's
        else if (getOp(test)!=null){
            if(getOp(test).precedence==11){//parenthetical
                //implied *
                if(!infix.isEmpty() && isNum(arr[startIndex-1])){
                    infix.push(getOp("*"));
                } else if (!infix.isEmpty() && infix.isOperatorTop()){
                    Operator o=(Operator)infix.top();
                    if(o.precedence==20){
                        infix.push(getOp("*"));
                    }
                }
                infix.push(getOp(test));
                infix.push("(");
            } else if(getOp(test).precedence==20){//constants
                //implied *
                if(!infix.isEmpty() && isNum(arr[startIndex-1])){
                    infix.push(getOp("*"));
                } else if (!infix.isEmpty() && infix.isOperatorTop()){
                    Operator o=(Operator)infix.top();
                    if(o.precedence==20){
                        infix.push(getOp("*"));
                    }
                }
                infix.push(getOp(test));
            } else if(getOp(test)==getOp("-")){
                if(i+1==arr.length || !isNum(arr[i+1])){
                    throw new SyntaxException("invalid negative sign");
                }
```

```
                infix.push(getOp(test));
            }
            else {
                infix.push(getOp(test));
            }
            startIndex=i+1;
        }
        //check if next iteration is still ok
        else if(i+1==arr.length || isNum(arr[i+1])){
            throw new SyntaxException("unsupported operation:"+test);
        }
    }
}
//infix is reversed
infix.reverse();
System.out.println("infix");
infix.print();
return infix;
}
//2.infix-postfix(shunting yard)
static StackC convert(StackC infix){
    //stacks
    StackC opStack=new StackC();
    StackC postfix=new StackC();
    //algo
    while(!infix.isEmpty()){
        if(infix.isDoubleTop()){
            postfix.push((Double)infix.top());
            infix.pop();
        } else if(infix.isOperatorTop()) {
            Operator temp=(Operator)infix.top();
            infix.pop();
            if(temp.precedence==100){//suffix
                opStack.push(temp);
                continue;
            } else if(temp.precedence==20){//constant
                postfix.push(temp);
                continue;
            }
            //init isOk
            boolean isOk=false;//loop validity check
            if(!opStack.isEmpty()){
                if(!opStack.isOperatorTop()){
                    String atemp=(String)opStack.top();
                    if(!atemp.equals("(")){
                        isOk=true;
                    }
```

```
            } else {
                Operator top=(Operator)opStack.top();
                if(!temp.isL2R() && top==temp){//in the case of "^"
                    isOk=false;//this is correct
                } else if(top.precedence>=temp.precedence){
                    isOk=true;
                }//else false
            }
        }
        //loop
        while(isOk){
            postfix.push((Operator)opStack.top());
            opStack.pop();
            //revalidate isOk
            isOk=false;
            if(!opStack.isEmpty()){
                if(!opStack.isOperatorTop()){
                    String atemp=(String)opStack.top();
                    if(!atemp.equals("(")){
                        isOk=true;
                    }
                } else {
                    Operator top=(Operator)opStack.top();
                    if(!temp.isL2R() && top==temp){//in the case of "^"
                        isOk=false;
                    } else if(top.precedence>=temp.precedence){
                        isOk=true;
                    }
                }
            }
        }
        //loop done
        opStack.push(temp);
    } else {
        String op=(String)infix.top();
        infix.pop();
        char o='\0';
        if(op.length()==1){
            o=op.charAt(0);
        }
        if(o=='('){
            opStack.push(op);
        } else if (o==')'){
            if(opStack.isEmpty()){
                throw new SyntaxException("Mismatched parenthesis:')'");
            }
            while(!opStack.top().equals("(")){
```

```
                postfix.push((Operator)opStack.top());
                opStack.pop();
                if(opStack.isEmpty()){
                   throw new SyntaxException("Mismatched parenthesis:')'");
                }
             }
             opStack.pop();//pops "(" from opStack
          } else {//operator
             throw new SyntaxException("Unparsed Operator:"+op);
          }
       }
    }
    //opStack.print();
    //infix empty
    while(!opStack.isEmpty()){
       if(!opStack.isOperatorTop()){
          opStack.pop();
          //next line removed to allow unclosed parentheses
          //throw new SyntaxException("Mismatched parenthesis:'('(ata)");
       }
       postfix.push((Operator)opStack.top());
       opStack.pop();
    }
    //convert "PREFIX" to postfix stack
    postfix.reverse();
    //return

    System.out.println("postfix");
    postfix.print();
    return postfix;
 }
 //3.postfix-ans
 static double compute(StackC postfix){
    Stack<Double> nums=new Stack<>();
    while(!postfix.isEmpty()){
       if(postfix.isDoubleTop()){
          nums.push((double)postfix.top());
          postfix.pop();
       } else {
          Operator temp=(Operator)postfix.top();//nums.print();//debug
          temp.exe(nums);//do the oper, and store in stack
          postfix.pop();
       }
    }
    if(nums.size()!=1){
       System.out.println("Error, ending stack size not 1, Items:");
       while(!nums.isEmpty()){
```

```
            System.out.print(nums.top()+",");
            nums.pop();
         }
         throw new SyntaxException("Stack Computation Error");
      } else {
         System.out.println("ans:"+nums.top());
         return nums.top();
      }
   }
   //ALL-IN-ONE
   static double calc(String inp) throws SyntaxException,MathException{
      if(inp==""){
         return 0;
      } else {
         return compute(convert(parse(inp)));
      }
   }
}
```

F) CollectionListSyncException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author student
 */
public class CollectionListSyncException extends RuntimeException{
   public CollectionListSyncException(String s){
      super();
   }
   public CollectionListSyncException(int kSiz,int oSiz){
      super("Error: mapping List Size mismatch, Key:"+kSiz+",Op:"+oSiz);
   }//end constructor
}
```

G) Field.java

```
package ph.edu.dlsu.chan.calculator;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
/*@author Patrick*/
public class Field extends GCompound{
    private Font txFont=new Font("Dialog", 0, 12);
    public Color bgColor = Color.GRAY;
    public Color txColor = Color.WHITE;
    private GRoundRect frame;
    private double width;
    private double height;
    private GLabel label;
    private String text;//field text
    private boolean isDot=false;
    private boolean isMaxDigits=false;
    private boolean isEditable=true;
    public boolean isNoInput=true;//if input is not zero
    public double M=0;//from std calcu M
    private String temp;//for mouse entered listener

    //constructor
    public Field(double width, double height){
        //init
        this.width=width;
        this.height=height;
        text="";
        //draw
        this.frame=new GRoundRect(width,height);
        this.frame.setFilled(true);
        this.frame.setColor(Color.black);
        this.frame.setFillColor(bgColor);
        this.label = new GLabel("");
        this.label.setFont(txFont);
        this.label.setColor(txColor);
        add(this.frame);
        add(this.label);
        clr();
        this.addMouseListener(new MouseListener() {
            @Override
            public void mouseClicked(MouseEvent e) {
                ;
            }

            @Override
            public void mousePressed(MouseEvent e) {
```

```
            if(temp=="" || temp==" ~Hello!~ "){
                temp=text;
            }
            label.setColor(txColor);
            show(temp);
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            if(temp=="" || temp==" ~Hello!~ "){
                temp=text;
            }
            label.setColor(txColor);
            show(temp);
        }

        @Override
        public void mouseEntered(MouseEvent e) {
            label.setColor(Color.ORANGE);
            show("value of M="+removeDoubleExcess(M));
        }

        @Override
        public void mouseExited(MouseEvent e) {
            if(temp=="" || temp==" ~Hello!~ "){
                temp=text;
            }
            label.setColor(txColor);
            show(temp);
        }
    });
}
//utility
private double rtXAt(double x){//rt-align
    return x-label.getWidth();
}
//method
public String getLabel(){
    return label.getLabel();
}
public boolean isDot(){
    return isDot;
}
public boolean isEditable(){
    return isEditable;
}
public void setEditable(boolean flag){
```

```
        isEditable=flag;
        if(!flag){//if false
            isDot=false;
            isNoInput=true;
        } else {//if true
            isMaxDigits=false;
        }
    }
    public static String removeDoubleExcess(Double result){
        String a=result.toString();
        if(a.endsWith(".0")){
            a=a.substring(0,a.length()-2);
        }
        return a;
    }




    //no edit text
    public void show(String tempoText){
        //setlabel
        this.label.setLabel(tempoText);
        //auto-sizing (changes font size)
        double size=this.label.getFont().getSize();
        double height=this.frame.getHeight();
        double width=this.frame.getWidth();
        double ratio1=size/height;
        double ratio2=this.label.getWidth()/width;
            //System.out.println(ratio1+","+ratio2+","+this.label.getBounds().getWidth());
        int fs;//font size
        if(ratio1<ratio2){
            fs=(int)((this.label.getFont().getSize())/ratio2*0.97);
        } else {
            fs=(int)((this.label.getFont().getSize())/ratio1*0.9);
        }
        //displaying text
        this.label.setFont("*-*-"+fs);
        double xAt=(frame.getWidth())*(1-(1/label.getWidth()));
        this.label.setLocation(rtXAt(xAt),height*0.5+this.label.getFont().getSize()*0.4);//this
is TAE
    }
    //show, but with setEditable=false, isZero=false
    public void result(String text){
        show(text);
        temp=text;
```

```
    setEditable(false);     //isNoInput=true;
}
public void result(double result){
    show(removeDoubleExcess(result));
    temp=removeDoubleExcess(result);
    setEditable(false);     //isNoInput=true;
}
//edits global var text
public void disp(String text){
    //zero if blank
    if(text=="" || text=="0"){//change isNoInput if input is 0 after this
        text="0";
        setEditable(false);
    }
    show(text);
    temp=text;
    //set global var
    this.text=text;
    //maximum input check (prevent too small a text)
    int fs=this.label.getFont().getSize();
    if(fs<=16){
        isMaxDigits=true;
    } else {
        isMaxDigits=false;
    }
}
//add to end
public void concat(char a){
    isNoInput=false;
    if(!isMaxDigits){
        if(!isEditable()){//not editable=new input
            if(a=='.'){
                disp("0.");
                isDot=true;
                setEditable(true);
            }else if(a!='0'){
                disp(a+"");
                setEditable(true);
            } else {//a is zero
                disp("");
                isNoInput=false;
            }
        }else{//actually concatenate
            if(a=='.'){
                if(isDot()){
                    return;//do nothing
                }
```

```
                isDot=true;
            }
            text=text.concat(a+"");
            disp(text);
        }
    }
}
public void concat(String a){
    isNoInput=false;
    if(!isMaxDigits){
        if(!isEditable()){//not editable=new input
            if(a.contains(".")){
                disp(a);
                isDot=true;
                setEditable(true);
            }else if(Double.parseDouble(a)!=0){
                disp(a+"");
                setEditable(true);
            } else {//is zero
                disp("");
                isNoInput=false;
            }
        }else{//actually concatenate
            if(a.contains(".")){
                if(isDot()){
                    return;//do nothing
                }
                isDot=true;
            }
            text=text.concat(a+"");
            disp(text);
        }
    }
}
//del a char
public void del(){
    if(!isEditable() || text.length()<=1){//remove display
        disp("");       //isNoInput=true;
    } else {
        if(text.endsWith(".")){
            isDot=false;
        }
        text=text.substring(0,text.length()-1);
        disp(text);
    }
}
//clear/reset to 0
```

```
    public void clr(){
        disp("");
        isDot=false;    //isNoInput=true;
    }
    //negate (in front)
    public void neg(){//doesn't mind if not editable
        if(text!="0"){
            isNoInput=false;
            if(text.charAt(0)=='-'){
                text=text.substring(1);
            } else {
                text="-"+text;
            }
            disp(text);
        }
    }
    //convert to double
    public double out(){
        if(label.getLabel().endsWith(".")){//if ends w/ "."
            label.getLabel().substring(0,label.getLabel().length()-1);
        }
        return Double.parseDouble(label.getLabel());
    }


    public void setM(double value){
        if(M!=value){
            M=value;
        }
    }
}
```

H) Field1.java

```
package ph.edu.dlsu.chan.calculator;
import acm.graphics.*;
import java.awt.*;
/*@author Patrick*/
public class Field1 extends GCompound{//input field
    private Font txFont=new Font("Dialog", 0, 12);
    public Color bgColor = Color.GRAY;
    public Color txColor = Color.WHITE;
    private GRoundRect frame;
    private double width;
    private double height;
```

```
    private GLabel label;
    private String text;//field text
    private String texttemp;//show
    private String curText;//field text w/ cursor
    //private boolean isMaxDigits=false;
    private boolean isEditable=true;
    //+
    private Hist<String> h=new Hist<>(20);//2 for demo, use 20
    int hIndex=1;//index h of current entry, hIndex=nSlots+1 if h isFilled.
    private int dispBeg=0;//index of string
    private int dispEnd=0;//index of string
    private int curIndex=0;//String(text) index before the cursor
       //(from -1 to length()-1)
    private String cursor="_";
    //private int maxChars=0;
    private int maxResizeChars=0;
    private int minFS=0;
    private int maxFS=0;

    //constructor
    public Field1(double width, double height){
       //init
       this.width=width;
       this.height=height;
       text="";
       //draw
       this.frame=new GRoundRect(width,height);
       this.frame.setFilled(true);
       this.frame.setColor(Color.black);
       this.frame.setFillColor(bgColor);
       this.label = new GLabel("");
       this.label.setFont(txFont);
       this.label.setColor(txColor);
       add(this.frame);
       add(this.label);
       //fs
       label.setLabel("W");//W is prob. the widest
       double size=this.label.getFont().getSize();
       double ratio1=size/height;
       maxFS=(int)(size/ratio1*0.9);
       minFS=(int)(0.5*maxFS);
       //maxChars computation
       this.label.setFont("*-*-"+maxFS);
       //maxChars=(int)(frame.getWidth()*0.97/label.getWidth());

//System.out.print(frame.getWidth()*0.97+";"+label.getWidth()+";"+frame.getWidth()*0.
97/label.getWidth()+";;");
```

```
        this.label.setFont("*-*-"+minFS);
        maxResizeChars=(int)(frame.getWidth()*0.97/label.getWidth());

//System.out.print(frame.getWidth()*0.97+";"+label.getWidth()+";"+frame.getWidth()*0.
97/label.getWidth());
        //clear
        clr();
    }
    //utility
    private double rtXAt(double x){//rt-align
        return x-label.getWidth();
    }
    //method
    public String getLabel(){
        return label.getLabel();
    }
    public String getText(){
        return text;
    }
    public boolean isEditable(){
        return isEditable;
    }
    public void setEditable(boolean flag){
        isEditable=flag;
        if(!flag){//if false
            curIndex=text.length()-1;
        } else {//if true
            hIndex=h.size()+1;
            curIndex=text.length()-1;//will place cursor
            text=texttemp;
            //isMaxDigits=false;
        }
    }
    public static String removeDoubleExcess(Double result){
        String a=result.toString();
        if(a.endsWith(".0")){
            a=a.substring(0,a.length()-2);
        }
        return a;
    }


    //no edit text (the naked command, no modifications, unli resize)
    public void show(String tempoText){
        //setlabel
        this.label.setLabel(tempoText);
        //auto-sizing (changes font size)
```

```
        double size=this.label.getFont().getSize();
        double height=this.frame.getHeight();
        double width=this.frame.getWidth();
        double ratio1=size/height;
        double ratio2=this.label.getWidth()/width;
            //System.out.println(ratio1+","+ratio2+","+this.label.getBounds().getWidth());
        int fs;//font size
        if(ratio1<ratio2){
            fs=(int)((this.label.getFont().getSize())/ratio2*0.97);
        } else {
            fs=(int)((this.label.getFont().getSize())/ratio1*0.9);
        }
        //displaying text
        this.label.setFont("*-*-"+fs);
        double xAt=(frame.getWidth())*(1-(1/label.getWidth()));
        this.label.setLocation(rtXAt(xAt),height*0.5+this.label.getFont().getSize()*0.4);//this
is TAE
    }
    //show Limited (scrolling) [still no edit text]
    public void showL(String tempoText,int dispBeginning){
        texttemp=tempoText;//uncut text label
        if(tempoText.length()>maxResizeChars){
            //note: this already takes indices' bases into account
            if(dispBeginning>tempoText.length()-1-maxResizeChars+1){
                dispBeginning=tempoText.length()-1-maxResizeChars+1;
            }
            dispBeg=dispBeginning;
            dispEnd=dispBeg+maxResizeChars-1;
            //setlabel
            show(tempoText.substring(dispBeg,dispEnd+1));
        } else {
            dispBeg=0;
            dispEnd=tempoText.length()-1;
            //displaying text
            show(tempoText);
        }
    }
    //showL(end), but with setEditable=false
    public void result(String text){
        showL(text,text.length());
        setEditable(false);     //isNoInput=true;
    }
    public void result(double result){
        String text=removeDoubleExcess(result);
        showL(text,text.length());
        setEditable(false);     //isNoInput=true;
    }
```

```
//edits global var text & places cursor & setEditable=true
public void disp(String text){//disp("") simply displays cursor
   if(!isEditable()){//if false
      setEditable(true);
   }
   //place cursor
   if(curIndex>text.length()-1){
      curIndex=text.length()-1;
   }
   this.text=text;
   curText=text.substring(0,curIndex+1)+cursor+text.substring(curIndex+1);
   //display
   showL(curText,dispBeg);//test for dispEnd (no repeat if cursor is w/in)
   if (dispBeg>curIndex){
      dispBeg=curIndex;
      if(dispBeg<0){
         dispBeg=0;
      }
      showL(curText,dispBeg);//for real this time
   } else if(dispEnd<curIndex+1){
      dispBeg=dispBeg+((curIndex+1)-dispEnd);//move dispBeg
      showL(curText,dispBeg);//for real this time
   }

}

//insert/input
public void inp(char a){
   if(isEditable()){//assumes char is one normal printable character
      //adds a on cursor location
      text=text.substring(0,curIndex+1)+a+text.substring(curIndex+1);
      curIndex++;
      disp(text);
   } else {
      //setEditable(true);
      disp(a+"");
   }
}
public void inp(String a){
   if(isEditable()){
      //adds a on cursor location
      text=text.substring(0,curIndex+1)+a+text.substring(curIndex+1);
      curIndex=curIndex+a.length();
      disp(text);
   } else {
      //setEditable(true);
      disp(a+"");
```

```
    }
  }
  //del a char
  public void del(){
    if(!isEditable() || text.length()<=1){//remove display
       disp("");      //isNoInput=true;
    } else {
       if(curIndex>-1){
          //removes char before cursor
          text=text.substring(0,curIndex)+text.substring(curIndex+1);
          curIndex--;
       } else {
          //removes char after cursor
          text=text.substring(1);
       }
      disp(text);
    }
  }
  //clear/reset to blank
  public void clr(){
    disp("");
  }
  //output the String formed
  public String out(){
    /*String a=label.getLabel();
    if(isEditable){
       //remove cursor
       if(curIndex<a.length()-1-1){//a.length=text.length+1
          return a.substring(0,curIndex+1)+a.substring(curIndex+2);
       } else {
          return a.substring(0,curIndex+1);
       }
    } else {
       return a;
    }*/
    if(texttemp.contains(cursor)){
       return text;
    } else {
       return texttemp;
    }
  }

  //save to HISTory
  public void save(){
    h.push(out());
    hIndex=h.size()+1;
    result(out());
```

```
    }
    //clr HISTory
    public void histClr(){
        h.clr();
        hIndex=1;
    }
    //move cursor
    public void L(){
        if(!isEditable){
            setEditable(true);

        }
        if(curIndex>-1){
            curIndex--;
            disp(text);
        }
    }
    public void R(){
        if(!isEditable){
            setEditable(true);
            curIndex=-1;
        }
        if(curIndex<text.length()-1){
            curIndex++;
            disp(text);
        }
    }
    //browse history
    public void U(){
        if(hIndex>1){
            hIndex--;
        }
        if(h.size()<hIndex){
            disp("");
        }else {
            result(h.get(hIndex));
        }
    }
    public void D(){
        if(hIndex<h.size()){
            hIndex++;
            result(h.get(hIndex));
        } else{
            hIndex=h.size()+1;
            disp("");
        }
    }
```

```
}
```

I) Field2.java

```
package ph.edu.dlsu.chan.calculator;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
/*@author Patrick*/
public class Field2 extends GCompound{//input field
    private Font txFont=new Font("Dialog", 0, 12);
    public Color bgColor = Color.GRAY;
    public Color txColor = Color.WHITE;
    private GRoundRect frame;
    private double width;
    private double height;
    private GLabel label;
    private String text;//field text
    private ButtonField bLeft;
    private ButtonField bRight;
    private static final double BWIDTH=20;
    private static final double BSPACING=5;
    //private boolean isMaxDigits=false;
    //private boolean isEditable=true;
    //+
    private int dispBeg=0;//index of string
    private int dispEnd=0;//index of string
    //private int maxChars=0;
    private int maxResizeChars=0;
    private int minFS=0;
    private int maxFS=0;

    //constructor
    public Field2(double width, double height){
        //init
        this.width=width-2*(BWIDTH+BSPACING);
        this.height=height;
        text="";
        //draw
        this.frame=new GRoundRect(this.width,height);
        this.frame.setFilled(true);
        this.frame.setColor(Color.black);
        this.frame.setFillColor(bgColor);
        this.label = new GLabel("");
```

```
        this.label.setFont(txFont);
        this.label.setColor(txColor);
        add(this.frame,BWIDTH+BSPACING,0);
        add(this.label);
        //buttons
        bLeft=new ButtonField("↞",BWIDTH,height);
        bRight=new ButtonField("↠",BWIDTH,height);
        add(bLeft);
        add(bRight,BWIDTH+BSPACING+this.width+BSPACING,0);
        //fs
        label.setLabel("M");//W is prob. the widest
        double size=this.label.getFont().getSize();
        double ratio1=size/height;
        maxFS=(int)(size/ratio1*0.9);
        minFS=15;//(int)(0.5*maxFS);
        //maxChars computation
        this.label.setFont("*-*-"+maxFS);
        //maxChars=(int)(frame.getWidth()*0.97/label.getWidth());

//System.out.print(frame.getWidth()*0.97+";"+label.getWidth()+";"+frame.getWidth()*0.
97/label.getWidth()+";;");
        this.label.setFont("*-*-"+minFS);
        maxResizeChars=(int)(frame.getWidth()*0.97/label.getWidth());

//System.out.print(frame.getWidth()*0.97+";"+label.getWidth()+";"+frame.getWidth()*0.
97/label.getWidth());
        //clear
        clr();
        //mouse listeners
        bLeft.addMouseListener(new MouseListener() {
            @Override
            public void mouseClicked(MouseEvent e) {;}
            @Override
            public void mousePressed(MouseEvent e) {
                bLeft.pressed();
            }
            @Override
            public void mouseReleased(MouseEvent e) {
                bLeft.released();
                L();
            }
            @Override
            public void mouseEntered(MouseEvent e) {;}
            @Override
            public void mouseExited(MouseEvent e) {;}
        });
        bRight.addMouseListener(new MouseListener() {
```

```
        @Override
        public void mouseClicked(MouseEvent e) {;}
        @Override
        public void mousePressed(MouseEvent e) {
            bRight.pressed();
        }
        @Override
        public void mouseReleased(MouseEvent e) {
            bRight.released();
            R();
        }
        @Override
        public void mouseEntered(MouseEvent e) {;}
        @Override
        public void mouseExited(MouseEvent e) {;}
    });
}
//utility
private double rtXAt(double x){//rt-align
    return x-label.getWidth();
}
//method
public String getLabel(){
    return label.getLabel();
}
public static String removeDoubleExcess(Double result){
    String a=result.toString();
    if(a.endsWith(".0")){
        a=a.substring(0,a.length()-2);
    }
    return a;
}


//no edit text (the naked command, no modifications, unli resize)
public void show(String tempoText){
    //setlabel
    this.label.setLabel(tempoText);
    //auto-sizing (changes font size)
    double size=this.label.getFont().getSize();
    double height=this.frame.getHeight();
    double width=this.frame.getWidth();
    double ratio1=size/height;
    double ratio2=this.label.getWidth()/width;
        //System.out.println(ratio1+","+ratio2+","+this.label.getBounds().getWidth());
    int fs;//font size
    if(ratio1<ratio2){
```

```
            fs=(int)((this.label.getFont().getSize())/ratio2*0.97);
        } else {
            fs=(int)((this.label.getFont().getSize())/ratio1*0.9);
        }
        //displaying text
        this.label.setFont("*-*-"+fs);
        double xAt=BWIDTH+BSPACING+(frame.getWidth())*(1-(1/label.getWidth()));
        this.label.setLocation(rtXAt(xAt),height*0.5+this.label.getFont().getSize()*0.4);//this
is TAE
    }
    //show Limited (scrolling) [still no edit text]
    public void showL(String tempoText,int dispBeginning){
        if(tempoText.length()>maxResizeChars){
            //note: this already takes indices' bases into account
            if(dispBeginning>tempoText.length()-1-maxResizeChars+1){
                dispBeginning=tempoText.length()-1-maxResizeChars+1;
            }
            dispBeg=dispBeginning;
            dispEnd=dispBeg+maxResizeChars-1;
            //setlabel
            show(tempoText.substring(dispBeg,dispEnd+1));
        } else {
            dispBeg=0;
            dispEnd=tempoText.length()-1;
            //displaying text
            show(tempoText);
        }
    }
    //edits global var text, then showL
    public void disp(String text){
        if(text==""){
            text="0";
        }
        this.text=text;
        showL(this.text,0);
    }
    public void disp(double text){
        this.text=removeDoubleExcess(text);
        showL(this.text,0);
    }
    public void disp(String text,int dispBeginning){
        if(text==""){
            text="0";
        }
        this.text=text;
        showL(this.text,dispBeginning);
    }
```

```
    public void disp(double text,int dispBeginning){
        this.text=removeDoubleExcess(text);
        showL(this.text,dispBeginning);
    }

    //clear/reset to zero
    public void clr(){
        disp("");
    }
    //convert to double
    public double out(){
        if(label.getLabel().endsWith(".")){//if ends w/ "."
            label.getLabel().substring(0,label.getLabel().length()-1);
        }
        return Double.parseDouble(label.getLabel());
    }


    //scroll
    public void L(){
        if(dispBeg>0){
            dispBeg--;
        }
        disp(text,dispBeg);
    }
    public void R(){
        disp(text,dispBeg);//parang test to get dispEnd
        if(dispEnd<text.length()-1){
            dispBeg++;
            disp(text,dispBeg);
        }
    }
}
```

J) Hist.java

```
package ph.edu.dlsu.chan.calculator;

/*@author Patrick*/
public class Hist<E> {
    private MyList<E> list=new MyList<>();
    private int nItems=0;//MAX ITEMS, actually


    public Hist(int noOfItems){
```

```
        list.createList(noOfItems);
        nItems=noOfItems;
    }

    public void push(E item){
        if(list.isFull()){
            list.remove(1);
            list.add(list.size()+1, item);
        } else {
            list.add(list.size()+1, item);
        }
    }

    public E get(int index){//1 based, 1 is oldest
        return list.get(index);
    }

    public void remove(int index){
        list.remove(index);
    }

    public boolean isEmpty(){
        return list.isEmpty();
    }

    public boolean isFilled(){
        return list.isFull();
    }

    public int size(){
        return list.size();
    }

    public int nSlots(){
        return nItems;
    }

    public void clr(){
        list.createList(nItems);
    }
}
```

K) List.java

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/*
 * File: List.java
 * ---------------
 * This is the List ADT definition
 */


public interface List<E>{


  public void createList();
  // precondition: none
  // postcondition: Create an empty list


  public void add(int index, E item) throws ListIndexOutOfBoundsException,
ListFullException;
// precondition: index (to be added) is within the position of the list of items,
1<=index<=size()+1
// postcondition: Insert item at position index of a list
// if 1<=index<= size()+1. If index <= size(), items
// at position index onwards are shifted one position
// to the right. Throws an exception when index is out of range
// or if the item cannot be placed on the list (list full).


 // public void replace(int index, E item) throws ListIndexOutOfBoundsException,
ListEmptyException;
// precondition: index (to be replaced) is within the position of the list of items,
1<=index<=size()+1
// postcondition: Replace item at position index of a list
// if 1<=index<= size()+1. If index <= size(), items
// at position index onwards are shifted one position
// to the right. Throws an exception when index is out of range
// or if the item cannot be placed on the list (list empty).


  public void remove(int index) throws ListIndexOutOfBoundsException;
// precondition: index (to be removed) is within the position of the list of items,
1<=index<=size()
// postcondition: Remove item at position index of a list
```

```
// if 1<=index<= size(). Items at position
// index+1 onwards are shifted one position to the left
// Throws an exception when index is out of range, or if list is empty.



 public boolean isEmpty();
// precondition: none
// postcondition: Determine if a list is empty



 public boolean isFull();
// precondition: none
// postcondition: Determine if a list is full



public E get(int index) throws ListIndexOutOfBoundsException;
// precondition: index is within the position of the list of items, 1<=index<=size()
// postcondition: Returns item at position index of
// a list if 1<=index<=size(). Throws an exception if index is out of range.



public int size();
// precondition: none
// postcondition: Returns number of items in a list

}
```

L) ListEmptyException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author Administrator
 */
class ListEmptyException extends RuntimeException{
   public ListEmptyException(String s){
```

```
        super(s);
    }//end constructor
}
```

M) ListFullException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author Administrator
 */
public class ListFullException extends RuntimeException{
    public ListFullException(String s){
        super(s);
    }//end constructor
} //end ListException
```

N) ListIndexOutOfBoundsException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author Administrator
 */
public class ListIndexOutOfBoundsException extends IndexOutOfBoundsException{
     public ListIndexOutOfBoundsException(String s){
        super(s);
     }//end constructor
} //end ListIndexOutofBoundsExceptio
```

O) MathException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author student
 */
public class MathException extends RuntimeException{
   public MathException(String s){
      super("Math Error:"+s);
   }//end constructor
}
```

P) MyList.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/*
 * File: MyList.java
 * -----------------
 * This is the List ADT implementation
 */


import acm.program.*;
import acm.util.*;
import java.util.Arrays;


public class MyList<E> implements List<E>{

    /// private data fields
    private final int DEF_MAX = 100;
    private int MAX_LIST = DEF_MAX;     // max length of list
    private E[] items;                  // array of list items
    private int NumItems;               // current size of list
```

```
    public void MyList(){
        items = (E[])new Object[MAX_LIST];
        NumItems = 0;
    }

    public void MyList(int MaxList){//RIP naming conventions haha for convenience
         MAX_LIST=MaxList;
        items = (E[])new Object[MAX_LIST];
        NumItems = 0;
    }

    /// list items are already allocated above with T items[MAX_LIST]
    @SuppressWarnings("unchecked")
    public void createList(){
        items = (E[])new Object[MAX_LIST];
        NumItems = 0;
    }

    public void createList(int MaxList){//RIP naming conventions haha for convenience
         MAX_LIST=MaxList;
        items = (E[])new Object[MAX_LIST];
        NumItems = 0;
    }

    public void add(int index, E item) throws ListIndexOutOfBoundsException,
ListFullException{
        if ( index > 0 && index <= NumItems + 1){
            if (isFull()){
                throw new ListFullException("ERROR: List Already Full");
            }
            else { // insert the element
                int j = NumItems;
                while(j >= index){
                   items[j] = items[j - 1];
                   j--;
                }
                items[index-1] = item;
                NumItems++;
            }
        }
        else
          throw new ListIndexOutOfBoundsException("ERROR: List Index Out Of
Bounds");
```

```
    }

    public void remove(int index) throws ListIndexOutOfBoundsException{
        if ( index > 0 && index <= NumItems){
                for(int i = index; i < NumItems; i++){
                   items[i-1] = items[i];
                }
                NumItems--;
           }
        else
         throw new ListIndexOutOfBoundsException("ERROR: List Index Out Of
Bounds");
   }

   public boolean isEmpty(){
         return NumItems == 0;
   }

   public boolean isFull(){
         return NumItems == MAX_LIST;
   }

   public E get(int index) throws ListIndexOutOfBoundsException{
       if ( index > 0 && index <= NumItems){
          return items[index-1];
       }
        else
         throw new ListIndexOutOfBoundsException("ERROR: List Index Out Of
Bounds");
   }

   public int size(){
         return NumItems;
   }


   public void resize(){
      resize(items.length*3/2);
   }
   public void resize(int newSize){
      if(newSize>items.length){
         int oldSize=items.length;//1 based
         MAX_LIST = newSize;//la lang
         items=Arrays.copyOf(items, newSize);
         System.out.println("List resized, from "+oldSize+" to "+MAX_LIST);
      }
   }
```

```
}
```

Q) Operator.java

```
package ph.edu.dlsu.chan.calculator;

/*@author Patrick*/
public abstract class Operator {//constructor does everything and pushes ans to stack
    int precedence;     //(higher precedence have higher values)
    boolean isLeftToRight=true;
    int numOperands;
    double opr[];//operands--actual
    String label;
    String indicator;//ex: "+","-",...

    //constructor
    public Operator(String indicator,int precedence,int numOfOperands){
        this(indicator,indicator,precedence,numOfOperands,true);
    }
    public Operator(String indicator,int precedence,int numOfOperands,boolean
isLeftToRightAssociativity){
        this(indicator,indicator,precedence,numOfOperands,isLeftToRightAssociativity);
    }
    public Operator(String label,String indicator,int precedence,int numOfOperands){
        this(label,indicator,precedence,numOfOperands,true);
    }
    public Operator(String label,String indicator,int precedence,int
numOfOperands,boolean isLeftToRightAssociativity){
        this.label=label;
        this.indicator=indicator;
        this.precedence=precedence;
        numOperands=numOfOperands;
        isLeftToRight=isLeftToRightAssociativity;
    }

    //do it all
    public double exe(Stack<Double> src){//execute (also returns oper's return)
        //operands
        if(src.size()<numOperands){
            throw new SyntaxException("only "+src.size()+"out of "
                    + numOperands +" needed operands available");
        } else {
            opr=new double[numOperands];
            for(int i=numOperands-1;i>=0;i--){
                opr[i]=src.top();
```

```
            src.pop();
          }
       }
       //operation &
       //push to stack
       double out=oper();
       src.push(out);
       return out;
    }

    //other methods
    //islefttoright
    public boolean isL2R(){
       return isLeftToRight;
    }
    public String ind(){
       return indicator;
    }
    //operatio it does
    abstract double oper();

}
```

R) Stack.java

```
package ph.edu.dlsu.chan.calculator;
public class Stack<E> extends StackBasic<E>{
    //constructor
    public Stack(){
       super();
    }
    public Stack(int s) {
       super(s);
    }
    //additional methods
    public int size(){//1-based
       return top+1;
    }
    public void clear(){
       top=-1;
    }
    public void reverse(){
       E[] temp=(E[])new Object[stackArray.length];
       for(int i=0;i<size();i++){
          temp[size()-1-i]=stackArray[i];
```

```
        }
        stackArray=temp;
    }
    public Stack<E> copyOf(){
        Stack<E> temp=new Stack<>(maxSize);
        for(int i=0;i<size();i++){
            temp.push(stackArray[i]);
        }
        return temp;
    }
    public int getMaxItems(){//++
        return maxSize;
    }
    public void print(){
        if(!isEmpty()){
            if(this.top() instanceof Operator){
                System.out.println("TOS-0:");
                for(int i=size()-1;i>=0;i--){
                    Operator temp=(Operator)stackArray[i];
                    if(temp==null){
                        System.out.print("(null)"+",");
                    } else {
                        System.out.print(temp.indicator+",");
                    }
                }
            } else {
                System.out.println("TOS-0:");
                for(int i=size()-1;i>=0;i--){
                    System.out.print(stackArray[i]+",");
                }
            }
        }
    }
}
```

S) StackBasic.java

```
package ph.edu.dlsu.chan.calculator;

import java.util.Arrays;

/*@author Patrick*/
public class StackBasic<E> implements StackInterface<E>{
    protected int maxSize;//1-based
    protected int DEF_SIZE=500;
    protected E[] stackArray;
```

```
protected int top;//TOS,0-based, -1 empty, size=TOS+1

//constructor
public StackBasic(){
   maxSize = DEF_SIZE;
   stackArray = (E[])new Object[maxSize];
   top = -1;//-1 empty
}
public StackBasic(int s) {
   maxSize = s;
   stackArray = (E[])new Object[maxSize];
   top = -1;
}
//methods
public void push(E j)/* throws StackFullException*/{
   if(isFull()){
      //throw new StackFullException("FULL!");
      resize();
   }
   top++;
   stackArray[top] = j;
}
public void pop() throws StackEmptyException{
   if(!isEmpty()){
      top--;
      //return stackArray[top+1];
   } else {
      throw new StackEmptyException("EMPTY!");
   }
}
public E top()  throws StackEmptyException{
   //System.out.println(stackArray[top]);
   if(!isEmpty()){
      return stackArray[top];
   } else {
      throw new StackEmptyException("EMPTY!");
   }
}
public boolean isEmpty() {
   return (top == -1);
}
public boolean isFull() {
   return (top == maxSize - 1);
}
```

```
    public void resize(){
        resize(stackArray.length*3/2);
    }
    public void resize(int newSize){
        if(newSize>stackArray.length){
            int oldSize=stackArray.length;//1 based
            maxSize = newSize;//la lang
            stackArray=Arrays.copyOf(stackArray, newSize);
            System.out.println("Stack resized, from "+oldSize+" to "+maxSize);
        }
    }
}
```

T) StackC.java

```
package ph.edu.dlsu.chan.calculator;

/*@author Patrick*/
public class StackC {//stack specifically for calcu (stores double and int)
    //as a stack
    protected int maxSize;//1-based
    protected static final int DEF_SIZE=500;
    //implementation variables
    protected Stack<Double> numbers;
    protected Stack<String> text;
    protected Stack<Operator> ops;
    //this directs which stack to look at,(somewhat sync)
    protected Stack<Integer> director;//0-double,1-string,2-operator


    //constructor
    public StackC(){
        this(DEF_SIZE);
    }
    public StackC(int s) {
        maxSize = s;
        numbers=new Stack<>(s);
        text=new Stack<>(s);
        ops=new Stack<>(s);
        director=new Stack<>(s);//true-double,false-string
    }

    //++
    private void checkIntegrity(String methodCalled) throws StackCSyncException{
        if(director.size()!=numbers.size()+text.size()+ops.size()){
```

```
        System.out.println(director.size()+"!="+numbers.size()+"+"
                +"+"+text.size()+"+"+ops.size());
        throw new StackCSyncException("from method:"+methodCalled+
                ";director:"+director.top()+",numbers:"+
                numbers.top()+",text:"+text.top()+",ops:"+ops.top());
    }
}
//methods (basic)
public void push(Double j)/* throws StackFullException*/{
    if(isFull()){
        //throw new StackFullException("FULL!");
        resize();
    }
    numbers.push(j);
    director.push(0);
    checkIntegrity("push");
}
public void push(String j)/* throws StackFullException*/{
    if(isFull()){
        //throw new StackFullException("FULL!");
        resize();
    }
    text.push(j);
    director.push(1);
    checkIntegrity("push");
}
public void push(Operator j)/* throws StackFullException*/{
    if(isFull()){
        //throw new StackFullException("FULL!");
        resize();
    }
    ops.push(j);
    director.push(2);
    checkIntegrity("push");
}
public void pop() throws StackEmptyException{
    if(!isEmpty()){
        if(director.top()==0){
            director.pop();
            numbers.pop();
        } else if(director.top()==1) {
            director.pop();
            text.pop();
        } else if(director.top()==2) {
            director.pop();
            ops.pop();
        }
```

```
            checkIntegrity("pop");
        } else {
            throw new StackEmptyException("StackC EMPTY!");
        }
    }
    public Object top()  throws StackEmptyException{//** how to handle return
type?--todo
        //System.out.println(stackArray[top]);
        if(!isEmpty()){
            checkIntegrity("before top");
            if(director.top()==0){
                return numbers.top();
            } else if(director.top()==1) {
                return text.top();
            } else if(director.top()==2) {
                return ops.top();
            } else {
                throw new RuntimeException("StackC director "
                        + "invalid value:"+director.top());
            }
        } else {
            throw new StackEmptyException("EMPTY!");
        }
    }
    public Boolean isDoubleTop(){
        return director.top()==0;
    }
    public Boolean isOperatorTop(){
        return director.top()==2;
    }
    public boolean isEmpty() {
        return director.isEmpty();
    }
    public boolean isFull() {
        return director.isFull();
    }

    protected void resize(){
        director.resize();
        numbers.resize();
        text.resize();
        ops.resize();
        maxSize = maxSize*3/2;//from Stack<E>
        checkIntegrity("resize");
    }
    protected void resize(int newSize){
        director.resize(newSize);
```

```
    numbers.resize(newSize);
    text.resize(newSize);
    ops.resize(newSize);
    if(newSize>maxSize){//from Stack<E>
       maxSize=newSize;
    }
    checkIntegrity("resize");
 }


 //methods (++)
 public int size(){
    return director.size();
 }
 public void clear(){
    director.clear();
    numbers.clear();
    text.clear();
    ops.clear();
    checkIntegrity("clear");
 }
 public void reverse(){
    director.reverse();
    numbers.reverse();
    text.reverse();
    ops.reverse();
    checkIntegrity("reverse");
 }
 public int getMaxItems(){//++
    return maxSize;
 }
 public void print(){
    Stack<Integer> d=director.copyOf();
    Stack<Double> n=numbers.copyOf();
    Stack<String> t=text.copyOf();
    Stack<Operator> o=ops.copyOf();
    System.out.println("StackC (TOS-0):");
    while(!director.isEmpty()){
       if(this.isOperatorTop()){
          if(this.top()!=null){
             Operator tempo=(Operator)this.top();
             System.out.print("(op:"+tempo.indicator+"),");
          }
       } else {
          System.out.print(this.top()+",");
       }
       this.pop();
```

```
        }
        director=d.copyOf();
        numbers=n.copyOf();
        text=t.copyOf();
        ops=o.copyOf();
        checkIntegrity("print");
        System.out.println();
    }
}
```

U) StackCSyncExcpetion.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author student
 */
public class StackCSyncException extends RuntimeException{
    public StackCSyncException(String s){
        super(s);
    }//end constructor
}
```

V) StackEmptyException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author student
 */
public class StackEmptyException extends RuntimeException{
```

```
    public StackEmptyException(String s){
        super(s);
    }//end constructor
}
```

W) StackFullException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author student
 */
public class StackFullException extends RuntimeException{
    public StackFullException(String s){
        super(s);
    }//end constructor
}
```

X) StackInterface.java

```
package ph.edu.dlsu.chan.calculator;
/*@author Patrick*/
public interface StackInterface<E> {
    //methods
    public void push(E j) throws StackFullException;
    public void pop() throws StackEmptyException;
    public E top()  throws StackEmptyException;
    public boolean isEmpty();
    public boolean isFull();
    //resize()
}
```

Y) SyntaxException.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
```

```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ph.edu.dlsu.chan.calculator;

/**
 *
 * @author student
 */
public class SyntaxException extends RuntimeException{
    public SyntaxException(String s){
        super("Syntax Error:"+s);
    }//end constructor
}
```