Program Project:

Calculator for Arithmetic Expressions

The design of Multimodal User Interface

Human Computer Interaction

University at Albany

Sena Busacker & George Tackie

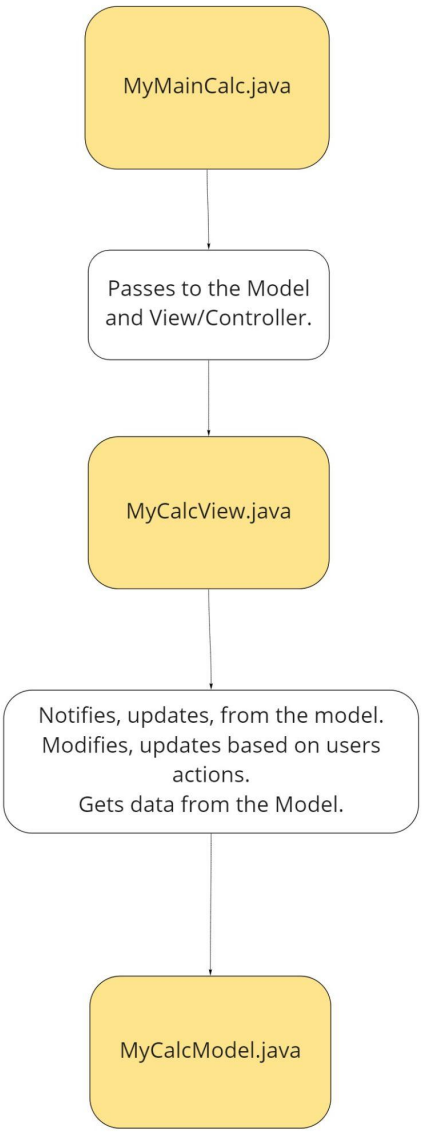
ICSI 407

TABLE OF CONTENTS

Data Flow Diagram for the System

Yellow: Class

White: Note



A list of routines and their brief descriptions

Three Classes are used throughout the design of Multimodal User Interface, the program is used the Model View Controller software Design Pattern. This is used to have actions to perform, design a display and calculations. Three classes are MyMainCalc.java MyCalcView.java MyCalcModel.java will be presenting a list of routines/functions with brief descriptions including names of what they are being used for. The color code will also be given. This will help out clear understanding of differences between comment, code and brief descriptions all throughout the pages.

Comment: Green
Code: Blue
Brief Descriptions: Black

MyMainCalc.java is programmed as a main program calling all classes and passing to the parts are needed. This will help to ensure only one be copied from each class.

```
class MyMainCalc {

The Main Method will get the system to update looks and feels and set the application.
public static void main(String args[]) {
   try {
       for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
           if ("Nimbus".equals(info.getName())) {
               javax.swing.UIManager.setLookAndFeel(info.getClassName());
               break;
           }//Close if
       }//Close for
   } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
   } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
   } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
   } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
   }//Close catch
   //</editor-fold>
```

Create and display the form by Tcalls both the MyCalView.java and MyCalModel.java to be parameters in the controller instance and handles all the eros for a backup.

```
   EventQueue.invokeLater(new Runnable() {
      public void run() {
          MyCalcModel theModel = new MyCalcModel();
          new MyCalcView(theModel);

      }//Close run
   });//Close invokeLater
}//Close Main Method
}//Close MyMainClac Class
```

MyCalcModel.java is created for expressions, addition, and multiplication of numbers respectively, as well as negative symbol is denoted for both binary operators of subtraction and the unary operators of taking the negatives. The Model's responsibilities are to provide access to the state of the system and the system's functionality of the non-negative power of expression, which is interpreted as X to the power of Y. This will have all the data and state all the calculations. as well as convert integers to double. In the code that is provided before for explicitly defending each name and all functions that are in each part.

MyCalcModel class takes a string and converts it to an expression takes the infix expression to a postfix expression.

```
public class MyCalcModel {
```

Combining both methods with their stacks to give calculation as a string

```
          public String stringToArithmetic(String string) {
                  double temp= suffixToArithmetic(infixToSuffix(string));
                  long x = (long)temp;
                  String str = Long.toString(x);
              return str;
            }//Close stringToArithmetic Method
```

Calculate operations only, each operators will react to other operations but this method takes unique operators such as unary operation, brackets and perform together and calculate.

```
          public static String infixToSuffix(String infix) {

               Stack< Character> stack = new Stack< Character>();
               String suffix = "";
               int length = infix.length();
               for (int i = 0; i < length; i++) {

                   Character temp ;
```

```
char c = infix.charAt(i);
switch (c) {

case '(':
   stack.push(c);
   break;

case '+':
          if (infix.charAt(i)=='+'&&( i==0||infix.charAt(i-1)=='(')){
            suffix += "0" + "";
   }//Close if

case '-':
   if (infix.charAt(i)=='-'&&( i==0||infix.charAt(i-1)=='(')){
            suffix += "0" + "";
   }//Close if
   while (stack.size() != 0) {
      temp = stack.pop();
      if (temp == '(') {
         stack.push('(');
         break;
      }//close if
      suffix += " " + temp;
   }//Close while
   stack.push(c);
   suffix += " ";
   break;
   theModel);



case '*':
   while (stack.size() != 0) {
      temp = stack.pop();
      if (temp == '(' || temp == '+' || temp == '-') {
         stack.push(temp);
         break;
      } else {
         suffix += " " + temp;
      }//Close else
   }//Close while
   stack.push(c);
   suffix += " ";
   break;

case '^':
   while (stack.size() != 0) {
      temp = stack.pop();
      if (temp == '*' ||temp == '(' || temp == '+' || temp == '-') {
         stack.push(temp);
```

```
                    break;
                } else {
                    suffix += " " + temp;
                }//Close else
            }//Close while
            stack.push(c);
            suffix += " ";
            break;

        case ')':
            while (stack.size() != 0) {
                temp = stack.pop();
                if (temp == '(') {
                    break;
                } else {
                    suffix += " " + temp;
                }//Close else
            }//Close while
            break;

        default:
            suffix += c;
        }//Close switch
    }//Close for
    while (stack.size() != 0) {
        suffix += " " + stack.pop();
    }//Close while
    return suffix;
}//Close infixToSuffix Method
```

Split each word to understand one by one string and use it as Stack method.

```
public static double suffixToArithmetic(String postfix) {
    Pattern pattern = Pattern.compile("\\d+");
    String strings[] = postfix.split(" ");
    for (int i = 0; i < strings.length; i++) {
        strings[i].trim();
    }//Close for
    Stack< Double> stack = new Stack< Double>();
    for (int i = 0; i < strings.length; i++) {
        if (strings[i].equals("")) {
            continue;
        }//Close if
        if ((pattern.matcher(strings[i])).matches()) {
            stack.push(Double.parseDouble(strings[i]));
        } else {
            double y = stack.pop();
            double x = stack.pop();
            stack.push(calculator(x, y, strings[i]));
        }//Close else
    }//Close for
```

```
            return stack.pop();
        }//Close suffixToArithmetic Method

Calculates each operator.
        private static double calculator(double x, double y, String symbol) {
            if (symbol.trim().equals("+")) {
                return x + y;
            }if (symbol.trim().equals("-")) {
                return x - y;
            }if (symbol.trim().equals("*")) {
                return x * y;
            } if (symbol.trim().equals("^")) {
                if (y>=0) {
                    return Math.pow(x,y);
                } else {
JOptionPane.showMessageDialog(null,"The power should be non-negative. Try again!");
                    System.out.println("The power should be non-negative. Try again!");
                }//Close else
                return 0;
            }//Close else
            return 0;
        }//Close calculator Method

}//Close MyCalcModel Class
```

Next, We will discuss how users will be able View and Control. The view would be able to create a frame that allows the user to see the user with a screen with a Model of a label displaying selectable buttons of numbers, able to quit by excessing in MenuBar, cancel input in a button of expressions, and operators and a text field in a single line able to edit the text. The user will view a simple application with a window also known as JFame. As for CalcController.java responsibility is to accept the user input such as button clicks, key press, mouse movements, can notify a user when they input a different operator or any alphabets, users will be notified with a message of error. This will be able to send messages to the model to notify the viewers. As for the two interaction modes, both View and the Controller are used to command *Speech to Text* and *Text to Speech.* Both can be switched between both and simultaneously reading the input out loud and able to speech to the application and write into a textfield both will read the total and the mathematical expressions. MyCalView.java also includes error handling. When the error occurs, a new panel with messages letting the user know what they did wrong and tell them what to do next. All of these performances of error, and convert string to int, able to *Speech to Tex*t, and *Text to Speech* will be dealt with ActionListener and actionPerformed methods.

In the View areas such as all the private and finals that are connected throughout the methods, the view for the calculator holds the buttons action listeners. As for the control, interact with the model.

```
public class MyCalcView {
```

Initializes will call the modelCal to calculate an expression by the numbers inserted into a textfield or use a button..

```
  public MyCalcView(MyCalcModel modelCon) {
```

Create a label name for the panel.

```
		JLabel lblNewLabel_1 = new JLabel("Calculator for Arithmetic Expressions ");
		lblNewLabel_1.setToolTipText("");
		lblNewLabel_1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
		GroupLayout gl_panel_1 = new GroupLayout(panel_1);
		gl_panel_1.setHorizontalGroup(
			gl_panel_1.createParallelGroup(Alignment.TRAILING)
				.addGap(0, 429, Short.MAX_VALUE)
				.addGroup(gl_panel_1.createSequentialGroup()
					.addContainerGap(87, Short.MAX_VALUE)
					.addComponent(lblNewLabel_1,
GroupLayout.PREFERRED_SIZE, 336, GroupLayout.PREFERRED_SIZE))
		);//Close setHorizontalGroup
		gl_panel_1.setVerticalGroup(
			gl_panel_1.createParallelGroup(Alignment.LEADING)
				.addGap(0, 42, Short.MAX_VALUE)
				.addComponent(lblNewLabel_1, Alignment.TRAILING,
GroupLayout.DEFAULT_SIZE, 36, Short.MAX_VALUE)
		);//Close setVerticalGroup
		panel_1.setLayout(gl_panel_1);


		zeroBtn = new JButton("0");
		zeroBtn.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				stringField.setText(stringField.getText()+"0");
			}//Close actionPerformed
		});//Close addActionListener
		zeroBtn.setBounds(39, 553, 70, 70);
		frame.getContentPane().add(zeroBtn);

		oneBtn = new JButton("1");
		oneBtn.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				stringField.setText(stringField.getText()+"1");
			}//Close actionPerformed
		});//Close addActionListener
		oneBtn.setBounds(39, 453, 70, 70);
		frame.getContentPane().add(oneBtn);

		twoBtn = new JButton("2");
		twoBtn.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				stringField.setText(stringField.getText()+"2");
			}//Close actionPerformed
```

```
});//Close addActionListener
twoBtn.setBounds(129, 453, 70, 70);
frame.getContentPane().add(twoBtn);

threeBtn = new JButton("3");
threeBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"3");
	}//Close actionPerformed
});//Close addActionListener
threeBtn.setBounds(219, 453, 70, 70);
frame.getContentPane().add(threeBtn);

fourBtn = new JButton("4");
fourBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"4");
	}//Close actionPerformed
});//Close addActionListener
fourBtn.setBounds(39, 353, 70, 70);
frame.getContentPane().add(fourBtn);

fiveBtn = new JButton("5");
fiveBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"5");
	}//Close actionPerformed
});//Close addActionListener
fiveBtn.setBounds(129, 353, 70, 70);
frame.getContentPane().add(fiveBtn);

sixBtn = new JButton("6");
sixBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"6");
	}//Close actionPerformed
});//Close addActionListener
sixBtn.setBounds(219, 353, 70, 70);
frame.getContentPane().add(sixBtn);

sevenBtn = new JButton("7");
sevenBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"7");
	}//Close actionPerformed
});//Close addActionListener
sevenBtn.setBounds(39, 253, 70, 70);
frame.getContentPane().add(sevenBtn);

eightBtn = new JButton("8");
```

```
eightBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"8");
	}//Close actionPerformed
});//Close addActionListener
eightBtn.setBounds(129, 253, 70, 70);
frame.getContentPane().add(eightBtn);

nineBtn = new JButton("9");
nineBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"9");
	}//Close actionPerformed
});//Close addActionListener
nineBtn.setBounds(219, 253, 70, 70);
frame.getContentPane().add(nineBtn);

addBtn = new JButton("+");
addBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"+");
	}//Close actionPerformed
});//Close addActionListener
addBtn.setBounds(309, 453, 70, 70);
frame.getContentPane().add(addBtn);

subBtn = new JButton("-");
subBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"-");
	}//Close actionPerformed
});//Close addActionListener
subBtn.setBounds(309, 353, 70, 70);
frame.getContentPane().add(subBtn);

multBtn = new JButton("*");
multBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"*");
	}//Close actionPerformed
});//Close addActionListener
multBtn.setBounds(309, 253, 70, 70);
frame.getContentPane().add(multBtn);

powBtn = new JButton("<html>x<SUP>y</SUP></html>");
powBtn.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		stringField.setText(stringField.getText()+"^");
	}//Close actionPerformed
});//Close addActionListener
```

```
                powBtn.setBounds(219, 153, 70, 70);
                frame.getContentPane().add(powBtn);
=
                leftPBtn = new JButton("(");
                leftPBtn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                stringField.setText(stringField.getText()+"(");
                        }//Close actionPerformed
                });//Close addActionListener
                leftPBtn.setBounds(39, 153, 70, 70);
                frame.getContentPane().add(leftPBtn);

                rightPBtn = new JButton(")");
                rightPBtn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                stringField.setText(stringField.getText()+")");
                        }//Close actionPerformed
                });//Close addActionListener
                rightPBtn.setBounds(129, 153, 70, 70);
                frame.getContentPane().add(rightPBtn);

                calcBtn = new JButton("Calculate");
                calcBtn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                note.setText("NOTE: It's been submitted.");
calcBtnActionPerformed(e);
                        }//Close submitbutton actionPerformed

                        private void calcBtnActionPerformed(ActionEvent evt) {
                  if (evt.getSource() == calcBtn) {
                        String expr = stringField.getText();
                  try {
                        outputbox.setText("= " +modelCon.stringToArithmetic(expr));
                }catch (Exception f) {
                                JOptionPane.showMessageDialog(null, "Wrong Expression.
Please try again!", "Error Message", JOptionPane.ERROR_MESSAGE);
                        System.out.println("Wrong Expression. Please try again!");
                        throw new IllegalStateException("Error");
                        }//Close else
                                }//Close if
                }//Close calcBtn ActionPerformed
                });//Close calcBtnActionPerformed addActionListener
                calcBtn.setBounds(219, 553, 160, 70);
                frame.getContentPane().add(calcBtn);

                clearBtn = new JButton("CLR");
                clearBtn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                stringField.setText(" ");
                                note.setText("");
```

```
                        }//Close actionPerformed
                    });//Close addActionListener
                    clearBtn.setBounds(129, 553, 70, 70);
                    frame.getContentPane().add(clearBtn);

                    textButton = new JRadioButton("Text to Speech");
                    textButton.addActionListener(new ActionListener() {
                            public void actionPerformed(ActionEvent e) {
                                    System.setProperty("freetts.voices",
"com.sun.speech.freetts.en.us.cmu_us_kal.KevinVoiceDirectory");
                    voice = VoiceManager.getInstance().getVoice("kevin16");
                    if(outputbox.getText().isEmpty() && outputbox.getText().isBlank()) {
                                            textButton.setEnabled(false); //To disable JTextField use
void setEnabled(boolean enabled)  method with false argument.
                                            JOptionPane.showMessageDialog(null, "You have not
entered the Calculate Button with an expression.\nPlease reload the application and try
again.", "Error Message", JOptionPane.ERROR_MESSAGE);
                                            System.out.println("Error Text Button.");
                    }else {
                    if (voice != null) {
                       voice.allocate();
                     if (e.getSource() == textButton) {
                    String expr = stringField.getText();
                       try {
                           voice.setRate(150);
                           voice.setPitch(120);
                           voice.setVolume(6);
                           voice.speak(stringField.getText() + " " + " " + outputbox.getText()); //
EMPTY QUOTE is to have the kevin to slowly speech, plause each worlds allowing user to
understand
                       }catch(Exception f) {
                            f.printStackTrace();
                    JOptionPane.showMessageDialog(null, "Wrong Expression. Please try again!",
"Error Message", JOptionPane.ERROR_MESSAGE);
                                                    System.out.println("Error in Text to Speech.");
                                                    throw new IllegalStateException("Cannot find
voice: kevin16");

                       }//Close catch
                     }//Close if
                    }//Close else
                                    }//Close else
                            }//Close actionPerformed
                    });//Close addActionListener
                    textButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
                    textButton.setBounds(299, 190, 119, 23);
                    frame.getContentPane().add(textButton);

                    speechButton = new JRadioButton("Speech to Text");
                    speechButton.addActionListener(new ActionListener() {
```

```
                        public void actionPerformed(ActionEvent e) {
                                note.setText("NOTE: Speech worked!");

                                logger.log(Level.INFO, "Loading..\n");

                                 Configuration configuration = new Configuration();

configuration.setAcousticModelPath("resource:/edu/cmu/sphinx/models/en-us/en-us");

configuration.setDictionaryPath("resource:/edu/cmu/sphinx/models/en-us/cmudict-en-us.dict");


                                        configuration.setGrammarPath("resource:/grammars");
                                        configuration.setGrammarName("grammar");
                                        configuration.setUseGrammar(true);

                                        logger.log(Level.INFO, "You can start to speak...\n");

                    try {
                        recognizer = new LiveSpeechRecognizer(configuration);
                        recognizer.startRecognition(true);
                        SpeechResult result = recognizer.getResult();

                    if (result != null) {
                        resultText = result.getHypothesis();

                        if (AudioSystem.isLineSupported(Port.Info.MICROPHONE)) {
                                                System.out.println("Microphone is available.");
                                                logger.log(Level.INFO, "Microphone is
available.\n");
                    } else {
                                logger.log(Level.INFO, "Microphone is not available.\n");
                                        }//Close else

                stringField.setText(wordsToNumbers(resultText));
outputbox.setText("= " + modelCon.stringToArithmetic(wordsToNumbers(resultText)));
System.out.println("You said: "+wordsToNumbers(resultText));
                        recognizer.stopRecognition();
                    }else
                                logger.log(Level.INFO, "I can't understand what you said.\n");
                                }catch (IOException e1) {
                        e1.printStackTrace();
                        JOptionPane.showMessageDialog(null, "ERROR", "Error Message",
JOptionPane.ERROR_MESSAGE);
                                        System.out.println("Error in Speech to Text.");
                                        logger.log(Level.WARNING, null, e1);
                                        logger.log(Level.WARNING, null, e1);
                                        resourcesThread.interrupt();
                    }//Close  catch
                    logger.log(Level.INFO, "SpeechThread has exited...");
```

```
                    }//Close actionPerformed
                });//Close addActionListener

                speechButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
                speechButton.setBounds(299, 219, 128, 23);
                frame.getContentPane().add(speechButton);

                separator = new JSeparator();
                separator.setBounds(310, 177, 103, 2);
                frame.getContentPane().add(separator);

                mb = new JMenuBar();
                mb.setBounds(0, 0, 449, 22);
                frame.getContentPane().add(mb);
                x = new JMenu("Menu");

        m1 = new JMenuItem("Help");
        m1.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_H,
ActionEvent.CTRL_MASK));
        m1.setBackground(SystemColor.inactiveCaption);
        m1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                     String what = e.getActionCommand();
                if(what.equals("Help")) {

                        JOptionPane.showMessageDialog(frame,
                                                "How do I use this?\r\n" + "Press any number
you wish to input and use operation and then clear to restart calculate\r\n\n"
                                                + "How do I exit out?\r\n" + "There are 3
ways user can exit out from the screen\r\n"
                                                + "1: Press Ctrl and hold down and press
Q to quit\r\n"
                                                + "2: Go to Menu bar and press Quit\r\n"
                                                + "3: Press X on top right corner"
                                                + "\n\nHow to use Text to Speech Button?
Seems like it's not doing anything.\n"
                                                + "If you are pressing "Text to Speech"
buttons first time when your opened the app, \nYou have not entered the Calculate Button
with a expression. So, the Button will not work due to not \nhaving any numbers and
operations in, please reload the application and try again with putting in numbers and
operations.");
                        System.out.println("Help Center now exited.");
                }else{
                        System.out.println("ERROR");
                }//Close else
        }//Close Help addActionListener
        });//Close Help addActionListener

        m2 = new JMenuItem("Quit");
        m2.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q,
```

```
ActionEvent.CTRL_MASK));
        m2.setBackground(SystemColor.inactiveCaption);
        m2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                    frame = new JFrame("Quit");
                    if(JOptionPane.showConfirmDialog(frame, "Confirm if you want to quit") ==
JOptionPane.YES_NO_OPTION) {
                                System.out.println("Good Bye.");
                               System.exit(0);
                    }//Close if
            }//Close Quit actionPerformed
        });//Close Quit addActionListener

        x.add(m1);
        x.add(m2);

        mb.add(x);
       note = new JLabel("");
        note.setBounds(27, 634, 409, 25);
        frame.getContentPane().add(note);

        outputbox = new JTextField();
        outputbox.setFont(new Font("Ink Free", Font.BOLD, 24));
        outputbox.setEditable(false);
        outputbox.setBounds(219, 86, 184, 50);
        frame.getContentPane().add(outputbox);

            }//Close MyCalcView Initialize
```

Deals with breaking a paragraph down into individual words into numbers and operators. There are some words and operators that are not written, This is due to whomever does use the application that does accept can still pick up numbers and operators and converts into the integers. Which could consist of a number of words.

```
  public String wordsToNumbers(String words) {
      newString = "";
      String[] splitWords = words.split(" ");
      for(String word : splitWords) {
          switch(word) {
              case "zero":
              newString += word.replace("zero", "0");
              break;
              case "hero":
              newString += word.replace("zero", "0");
              break;
              case "one":
              newString += word.replace("one", "1");
              break;
              case "two":
              newString += word.replace("two", "2");
              break;
```

```
            case "too":
            newString += word.replace("too", "2");
            break;
            case "three":
            newString += word.replace("three", "3");
            break;
            case "tree":
            newString += word.replace("tree", "3");
            break;
            case "free":
            newString += word.replace("free", "3");
            break;
            case "four":
            newString += word.replace("four", "4");
            break;
            case "for":
            newString += word.replace("for", "4");
            break;
            case "five":
            newString += word.replace("five", "5");
            break;
            case "six":
            newString += word.replace("six", "6");
            break;
            case "sex":
            newString += word.replace("sex", "6");
            break;
            case "seven":
            newString += word.replace("seven", "7");
            break;
            case "eight":
            newString += word.replace("eight", "8");
            break;
            case "ate":
            newString += word.replace("ate", "8");
            break;
            case "nine":
            newString += word.replace("nine", "9");
            break;
            case "ten":
            newString += word.replace("ten", "10");
            break;
           case "plus":
            newString += word.replace("plus", "+");
            break;
            case "minus":
            newString += word.replace("minus", "-");
            break;
            case "times":
            newString += word.replace("times", "*");
```

```
            break;
            case "multiply":
            newString += word.replace("multiply", "*");
            break;
            case "power":
            newString += word.replace("power", "^");
            break;
            case "powder":
            newString += word.replace("powder", "^");
            break;
            case "carrot":
            newString += word.replace("carrot", "^");
            break;
            case "left":
            newString += word.replace("left", "(");
            break;
            case "bracket":
            newString += word.replace("bracket", "(");
            break;
            case "right":
            newString += word.replace("right", ")");
            break;
            case "write":
            newString += word.replace("write", ")");
            break;
        }//Close switch
    }//Close for loop
    return newString;
  }//Close wordsToNumbers

}//Close MyCalcView class
```

Implementation details

This section will give an understanding of what types of implementations such as data structures and specific algorithms and others are being used. I'll be using the same program earlier on in page 4 As well as code coloring down below.

Comment: Green
Code: Blue
Details: Black

MyMainCalc.java

```
class MyMainCalc {

public static void main(String args[]) {
```

Try and Catch statement will check for any errors in the MyCalView.java called by inside the mainstream. If an error is thrown then the cross platform will be called else the catch will catch any errors and try to handle the problem.

```
   try {
       for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
           if ("Nimbus".equals(info.getName())) {
               javax.swing.UIManager.setLookAndFeel(info.getClassName());
               break;
           }//Close if
       }//Close for
   } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
   } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
   } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
   } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
   }//Close catch

   EventQueue.invokeLater(new Runnable() {
       public void run() {
```

Call the method

```
           MyCalcModel theModel = new MyCalcModel();
            new MyCalcView(theModel);
```

```
      }//Close run
   });//Close invokeLater
}//Close Main Method
}//Close MyMainCalc Class
```

MyCalcModel.java

```
public class MyCalcModel {

        public String stringToArithmetic(String string) {
Setting a variable double into temp, this allows to call the all the method and into one.
                double temp= suffixToArithmetic(infixToSuffix(string));
Convert double into long since the bits are held bigger. Can hold all the numbers and
operators.
                long x = (long)temp;
Converts into string, this allows the text field to calculate and be able to use in getText, setText
for Text to Speech and Speech to text. setText only allows strings input inside, so this is why
to convert into strings.
                String str = Long.toString(x);
Return str so it can be printed out in the textfield.
              return str;
            }//Close stringToArithmetic Method

        public static String infixToSuffix(String infix) {
Stack will contain character objects in the and cast char 's' to character.
              Stack< Character> stack = new Stack< Character>();
Suffix allows it to initialize postfix as an empty string.
              String suffix = "";
Length will help to loop through array , as one character at a time until the reaches the end of
the string
              int length = infix.length();
              for (int i = 0; i < length; i++) {
Assigning each operations.
                Character temp ;
Assigning character into the stacks.
                char c = infix.charAt(i);
If a user typed any one of these operators or more, it will go through each and set into proper
math order. In another words, this is setting it into PEMDAS.
                switch (c) {

If user typed in, it will scan the character '(' and push  it to the stack
                case '(':
                   stack.push(c);
                   break;
If the scanned character '+' is chosen then the if statement will print any possible operators
may come up with in the unary operators or the brackets.
```

```
                case '+':
                        if (infix.charAt(i)=='+'&&( i==0||infix.charAt(i-1)=='(')){
                          suffix += "0" + "";
                    }//Close if
```

If the scanned character '-' is chosen then the if statement will handle unary and breasts as will as all others operators to deal with unary and binary operators everytime in while loop, if the size is not equal to zero or non at all.

```
                case '-':
                    if (infix.charAt(i)=='-'&&( i==0||infix.charAt(i-1)=='(')){
                            suffix += "0" + "";
                    }//Close if
                    while (stack.size() != 0) {
                        temp = stack.pop();
                        if (temp == '(') {
                            stack.push('(');
                            break;
                        }//close if
                        suffix += " " + temp;
                    }//Close while
                    stack.push(c);
                    suffix += " ";
                    break;
```

If the scanned character '*' is chosen then the if statement will print any possible operators may come up with in the unary operators or the brackets.

```
                case '*':
                    while (stack.size() != 0) {
                        temp = stack.pop();
                        if (temp == '(' || temp == '+' || temp == '-') {
                            stack.push(temp);
                            break;
                        } else {
                            suffix += " " + temp;
                        }//Close else
                    }//Close while
                    stack.push(c);
                    suffix += " ";
                    break;
```

If the scanned character '^' is chosen then the if statement will print any possible operators may come up with in the unary operators or the brackets.

```
                case '^':
                    while (stack.size() != 0) {
                        temp = stack.pop();
                        if (temp == '*' ||temp == '(' || temp == '+' || temp == '-') {
                            stack.push(temp);
                            break;
                        } else {
                            suffix += " " + temp;
                        }//Close else
                    }//Close while
                    stack.push(c);
```

```
                    suffix += " ";
                    break;
```
If the scanned character ‘)’ is chosen then the if statement will print any possible operators may come up with in the unary operators or the brackets.
```
                case ')':
                    while (stack.size() != 0) {
                        temp = stack.pop();
                        if (temp == '(') {
                            break;
                        } else {
                            suffix += " " + temp;
                        }//Close else
                    }//Close while
                    break;
```
If any operators are defiant, suffix will be assigned to character.
```
                default:
                    suffix += c;
                }//Close switch
            }//Close for
```
If the stack size is zero or nothing at all, it will check each time and then pop.
```
            while (stack.size() != 0) {
                suffix += " " + stack.pop();
            }//Close while
            return suffix;
        }//Close infixToSuffix Method

        public static double suffixToArithmetic(String postfix) {
            Pattern pattern = Pattern.compile("\\d+");
```
Split each word to understand one by one string
```
            String strings[] = postfix.split(" ");
```
Cut Strings individually and trim them.
```
            for (int i = 0; i < strings.length; i++) {
                strings[i].trim();
            }//Close for
```
Assigning stack into double allowing this to connect and use it to pop from the top of the stack.
```
            Stack< Double> stack = new Stack< Double>();
```
Checks every strings
```
            for (int i = 0; i < strings.length; i++) {
                if (strings[i].equals("")) {
                    continue;
                }//Close if
```
Match the patterns, stitch it back together.
```
                if ((pattern.matcher(strings[i])).matches()) {
                    stack.push(Double.parseDouble(strings[i]));
                } else {
```
X is for first integer and Y is for second, and strings will be used for the operator.
```
                    double y = stack.pop();
                    double x = stack.pop();
                    stack.push(calculator(x, y, strings[i]));
```

```
                }//Close else
            }//Close for
            return stack.pop();
        }//Close suffixToArithmetic Method

        private static double calculator(double x, double y, String symbol) {
Calculator addition and returns so that it's able to print out on display.
            if (symbol.trim().equals("+")) {
                return x + y;
Calculator subtraction and returns so that it's able to print out on display.
            }if (symbol.trim().equals("-")) {
                return x - y;
Calculator multiplication  and returns so that it's able to print out on display.
            }if (symbol.trim().equals("*")) {
                return x * y;
Calculator power and returns so that it's able to print out on display.
            } if (symbol.trim().equals("^")) {
                if (y>=0) {
                    return Math.pow(x,y);
                } else {
Prints out an error message if the power is negative in both display and the console.
    JOptionPane.showMessageDialog(null,"The power should be non-negative. Try again!");
                    System.out.println("The power should be non-negative. Try again!");
                }//Close else
                return 0;
            }//Close else
            return 0;
        }//Close calculator Method

}//Close MyCalcModel Class
```

MyCalView.java

```
public class MyCalcView {
        Logger allows to let the user know when to talk, when the program started running and
when it stopped, onl in the Speech to Text settings.
        private Logger logger = Logger.getLogger(getClass().getName()); // Logger

Setting it to private, it does not allow any outsider of CalView class aren't allow to access the
inside the CalView.
Setting it to final, it does entity that can only be assigned to once, once the final is used, it will
always contain the same value.
These down below are to set all the users interface components such as buttons is Jbutton.
Able to press buttons, labels are JLable. Show text on the screen. JLable is an unselectable
text and images.Jmenu allows to have a menu where the user can choose from the help to
quit.  Lastly, stringField. Outputbox are known as JTextField in java. This allows a single line of
input that is editable into a stringField and enabled to edit into the outputbox.


        private JFrame frame;
```

```
        private JSeparator separator;
        private String newString;
        private JTextField outputbox;
        final JTextField stringField;
        final JRadioButton speechButton;
        final JRadioButton textButton;
        final JLabel ModeSelection;
        final JButton clearBtn;
        final JButton calcBtn;
        final JButton rightPBtn;
        final JButton leftPBtn;
        final JButton powBtn;
        final JButton multBtn;
        final JButton subBtn;
        final JButton addBtn;
        final JButton nineBtn;
        final JButton eightBtn;
        final JButton sevenBtn;
        final JButton sixBtn;
        final JButton fiveBtn;
        final JButton fourBtn;
        final JButton threeBtn;
        final JButton twoBtn;
        final JButton oneBtn;
        final JButton zeroBtn;
        private JMenu x; // JMenu
        private JMenuItem m1, m2;  // Menu items
```

Letting the user know when pressing “calculate”, will notify the user that the calculation is wrong. Also is being used in the speech text if the microphone catches the user’s voice it will notify the user that speech as worked.

```
        private JLabel note;
```

Get results into the text box from Speech to Text.

```
      private String resultText;
        private JMenuBar mb;

/*Voice*/
```

Setting a voice allowing it to *Text to Speech*

```
        Voice voice;

/*LiveRecognizer*/
```

When a user starts to speech, LiveRecongizer will pick up any sounds

```
        private LiveSpeechRecognizer recognizer;

/*Threads*/
```

resourcesTread allows to not other interpret to input into the microphone and pick up other backgrounds.

```
                Thread resourcesThread;

        /**
```

```
        * Initialize the contents of the frame.
        * @param theModel
        */
  public MyCalcView(MyCalcModel modelCon) {
```

Set up the frame

```
                frame = new JFrame("Arithmetic Expressions Calculator");
                frame.setBounds(100, 100, 463, 697);
                frame.setResizable(false);
                frame.getContentPane().setFont(new Font("Times New Roman", Font.PLAIN,
20));
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.getContentPane().setLayout(null);
      frame.setVisible(true);
      frame.setLocationRelativeTo(null);
```

Create a panel

```
                JPanel panel_1 = new JPanel();
                panel_1.setBorder(new LineBorder(new Color(0, 0, 0), 3));
                panel_1.setBounds(10, 33, 429, 42);
                frame.getContentPane().add(panel_1);
```

Create label name for the panel

```
                JLabel lblNewLabel_1 = new JLabel("Calculator for Arithmetic Expressions ");
                lblNewLabel_1.setToolTipText("");
                lblNewLabel_1.setFont(new Font("Times New Roman", Font.PLAIN, 16));
                GroupLayout gl_panel_1 = new GroupLayout(panel_1);
                gl_panel_1.setHorizontalGroup(
                        gl_panel_1.createParallelGroup(Alignment.TRAILING)
                                .addGap(0, 429, Short.MAX_VALUE)
                                .addGroup(gl_panel_1.createSequentialGroup()
                                        .addContainerGap(87, Short.MAX_VALUE)
                                        .addComponent(lblNewLabel_1,
GroupLayout.PREFERRED_SIZE, 336, GroupLayout.PREFERRED_SIZE))
                );//Close setHorizontalGroup
                gl_panel_1.setVerticalGroup(
                        gl_panel_1.createParallelGroup(Alignment.LEADING)
                                .addGap(0, 42, Short.MAX_VALUE)
                                .addComponent(lblNewLabel_1, Alignment.TRAILING,
GroupLayout.DEFAULT_SIZE, 36, Short.MAX_VALUE)
                );//Close setVerticalGroup
                panel_1.setLayout(gl_panel_1);
```

To create a Text Field for input for user

```
                stringField = new JTextField();
                stringField.setFont(new Font("Ink Free", Font.BOLD, 24));
                stringField.setBounds(39, 86, 184, 50);
                frame.getContentPane().add(stringField);
                stringField.setEditable(true);
```

To create a Label for Mode Selection

```
ModeSelection = new JLabel("Mode Selection");
ModeSelection.setHorizontalAlignment(SwingConstants.CENTER);
ModeSelection.setFont(new Font("Yu Gothic Medium", Font.PLAIN, 16));
ModeSelection.setBounds(300, 154, 127, 23);
frame.getContentPane().add(ModeSelection);
```

To create a Button for with a words “zero” in a box

```
zeroBtn = new JButton("0");
zeroBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+"0");
        }//Close actionPerformed
});//Close addActionListener
zeroBtn.setBounds(39, 553, 70, 70);
frame.getContentPane().add(zeroBtn);
```

To create a Button for with a words “one” in a box

```
oneBtn = new JButton("1");
oneBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+"1");
        }//Close actionPerformed
});//Close addActionListener
oneBtn.setBounds(39, 453, 70, 70);
frame.getContentPane().add(oneBtn);
```

To create a Button for with a words “two” in a box

```
twoBtn = new JButton("2");
twoBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+"2");
        }//Close actionPerformed
});//Close addActionListener
twoBtn.setBounds(129, 453, 70, 70);
frame.getContentPane().add(twoBtn);
```

To create a Button for with a words “three” in a box

```
threeBtn = new JButton("3");
threeBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+"3");
```

```
                }//Close actionPerformed
            });//Close addActionListener
            threeBtn.setBounds(219, 453, 70, 70);
            frame.getContentPane().add(threeBtn);
```

To create a Button for with a words “four” in a box

```
            fourBtn = new JButton("4");
            fourBtn.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                    stringField.setText(stringField.getText()+"4");
                }//Close actionPerformed
            });//Close addActionListener
            fourBtn.setBounds(39, 353, 70, 70);
            frame.getContentPane().add(fourBtn);
```

To create a Button for with a words “five” in a box

```
            fiveBtn = new JButton("5");
            fiveBtn.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                    stringField.setText(stringField.getText()+"5");
                }//Close actionPerformed
            });//Close addActionListener
            fiveBtn.setBounds(129, 353, 70, 70);
            frame.getContentPane().add(fiveBtn);
```

To create a Button for with a words “six” in a box

```
            sixBtn = new JButton("6");
            sixBtn.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                    stringField.setText(stringField.getText()+"6");
                }//Close actionPerformed
            });//Close addActionListener
            sixBtn.setBounds(219, 353, 70, 70);
            frame.getContentPane().add(sixBtn);
```

To create a Button for with a words “seve” in a box

```
            sevenBtn = new JButton("7");
            sevenBtn.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                    stringField.setText(stringField.getText()+"7");
                }//Close actionPerformed
            });//Close addActionListener
```

```
sevenBtn.setBounds(39, 253, 70, 70);
frame.getContentPane().add(sevenBtn);
```

To create a Button for with a words “eight” in a box

```
eightBtn = new JButton("8");
eightBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                stringField.setText(stringField.getText()+"8");
        }//Close actionPerformed
});//Close addActionListener
eightBtn.setBounds(129, 253, 70, 70);
frame.getContentPane().add(eightBtn);
```

To create a Button for with a words “nine” in a box

```
nineBtn = new JButton("9");
nineBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+"9");
        }//Close actionPerformed
});//Close addActionListener
nineBtn.setBounds(219, 253, 70, 70);
frame.getContentPane().add(nineBtn);
```

To create a Button for with a character of operation “add” in a box

```
addBtn = new JButton("+");
addBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+"+");
        }//Close actionPerformed
});//Close addActionListener
addBtn.setBounds(309, 453, 70, 70);
frame.getContentPane().add(addBtn);
```

To create a Button for with a character of operation “subtract” in a box

```
subBtn = new JButton("-");
subBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+"-");
        }//Close actionPerformed
});//Close addActionListener
subBtn.setBounds(309, 353, 70, 70);
frame.getContentPane().add(subBtn);
```

To create a Button for with a character of operation “multiplication” in a box

```
multBtn = new JButton("*");
multBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                stringField.setText(stringField.getText()+"*");
        }//Close actionPerformed
});//Close addActionListener
multBtn.setBounds(309, 253, 70, 70);
frame.getContentPane().add(multBtn);
```

To create a Button for with a character of operation “power” in a box

```
powBtn = new JButton("<html>x<SUP>y</SUP></html>");
powBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                stringField.setText(stringField.getText()+"^");
        }//Close actionPerformed
});//Close addActionListener
powBtn.setBounds(219, 153, 70, 70);
frame.getContentPane().add(powBtn);
```

To create a Button for with a character of operation “ left bracket” in a box

```
leftPBtn = new JButton("(");
leftPBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+"(");
        }//Close actionPerformed
});//Close addActionListener
leftPBtn.setBounds(39, 153, 70, 70);
frame.getContentPane().add(leftPBtn);
```

To create a Button for with a character of operation “right bracket” in a box

```
rightPBtn = new JButton(")");
rightPBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

When a user presses the button it will show what pressed and output the result into the textfield.

```
                stringField.setText(stringField.getText()+")");
        }//Close actionPerformed
});//Close addActionListener
rightPBtn.setBounds(129, 153, 70, 70);
frame.getContentPane().add(rightPBtn);
```

To create a Button for with a words “calculator” in a box

```
calcBtn = new JButton("Calculate");
calcBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

WHen user does press the button, will show its been submitted to the user

```
                note.setText("NOTE: It's been submitted."); //Tell the user input
was put in.
```

Calling a method constructActionPerformed(evt)

```
                    calcBtnActionPerformed(e)
                }//Close submitbutton actionPerformed

                private void calcBtnActionPerformed(ActionEvent evt) {
            if (evt.getSource() == calcBtn) {
```

Try and Catch statement will check for any errors in the strings. The string such as any alphabets or non related to calculation operation and division will give an error message as well as in the console, saying “Wrong Expression. Please try again!”. The error message is put into the answer and the actionPerformed or as is listen down as the ActionListener method ends.

```
        String expr = stringField.getText();
            try {
                outputbox.setText("= " +modelCon.stringToArithmetic(expr));
                //outputbox.setText("= " +modelCon.stringToArithmetic(expr));
            }catch (Exception f) {
                    JOptionPane.showMessageDialog(null, "Wrong Expression.
Please try again!", "Error Message", JOptionPane.ERROR_MESSAGE);
                System.out.println("Wrong Expression. Please try again!");
                throw new IllegalStateException("Error");
                }//Close else
                    }//Close if
            }//Close calcBtn ActionPerformed
            });//CLose calcBtnActionPerformed addActionListener
            calcBtn.setBounds(219, 553, 160, 70);
            frame.getContentPane().add(calcBtn);
```

To create a Button for with a words “clear” in a box

```
            clearBtn = new JButton("CLR");
            clearBtn.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
Empty set will clear out in both text field called stringField and note.
                    stringField.setText(" ");
                    note.setText("");
                }//Close actionPerformed
            });//Close addActionListener
            clearBtn.setBounds(129, 553, 70, 70);
            frame.getContentPane().add(clearBtn);
```

To create a Button for with word “Text to Speech”

```
            textButton = new JRadioButton("Text to Speech");
            textButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    System.setProperty("freetts.voices",
"com.sun.speech.freetts.en.us.cmu_us_kal.KevinVoiceDirectory");
```

Default voice values, inside its name or the voice/

```
            voice = VoiceManager.getInstance().getVoice("kevin16");
```

When there is nothing in both textbox, it will send an error messages saying there is nothing in the bo, so you must restart the application.

```
            if(outputbox.getText().isEmpty() && outputbox.getText().isBlank()) {
```

Disable the textfield, not able to edit into the text box, so they won't change the total of the

calculation.

```
                                        textButton.setEnabled(false); //To disable JTextField use
void setEnabled(boolean enabled)  method with false argument.
                                        JOptionPane.showMessageDialog(null, "You have not
entered the Calculate Button with an expression.\nPlease reload the application and try again.",
"Error Message", JOptionPane.ERROR_MESSAGE);
                                        System.out.println("Error Text Button.");
```

If user did type or pressed to calculate and press the calciation, than it will send out a voice to user.

```
                }else {
                if (voice != null) {
```

Loads the voice.

```
                    voice.allocate();
```

Try to catch al the rate , pitch and volume as well as speech what evver in stringfield and outputbox textfield they are else catches all the errors.

```
                 if (e.getSource() == textButton) {
                    try {
                       voice.setRate(150);
                       voice.setPitch(120);
                       voice.setVolume(6);
                       voice.speak(stringField.getText() + " " + " " + outputbox.getText()); //
EMPTY QUOTE is to have the kevin to slowly speech, plause each worlds allowing user to
understand
                    }catch(Exception f) {
                          f.printStackTrace();
                JOptionPane.showMessageDialog(null, "Wrong Expression. Please try again!",
"Error Message", JOptionPane.ERROR_MESSAGE);
                                                System.out.println("Error in Text to Speech.");
                                                throw new IllegalStateException("Cannot find
voice: kevin16");

                    }//Close catch
                 }//Close if
                }//Close else
                                }//Close else
                           }//Close actionPerformed
                });//Close addActionListener
                textButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
                textButton.setBounds(299, 190, 119, 23);
                frame.getContentPane().add(textButton);
```

To create a Button for with words “Speech to Text”

```
                speechButton = new JRadioButton("Speech to Text");
                speechButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                note.setText("NOTE: Speech worked!");
```

Loading Message only when the Speech to Text is pressed.

```
                logger.log(Level.INFO, "Loading..\n");
```

Configuration allows the application not specify the properties.

```
                Configuration configuration = new Configuration();
```
Load model from the jar
```
configuration.setAcousticModelPath("resource:/edu/cmu/sphinx/models/en-us/en-us");

configuration.setDictionaryPath("resource:/edu/cmu/sphinx/models/en-us/cmudict-en-us.dict");
```
Grammar helps for both the user and the program. Only finds words are simarr and finds it in a grammar file and sets it to true.
```
                                        configuration.setGrammarPath("resource:/grammars");
                                        configuration.setGrammarName("grammar");
                                        configuration.setUseGrammar(true);
```
Start talking only when the Speech to Text is pressed.
```
                                        logger.log(Level.INFO, "You can start to speak...\n");
```
Try and Catch statement will check for any errors in speech. The string such as any alphabets or non related to calculation operation and division will give an error message as well as in the console, saying “Wrong Expression. Please try again!”. The error message is put into the answer and the actionPerformed or as is listen down as the ActionListener method ends.
```
                try {
                    recognizer = new LiveSpeechRecognizer(configuration);
                    recognizer.startRecognition(true);
                    SpeechResult result = recognizer.getResult();

                if (result != null) {
                    resultText = result.getHypothesis();
```
Letting user know if microphone is on or not
```
                    if (AudioSystem.isLineSupported(Port.Info.MICROPHONE)) {
```
Creating a print out in console to let the user know if microphone is on or not.
```
                                            System.out.println("Microphone is available.");
                                            logger.log(Level.INFO, "Microphone is
available.\n");
                                        } else {
                                            logger.log(Level.INFO, "Microphone is not
available.\n");
                                        }//Close else
```
Output the result of calculation in console and the application
```
                    stringField.setText(wordsToNumbers(resultText)); //words user said
                    outputbox.setText("= " +
modelCon.stringToArithmetic(wordsToNumbers(resultText))); //print the result
                    System.out.println("You said: "+wordsToNumbers(resultText));
                    recognizer.stopRecognition();
```
Handle errors - Microphone didn't pick up any sounds, inform the user.
```
                }else
                                logger.log(Level.INFO, "I can't understand what you said.\n");
                        }catch (IOException e1) {
                    e1.printStackTrace();
                    JOptionPane.showMessageDialog(null, "ERROR", "Error Message",
JOptionPane.ERROR_MESSAGE);
```

```
                                        System.out.println("Error in Speech to Text.");
                                        logger.log(Level.WARNING, null, e1);
                                        logger.log(Level.WARNING, null, e1);
                                        resourcesThread.interrupt();
                }//Close  catch
                logger.log(Level.INFO, "SpeechThread has exited...");
                        }//Close actionPerformed
                });//Close addActionListener
```

To Create font, size, and frame for Speech to Text button

```
                speechButton.setFont(new Font("Tahoma", Font.PLAIN, 14));
                speechButton.setBounds(299, 219, 128, 23);
                frame.getContentPane().add(speechButton);
```

To create separator to separate between Jlabel and buttons

```
                separator = new JSeparator();
                separator.setBounds(310, 177, 103, 2);
                frame.getContentPane().add(separator);
```

To Create Menu Bar

```
                mb = new JMenuBar();
                mb.setBounds(0, 0, 449, 22);
                frame.getContentPane().add(mb);
                x = new JMenu("Menu");
```

To Create Menu Items for Help

```
        m1 = new JMenuItem("Help");
        m1.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_H, ActionEvent.CTRL_MASK));
        m1.setBackground(SystemColor.inactiveCaption);
        m1.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
                    String what = e.getActionCommand();
                if(what.equals("Help")) {

                        JOptionPane.showMessageDialog(frame,
                                                "How do I use this?\r\n" + "Press any number you
wish to input and use operation and then clear to restart calculate\r\n\n"
                                                + "How do I exit out?\r\n" + "There are 3
ways user can exit out from the screen\r\n"
                                                + "1: Press Ctrl and hold down and press Q
to quit\r\n"
                                                + "2: Go to Menu bar and press Quit\r\n"
                                                + "3: Press X on top right corner"
                                                + "\n\nHow to use Text to Speech Button?
Seems like it's not doing anything.\n"
                                                + "If you are pressing "Text to Speech"
buttons first time when your opened the app, \nYou have not entered the Calculate Button with
a expression. So, the Button will not work due to not \nhaving any numbers and operations in,
please reload the application and try again with putting in numbers and operations.");
                        System.out.println("Help Center now exited.");
                }else{
```

```
                        System.out.println("ERROR");
                    }//Close else
        }//Close Help addActionListener
        });//Close Help addActionListener

 To Create Menu Items for Help
        m2 = new JMenuItem("Quit");
        m2.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q, ActionEvent.CTRL_MASK));
        m2.setBackground(SystemColor.inactiveCaption);
        m2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                    frame = new JFrame("Quit");
To confirm with the user if that's what they really want to do.
                    if(JOptionPane.showConfirmDialog(frame, "Confirm if you want to quit") ==
JOptionPane.YES_NO_OPTION) {
                        System.out.println("Good Bye.");
                        System.exit(0);
                    }//Close if
            }//Close Quit actionPerformed
        });//Close Quit addActionListener

Add menu items to menu
        x.add(m1);
        x.add(m2);

Add menu to menu bar
        mb.add(x);

To create Label for the NOTE, letting user know the input has successfully went through
        note = new JLabel("");
        note.setBounds(27, 634, 409, 25);
        frame.getContentPane().add(note);

To create a text field for Outputbox and is disabled for editing into the textfield.
        outputbox = new JTextField();
        outputbox.setFont(new Font("Ink Free", Font.BOLD, 24));
        outputbox.setEditable(false);
        outputbox.setBounds(219, 86, 184, 50);
        frame.getContentPane().add(outputbox);

            }//Close MyCalcView Initialize

To Convert String to int whenever user says its says as of string so it will now replace to integer
    public String wordsToNumbers(String words) {
        newString = "";
        String[] splitWords = words.split(" ");
        for(String word : splitWords) {
            switch(word) {
                case "zero":
                newString += word.replace("zero", "0");
```

```
            break;
            case "hero":
            newString += word.replace("zero", "0");
            break;
            case "one":
            newString += word.replace("one", "1");
            break;
            case "two":
            newString += word.replace("two", "2");
            break;
            case "too":
            newString += word.replace("too", "2");
            break;
            case "three":
            newString += word.replace("three", "3");
            break;
            case "tree":
            newString += word.replace("tree", "3");
            break;
            case "free":
            newString += word.replace("free", "3");
            break;
            case "four":
            newString += word.replace("four", "4");
            break;
            case "for":
            newString += word.replace("for", "4");
            break;
            case "five":
            newString += word.replace("five", "5");
            break;
            case "six":
            newString += word.replace("six", "6");
            break;
            case "sex":
            newString += word.replace("sex", "6");
            break;
            case "seven":
            newString += word.replace("seven", "7");
            break;
            case "eight":
            newString += word.replace("eight", "8");
            break;
            case "ate":
            newString += word.replace("ate", "8");
            break;
            case "nine":
            newString += word.replace("nine", "9");
            break;
            case "ten":
```

```
            newString += word.replace("ten", "10");
            break;
           case "plus":
            newString += word.replace("plus", "+");
            break;
            case "minus":
            newString += word.replace("minus", "-");
            break;
            case "times":
            newString += word.replace("times", "*");
            break;
            case "multiply":
            newString += word.replace("multiply", "*");
            break;
            case "power":
            newString += word.replace("power", "^");
            break;
            case "powder":
            newString += word.replace("powder", "^");
            break;
            case "carrot":
            newString += word.replace("carrot", "^");
            break;
            case "left":
            newString += word.replace("left", "(");
            break;
            case "bracket":
            newString += word.replace("bracket", "(");
            break;
            case "right":
            newString += word.replace("right", ")");
            break;
            case "write":
            newString += word.replace("write", ")");
            break;
         }//Close switch
      }//Close for loop
      return newString; Returns a string allowing it to print into the speech to text area to perform
an action.
   }//Close wordsToNumbers

}//Close MyCalcView class
```

How You Tested Your Program

Programming a calculator for arithmetic expression for the design of a multimodal user interface was challenging. Learning while taking new design strategies had to be within creativity, ideas, and put anything I've so far known from previous classes implying to the calculator application. We had the excitement to create a challenging application, but yet, all the sweat and tears for a month working on it, so we went looking for a partner but didn't know anyone, so we went to log on to the discord and try to find the best candidate for my partner. We found a guy who seems to be good with time management. So we sent him a message to be a partner. Throughout, we found out that we were both stuck on different things. My partner George Tackie was stuck on programming error handling, unary operators, and modeling the output design while struggling with speech to text, converting string to int for speech to text. So, working together helped each other grow knowledge of what each of us knows, and speed has increased since we both were best fit for each other. We did manage to work on the project together within nine days.

We used the Model View Controller as a software design pattern to develop the user interface for our project. Model for the calculation, View for the design, and Controller to manage to send errors, action listener, and create Speech and Text & Text to Speech.

Programming the Viewing was the first step I've begun working on within eclipse. (we did use a working builder in eclipse where they provide.) This did help a lot to speed up the process much faster as well as being able to follow the requirements such as quit, clear, Speech to Text & Text to Speech implying the design. The designs weren't perfect at the beginning since calculations in the mode were not made yet. When it did, we had created multiple design outputs and tried which one fits the best.

The Control was the most stressful. It had lots of research, days to figure out and understand. And to be honest, this part was all on George. We do thank him for working on the Control part of this assignment. If it weren't for George and Google, we definitely would've lost my mind but having a partner was great. He was accommodating and flexible (sometimes). George was very grateful to find Sena who was an amazing partner. George was very confused about figuring out his part of our project. Sena finished the error handling very quickly and she helped George to figure out the problems he was dealing with.

Control has cleared the text field, input the clarion and output the total, have a quit button in the Menu Bar. Also, allowing us to listen to "Kevin," our voice speaker, read the mathematics and the total and convert words to numbers. These are tactics we have implying to our code in java and working with Mode and the View altogether.

Viewing and Control are in the same class; this helps to apply the error handling. Inside the action listed, the global variables are called in to help fix and send errors to the program, telling users that some things are wrong in the input and asking them to try again.

Lastly, The Model is the calculator. Programming a calculator is from the stack. Stack helps to set the order of operation. We first thought inputting a number in from stack and calling all the methods as a string will work, but It didn't. So, having to convert from double to long. Long has the most bits in the prime data type that can hold numbers and characters. Using long to convert double to long helped to convert to string. Whenever the double contains it, it can convert from the largest bits and hold that in and move it all into a string. This was a no issue. Conversion helped with voice recognizer, and the reader can understand from the string. I've tried using integer variables, but it just ends up giving errors.

Lastly, The Main method was simple since it was just needed to call all the classes into the main method and use try and catch to fix the errors when the panel opens and calls from other classes.

We enjoyed working with the Model View Controller (MVC). Allow learning how to handle a problem when a user puts in illegal numbers and operations during calculation. Also, we learned what grammar did in the program and discovered a new programming language called XML. Testing a program is always a struggle, but keep changing around, commenting sections to see which one is which, sharing the code, and finding out that sending 30 images of screenshots to show your partner doesn't allow since it contains so much MB. Programming a calculator for arithmetic expression for designing a multimodal user interface was complicated and difficult for sure, but giving up and trying again is the best way to test a program and find out which one works the best without a doubt.

We first test decided to be the user to try out my calculator. We used basic equations such as 2+2 = 4, 6-8 = -2, and so on. We tested the text to speech function for my operations. We also made sure my voice could be understood.

Next my female housemate tested my calculator. She put in equations like mine, but she used the parentheses and included the power button as well. (4+3) = 7, 8^2= 64. Since she had an accent, the microphone couldn’t understand her properly, so we had to go back and try fixing the code for it. My partner realized that we were using only the English dictionary given by shinx4 as the resource, so she created a gram file to get the code to work. And it was able to pick up my female housemate.

Lastly my male housemate tested the calculator with non-equations to see whether the error messages would work. He also tested his voice in the speech to text.

Testing Outputs-

2+2 = 4

6-8 = -2

(4+3) = 7

8^2 = 64

(1+3)-(1+1) = 2

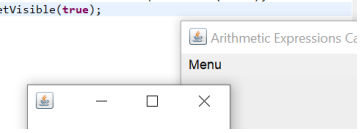
Text to speech button was a success

Speech to text button was a success

Testing outputs

Main Method

```
public class Main {
    //Main Method
        public static void main(String[] args) {
            simplemodel model = new simplemodel();
            SimpleJButton view =  new SimpleJButton(model);
            view.setVisible(true);
        }
}//Close Main
```



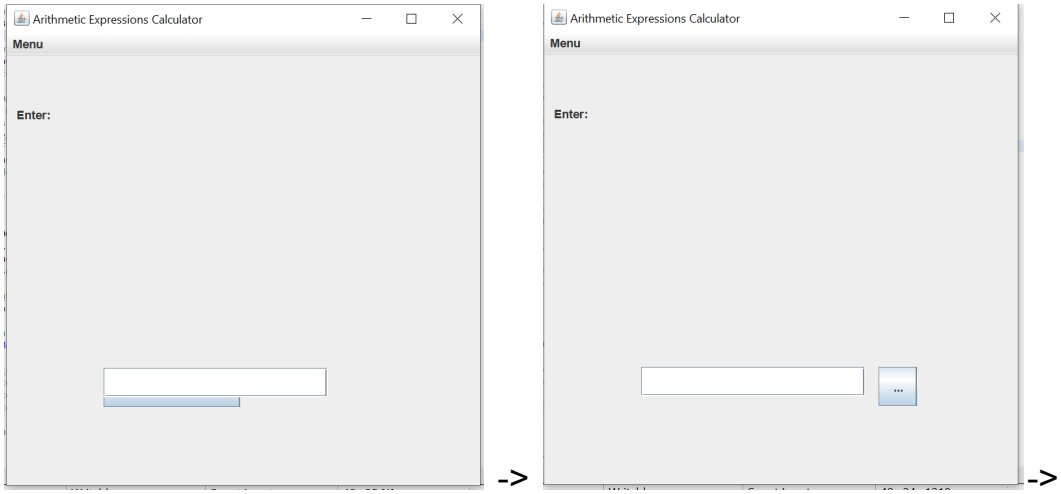
->

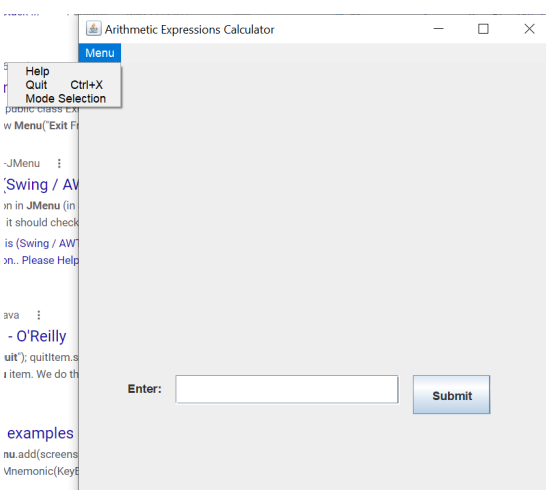
```
    }

    /**
     * Launch the application.
     */
        public static void main(String[] args)throws IOException {

            try {

                calculator theModel = new calculator();
                //windowbuilder window = new windowbuilder();
                windowbuilder theView = new windowbuilder(theModel);

                theView.setVisible(true);

            } catch (Exception e) {
                e.printStackTrace();
        }
        }
```

->

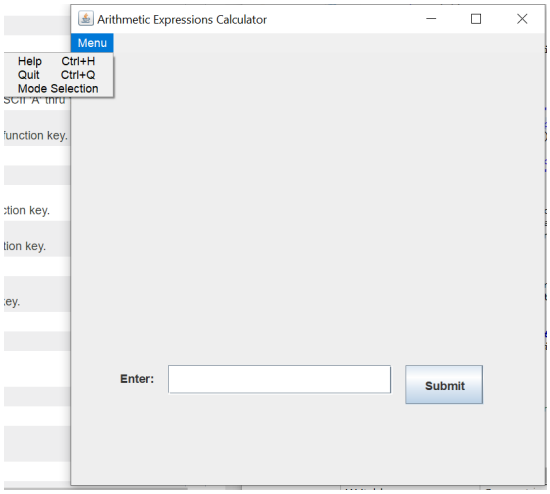
```
package calculator;

import java.awt.EventQueue;

class MyMainCalc {
/**
 * Launch the application.
 * @param args the command line arguments
 * @wbp.parser.entryPoint
 */
//Runs the calculator calls both the view and model to be parameters in the controller instance
public static void main(String args[]) {
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }//Close if
        }//Close for
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(MyCalcView.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }//Close catch
    //</editor-fold>

/*Create and display the form */
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            MyCalcModel theModel = new MyCalcModel();
            new MyCalcView(theModel);

        }//Close run
    });//Close invokeLater
}//Close Main Method
}//Close MyMainClac Class
```

The View

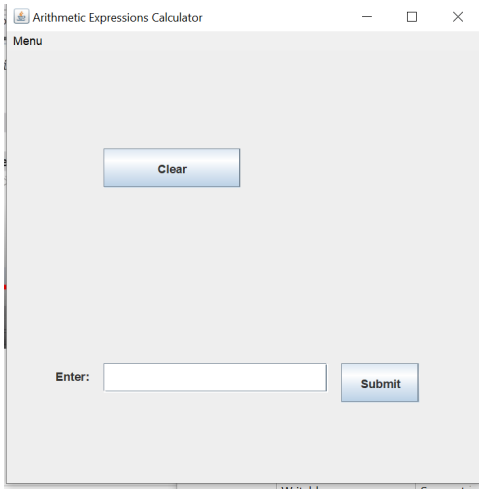

Arithmetic Expressions Calculator
Menu
Enter:
...
->
->
Arithmetic Expressions Calculator
Menu
Enter:
Submit
Arithmetic Expressions Calculator
Menu
Enter:
Submit
Writable
Smart Insert
44 : 26 : 1381
->
->
Arithmetic Expressions Calculator
Menu
Mode Selection
Quit
Mode Selection
Enter:
Submit
Arithmetic Expressions Calculator
Menu
Exit Ctrl+X
Mode Selection
Enter:
Submit
->
->



->

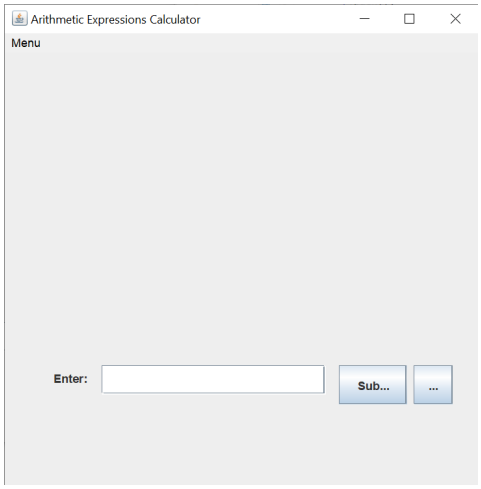


->

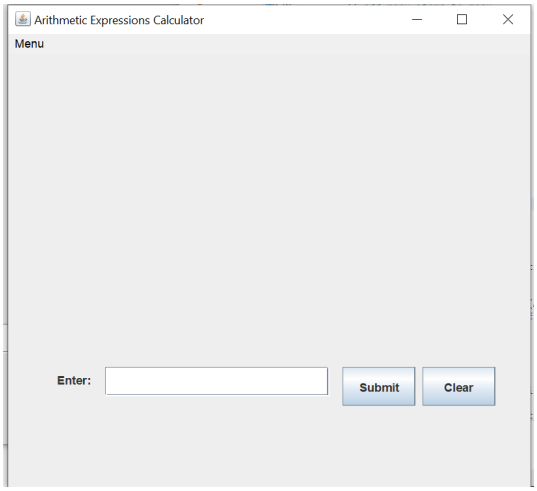


(Adding clear in to design) ->

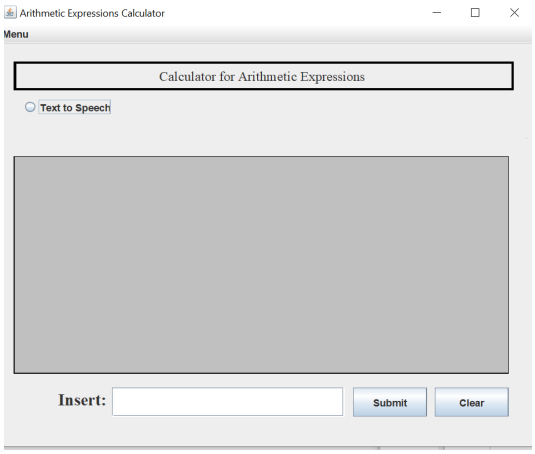
->



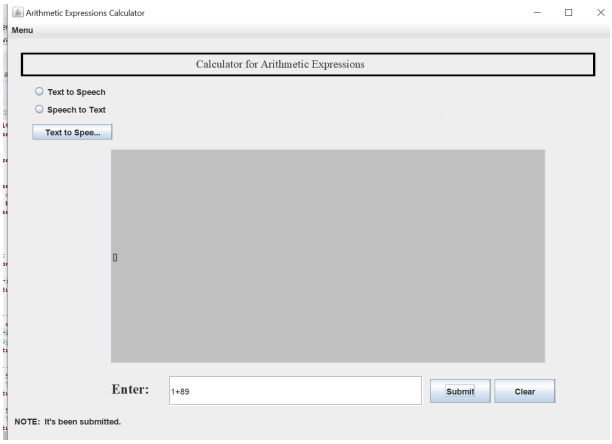
->



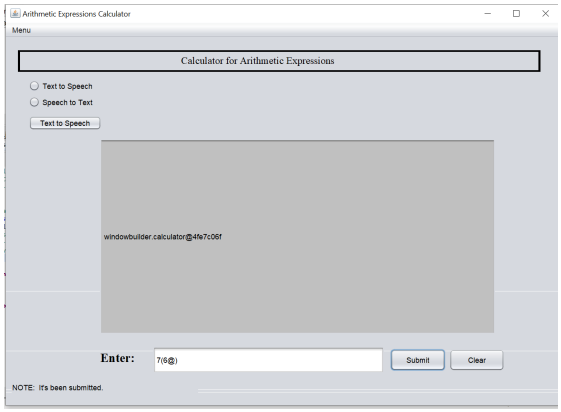
-> (adding textfield, button and title and change enter to insert) ->



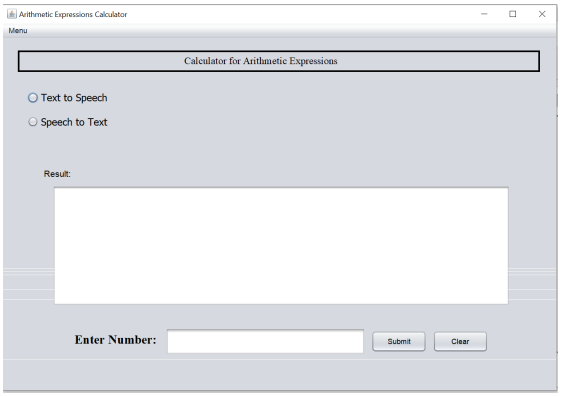
->



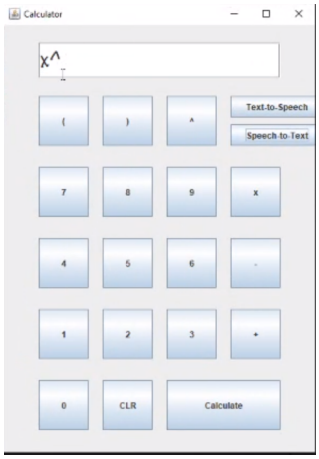
->



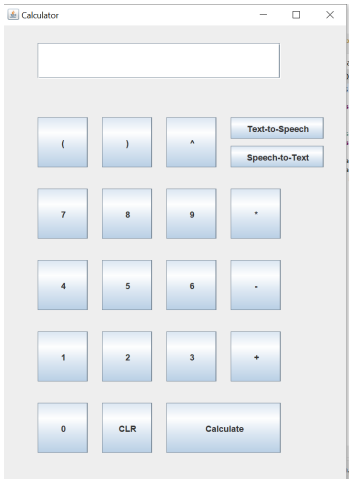
->



-> This was the final design we thought was going to be done with the project of the display... but then when it came to the calculator, only the string giving the bytes numbers, and not the total. So, I've restarted the design. ->

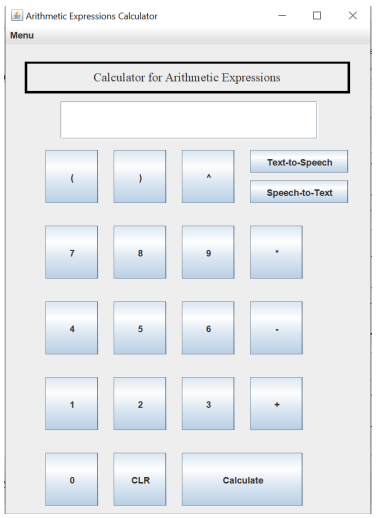


->

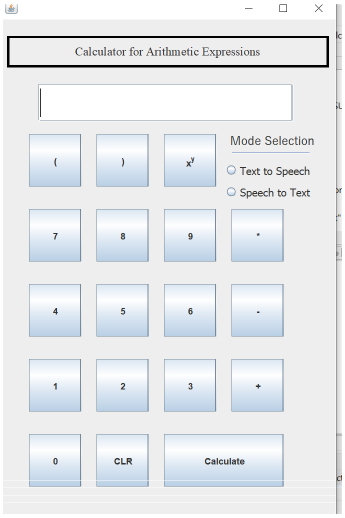


->

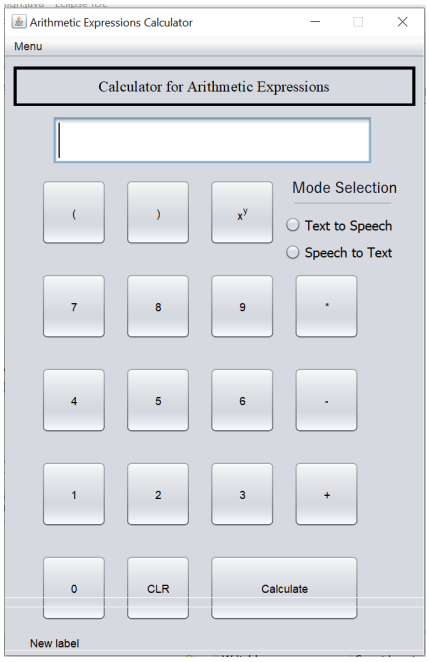
->



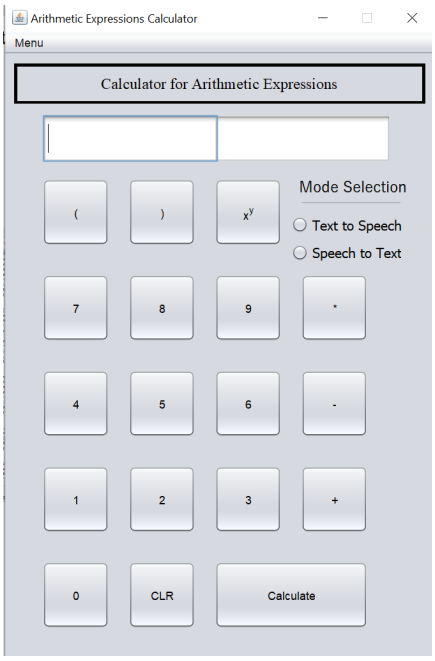
->



-> ->

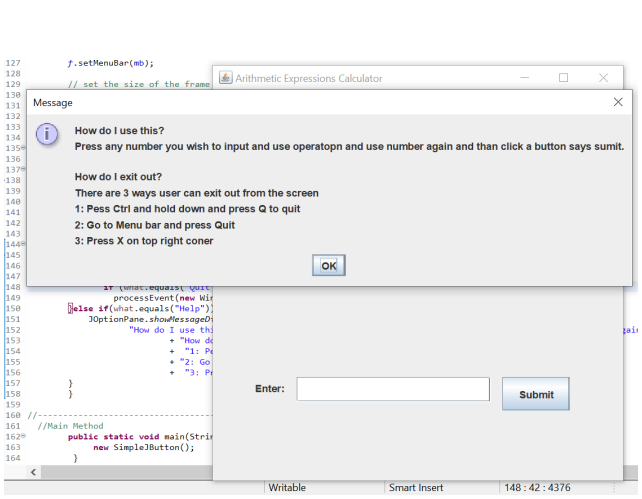


-> (FINAL DESIGN) ->

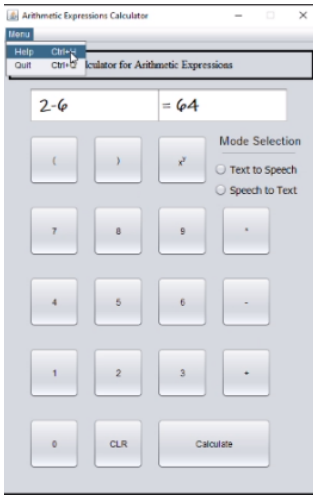


The Control

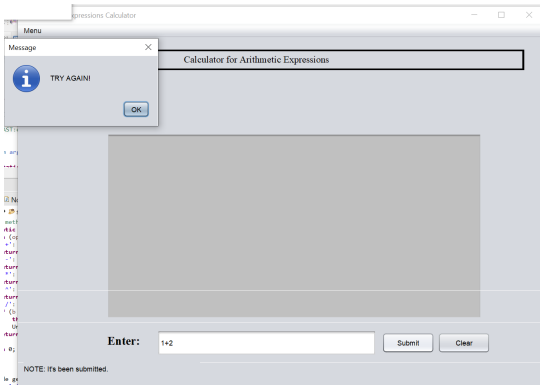
Help Button action:



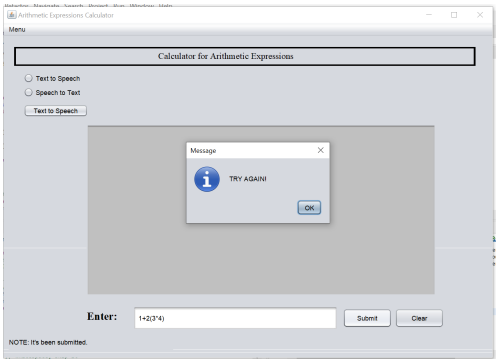
->(FINAL) ->



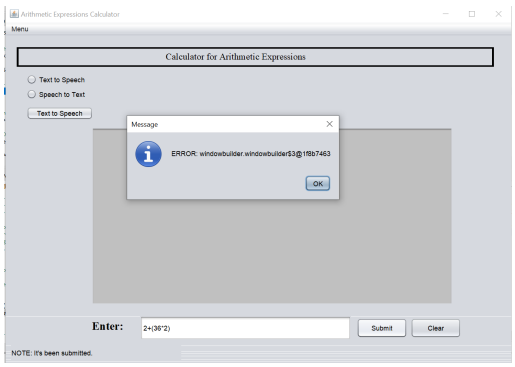
Error Handling



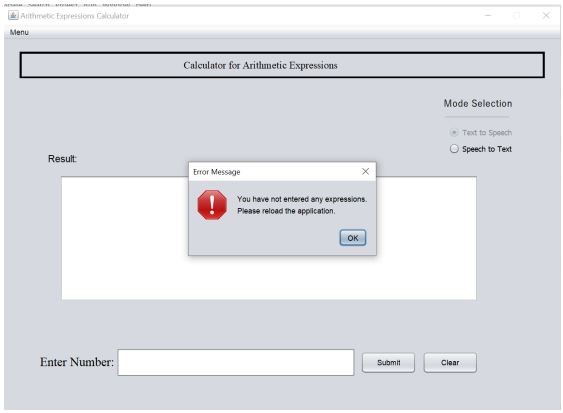
->

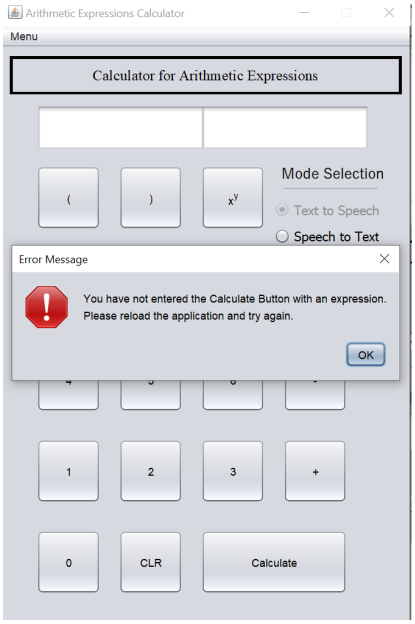


->

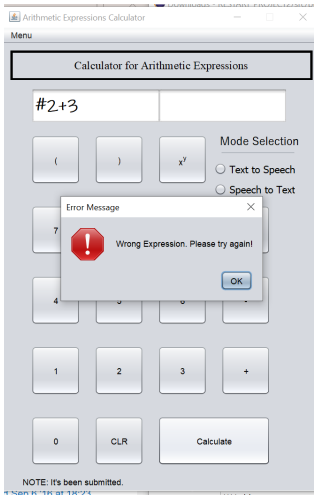


->

-> 

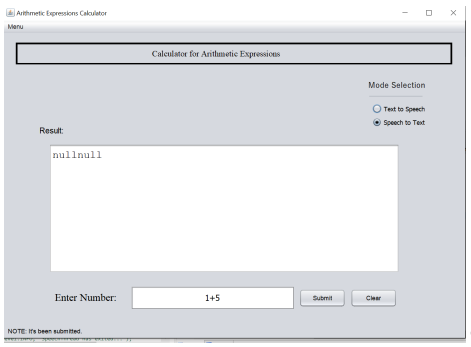
-> (FINAL) -> 

Text to Speech

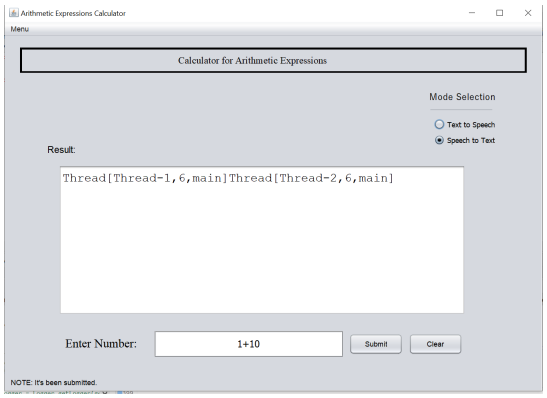


and Text to Speech button action is reading text out loud and there is only one way to show its text output, the video. The video will be provided with the final text output in the file, Please do check out!

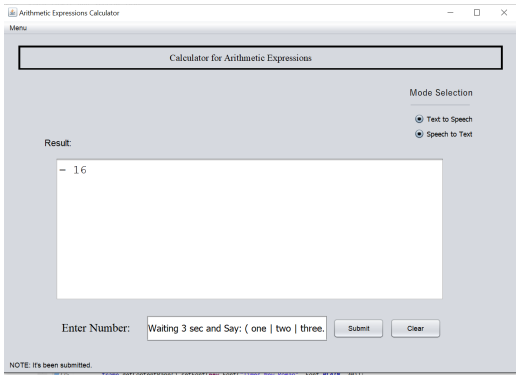
Speech to Text:



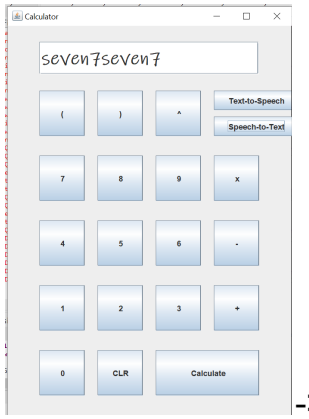
->



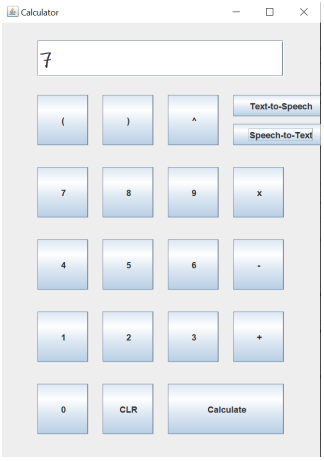
->



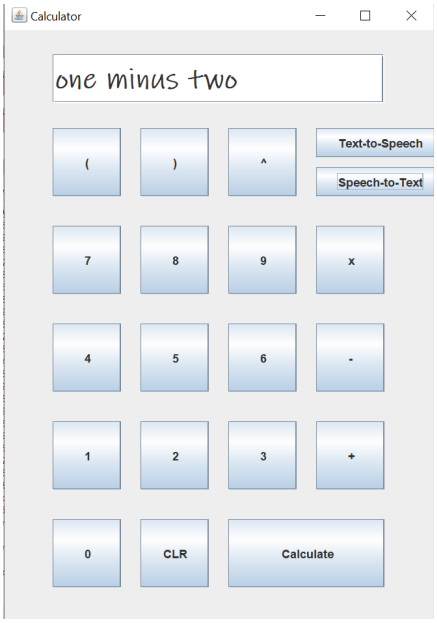
->



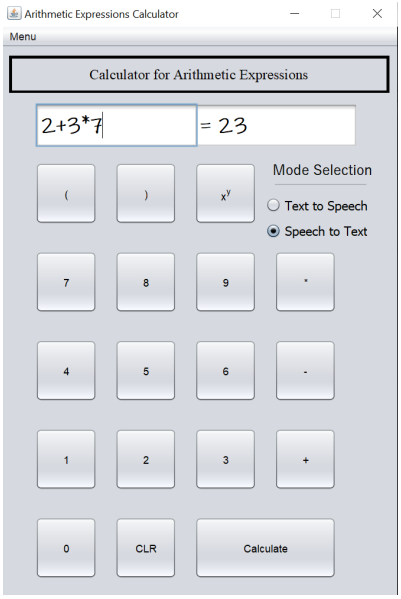
->



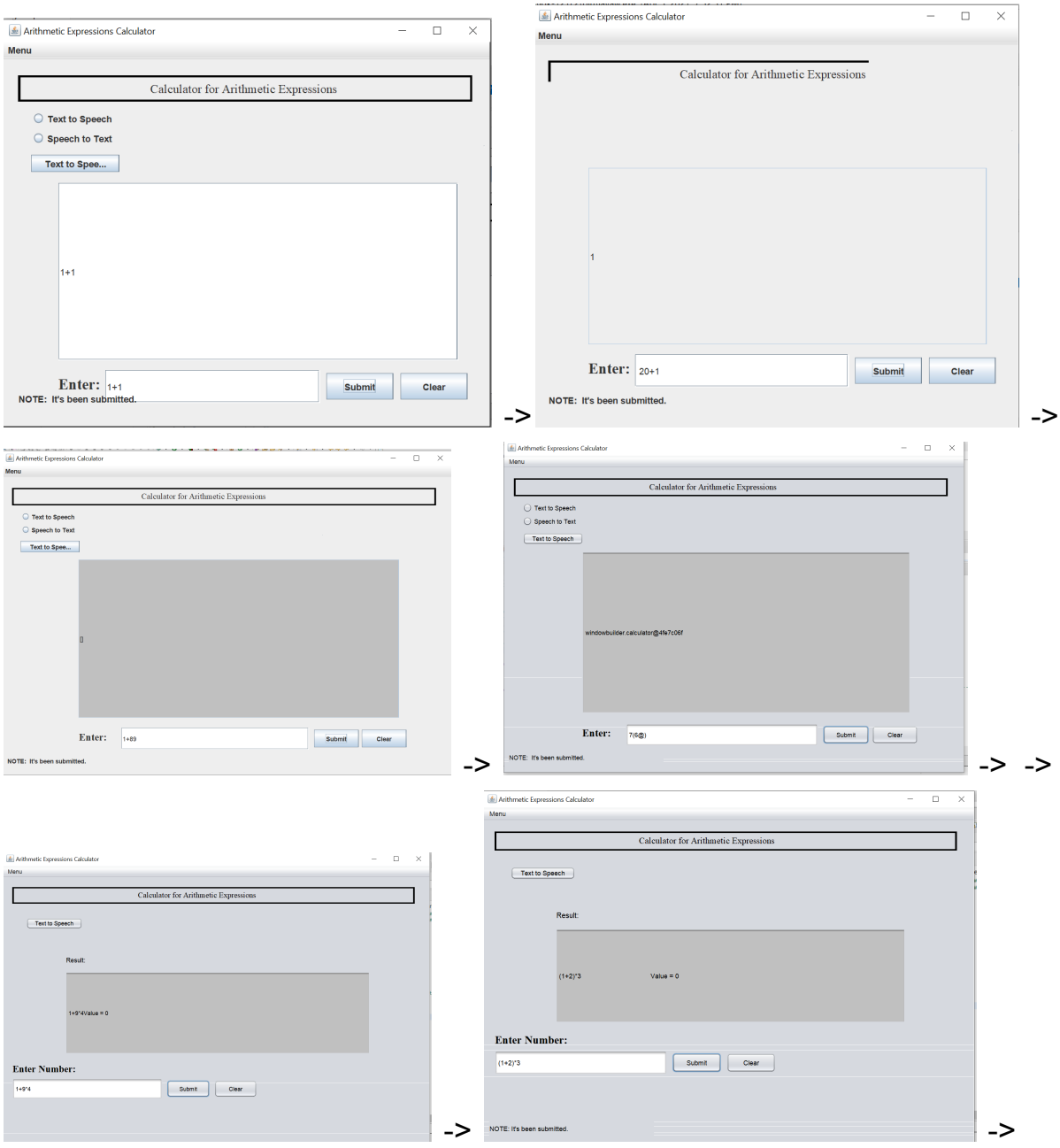
->

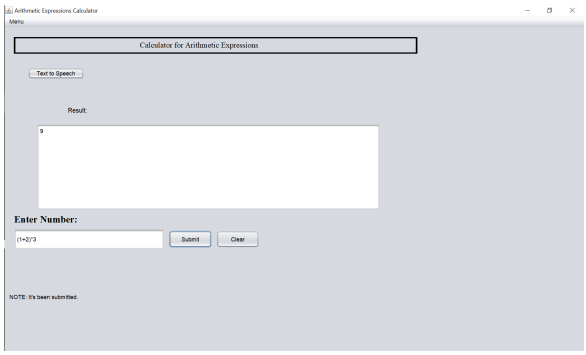


-> -> (FINAL) ->

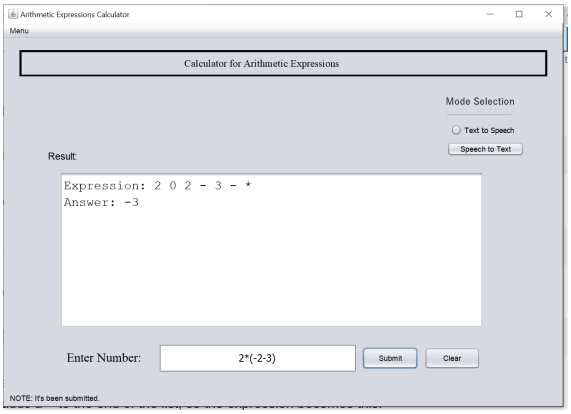


The Model

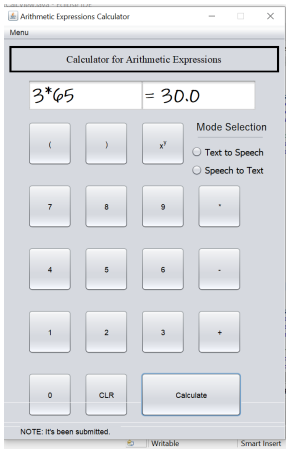

Arithmetic Expressions Calculator
Menu
Calculator for Arithmetic Expressions
Text to Speech
Speech to Text
Text to Spee...
1+1
Enter:
1+1
Submit
Clear
NOTE: It's been submitted.
->
Arithmetic Expressions Calculator
Menu
Calculator for Arithmetic Expressions
1
Enter:
20+1
Submit
Clear
NOTE: It's been submitted.
->
Arithmetic Expressions Calculator
Menu
Calculator for Arithmetic Expressions
Text to Speech
Speech to Text
Text to Spee...
Enter:
1+89
Submit
Clear
NOTE: It's been submitted.
->
Arithmetic Expressions Calculator
Menu
Calculator for Arithmetic Expressions
Text to Speech
Speech to Text
Text to Speech
windowbuilder.calculator@4fe7c06f
Enter:
7(6@)
Submit
Clear
NOTE: It's been submitted.
-> ->
Arithmetic Expressions Calculator
Menu
Calculator for Arithmetic Expressions
Text to Speech
Result:
1+9*4Value = 0
Enter Number:
1+9*4
Submit
Clear
->
Arithmetic Expressions Calculator
Menu
Calculator for Arithmetic Expressions
Text to Speech
Result:
(1+2)^3
Value = 0
Enter Number:
(1+2)^3
Submit
Clear
NOTE: It's been submitted.
->
Arithmetic Expressions Calculator
Menu
Calculator for Arithmetic Expressions
Text to Speech
Result:
(1+2)^3
Value = 0
Enter Number:
(1+2)^3
Submit
Clear
NOTE: It's been submitted.



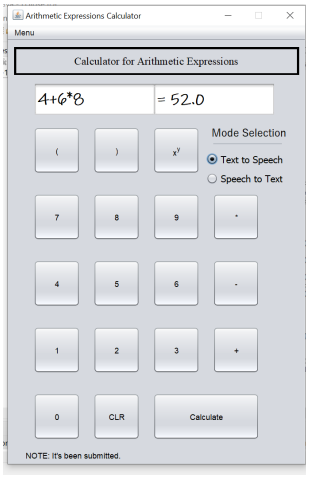
->



(Every operator worked expect for unary, so changed to new) ->

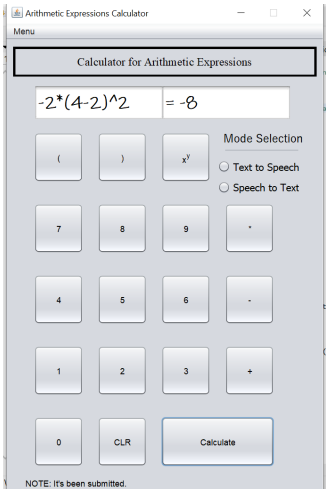


(issue with calculator, so had to fix)->



(WORKED!)

and also unary ->



.

How to run your program

Arithmetic Expression Calculator is a Multimodal User interface calculator to calculate addition, subtraction, power, and allow use of brackets for all numbers. Division is not allowed in the calciation, so there will be a message pop on the screen if you do try. You are to put in any integers from zero to how many as you wish to get the total number. Here are steps for you to follow to understand how to use the Arithmetic Expression Calculator.

REMINDER: Any alphabet or division or any other operators that are not in the button or inputs will be invaild, as a number or calculation. If so, put in any illegal numbers, and operations notification will show up on your screen.

Step 1. You may press any buttons or type in the first left box to “Input”, type in any first number and operator such as plus, munus, multiplication, and power symbols to calculate.

Step 2. Click “calculate”. This will set a total number on the right side of the text you have entered. If you wish to have Kevin read out your calculation and the total, please press “Text to Speech" Button. Or if you wish to talk, you may Press “Speech to Text” and tell Kevin, what you want to calculate. (Please do wait three seconds! Let Kevin get ready for you to calculate!)

Step 3. Clear for all use of pressing and buttons or retype the numbers and operations. “Text to Speech can be used multiple times but Kevin only can listen to you one time. Please reload the application and start talking to Kevin again! He will be available to listen to you and calculate only once at a time in the application.

Step 4. If you need help and have questions about the application, please press “CTRL + H” or go to “menu” -> Click “help”. There will be a question you may have and help you out.

Step 5. Quit can be used by pressing “CTRL + Q” or go to “Menu”. Please click “Menu” and then click “Quit”. If you wish to quit, press “yes” otherwise press “Cancel” or No”.

As you can see, the Calculation does work as well as able to Talk to Kevin and Kevin reads out loud the Calculation. Clicking on “Calculation” is shown of the two numbers with operator in between you had put into the box and have now completed the task.

Describe Parameter

Only Parameters will be shown in each class, others will be deleted. This will help the reader easily to read. The color code will also be given. This will help out clear understanding of differences between comment, code and brief descriptions all throughout the pages.

Comment: Green
Code: Blue
Brief Descriptions: Black

MyCalcView.java

```
public class MyCalcView {

Ac calling MyCalModel class is a parameter. THis is also known as a call by reference. Calling class from different class through parameter passing and na,ing an object as a modelCon.
  public MyCalcView(MyCalcModel modelCon) {

        }//Close MyCalcView Initialize

String “words” is a parameter. Calling all the words in the string be replicated by numbers of string. This only relies on and replace prior to this deal with breaking a paragraph down into individual strings of numbers, consisting of a number of words.
   public String wordsToNumbers(String words) {

        }//Close wordsToNumbers
```

MyCalcModel.java

```
public class MyCalcModel {

Combining both methods with their stacks to give calculation as a string called string
          public String stringToArithmetic(String string) {

            }//Close stringToArithmetic Method

Calculate operations only, each operators will react to other operations but this
 method takes an unique operators such as unary operation, brackets and perform together and calculate by using infix
 public static String infixToSuffix(String infix) {

 }//Close infixToSuffix Method

Split each words to understand one by one string  called postfix
 public static double suffixToArithmetic(String postfix) {
```

```
        }//Close suffixToArithmetic Method

Calculates each operator as a symbol by taking as string and as for numbers it will calculate
double ca;;ed x and y. X will be used first and y is the second number to calculate .
   private static double calculator(double x, double y, String symbol) {

} //Close calculator Method

} //Close MyCalcModel Class
```

Description of all software packages used

Comment: Green
Code: Blue
Brief Descriptions: Black

MyMainCalc.java

Used in the Main. This allows the Swing presse when done in a tread. EventQueue helps during the computing a long lasting calculations model with the GUI. ALso helps to update the application and posts an event at the end of the Swings event list and procoee after all the previous GUI events that are processed.
 import java.awt.EventQueue;

MyCalcModel.java

Stack package is used during the calculate operations only, when each operator is being calculated, the stack will help and push and pop in order of the operator of mathematics.

Stack is also used to split each word and split and match packs together.
import java.util.Stack;

Regex.Pattern is used to create a pattern from the regular expression passed through a matcher. When ever the string and operator are needed to match to the text against a regular expression pattern more than one time, the regex.Pattern will comes in and instance to help out
import java.util.regex.Pattern;

Will show an error message if the power of the operator is non-negative.
import javax.swing.JOptionPane;

MyCalcView.java

To create a Frame for the panel.
import javax.swing.JFrame;
To create a Panel.
import javax.swing.JPanel;
To create a line border between the mode selection and Speech to Text & Text to Speech to so it's easy to see.
import javax.swing.border.LineBorder;
To create Color in the panel of the line border.
import java.awt.Color;
To let the user know if the microphone is on or not, so this will open up in the computer system to double check.
import javax.sound.sampled.AudioSystem;
To take information from the microphone system in the computer system and letting user

know what's going on.
import javax.sound.sampled.Port;
To design horizontal and vertical layouts in the panel
import javax.swing.GroupLayout;
For the horizontal and vertical axis with the right and right edges.
import javax.swing.GroupLayout.Alignment;
To createLabels for Model Selection, and note.
import javax.swing.JLabel;
To create a menu for Help and Quit.
import javax.swing.JMenu;
To create a Menu Bar for Help and Quit.
import javax.swing.JMenuBar;
To create Menu Items named Help and Quit.
import javax.swing.JMenuItem;
To create messages for errors.
import javax.swing.JOptionPane;
To create fonts for label, men, frame.
import java.awt.Font;
To create colors for Quit and Help
import java.awt.SystemColor;
To create action such as allowing to talk to Kevin and read out loud the text field and able to calculate and convert words to numbers.
import java.awt.event.ActionEvent;
To create listings for the action such as allowing to talk to Kevin and read out loud the text field and being able to calculate and convert words to numbers.
import java.awt.event.ActionListener;
To create pressing buttons for easy access to quit the panel and to go to help center.
import java.awt.event.KeyEvent;
To create Text field or input and output of the calculator.
import javax.swing.JTextField;
To create a keyboard streak for pressing buttons for easy access to quit the panel and to go to the help center.
import javax.swing.KeyStroke;
To create Buttons for numbers, operators and brackets, Speech to Text & Text to Speech
import javax.swing.JButton;
To create a position center on the screen.
import javax.swing.SwingConstants;
To create buttons only for Mode Selections: Speech to Text & Text to Speech
import javax.swing.JRadioButton;
To create a line to separate between the label and the buttons
import javax.swing.JSeparator;
To create Catch in the parameter allowing to catch any errors and send an message
import java.io.IOException;
To send an info message in when Speech to Text is pressed and only works on Speech to Text.
import java.util.logging.Level;
To send an info message in when Speech to Text is pressed and only works on Speech to Text
import java.util.logging.Logger;
To create Kevin alive, reads a text field off calculator. Also known as using for Text to Speech.

```
import com.sun.speech.freetts.Voice;
import com.sun.speech.freetts.VoiceManager;
To Create Speech to Text. Also used for grammars as well.
import edu.cmu.sphinx.api.Configuration;
import edu.cmu.sphinx.api.SpeechResult;
import edu.cmu.sphinx.api.LiveSpeechRecognizer;
```

All the Software packages used in MyMainCalc.java, MyCalcModel.java, and MyCalcView.java

1. cmu_time_awb.jar
2. cmu_us_kal.jar
3. cmudict04.jar
4. cmulex.jar
5. cmutimelex.jar
6. en_us.jar
7. freetts-jsapi10.jar
8. freetts.jar
9. mbrola.jar
10. sphinx4-core-5prealpha-20160628.232526-10-javadoc.jar
11. sphinx4-core-5prealpha-20160628.232526-10.jar
12. sphinx4-core-5prealpha-20160628.232526-10-sources.jar
13. sphinx4-data-5prealpha-20160628.232535-10-javadoc.jar
14. sphinx4-data-5prealpha-20160628.232535-10.jar
15. sphinx4-data-5prealpha-20160628.232535-10-sources.jar

Resources

https://miro.com/app/board/o9J_lJKGHQY=/

https://www.geeksforgeeks.org/pattern-compilestring-method-in-java-with-examples/