Java Projects With Swing GUI

Miles Jennings

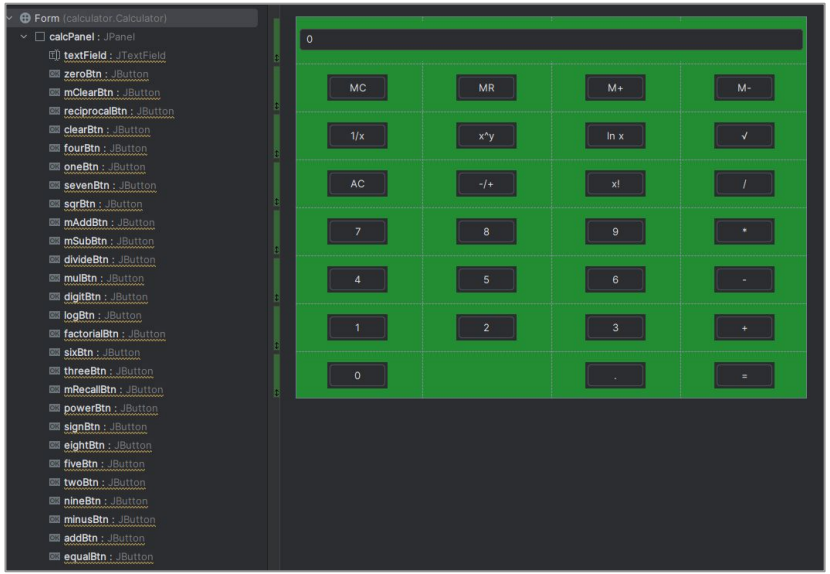
Calculator Overview:

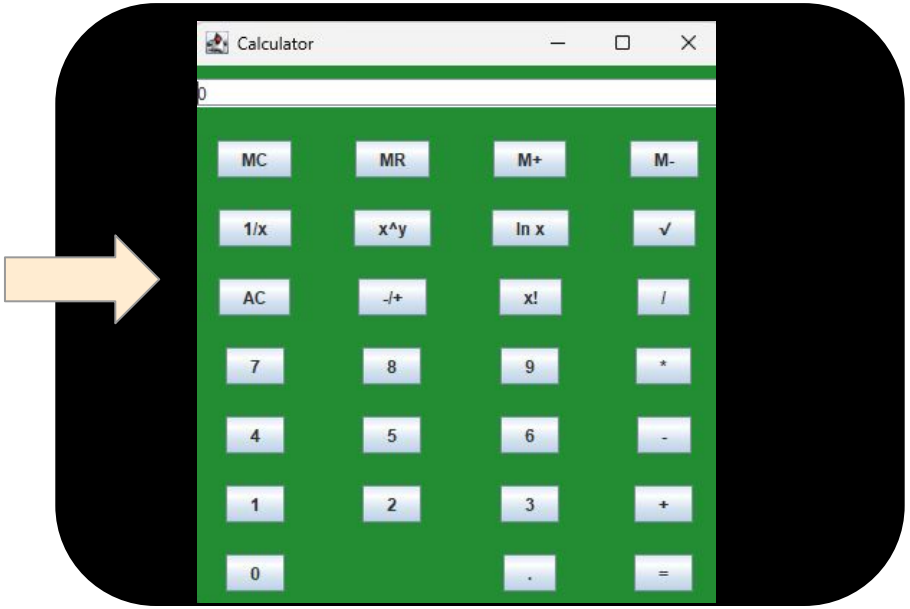
- This Java-based calculator program is built using Swing GUI, designed to perform both basic and advanced mathematical operations. The program includes a user-friendly interface with buttons for numerical input and various functions. It supports operations such as addition, subtraction, multiplication, division, square roots, logarithms, factorials, and more.
- In addition to the basic arithmetic, this program features memory functions (such as memory recall and memory operations) and provides error handling for common mathematical issues like division by zero.

Requirements:

- JDK 21 or higher for smooth operation. Java Swing library for GUI components.

Created buttons in Swing to create an interactive GUI





My Approach to Implementing the Logic of the Buttons

- Implemented a base class (numberBtnClicked) that extends ActionListener, allowing number buttons to inherit common behavior while updating the calculator display dynamically.
- Created individual subclasses (zeroBtnClicked, oneBtnClicked, etc.) that extend numberBtnClicked, each associating a specific number value with its respective button.
- Assigned each button an instance of its corresponding event handler subclass, ensuring that clicking a button updates the JTextField display with the correct digit.
- Checked if the text field contains "0" before appending new values, preventing unnecessary leading zeros in user input.
- Utilized object-oriented design principles, such as inheritance, to simplify event handling and reduce redundant code while maintaining flexibility for future expansion.

```
public class Calculator {
    //Main panel for the swing GUI
    private JPanel calcPanel;  2 usages
    //Text field to display the buttons/result calculated
    private JTextField textField;  67 usages
    //buttons that make up the calculator
    private JButton zeroBtn;  3 usages
    private JButton mClearBtn;  3 usages
    private JButton reciprocalBtn;  3 usages
    private JButton clearBtn;  3 usages
    private JButton fourBtn;  3 usages
    private JButton oneBtn;  3 usages
    private JButton sevenBtn;  3 usages
    private JButton mRecallBtn;  3 usages
    private JButton sqrBtn;  3 usages
    private JButton divideBtn;  3 usages
    private JButton mulBtn;  3 usages
    private JButton minusBtn;  3 usages
    private JButton addBtn;  3 usages
    private JButton mAddBtn;  3 usages
    private JButton powerBtn;  3 usages
    private JButton signBtn;  3 usages
    private JButton eightBtn;  3 usages
    private JButton fiveBtn;  3 usages
    private JButton twoBtn;  3 usages
    private JButton mSubBtn;  3 usages
    private JButton digitBtn;  4 usages
    private JButton logBtn;  3 usages
    private JButton equalBtn;  3 usages
    private JButton factorialBtn;  3 usages
    private JButton nineBtn;  3 usages
    private JButton sixBtn;  3 usages
    private JButton threeBtn;  3 usages
```

Code Samples

```java
/**
 * Class made to be the base class to be inherited from for the number buttons
 */
class numberBtnClicked implements ActionListener {  20 usages  10 inheritors
    // The value associated with the number button
    private final int value;  3 usages
    private final JTextField textField;  5 usages

    /**
     * Constructor that sets the value of the number button and the text field.
     *
     * @param value The number value as an integer (e.g., 1, 2, 3).
     * @param textField The text field to update when the button is clicked.
     */
    public numberBtnClicked(int value, JTextField textField) {  20 usages
        this.value = value;
        this.textField = textField;
    }

    /**
     * Appends the button's value to the calculator's display when clicked.
     *
     * @param e is the event triggered by the button click.
     */
    @Override
    public void actionPerformed(ActionEvent e) {
        if(textField.getText().equals("0")){
            textField.setText(String.valueOf(value));
        } else {
            textField.setText(textField.getText() + value);
        }
```

```java
/**
 * Constructor made to set up all button actionlisteners and logic.
 */
public Calculator() {  1 usage

    // Initialize all the number buttons using numberBtnClicked
    zeroBtn.addActionListener(new numberBtnClicked( value: 0, textField));
    oneBtn.addActionListener(new numberBtnClicked( value: 1, textField));
    twoBtn.addActionListener(new numberBtnClicked( value: 2, textField));
    threeBtn.addActionListener(new numberBtnClicked( value: 3, textField));
    fourBtn.addActionListener(new numberBtnClicked( value: 4, textField));
    fiveBtn.addActionListener(new numberBtnClicked( value: 5, textField));
    sixBtn.addActionListener(new numberBtnClicked( value: 6, textField));
    sevenBtn.addActionListener(new numberBtnClicked( value: 7, textField));
    eightBtn.addActionListener(new numberBtnClicked( value: 8, textField));
    nineBtn.addActionListener(new numberBtnClicked( value: 9, textField));

    //actionlisteners for the rest of the buttons on calculator
    /**
     * Memory clear actionListener to clear the memory variable
     */
    mClearBtn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            memory = 0.0;

        }
    });
    /**
     * Reciprocal actionListener to get reciprocal of num1 when button is pressed
     * Outputs error to textField when trying to take reciprocal of 0
     */
    reciprocalBtn.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
```

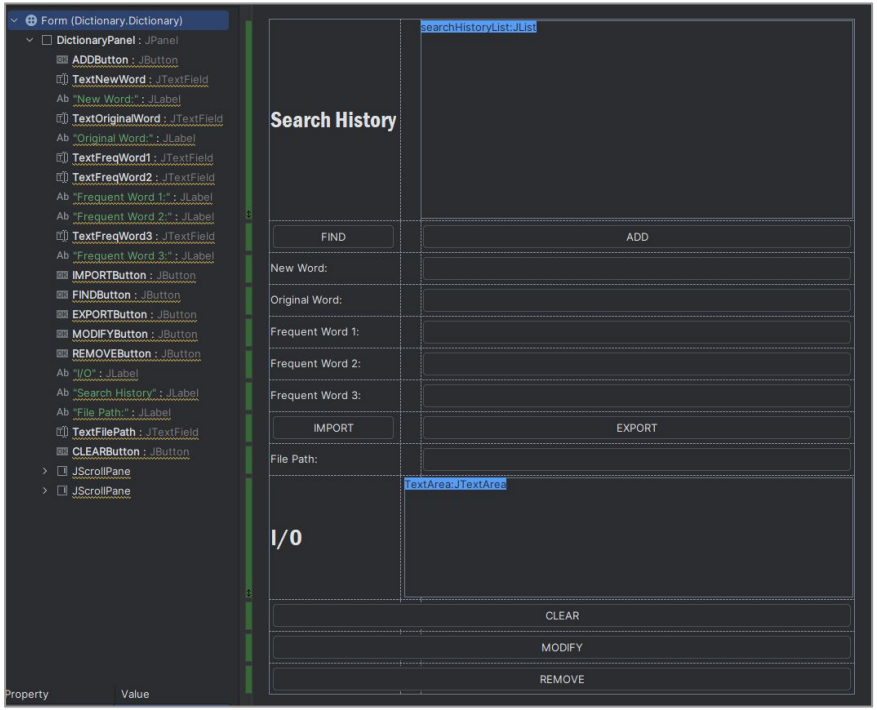
Dictionary Overview:

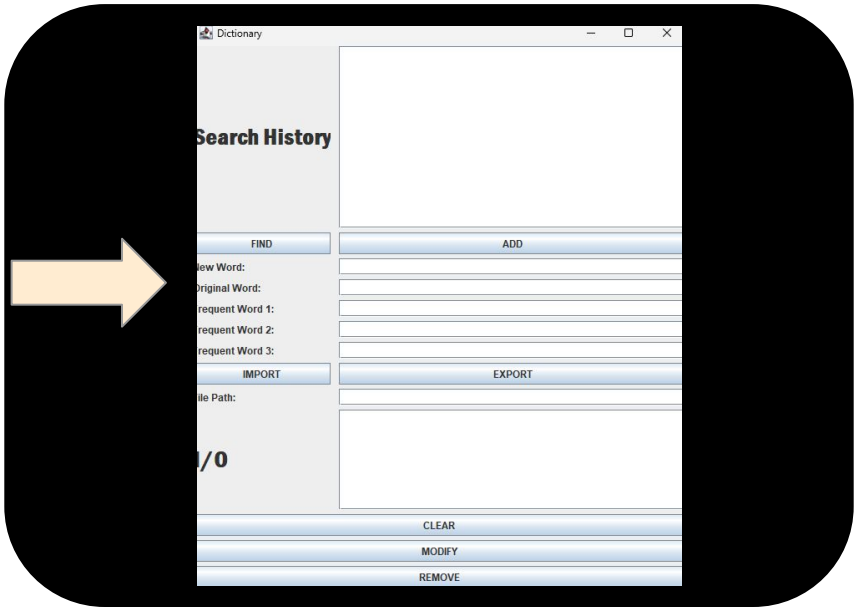
- This Java program implements a dictionary system where users can add, modify, remove, and find words along with their definitions.
- The program supports importing and exporting words to/from a .txt file, displaying words with their frequency of use, and handling exceptions for invalid or duplicated words.
- The program features a graphical user interface (GUI) built using Swing.

Requirements:

- JDK 21 (Recommended for compatibility and smooth operation) Swing library for GUI components Java 8 or higher

Created buttons in Swing to create an interactive GUI





My Approach to Implementing the Logic of the Buttons

- Utilized Swing's JButton components to create interactive buttons for key dictionary functions such as Add, Find, Modify, Remove, Import, and Export.
- Implemented ActionListener for each button, enabling dynamic user interactions like adding words, updating definitions, and searching the dictionary.
- Connected text fields (JTextField, JTextArea) to the button event handlers, allowing users to input words, definitions, and file paths for dictionary operations.
- Integrated exception handling (try-catch blocks) within ActionListeners, ensuring proper validation of word input and preventing errors such as duplicate words or invalid characters.
- Used a HashMap<String, Word_Data> as the core data structure, mapping words to their meanings and frequencies, enabling efficient dictionary management.
- Maintained a search history using an ArrayList<String>, dynamically updating the JList display to show recent lookups for an improved user experience.

```
public class Dictionary {
    //Elements from Swing
    public JButton FINDButton;  3 usages
    public JButton ADDButton;  3 usages
    public JButton MODIFYButton;  3 usages
    public JButton REMOVEButton;  3 usages
    public JButton CLEARButton;  3 usages
    public JButton IMPORTButton;  3 usages
    public JButton EXPORTButton;  3 usages
    public JTextField TextNewWord;  11 usages
    public JTextField TextOriginalWord;  5 usages
    public JTextField TextFreqWord1;  6 usages
    public JTextField TextFreqWord2;  6 usages
    public JTextField TextFreqWord3;  6 usages
    public JTextArea TextArea;  27 usages
    public JList searchHistoryList;  4 usages
    public JTextField TextFilePath;  5 usages
    public JPanel DictionaryPanel;  3 usages
```

Code Samples

```java
public final HashMap<String, Word_Data> dictionary = new HashMap<>();  15 usages
public final ArrayList<Map.Entry<String, Word_Data>> temp_compare = new ArrayLis
public ArrayList<String> searchHistory = new ArrayList<>();     //Utilizes a bui
public final int maxHistorySize = 10; // Maximum entries in search history  1 usag
/*HELPER METHOD FOR SEARCH HISTORY */
/**
 * Updates the search history, ensuring the most recent search is at the top.
 * Ignores null or empty words and removes any words past the maxHistorySize len
 * If the word being added is already in the search history, it will remove it a
 * @param word The word to add to the search history.
 */
    public void updateSearchHistory(String word) {  2 usages
        if (word == null || word.isEmpty()) {
            return; // Ignore null or empty words
        }

        if (searchHistory.contains(word)) {
            searchHistory.remove(word);
        }
        searchHistory.add( index: 0, word);

        while (searchHistory.size() > maxHistorySize) {
            searchHistory.remove( index: searchHistory.size() - 1);
        }

        searchHistoryList.setListData(searchHistory.toArray(new String[0]));
    }
```

```java
ADDButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try{
            word = TextNewWord.getText().trim();    //.trim() added to clear all trailing spaces for cleaner formatting
            definition = TextArea.getText().trim();

            if (!ValidWordCheck(word)) {        // Use the helper method to check validity
                throw new InvalidWordError();
            }
            if (word.isEmpty()) {
                throw new InvalidWordError();
            }
            if (definition.isEmpty()) {
                throw new InvalidWordError();
            }

            if (dictionary.containsKey(word)) {     //very helpful hashmap utility, checks if the key already exists
                throw new WordDuplicatedError();
            }

            dictionary.put(word, new Word_Data(definition));        //adds the new word to the dictionary
            TextArea.setText("Success: '" + word + "' has been added to the dictionary\n With definition: " + definition);
            TextNewWord.setText("");

        }
        catch (InvalidWordError ex) {
            TextArea.setText( ex.getMessage());
            System.out.println(ex.getMessage());
            throw new InvalidWordError();
        } catch (WordDuplicatedError ex) {
            TextArea.setText( ex.getMessage());
            System.out.println(ex.getMessage());
            throw new WordDuplicatedError();
```