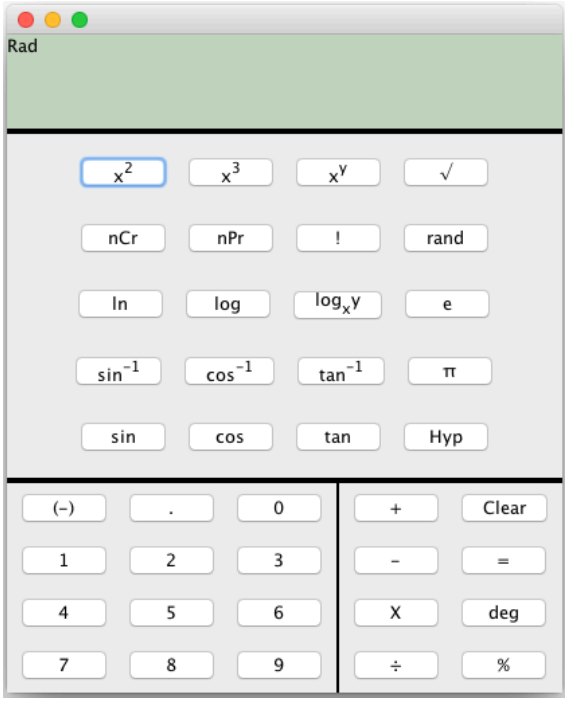
Project: GUI Scientific Calculator

Author: Anthony Bader
Language: Java
Lines of Source Code: 1113
IDE Used: NetBeans 8.2

Overview

The purpose of the programming project is to have a fully functioning scientific calculator, which emulates the user experience of a standard scientific calculator such as one designed by Texas Instruments, Casio, Canon, etc. The actual interface is a custom design, with functions similar to those stated above. As a learning experience geared towards object oriented programming, the calculator application employs many concepts and techniques such as classes, objects, methods, exception handling, inheritance, polymorphism, GUIs with event handling, Layout Manager, and APIs, particularly Java's Swing and AWT library for GUIs and Java's Math class. It serves as an example of my ability to program in a high level object oriented programming language, general coding knowledge, and an example of analytical problem solving in action.



The completed project as shown on the left. The screen uses several nested JPanels (from Java's Swing Library) with numerous other JComponents such as a JLabel and a JFrame. The user can press buttons using a mouse, generating the users input on the top JLabel (top green rectangle that looks like a digital screen). If you use a function that takes two inputs, they must be input in the correct order. If the operation performed is a single input, such as x^2 or sin(x), the number must be input first. If no input is required, such as "e" or "rand," the number will simply generate on the screen. A video demonstration of the application in use is provided below, the source code saved in a separate file (.rtf).

Project Details

I. *Brief Summary of How it Works*

The program essentially re-writes a JLabel based on the user's interaction and function called. There are two classes utilized, a calculator class that is most of the code and a tester

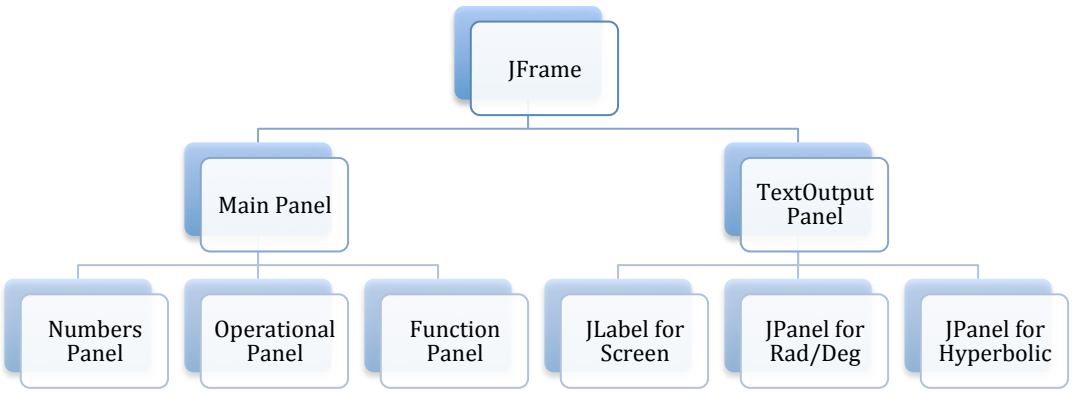
class to instantiate a calculator object. There are also many inner classes within the calculator class, one for each function on the calculator. The math engine is a combination of several methods in Java (one for each operation) and the Clear() and Equals() methods. Each function of the calculator gets its own dedicated listener (the inner classes in the calculator class), which fires when the button it's assigned to is pressed. When the user inputs a value it is stored in an ArrayList instance variable to the instance of calculator class. The operation the user selects is stored as an integer index value in an integer called OperationChosen. The Eqauls() method is where the actual math is performed. As it is a function bearing button on the calculator, it has its own dedicated listener, which can allow the user to call the Equals() method manually when appropriate. It can also be called automatically in the program based on the same design scheme many ordinary calculators follow. The Equals() method appropriately updates the ArrayList holding the user input values, and runs a switch. The switch runs cases based on the OperationChosen integer value. Each index (e.g. 1 is addition, 2 is subtraction, 4 is division) then calls the method associated with it, passing the elements of the ArrayList into the method as arguments.

II. *Managing "State" and Making User Interaction Clean*

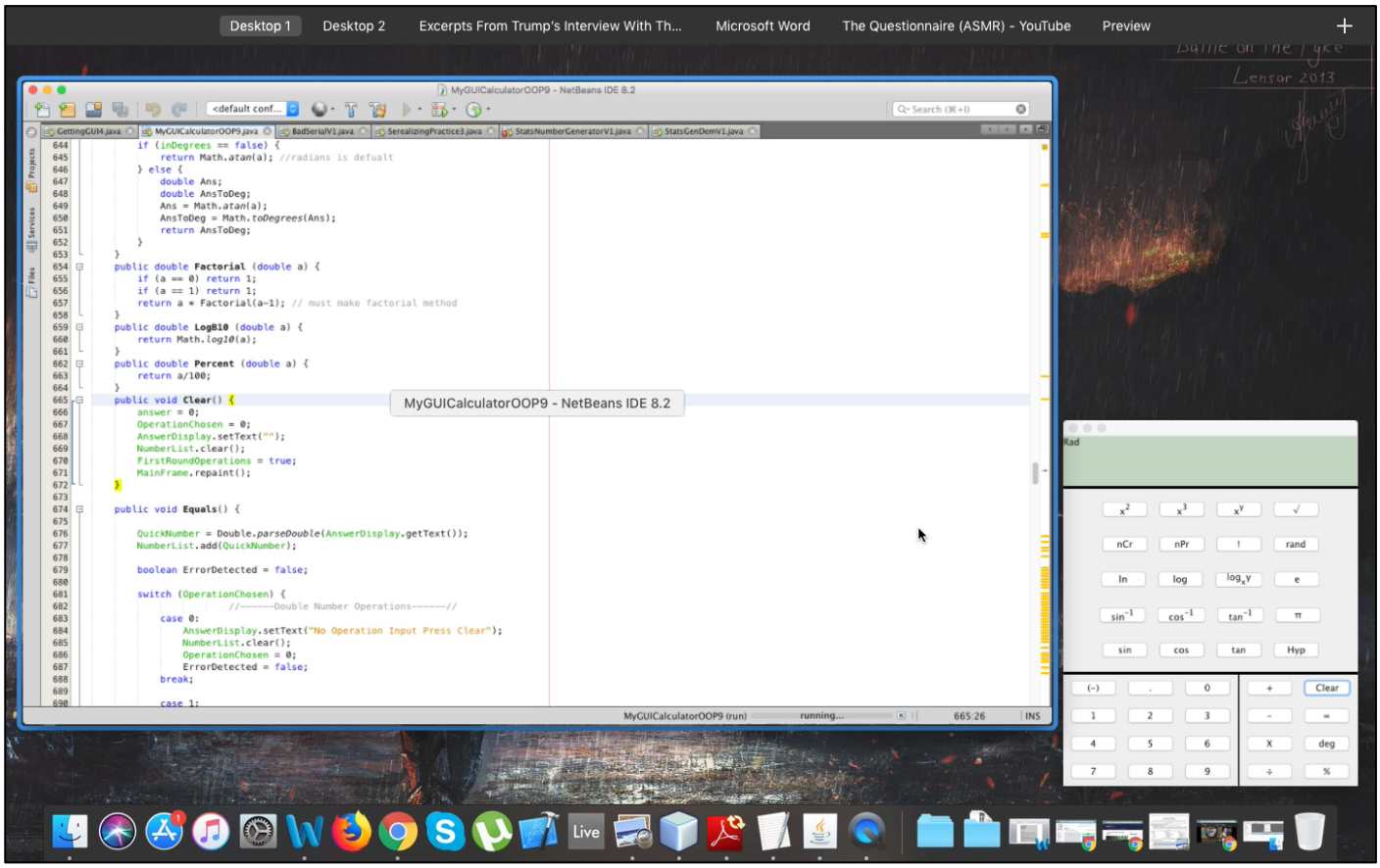
In order to maintain seamless operation by the user the idea of "state" is used in the design. The variables that control the state of the calculator are primarily two Booleans. One Boolean is set to true if it is time for the calculator user to input a new number while the other is set to true if it is the first round of operations since the Equals () method has been called. This allows for the rightly timed clearing of the screen, knowledge of when to reset the ArrayList of input values, and many more things in the background to keep the operation from the user end of things smooth and elegant.

III. *Managing Graphics Components and Using Layout Manager*

Many JComponents were utilized in order to have a clean design as well as showcase Layout Manager's three major layouts (Border Layout – which has 5 regions, all edges with a center, Box Layout – which allows for vertical stacking of JComponents, and Flow Layout – which has dynamically changing sizes reading from left to right). The hierarchy and nesting of major JComponents used is shown below.



Video Demo


Desktop 1
Desktop 2
Excerpts From Trump's Interview With Th...
Microsoft Word
The Questionnaire (ASMR) - YouTube
Preview
MyGUICalculatorOOP9 - NetBeans IDE 8.2
MyGUICalculatorOOP9 - NetBeans IDE 8.2