RHE Calculator Tutorial

Dave Minkowski

CIS 163AA - Spring 2018 - Section: 20695

Dr. Elliot Cherner

The program I’ve written for my final project came is a very simple calculator to convert my hourly pay rates at work. My company recently changed the way they pay us, giving field employees more schedule options to choose from while making the same annual salary. Before the changes, it would behoove you to work a schedule consisting of 24 hour shifts to pack the most hours into your paycheck. Now, your pay rate changes inversely with the shift’s RHE value (regular hourly equivalent), meaning you can work fewer hours per shift while still making the same annual pay. Without wasting too much more time talking about the particulars of my workplace, this RHE change made overtime pay slightly more difficult to calculate with all these different pay rates. The aim of this program is to be able to help make that calculation easier.

Looking first into code of the program, you can see I’ve imported several of java’s swing GUI components to help build a user-friendly graphical interface that’s more aesthetically pleasing and probably more comfortable for the average person to use (my coworkers, for example). Each of these components gives me formatting and control options to implement in the gui.

The RHE Calculator class extends the JFrame class, retaining the functionality of the JFrame options while allowing us to define elements of the class. It’s also ‘implements’ ActionListener, allowing us to associate a component of the gui to trigger an action when activated.

I’ve then defined the various labels and fields for the gui. The labels are the text you see next to each textbox (field) which will either accept input or display the output of the calculation once the calculate button is pressed.

Next, we have the definitions for all the elements within the RHECalculator() frame. The layoutConstraints variable is initialized as null. The labels and fields are created as objects and given default sizes and text to display, and the Calculate button is given some text and assigned the actionListener interface. The text fields are assigned a columnar width and depending on the field, they are set to be non-editable. The hourly rate field is one of two editable fields, however the hourlyRateField uses NumberFormat and the JFormattedTextField properties to extract the value of the field rather than the text string.

Each element of the layout is assigned a position in the X,Y coordinate plane of the gridbagLayout and assigned padding on each side (top, left, bottom and right). The layoutConst variable is created for each component which is added to the layout with the last line of each entry (ie: calcButton, layoutConst).

Finally, the actionPerformed method is called when the actionListener is triggered (ActionEvent event). This is the main function of the program here, to calculate RHE and annual salary on a button click. There are variables assigned for each input and output. There are only two input fields: the baseRHE and the hourlyRate. The hourlyRate is obtained using the getValue method on the formatted hourlyRate text field while the baseRHE value is obtained by parsing the string contained in the baseRHE field to an integer. It’s two ways to perform the same function, which I was fiddling with as I was writing this program. The calculation is performed, the values are set to double variables, and the associated values are set to their respective fields using a 2 decimal place format on each string conversion. The main method calls the frame and makes it visible when the program is run.

Using the program is simple. Each employee knows his or her hourly pay rate, and each employee is assigned to a shift with a designated RHE, so any user who uses this calculator would already know both their RHE and their pay rate. The RHE conversion requires an annual salary to compute, so the program determines the employee’s annual salary based on their RHE and hourly rate. The fields labeled “RHE 40 Rate:” or “RHE 64 Rate:” etc, are the hourly rates that employee would earn at while working a shift of that RHE. The lower the RHE, the higher the pay rate will be.

The default RHE is set to 44, which is the most common RHE for our field employees. Whichever RHE and hourly rate the user enters in the calculator should display identically after the calculation. (If I calculate an hourly rate of 25 dollars at an RHE of 44, the “RHE 44 Rate:” field should also be 25.00).

My final thoughts… I’d like to implement some control for valid RHE rates – we only have 4 at my workplace, so any RHE value that isn’t 40, 44, 52 or 64 should be ignored, or at least, should remind the user to enter a correct RHE. Ideally, I would have liked to implement radio controls to select an RHE instead of typing it in. I didn’t quite learn how to do that so we’re using typed input for now. I’d also like to add options to enter multiple shifts at multiple RHE’s instead of displaying rates for working one specific RHE multiple times. Additionally, I’m writing this on the due date of the final and have yet to implement the actual overtime calculation and display, and that’s something I’ll work on after I finish this final paragraph. As of now, the program functions as an RHE conversion calculator, which an employee can use to figure their overtime while working other shifts