Assignment 4

Due Monday, Oct 18, 11:30 PM
50 points

Purpose

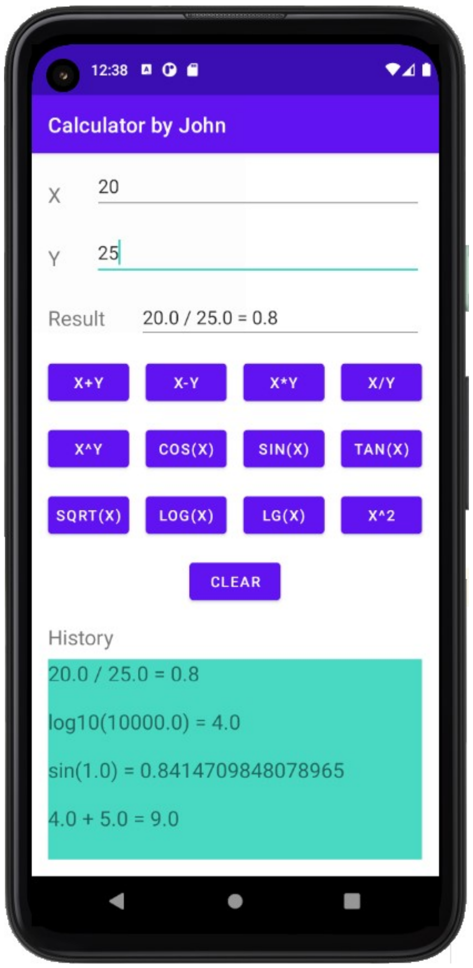
The purpose of this assignment is to

- use **`ConstraintLayout`** to create a layout
- work with multiple **`Button`**, **`TextView`**, and **`EditText`** widgets in layout and code.
- create a custom launcher icon
- continue learning Kotlin

Background

In this assignment, you will create a simplified calculator similar to the one show in the screen shot on the right. In this calculator, there are two fields for **X** and **Y**. The user can enter values in these fields and then push one of the buttons. The result is shown in **Result** as an equation that uses the values from the **X** and **Y** fields and shows the result of the calculation.

Each time the user does a calculation, the result is added to the top of the history area at the bottom of the screen. The **CLEAR** button, clears the **X**, **Y**, and **Result** fields, but it does not clear the history. The history is only cleared when the app is closed. This history is scrollable, and so the user can view many past results.



```
X       20
Y       25
Result  20.0 / 25.0 = 0.8

History
20.0 / 25.0 = 0.8
log10(10000.0) = 4.0
sin(1.0) = 0.8414709848078965
4.0 + 5.0 = 9.0
```

Calculator design

Instructions

Create a new project named **Assignment 4**. Your code and project should be set up and work according to the following requirements.

1. App label

Edit **`app/res/values/string.xml`** and change the label to say **Calculator by <name>**, where the name is your first name. For example, I changed mine to **Calculator by John**.

2. Launcher icon

Create a custom launcher icon to replace the default icon created by Android Studio when the project is first created.

3. Design and layout

Place all the widgets in a `ConstraintLayout`. Your design should look similar to the image. You should follow these special requirements as well

- The **X**, **Y**, and **Result** are labels for three `EditText` widgets. The user can type in the ones for **X** and **Y**, however, the user cannot type in the **Result** `EditText`. To prevent the user from typing in an `EditText`, configure these two attributes in the widget:
    1. `android:focusable="false"`
    2. `android:focusableInTouchMode="false"`

  The user should be able to type only decimal numbers in the `EditText` for the **X** and the **Y**. For example, `10`, `42`, or `1.45`.
- The green-ish area at the bottom is a `ScrollView`. When you put the `ScrollView` in the `ConstraintLayout`, it will automatically add a `LinearLayout`. Once the `ScrollView` is set up, add a `TextView` in the `LinearLayout`. Use that `TextView` when writing out the history. Setting up the `ScrollView`, its `LinearLayout`, and the `TextView` is the same as in assignment 2. The difference here is that the `ScrollView` area will be smaller. You will need to set the `layout_width` and the `layout_height` of the `ScrollView` to `0dp`
- Widgets should not touch each other or the side of the `ConstraintLayout`. This means everything should have a value for `layout_margin` (start, end, top, and/or bottom) on any side that has a constraint.
- Notice the buttons are all the same size and are spaced evenly. You can do this by have constraints from A to B and B to A, and then setting the `layout_margin` to the same value for each one. For example, to set up the spacing between the `COS(X)` button and the `SIN(X)` button, constrain the left side of the `COS(X)` button to the `SIN(X)` button, and then make a second connection, from the `SIN(X)` button to the `COS(X)` button. Set both margins to a value like `4dp`, which would make the two buttons be separated by a total of `8dp`.
- In the Android Studio layout designer, the layout should work for both the **Pixel** phone and the **Pixel 5** portrait mode. If the constraints, margins, and sizes are set correctly, the sizes of images will change, for example, the `ScrollView` will take up more space in the **Pixel 5** than in the **Pixel**, but all spacing (margins) should remain the relatively the same.

4. Code

Set up the code so that there is one `onClick()` function that handles every button press.

- **`EditText` fields**

  When reading values from the **X** and **Y** `EditText` fields, it is possible that the user left them empty. Make sure to handle that situation, otherwise, your app may crash.
- **Clear button**

    - When the user presses the **Clear** button, the **X**, **Y** and **Result `EditText`** fields should all be cleared. Do not clear the history.
- **Calculation buttons**
    - General rules for all the calculation buttons
        - Each button should read in the appropriate fields, do the calculation, and display the result as an equation in the **Result** field (see below).
        - For buttons that do calculations that use both **X** and **Y**, the values in each field should be checked before doing any calculation. If one of the fields is empty, do not do the calculation. Instead, you should place a warning in the **Result** field with an appropriate message. For example, if the user places 5 in the **X** field but leaves the **Y** empty and pushes the **X+Y** button, the **Result** field should have the warning message **WARNING: the Y field is empty**. Do not put warnings in the history.
        - For buttons that do calculations that use only the **X** field, the code should clear the **Y** field and then ignore it. The value in the **X** field should be checked before doing any calculation. If it is empty, do not do the calculation, Instead, you should place a warning in the **Result** field with an appropriate message. For example, if the user leaves the **X** field empty and pushes the **COS(X)**, the **Y** should be cleared and the the **Result** field should have the warning message **WARNING: the X field is empty**. Do not put warnings in the history.
        - Each result that is not a warning should be placed at the top of the history. Separate each result by a blank line for readability. For example, if the user started the app, the history would be empty. If the user added 4 and 5, the history would then look like this:

          ```
          4.0 + 5.0 = 9.0
          ```

          If the user then multiplied 2 and 3, the history would be updated to:

          ```
          2.0 * 3.0 = 6.0

          4.0 + 5.0 = 9.0
          ```

          If the user then added 42 and 10.5, the history would be updated to:

          ```
          42.0 + 10.5 = 52.5

          2.0 * 3.0 = 6.0

          4.0 + 5.0 = 9.0
          ```

        - Many calculations will result in long numbers. For example, it you divide 1 by 3 you will get .333333333333333 . Others may cause a math error. For example, 3 divided by 0 will give you a `NaN` value (Not a Number). You don't have to do anything special to handle these situations. Just display the result of the calculation. In some cases, the result may span 2 lines in the **Result** and the history. Again, this behavior is ok and you don't need to do anything special with it.
        - Some calculations require you to use the **`Math`** class, which works the same in Kotlin as it does in Java.
    - What the individual buttons should calculate and display in the **Result** field

- The **X+Y** button should use the numbers in the **X** field and the **Y** field, add them together, and show the result as an equation in the **Result** field. For example, if **X** has 3.4 and **Y** has 1.2, then the **Result**, would show

```
3.4 + 1.2 = 4.6
```

- The **X-Y** button should use the numbers in the **X** field and the **Y** field, subtract them, and show the result as an equation in the **Result** field. For example, if **X** has 3.4 and **Y** has 1.2, then the **Result**, would show

```
3.4 - 1.2 = 2.2
```

- The **X*Y** button should use the numbers in the **X** field and the **Y** field, multiply them, and show the result as an equation in the **Result** field. For example, if **X** has 3.4 and **Y** has 1.2, then the **Result**, would show

```
3.4 * 1.2 = 4.08
```

- The **X/Y** button should use the numbers in the **X** field and the **Y** field, divide them, and show the result as an equation in the **Result** field. For example, if **X** has 3 and **Y** has 1.5, then the **Result**, would show

```
3.0 / 1.5 = 2.0
```

- The **X^Y** button should use the numbers in the **X** field and the **Y** field. Use the numbers to calculate X^Y, that is "x to the power of y". Show the result as an equation in the **Result** field. For example, if **X** has 1.5 and **Y** has 3, then the **Result**, would show

```
1.5 ^ 3 = 3.375
```

- The **COS(X)** button should use the number in the **X** field to calculate the cosine of **X**. Show the result as an equation in the **Result** field. For example, if **X** has the value 0, then the **Result**, would show

```
cos(0.0) = 1.0
```

- The **SIN(X)** button should use the number in the **X** field to calculate the sine of **X**. Show the result as an equation in the **Result** field. For example, if **X** has the value 0, then the **Result**, would show

```
sin(0.0) = 0.0
```

- The **TAN(X)** button should use the number in the **X** field to calculate the tangent of **X**. Show the result as an equation in the **Result** field. For example, if **X** has the value -1, then the **Result**, would show

```
tan(-1.0) = -1.5574077246549023
```

- The **SQRT(X)** button should use the number in the **X** field to calculate the square root of **X**. Show the result as an equation in the **Result** field. For example, if **X** has the value 100, then the **Result**, would show

```
sqrt(100.0) = 10.0
```

- The **LOG(X)** button should use the number in the **X** field to calculate the log base 10 of **X**. Show the result as an equation in the **Result** field. For example, if **X** has the value 100, then the **Result**, would show

  ```
  log(100.0) = 2.0
  ```

- The **LG(X)** button should use the number in the **X** field to calculate the natural log of **X**. Show the result as an equation in the **Result** field. For example, if **X** has the value 2.71828182846, then the **Result**, would show

  ```
  lg(2.71828182846) = 1.0000000000003513
  ```

- The **X^2** button should use the number in the **X** field to calculate **X** squared. Show the result as an equation in the **Result** field. For example, if **X** has the value 5, then the **Result**, would show

  ```
  5.0 ^ 2 = 25.0
  ```

Submission

There one item to turn in. Follow the instructions in **Module 4 Tools overview** on how to submit assignments. Upload the ZIP for your Kotlin Practice project. It should be uploaded to the Assignment 4 DropBox in D2L, which is in **Tasks** > **Assignments**.

If you have any problems uploading the ZIP to D2L, email me immediately and give me any error messages the D2L gives you. Do not email me your project unless I specifically ask you to do so.