Learning Objectives:

- Gain experience creating a new Android application
- Gain experience with Android SDK event-driven programming
- Practice object-oriented programming
- (Optional) Gain experience using GitHub integration in Android Studio

Project 1

There will be no "live" demos of Project #1 because we are remote this term.
Please include a video demonstrating that your app is working correctly
Deliverables are due to D2L by 10:00 PM on Saturday, 30-Jan-2021

NO ASSIGNMENTS WILL BE ACCEPTED AFTER 3:00 PM ON MONDAY, 01-FEB-2021

Introduction

This assignment builds Ohms Law calculator app we studied in class and on the experience you've gained (or are gaining) by implementing `GeoQuiz` from BNRG. In this project you will create a straightforward Android app using Android Studio or IntellIJ with the Android SDK plug-in. The structure of this app is similar to that of `OhmsLawCalculator_r0` so the source code for that app could potentially provide reference code to work from. The Project #1 app is a basic calculator (the slang term for this is a "four-banger" calculator):

"Many of the first electronic desktop calculators cost over $1,000. I'm talking about one thousand 1966 dollars. In fact, a new calculator could cost more than a new car! 'What do you think honey, a new car or a new calculator?' Today this seems truly absurd. By the dawn of the pocket electronic calculators in the early 1970's prices had fallen significantly but a simple four-function model could still set you back more than a week's salary. Why were the early machines so expensive? Were manufacturers cranking them out for pennies and selling them for huge profits? No. In reality the early electronic calculator business was highly competitive and prices were closely linked to the cost of raw material. As the cost of raw materials fell, calculator prices plummeted and profit margins shrunk. By the 1980's four-function calculator prices broke through the $10 barrier. This would have seemed impossible a decade earlier." [Editor's note: I priced a brand new solar-powered basic calculator for $3.99 on Amazon].

Source: http://www.vintagecalculators.com. Written by Bruce Flamm (Bflamm@ix.netcom.com. Edited for brevity by R. Kravitz.

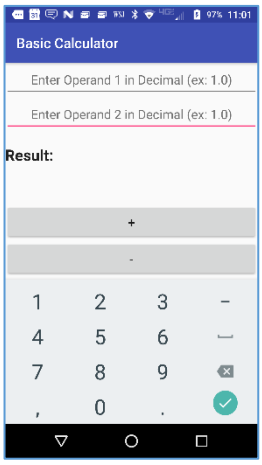
Use of Git and GitHub Classroom

While I strongly encourage you to use the Android Studio or IntelliJ integration of Git and GitHub on Project #1, using it is optional for the project. There is a short time to complete this project and with the transition from pure Java to Android I don't want to burden you with learning yet another tool. Project #2, Project #3, and the final project will make use of GitHub classroom to distribute and grade your project and GitHub classroom depends on you being competent in using Git and GitHub. Since we are not teaching Git or GitHub in this course I have included a file in the project release which gives you links to good tutorials (including a few video tutorials) on installing and configuring Git and GitHub, on using Git and GitHub, and on using the Android Studio integration with GitHub.

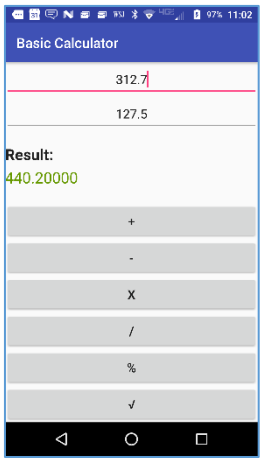
Note: It is easy to create a `.zip` file from a GitHub repository (one of your deliverables for this project) to submit as part of your deliverables for the project. In GitHub select the repository for your project, click on the ***Clone or Download button*** and select Download ZIP. You can do the same if you keep your project in a local repository on your host machine. Just compress (Windows) or run WinZip or 7zip on your project directory.

Specification:

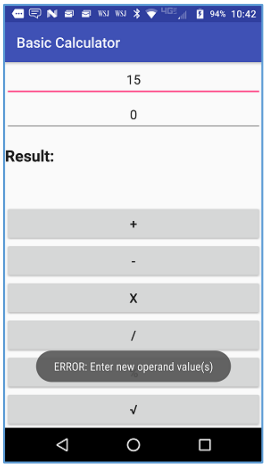
The calculator should accept two decimal (non-negative, decimal point allowed) operand values from the user and perform one of 6 functions (Add, Subtract, Multiply, Divide, Percent, and Square Root). The first four operations take both operands. Percent and square root only take a single operand. In addition to the `EditText` fields for the operand(s), the user interface should include a result with a label (both implemented as `TextView` widgets) and 6 buttons, one for each calculator function. My app's user interface looks like this:



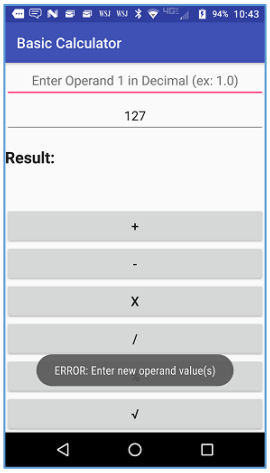
Waiting for user input



Successful Add Operation



Divide by 0



Missing Operand

As you can see, on a successful operation (e.g. valid operand(s)) the answer is displayed under the "`Result:`" label and the app resumes waiting for the user to enter new operands or click on a different operation using the same operands. The result should stand out. I did this by displaying the result in a different color but you could use a different font or a different font size for a similar effect.

As the author and implementer of the app you can design a different user interface but the release for the assignment includes a figure showing the attributes of my layout if you'd like to implement it.

Functionally, there's not much to say about a calculator. The user enters one or two operands using the keypad on the device and then clicks on one of the 6 buttons to select the operation to perform. The app should also detect, and properly handle, error conditions.

On error conditions, such as divide by 0 or a missing operand, an Android *`Toast`* is displayed, the app erases the errant operands, sets the focus (e.g. the spot where the user can interact with the app) on the first operand and resumes waiting for the user to perform the next calculation. Both operands and the result are Java *`double`* variables.

The following table summarizes the operations of the calculator:

Basic Arithmetic Operations			
Function Button	**Operand 1**	**Operand 2**	**Operation**
+ (ADD), - (SUB), X (MULT), / (DIV)			
	no entry	no entry	ERROR
	no entry	valid decimal	ERROR
	valid decimal	no entry	ERROR
+ (add)	valid decimal	valid decimal	op1 + op2
- (subtract)	valid decimal	valid decimal	op1 - op2
X (multiply)	valid decimal	valid decimal	op1 * op2
/ (divide)	valid decimal	valid decimal	op1 / op2
Extended Arithmetic Operations			
% (percent)	no entry	no entry	ERROR
	no entry	decimal num	ERROR
	valid decimal	no entry	op1 / 100.0
	valid decimal	valid decimal	op1 / 100.0
√ (square root)	no entry	no entry	ERROR
	no entry	valid decimal	ERROR
	valid decimal	no entry	math.sqrt(op1)
	valid decimal	valid decimal	math.sqrt(op1)

This app can easily be implemented in a single activity. Your app does not need to deal with configuration changes like screen rotations, nor does it need a different layout when the device is rotated. Your app does not need to concern itself with localization changes, either, but it is good practice to use string references (in `strings.xml`) instead of "hardwiring" text into the layout.

Demo:

You will include a video that demonstrates your application on your own Android device.

Deliverables:

Submit a single `.zip` or `.rar` file (`ece558proj1_yourname.zip`) to your D2L dropbox for Project 1. The file should contain:

- An archive of the final version of your app. Android Studio and IntelliJ use the file manager and filesystem of the host PC so create a .zip or .rar file for the entire project directory. "Neatness counts" for this project - we will grade your code for readability. You code should be well structured, indented consistently (Android Studio and IntelliJ help with that) and you should include comments describing what long sections of your code do. Comments should be descriptive rather than explain the obvious (ex: //set a to b when the actual code says a = b; does not provide any value-added). NOTE: Make sure that your archive includes the source code you want to submit for grading and not some earlier version. You do not need to submit

JavaDoc with this project but it is good practice to include JavaDoc-formatted comments for all of your key methods in your source code.

- A signed `.apk` (Android Application Package) file that can be downloaded and installed on an Android device. You can generate an `.apk` file from Android Studio by selecting `Build/Generate Signed APK`. It should be noted that using Android Studio to generate an `.apk` file will require that you "sign" your app with a keystore. The `Generate Signed APK` wizard will help you through the process.

Grading:

You will be graded on the following:

- Functionality of your app during the demo 75 pts
- Code quality (readability, comments, etc.) 15 pts
- Signed .apk (We may install your app on our device) 10 pts

Acknowledgements:

- Bill Phillips, Brian Hardy, Kristin Marsicano and the other authors, developers and instructors at Big Nerd Ranch, Inc. for creating excellent "how to" projects that show how to implement many of the features that have become standard on today's mobile devices.
- Kristin Marsicano, my Android Boot camp instructor (Aug 2013), who spent several hours with me, discussing ways to teach the SDK and Android programming to those of us who may not be familiar with, and/or comfortable with events, Java, objects, and object oriented programming.
- The Android developer Community and Google for providing tutorials, explanations and Android program examples. There is a wealth of information "out there" and it's only a few searches away.

<finis>