CST2335 – Graphical Interface Programming
Lab 1

Introduction:
The goal of this lab is to ensure that you can install Android Studio, create a sample project, and commit it to a Git repository. This repository will initially be on your computer, but you are encouraged to create a Github repository as a backup location.
You will also learn how to use string resources to support multi-lingual applications. This will show you how the Java files integrate with XML files to build your application.

Reference:

Look at the week 1 lecture slides and example code on Github at:
https://github.com/etorunski/Winter2021_Examples.git
https://www.tutlane.com/tutorial/android/android-localization-multi-language-with-examples

Steps:

1. If you use Windows and have never installed Git before, then go to https://git-scm.com/downloads and download Git.

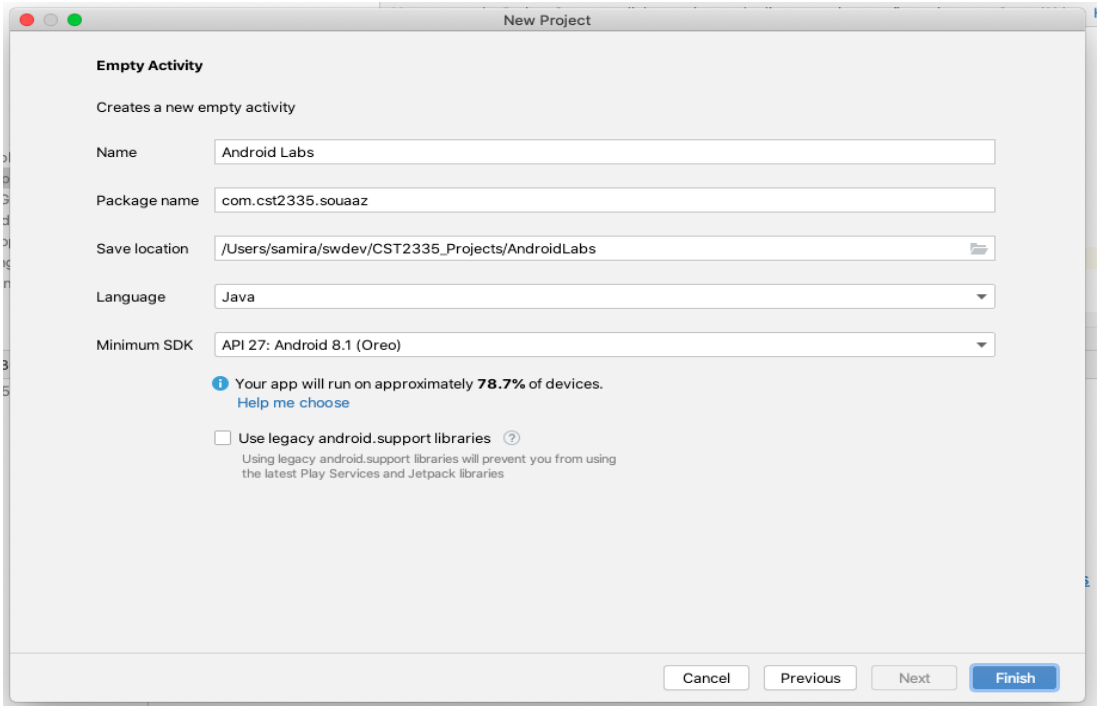
2. Install Android Studio version 4.2, from here: https://developer.android.com/studio

3. Start Android Studio

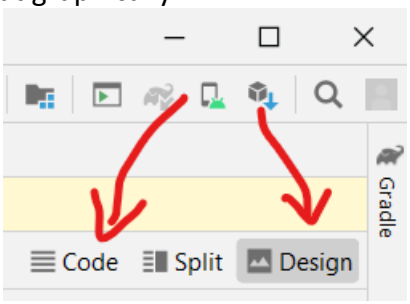
4. Create an Android Virtual Device (AVD):
   https://developer.android.com/studio/run/managing-avds.html
   Create a device for Android-27, Oreo Android 8.1. It is recommended to select a Nexus device such as Nexus 4 to conserve memory. Make sure you install the Android 8.1 system image in addition to whatever the instructions tell you.

5. Create a project using “Start a New Android Project”. Select the Phone and Tablet tab. For the activity type, select “Empty Activity” and click “Next”.

6. To configure your project, give your project the name “Android Labs”. Select the package name to be “com.cst2335.***abcd***, but replace ***abcd*** with your ***Algonquin ID***. Pick either Java or Kotlin as your programming language, set the minimum API level to be API 27: Android 8.1 and click “Finish”.



7. Once the project has been created, and Gradle has finished compiling, go to the folders: “app” -> “res” -> “layout” and click on “activity_main.xml”. Select the “Design” tab at the top right to edit the layout graphically.



There should be a TextView saying “Hello world!”. This string is hard-coded, which is a bad idea. Go to the “res” -> “values” folder and open the file “strings.xml”. Between the resources tags, put a new string value, whose name is “hello_message”, and value is “Hello World!”. You can copy and paste the existing <string> entry for “Android Labs”, but change the name parameter, and value between the tags.

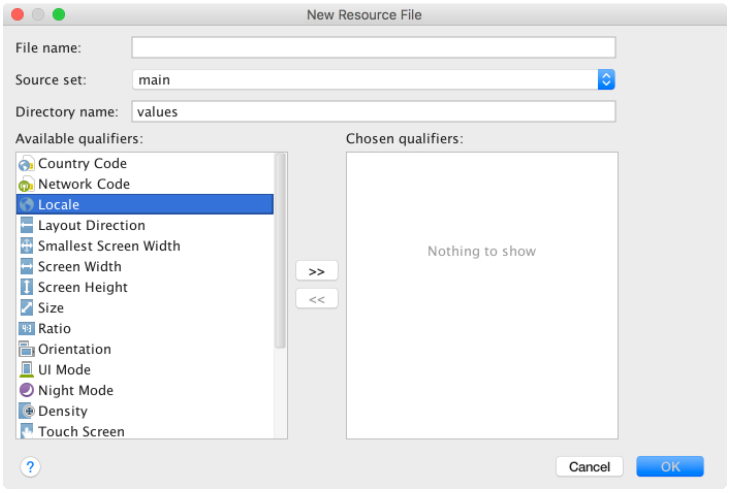
```
<resources>
    <string name="app_name">Lab1</string>
    <string name="hello_message">Hello World!</string>
</resources>
```

8. Go back to the activity_main.xml file and click the “Code” tab beside the Design tab to change the *android:text=”Hello World!”* to: *android:text=”@string/hello_message”*. Android Studio should offer this value as a quick-suggest value. Anything with an @

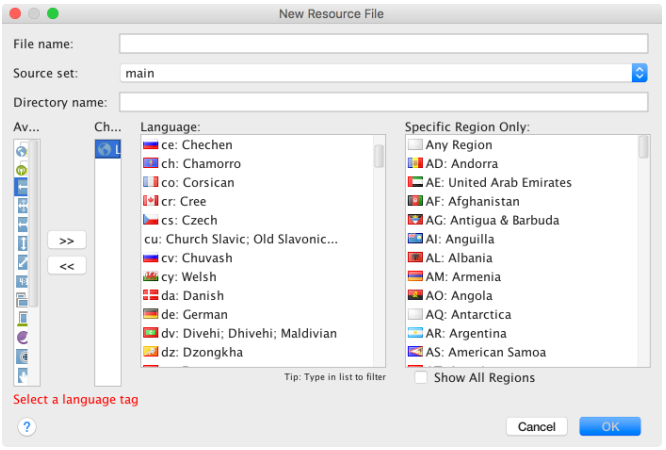
symbol means that it is a resource value. @string/hello_message means go to strings.xml, and look for the <string> tag whose name equals “hello_message”.

9. Switch back to the “Design” tab and see that the TextView still shows “Hello World”.

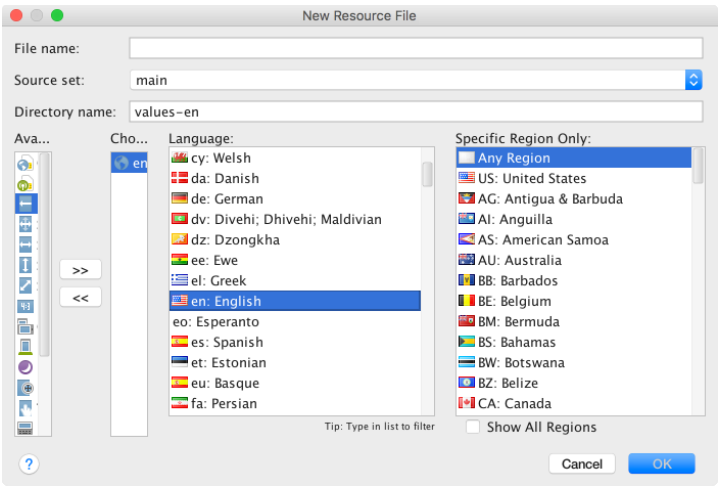
10. Right-click on the “app” folder and select “New” -> “Android Resource File”. Select “Locale” from the list and click the “>>” button to choose which locale.



11. From the list of languages, select any language other than English that you can speak:



12. If you only know how to speak English, then select en:English, and select a country such as Canada, or Australia:



13. Set the file name as “strings” and click “Ok”. This should create a second “strings.xml” file in a different “values-XX-rYY” folder, where the first XX is the language, and the second rYY is the specific region. You should leave the Specific Region to “Any Region”. Copy the tags between the <resources> tags of the original “values/strings.xml” file and paste them between the <resources> tags of the new “values-XX-rYY/strings.xml” file.

14. In the strings.xml file for the second language, modify the strings to the new language. For example, replace “Hello World!” with “Bonjour!”, or “Guten Tag!”, or whatever you want to translate the message to.

15. Go back to the activity_main.xml, and make sure you are in the “Code” mode. Change the root XML tag from **<androidx.constraintlayout.widget.ConstraintLayout>** to **<LinearLayout>**. Add the attribute **android:orientation=”vertical”**. Now switch to the “Design” mode. Click on the Button widget and drag it over to your interface, under the “Hello World!” text view.

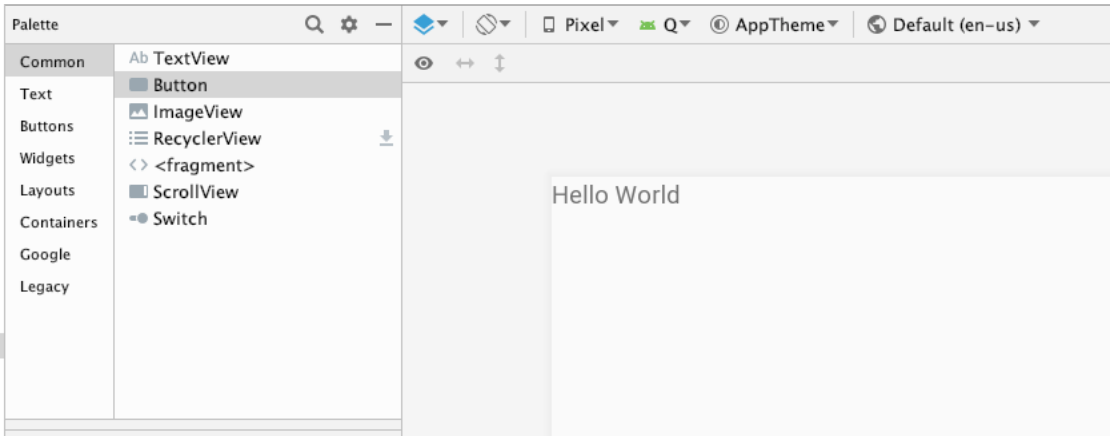
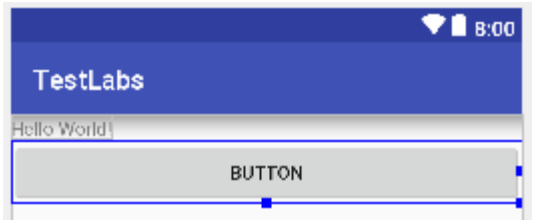


Figure 1 The widgets are on the left side. Drag a button under "Hello World"



Change your view to “Code” mode, and find the <Button> tag. Somewhere in the tag is the line： android:text="Button"
Change it to: android:text="@string/ButtonText"
This means that Android will look in your strings.xml file and look for the <string name=”ButtonText”> tag. It doesn’t exist yet so you must go to both strings.xml files and add a string tag for your button: ***<string name=”ButtonText”>I\’m a button</string>***
The backslash is required before the apostrophe (**\’**) so that XML doesn’t think you are declaring a string. Make sure to add one in the default strings.xml, and your second language strings.xml file.

16. Once you are done, click Debug, and create a new instance of an emulator. This will launch an emulator, then install and launch your application. Demonstrate that your application runs and shows a TextView and Button. Also, change the device’s language in the “Settings” -> “Language & Input” -> “Language” menus. Set the language of the phone to whatever you chose as your second language. Your application should automatically display the strings from your other strings.xml file. If you have an Android phone or tablet, you can use it to demonstrate instead of the emulator. Here are the instructions how to use your device:

    http://developer.android.com/tools/debugging/debugging-studio.html
    http://developer.android.com/tools/device.html

17. Select the “VCS” menu and click “Enable Version Control Integration”. Select “Git” for the Version Control System and click Ok. Now under the “VCS” menu, there should be “Git” option. If there isn’t, then tell AndroidStudio where Git is installed through “File->Settings->Version Control->Git->Git executable”.

18. Create an account on Github (http://www.github.com). Click the “Sign up” button on the top right. Make sure to use your Algonquin College email. Next click on the “Select a Plan” button. Make sure you select the free plan. You will get an email to confirm your email address. Check your email account for an email from github. You will have click a link to activate your account.

19. From the VCS menu, select “Import into Version Control”, and select “Share project on Github”. Call the repository “AndroidLabs” and select the “Private” checkbox so that no one else can see your labs on Github. Enter your login name and password for Github. If you see a window “Setup Master Password”, this is a password to create a login database on your computer which will store your login names and passwords for all

your accounts. The password you create doesn't have to be the same as your Github password.

20. You can now right-click on any files in your project, such as strings.xml or MainActivity.java, and there should be a "Git" menu. From this menu, you can commit the file if you make any new changes (which you will in this course), you can view the history of your commits, and compare your file with previous versions.

21. Other things to try are:

    a. to go to the "app/src/main/res/mip-map*" folders and edit the ic_launcher.png images. This will change the application's icon when installed on Android.
    b. You should also see in the "app/src/main/res" folder that there is a new "values-xx-xxx" folder that you created with your new strings.xml file.

Marks:

The app starts without crashing	+1
The app has a text label saying "Hello world"	+1
The app has a button saying "I'm a button"	+1
The app supports a second language	+1
Your code has been uploaded to your github account	+1