ATLS 4120/5120: Mobile Application Development
Week 15: Device Environments

In the Finder make a copy of your coffee project – right click Duplicate and change the name of the folder (Coffee Adapt)

Supporting different screens

https://developer.android.com/training/basics/supporting-devices/screens.html
Android categorizes device screens using two general properties: size and density. The screens orientation (landscape or portrait) is considered a variation of screen size. Android automatically scales your layout in order to properly fit the screen. So your layouts for different screen sizes don't need to worry about the absolute size of UI elements but instead focus on the layout structure that affects the user experience (such as the size or position of important views relative to sibling views). If you can't define one layout that works on all of these you need to define different layouts for each screen size and they need to be saved in a folder with the screen size name(layout-large, layout-land). The default layout is in the res/layout folder.

Go into our coffee app and see how the first activity looks in landscape mode.
Using constraint layout it seems that all the constraints were to the margins so in landscape the button and image were up too high. I changed a few constraints by deleting the constraints to the margins and replaced them so that they were relative to other widgets using
app:layout_constraintTop_toBottomOf and that fixed them a bit.

Now we're going to see how you define a new layout for landscape mode.
https://developer.android.com/studio/write/layout-editor.html#create-variant

Open the layout file
Design mode or preview
Click the Orientation in Editor in the toolbar.

Create landscape variation
In res/layout/activity_find_coffee.xml it creates a new layout file
In the Project view you can see how this is really stored in app/src/main/res/layout-land
Edit this so you have a layout that looks good in landscape

I did this by putting the button to the right of the spinner and the image view to the right of the button. First delete all constraints for the button and image view and then move them. With Autoconnect on it will then add the constraints. I then had to add vertical constraints.
Note: Instant Run did not seem to apply these constraint changes and I had to stop and rerun the app.

Run it and try it in both orientations. (command arrow on the Mac to rotate)

Supporting different languages

We've been using IDs in our layout xml files and assigning those strings in our strings.xml file
res/values/strings.xml is the default strings file. If this default file is absent, or if it is missing a string that your application needs, then your application will not run and will show an error.

To add support for more languages, create additional values directories inside res/ that include a hyphen and the ISO language code at the end of the directory name. For example, values-es/ is the directory containing simple resources for the Locales with the language code "es". Android loads the appropriate resources according to the locale settings of the device at run time. You can do this for drawables too.

In the Project view go to app/src/main/res right click New | Android Resource Directory values-fr
(French fr, Spanish es, Japanese ja, German de. Other ISO language codes
http://www.loc.gov/standards/iso639-2/php/code_list.php)
Right click on your new folder New | Values resource file
strings.xml
Open your original strings.xml file and copy all the strings you have.
Go into your new fr/strings.xml file and paste them in the <resources> tag.
Change the string values for your new language. (There are services that do this for a fee)
(hippie had no translation so I just left it as I don't really know French.)

When a user runs your app Android selects which resources to load based on the device's locale.
If it finds a resource for that locale it will use the resources in it. If it doesn't find a resource in that file it will look in the default resource file.

To test this in the emulator you need to change the language of the device.
Settings | Personal | Language & Input
Change the language to the language you're localizing to (Francais(France))
(Android also lets you define a custom locale, but we shouldn't need that)
Go back to the screen of apps and you'll notice that the apps that come with the phone, Settings, Camera, Phone etc are now all in the language you picked.

When you run your app you should see all your strings in the new language, but your second activity still has English.

To fix this we should make strings in our Java file IDs and define the string in the xml file just as we did with our layout. In your strings files add
`<string name="message">You should check out </string>`
and the French version `<string name="message">Tu aimerais </string>`

In ReceiveCoffeeActivity.java let's get that string resource and use it in our TextView.
```
String message = getString(R.string.message);
messageView.setText(message + coffeeShop);
```

Now run your app.
Add in a space in your message
```
messageView.setText(message + " " + coffeeShop);
```