ATLS 4120: Mobile Application Development
Week 13: Device Environments

With Android running on so many different size devices your app layouts must be flexible and adapt to different size screens and orientations.

Supporting different screens

https://developer.android.com/training/multiscreen/screensizes
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).

Constraint Layout
Constraint layout creates a layout responsive for different screen sizes.
Avoid hard coded layout values

- "wrap_content" tells the view to set its size to whatever is necessary to fit the content within that view.
- "match_parent" makes the view expand to as much as possible within the parent view.

Create alternative layouts
If you can't define one layout that works for all screen sizes or orientations you'll need to define different layouts for each configuration orientation.
Remember that the default layout is in the res/layout folder.

Screen size qualifiers:
You can provide screen-specific layouts by creating additional res/layout/ directories and then append a screen configuration qualifier to the layout directory name for each screen configuration that requires a different layout

- layout-w600dp for screens that have 600dp of available width

You can also use the smallest width qualifier(sw) to provide alternative layouts for screens that have a minimum width measured in density-independent pixels

- layout-sw600dp for screens that have a minimum available width of 600dp

Orientation qualifier:
To provide different layouts based on the device's orientation you can add the "port" or "land" qualifiers to your resource directory names. Just be sure these come *after* the other size qualifiers.

- layout-land for devices in landscape orientation
- layout-sw600dp-land for devices with 600dp wide and larger and in landscape orientation

Supporting different languages

We've been using IDs in our layout xml files and assigning those strings in our res/values/strings.xml file, which 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. Android loads the appropriate resources according to the locale settings of the device at run time. You can do this for drawables too.

- values-es/ is the directory containing simple resources for the locale with the language code "es".

When a user runs an 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. If it can't the resource in any file the app won't load.

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

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 and that fixed them a bit.
Aligning the image to the start and the end of the button also helped keep it small.

Landscape variation
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_main it creates a new layout file.
In Android view you see (land) next to it.
In the Project view you can see how this is really stored in app/src/main/res/layout-land
The title of the tab has the prefix "land/"
Edit this so you have a layout that looks good in landscape.
Be careful that you know which file you're editing.
The changes you make here are only for this file. So it makes sense to do it at the end once the app is working because if you want to add new widgets or change anything for the existing widgets, you'll need to do it twice, in each file. They are in no way connected.

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. Then use Infer Constraints for both and modify as needed.

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

Localization
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.)

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)
On the Pixel: Settings | System | Languages | Add a language
Then in the list move the new language above English.
Now 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 CoffeeActivity.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**);