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
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. We could try to fix it, but instead let's go ahead and define a new layout for landscape mode.

Open the layout file
Design mode or preview
Click the left most button in the toolbar right above the phone image
Create landscape variation
Creates new layout file in a land folder
Can't see the folder in the Android view, change to Project view
app/src/main/res now you can see the new layout-land folder with the newly created layout xml file
Edit this so you have a layout that looks good in landscape

I did this by putting the Spinner and ImageView next to each other in a Linear Layout and put the button below.

```
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   android:layout_below="@+id/textView2"
   android:id="@+id/linear">

<Spinner
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/spinner"
   android:entries="@array/crowd"
   android:paddingLeft="50dp"
   android:layout_marginTop="20dp" />
```

```
    <ImageView
        android:layout_width="135dp"
        android:layout_height="137dp"
        android:id="@+id/imageView"
        android:src="@drawable/coffee"
        android:contentDescription="@string/coffee_image"
        android:layout_weight="1"
        android:paddingLeft="100dp" />
</LinearLayout>
```

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)

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.

Now run your app in the emulator.

You should see all your strings in the new language, but there are a few problems.
1. The right coffee shop is not being picked
2. In your second activity there's still English

#1 is because we wrote our Java to test for the English strings. So instead of checking for the string let's use the position in the string array.
In FindCoffeeActivity.java change String crowd = String.*valueOf*(crowdSpinner.getSelectedItem()); TO Integer crowd = crowdSpinner.getSelectedItemPosition();

Now we need to change CoffeeShop.java so setCoffeeShop(), setCoffeeShopURL(), and setCoffeeInfo() take integers.

```
public void setCoffeeShop(Integer coffeeCrowd)
public void setCoffeeShopURL(Integer coffeeCrowd)
private void setCoffeeInfo(Integer coffeeCrowd)
```

Then we have to change our case statements to use the array position (starting at 0) instead of the string.

```
private void setCoffeeInfo(Integer coffeeCrowd){
   switch (coffeeCrowd){
      case 1:
         coffeeShop="Amante";
         coffeeShopURL="https://www.amantecoffee.com";
         break;
      case 0:
         coffeeShop="Starbucks";
         coffeeShopURL="https://www.starbucks.com";
         break;
      case 2:
         coffeeShop="Ozo";
         coffeeShopURL="https://ozocoffee.com";
         break;
      case 4:
         coffeeShop="Trident";
         coffeeShopURL="http://www.tridentcafe.com";
         break;
      case 3:
         coffeeShop="Pekoe";
         coffeeShopURL="http://www.pekoesiphouse.com";
         break;
      case 5:
         coffeeShop="Buchanans";
         coffeeShopURL="http://www.buchananscoffeepub.com";
         break;
      default:
         coffeeShop="none";
         coffeeShopURL="https://www.google.com/search?q=boulder+coffee+shops&ie=utf-
8&oe=utf-8";
   }
}
```

To fix #2 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**);