Girl Develop It, Class 3

- res/
  - values/styles.xml
    ```
    <style="NumberKey" parent="Wrap">
      <item="android:textColor">@color/white</item>
    </style>
    ```
    - Group of reusable widget concepts that can be applied to multiple XML views.
    - They are useful for simplifying XML layouts because it reduces duplication and places that you need to update when things change.
    - Can inherit from other styles (using parent="")
    - Themes are a special styles that are applied to the entire application. This is good for things like setting the application's background color, whether to show a title bar or not, etc.
  - values/dimens.xml
    - Dimensions are used to describe width and heights of views. They can also be used to specify text sizes or margins.
    - It's good practice to define dimensions in values/dimens.xml instead of hard coded on your XML widgets. This allows you to change them quickly and define different sizes for different screens.
    - Sizes
      - match_parent (deprecated: fill_parent) - this view should fill the entire screen.
      - wrap_content - make this view no bigger than it needs to be to contain this element.
      - You can also define custom sizes in dp's (density independent pixels).
        ```
        <dimen name="NumberButtonHeight">30dp</dimen>
        ```
      - Font sizes are specified in sp (not dp), which gives the benefits of dp, but text size also changes when the user changes their text viewing settings.
        Example: Extra large font sizes may be turned on for the visually impaired and you would want your app to use large fonts also.
  - values/strings.xml
    - You put strings like error messages, labels, and other static messages into the strings.xml file.
    - This makes it easy to internationalize, reuse, and update if they are centralized in one place.
    - Can use <b> <i> and <u> to style them
      ```
      <string name="BOLD_STRING">Hello <b>there</b></string>
      ```
    - Can also use a format string
      ```
      <string name="BROKEN_THINGS">%d of your things are broken.</string>
      ```
  - values/colors.xml
    - You can define your own colors in values/colors.xml.
    - There are several ways to specify them, but the simplest is #RRGGBB.
      ```
      <color name="red">#FF0000</color>
      ```
    - IntelliJ has a GUI color picker.
  - drawable/
    - This is where you store images that will be used in the application. You can use JPG and PNG, but PNG is preferred.
      Note: File names must be all lower case, alpha-numeric, or with underscores.
    - Can define shapes in XML or use special 9-patch files that stretch as specified.
    - Can also describe states of an image or button. The states can refer to images, colors, or XML shapes.
      ```
      <selector>
        <!--Pressed-->
        <item android:state_pressed="true" android:drawable="@color/selected_brown"/>

        <!--Focused-->
        <item android:state_focused="true" android:drawable="@color/focused_yellow"/>

        <!--Normal-->
        <item android:drawable="@color/brown"/>
      </selector>
      ```
- Making interfaces that scale http://developer.android.com/guide/practices/screens_support.html
  - Best Practices
    - Use wrap_content, match_parent, or dp units when specifying dimensions.
    - Don't use exact pixel sizes when defining a view because it won't scale or look good on other devices.

- Create drawables for different densities so the images don't look bad when scaling
  xhdpi, ldpi, mdpi, hdpi
- Create styles and use dimensions to scale your views. Sometimes you don't need a whole new layout to make your app look good on larger screens. For example, with the calculator app, we just want the buttons to look bigger. You can create a style that uses a dimension. Then using the qualifier system (described below), you can specify how big the dimension is on the two different screens.

- Using Android's qualifier system.
  - Qualifiers are used on folder names to specify if it relates to a specific screen size, orientation, or SDK version.
    - layout-small, layout-medium, layout-large, layout-xlarge for different screen sizes
    - layout-land for landscape version of your view
  - Android magically shows the correct view for you if you put it in the right folder.
  - Responsive: You can create buckets for custom screen widths or heights. For example, if your current layout looks great up to a 300dp screen width you create a layout for the smaller screen and then another one for the larger size. Android knows which to sjow.