Mobile Application Development
Week 4: Advanced Adaptive Layout

Adaptive Layout

The goal of adaptive layouts is for your user interface to look good and work well on all iOS devices by adapting to the user's environment.

- Orientation
- Device size and settings
- Locale

Size classes
https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/adaptivity-and-layout/
Instead of thinking in terms of different device sizes, Apple generalized these into size classes.
Each iOS device has a default size class dependent on orientation and screen real estate (think split screen)
Size classes define height and width dimensions as either compact or regular and these two dimensions for a trait collection.

- Size classes enable a layout to work with all iOS device screen sizes and orientations
- Build your interface as it will look on most devices and orientations (for iPhone that's w Compact h Regular)
- If you can't get the layout to work on all sizes and orientations, or you want a different layout, you can specify a different layout for a certain size class.
- Change the interface for different size classes by varying traits for a size class
    - Add a variance to change an attribute
    - Uninstall a view or constraint
        - NEVER delete a constraint as that will delete it from all size classes
    - Create new constraints
- When you vary for traits the variations are only for that size class

daVinci

(daVinci autolayout size classes)
I'm going to change the layout for the size class w Compact h Compact (such as iPhone SE landscape).
View the storyboard as iPhone SE landscape orientation and click Vary for Traits and select both width and height to target compact width and height size classes.
Notice the bottom bar turns blue meaning we are editing ONLY for this size class.
Move the views around so the label is still centered but the buttons are on the left and the image on the right instead of being stacked. Ignore all the auto layout warnings, we'll get to them.
Stack view:
Select the stack view (document outline) and in the size inspector for any constraint double click on it, click on the + next to installed, chose our current size class, Width Compact | Height Compact, and Add Variation
This adds a row for this size class. Now uncheck installed for this size class. Notice the constraint is still installed for other size classes.
Never delete a constraint or it will be deleted for ALL size classes. You must add a variation and then uncheck install for the size class where you don't want it applied.
Do this for all the stack view constraints.
If you want to change the stack view to be vertical go into the attributes inspector and click the + next to Axis and create a variation for the wC hC size class and change it to vertical.
Now we need constraints for the stack view for this size class.

Add constraints:
X position: Leading space to superview(100)
Y position: Align vertically in container
Notice that the new constraints are only installed for the wC hC size class.
If you need to change the spacing you must do the same thing. Click on the + next to spacing, chose our current size class, Width Compact | Height Compact, and Add Variation. Then change the value just for this size class.
You use this process for all the attributes with a + next to them so you're changing the value only for the size class.

Image:
Repeat what we did above to uninstall the Align Center X, Equal Width to Superview, and Align Bottom to Safe Area constraints.
The width, height, and aspect ratios remain unchanged.
Add constraints:
X position: Trailing space to safe area (50)
Y position: Align vertically in container
That should get rid of any errors or warnings you had.
To get the image to scale like before, we add the following vertical constraints:
Top space to the label (10) and make this High priority (750)
Align Bottom to Safe Area (10) and make this High priority (750)
Notice these new constraints are not installed for all the other size classes.
This will let it grow towards its width and height constraints but not cover the label.

When you're done editing for this size class click Done Varying. The bottom bar should no longer be blue which means your back to editing for all size classes.

This should make the layout in two columns for size class Width Compact | Height Compact and not change the others.

Beatles
The layout we came up with is very vertical and doesn't work in landscape so we'll vary the layout for the Width Compact | Height Compact size class.
Change the size class to w Compact h Compact (such as iPhone SE landscape).
Click Vary for Traits and select both width and height to target compact width and height size classes.
Notice the bottom bar turns blue meaning we are editing ONLY for this size class.
Select the top level stack view and go into the attributes inspector.
Click the + next to axis to add a variance for the Width Compact | Height Compact size class and change it to horizontal.
Now it will lay out the two embedded stack view horizontally and in that one step the layout works.
But to make it even better uninstall all the constraints on the top level stack view except the bottom to safe area constraint.
Add a vertical in container constraint to center it vertically and do the same horizontally as well.
If you want the stack view that has the buttons, segmented control, and slider to be spaced out more, add a variance for Spacing (60).
If the text can grow larger than the size of the label you might need a width constraint for the label (260).

When you’re done editing for this size class click Done Varying. The bottom bar should no longer be blue which means your back to editing for all size classes.