ATLS 4120: 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
(slide)
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
    - Size or position of a view
    - Add or uninstall a view or constraint
- 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 iPhones (other than plus) in landscape.
Change the size class to w Compact h Compact (such as iPhone 8 landscape).
Click Vary for Traits and select both width and height to target compact width and height size classes.
This should show 4 phones in landscape orientation.
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're getting there.
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, Compact width | Compact height, 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.
Now we need constraints for this size class.
Add constraints:
X position: Leading space to superview(40)

Y position: Align vertically in container
Notice that the new constraints are only installed for the wC hC size class.
Update frames (just for the stack view) to see where it will be placed.
If you need to change the spacing you must do the same thing. Click on the + next to spacing, chose our current size class, Compact width | Compact height, 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 chaning 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.
Add constraints:
X position: Trailing space to safe area (40)
Y position: Top space to label bottom(20) and Bottom space to bottom safe area (>= 20) to ensure there's a border.
Width: pin to 180 but make it high(not required) priority so it will make it bigger if it can.

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 compact | 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 w Compact h Compact size class.
Change the size class to w Compact h Compact (such as iPhone 4s 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 this size class and change it to horizontal.
Now if will lay out the two embedded stack view horizontally and in that one step the layout works.
But to make it perfect uninstall the constraints on the top level stack view.
Add a vertical in container constraint to center it vertically.
For horizontal add a leading constraint. That might cause the right side to off the right on the smaller phones.
Add a trailing constraint. This will give you an error because it can't do both. Make the trailing constraint <= and it should resolve the error and look good.
If you want the buttons to align to the top of the image, and the slider to be aligned with the text label you could add a variance for the stack view's alignment and change it to fill, but this will stretch the buttons.
Instead, select the stack view that has the buttons, segmented control, and slider and add a variance for Spacing. Increase that value until the stack view has a height that matches the stack view with the image and text and it will all be aligned.