**Topic: Adaptive Layout with Size Classes**  
(Previous topic: More UI controls)

Intro

* See how everyone is doing on the lab assignment
* Discuss the code review process and due date

Overview

* Auto-layout constraints can only do so much- they can’t move controls around and give you a different arrangement.
* Size-classes let you have different control arrangements for different orientations and screen sizes.

Orientations

* Ch. 5 Example from textbook: Orientations
  + Specify supported orientations at the app level – info.plist, iPhone /iPod deployment Info &   
    iPad Deployment Info.
  + Specify supported orientations at the ViewController level – override GetSupportedInterfaceOrientations.

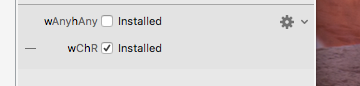
Auto-Layout (Review)

* Layout example
  + Labels are all positioned relative to the super-view using constraints. They stay in the correct relative positions for all sizes and rotations.
  + Note that this example uses:
    - Constraints from edges to adjacent edges
    - Constraints from edges to the superview
    - Constraints from the center to the X axis (align center Y)
    - Width constraints between two labels (maintain equal width)

Size-Classes

* Four combinations cover all devices and orientations (pg. 159):

|  |  |  |
| --- | --- | --- |
| **Width** | **Height** | **Device and Orientation** |
| Compact | Compact | All iPhones in landscape, except 6 Plus |
| Compact | Regular | All iPhones in portrait |
| Regular | Compact | iPhone 6 Plus in landscape |
| Regular | Regular | All iPads, both orientations |

* wAny, hAny is used as a base configuration
* Add other size class combinations for some or all of the four categories of device and orientation.
* To over-ride a constraint in wAny or hAny
  + Switch to a new size class
  + Add the constraint,
  + Select it, and in its properties, add a check box for the new size class, then uncheck the size class with “Any”.   
    
* Restructure example
  + Demonstrates Size Classes