UnitConverter
Lesson 7

Description

Add a new `TemperatureRange` "view model" to the project to encapsulate the range of temperatures used by the picker view.

Learning Outcomes

- Analyze code and categorize its role within the MVC pattern.
- Discover the concept of a "view model," and how to add custom objects to an app interface using Interface Builder.
- Compare and contrast domain-specific models with "view models."
- Practice using the Identity Inspector and Connections Inspector to define object attributes.

Vocabulary

Model-View-Controller	model	controller
problem domain	view model	Document Outline
Identity Inspector	Connections Inspector	

Materials

- **UnitConverter Lesson 7** Xcode project

Opening

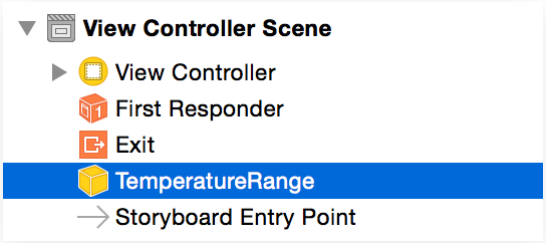
There is another model "hidden" in our code, do you see it?

Agenda

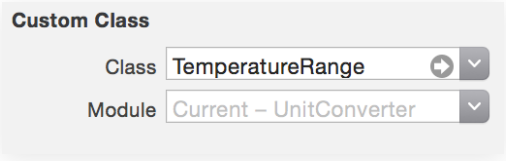
- Discuss the `ViewController` `temperatureValues` property and how the `ViewController` is serving as the `UIPickerViewDataSource`.
- Discuss whether a range of temperature values has anything to do with unit conversion, and whether the `UnitConverter` model should generate a range of temperatures.
- Discuss the concept of a "view model": a model object whose sole purpose is to serve the view.
- Add a new Swift class (⌘N) to the project for a `TemperatureRange` model.

```
import Foundation

class TemperatureRange {

}
```

- Discuss a plan for establishing a `TemperatureRange` object as the picker view's `dataSource`, including the adoption of the `UIPickerViewDataSource` protocol, and changing the picker view's `dataSource` connection from the controller to a new `TemperatureRange` object.
- Using Interface Builder and the Object Library (⌥⌘L), drag an Object to the View Controller Scene in the Document Outline (⧈).
- In the Document Outline (⧈), rename the Object to **TemperatureRange**.



- With the TemperatureRange object selected, use the Identity Inspector (⌥⌘3) to set the Class to `TemperatureRange`.



- Using Interface Builder, select the picker view and use the Connections Inspector (⌥⌘6) to delete the `dataSource` connection between the picker view and the controller.
- Use the Connections Inspector (⌥⌘6) to drag a new connection from the picker view's `dataSource` to the TemperatureRange object in the Document Outline.
- Run the app (⌘R), observe the crash, and inspect the error displayed in the console.
- Discuss how the `TemperatureRange` model should now adopt the `UIPickerViewDataSource` protocol.
- Discuss how the existing controller code might be extracted into the `TemperatureRange` model, and define a plan for completing the change.

Closing

What are the similarities and differences between a model and a view model?

Modifications and Extensions

- Remove the `TemperatureRange` object from the Document Outline, and figure out how to establish the connection between the picker view and a `TemperatureRange` object using code in the controller. Contrast the benefits and drawbacks of both approaches.

Resources

Start Developing iOS Apps Today: Using Design Patterns https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/DesignPatterns.html

Cocoa Core Competencies: Model-View-Controller https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

Cocoa Core Competencies: Model Object https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/ModelObject.html

The Swift Programming Language: Classes and Structures https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/ClassesAndStructures.html

Interface Builder Object and Media Help: Adding a Custom Object https://developer.apple.com/library/ios/recipes/xcode_help-IB_objects_media/Chapters/CustomObject.html

UIKit User Interface Catalog: Picker Views https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/UIPickerView.html