UnitConverter
Lesson 10

Description

Address another usability issue when restarting the app, by saving and restoring the last selected temperature.

Learning Outcomes

- Analyze the user experience of an app to identify flaws in its usability.
- Discover how user preferences in iOS apps are persisted, and describe how property list files represent key-value data stores.
- Apply method extraction to refactor code into well-named methods that carry out a single task.
- Recognize the drawback of buried string literals, and apply constants to prevent literal values from being obfuscated within a growing codebase.

Vocabulary

user experience	usability	life cycle methods
data persistence	`NSUserDefaults`	property list
plist file	key-value data structure	refactor
method extraction	helper method	constant

Materials

- **UnitConverter Lesson 10** Xcode project

- **Saving User Preferences** presentation

Opening

How can we get the picker view to remember the last selected temperature when we restart the app?

Agenda

- Run the app (⌘R), select a temperature, background the app, then foreground the app. Notice how the last selected temperature is still displayed.
- Using the multitasking bar (⇧⌘H twice quickly), quit the app, then start the app.
- Discuss the user experience when restarting the app, how the app "forgets" the last selected temperature, and displays the default temperature in the picker view. Consider how this affects user experience.
- Discuss the desire for the app to "remember" the last selected temperature, and to use that temperature when it starts, if a last-selected temperature is known.
- Discuss the benefits and drawbacks of saving the last-selected temperature with controller life cycle methods, compared with saving the selected temperature when a new temperature is selected.
- Present the concepts of `NSUserDefaults` and property list (**.plist**) files.
- Explore the plist files the Simulator utilizes in **~/Library/Developer/CoreSimulator/Devices/[DEVICE_ID]/data/Containers/Data/Application/[APP_ID]/Library/Preferences.**
- Enhance `pickerView:didSelectRow:inComponent:` to save the picker view's selected row index.

```
let defaults = NSUserDefaults.standardUserDefaults()
defaults.setInteger(row, forKey: "defaultCelsiusPickerRow")
defaults.synchronize()
```

- Run the app (⌘R), select a temperature, and use the OS X Finder's preview feature to explore the **edu.yourschool.Preferences.plist** file, to verify the appropriate value is saved when a new temperature is selected.
- Discuss how the controller method `pickerView:didSelectRow:inComponent:` now has two responsibilities: updating the temperature label and saving the last-selected row.
- Extract the code for each respective task into two separate, well-named controller methods.

```
func displayConvertedTemperatureForRow(row: Int) {
   let degreesCelsius = temperatureRange.values[row]
   temperatureLabel.text =
      "\(converter.degreesFahrenheit(degreesCelsius))°F"
}

func saveSelectedRow(row: Int) {
   let defaults = NSUserDefaults.standardUserDefaults()
   defaults.setInteger(row, forKey: "defaultCelsiusPickerRow")
   defaults.synchronize()
}
```

- Update `pickerView:didSelectRow:inComponent:` to call the two new methods.

```
func pickerView(pickerView: UIPickerView, didSelectRow row: Int,
   inComponent component: Int) {
   displayConvertedTemperatureForRow(row)
   saveSelectedRow(row)
}
```

- Discuss the resulting expressiveness of `pickerView:didSelectRow:inComponent:`, and how the extracted helper methods `displayConvertedTemperatureForRow:` and `saveLastSelectedRow:` each carry out one specific task.
- Discuss how the number of lines of code within the controller are increasing, and discuss best practices for handling string literals such as `"defaultCelsiusPickerRow"`, found in the `saveSelectedRow:` method, which become buried as a codebase grows.
- Extract the buried string into a constant placed near the top of the ViewController class.

```
let userDefaultsLastRowKey = "defaultCelsiusPickerRow"
```

- Refactor `saveSelectedRow:` to use the `userDefaultsLastRowKey` constant.

```
 defaults.setInteger(row, forKey: userDefaultsLastRowKey)
```

- Run the app (⌘R), select a temperature, and explore the **edu.yourschool.Preferences.plist** file to verify the appropriate value is saved. Using the multitasking bar (⇧⌘H twice quickly), quit the app, start the app again, and observe that, despite saving the last selected picker row, the default row is selected.

Closing

Why isn't the temperature picker displaying the last selected row when we start the app?

Modifications and Extensions

- Extract the configuration of the picker view and label in `viewDidLoad` into a well-named method.
- Implement a way to customize the range of temperatures displayed in the picker view, rather than the existing range of `-100` to `100`.

Resources

Preferences and Settings Programming Guide https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/UserDefaults/

NSUserDefaults Class Reference https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSUserDefaults_Class/