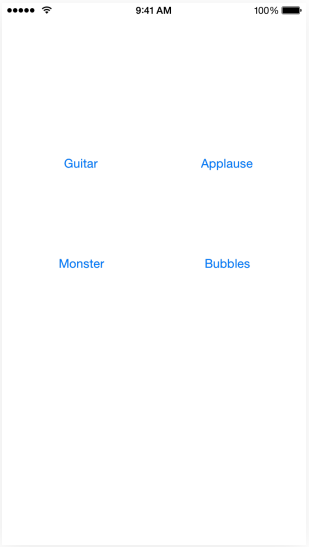
NoiseMaker
Lesson 6

Description

Extract the `AVAudioPlayer` property initialization out of the playback methods and into and initializer.



Learning Outcomes

- Analyze existing code to determine the frequency of unnecessary object instantiation.
- Practice implementing an initializer.
- Distinguish between constants and variables, and between optional and non-optional types.

Vocabulary

instantiation	initialization	initializer
`init`	constant	`let`
optional	forced unwrapping	

Materials

- **NoiseMaker Lesson 6** Xcode project
- **Initialization** presentation

Opening

How many times are `AVAudioPlayer` objects created when we play sounds with our app?

Agenda

- Discuss the existing implementation of the `NoiseMaker` model.
- Discuss how a new `AVAudioPlayer` object is instantiated every time a "play" method is called.
- Discuss how the `AVAudioPlayer` instantiations can be reduced, by creating each `AVAudioPlayer` object once, when a `NoiseMaker` object is created.
- In the `NoiseMaker` class, extract the `AVAudioPlayer` instantiations into a new initializer.

```
init() {
   let url = NSBundle.mainBundle().URLForResource("guitar",
      withExtension: "wav")
   guitarPlayer = AVAudioPlayer(contentsOfURL: url, error: nil)
   let url2 = NSBundle.mainBundle().URLForResource("applause",
      withExtension: "wav")
   applausePlayer = AVAudioPlayer(contentsOfURL: url2, error: nil)
   let url3 = NSBundle.mainBundle().URLForResource("monster",
      withExtension: "wav")
   monsterPlayer = AVAudioPlayer(contentsOfURL: url3, error: nil)
   let url4 = NSBundle.mainBundle().URLForResource("bubbles",
      withExtension: "wav")
   bubblesPlayer = AVAudioPlayer(contentsOfURL: url4, error: nil)
}
```

- Present the concept of initialization.
- Discuss how each `AVAudioPlayer?` property no longer needs to be a variable, nor optional.
- Update the `NoiseMaker` property declarations, changing `var` to `let` and removing the `?` optional symbol.

```
private let guitarPlayer: AVAudioPlayer
...
private let bubblesPlayer: AVAudioPlayer
```

- Update each "play" method such that they only call the `play` method on each respective `AVAudioPlayer` property, and remove the `!` forced unwrapping.

```
func playGuitarSound() {
   guitarPlayer.play()
}

func playApplauseSound() {
   applausePlayer.play()
}

func playMonsterSound() {
   monsterPlayer.play()
}

func playBubblesSound() {
   bubblesPlayer.play()
}
```

- Run the app (⌘R), and tap the buttons to play each sound.
- Discuss how the controller instantiates the `NoiseMaker` model once, and how the `NoiseMaker` model instantiates each of its `AVAudioPlayer` properties only once.
- Discuss how tapping each button no longer instantiates a new `AVAudioPlayer` before playing each sound.

Closing

Repetitive code is often referred to as a "code smell." What repetitive code do you smell? How do you think we can reduce the repetitive code in our model?

Modifications And Extensions

- Implement a custom initializer called `initWithSoundFileNames:` that receives an array of sound file names, and uses the file names in the array to prepare each `AVAudioPlayer` property. Refactor the existing initializer to use `initWithSoundFileNames:` as the designated initializer.
- Observe how the app has four buttons, four controller actions, four model methods, and four `AVAudioPlayer` properties. Investigate how the text property of each button might be used to prepare the `AVAudioPlayer` properties and to cause the respective `AVAudioPlayer` object to play the appropriate sound.

Resources

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

The Swift Programming Language: Initialization https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Initialization.html

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

The Swift Programming Language: Properties https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Properties.html