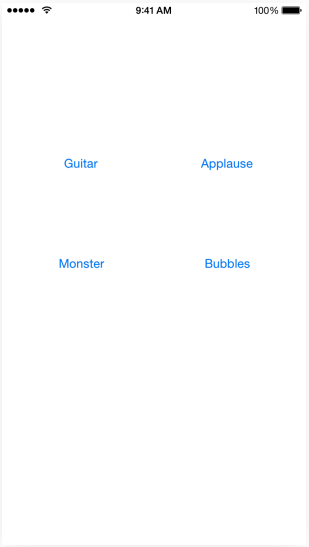
NoiseMaker
Lesson 8

Description

Refactor the four "play" methods in the model into a single `play:` method.



Learning Outcomes

- Recognize similarities and differences in repetitive code, and practice extracting varying values into method parameters.
- Recognize potential flaws in code that can lead to runtime errors.
- Formulate a logical condition as a Boolean expression.

Vocabulary

refactor	method	parameter
argument	array subscripting	`if`

Materials

- **NoiseMaker Lesson 8** Xcode project

Opening

How might we refactor the four "play" methods into one?

Agenda

- Discuss the how the play methods in the `NoiseMaker` model are identical except for the numeric array index.
- Discuss combining the four separate methods into one method that receives an array index as its parameter.
- In the `NoiseMaker` class, delete the four independent "play" methods and implement a single `play:` method.

```
func play(index: Int) {
   players[index].play()
}
```

- Discuss how the `play:` method expects to receive an `Int` as its argument, which is used to access a particular player in the `players` array.
- Discuss how the `NoiseMaker` model implementation is now more concise and does not contain repetitive code.
- Discuss how the implementation of `play:` is vulnerable to a runtime error if it receives an `Int` value outside of the bounds of the array, and discuss the conditions to check to ensure safer array subscripting.
- Update the `play:` method with increased safety.

```
func play(index: Int) {
   if !players.isEmpty && index >= 0 && index < players.count {
      players[index].play()
   }
}
```

- Discuss the changes in the `ViewController` that are necessary, in order to take advantage of the new `NoiseMaker play:` method.
- Update the four controller actions to call the `NoiseMaker play:` method.

```
@IBAction func playGuitar(sender: UIButton) {
   noiseMaker.play(0)
}

@IBAction func playApplause(sender: UIButton) {
   noiseMaker.play(1)
}

@IBAction func playMonster(sender: UIButton) {
   noiseMaker.play(2)
}
```

```
@IBAction func playBubbles(sender: UIButton) {
    noiseMaker.play(3)
}
```

- Run the app (⌘R), tap the buttons, and verify that the functionality remains unchanged.

Closing

What do you think about the repetitive code in the controller? Might there be a way to refactor these four methods into one? Can you think of a way to implement this one method without using a long `if` statement?

Modifications And Extensions

- Add an `else` clause to the `if` statement in the `NoiseMaker play:` method that plays a default sound.
- Extract the condition within the `NoiseMaker play:` method into a well-named `(Int) -> Bool` method to encapsulate the checking of the `index` value.

Resources

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

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

Swift Standard Library Reference: Array https://developer.apple.com/library/ios/documentation/General/Reference/SwiftStandardLibraryReference/Array.html

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