SpaceAdventure
Lesson 5

Description

Add a `start` method implementation to the `SpaceAdventure` class.

```
Welcome to our solar system!
There are 8 planets to explore.
You are currently on Earth, which has a circumference of 24859.82 miles.
What is your name?
Jane
Nice to meet you, Jane. My name is Eliza, I'm an old friend of Siri.
Let's go on an adventure!
Shall I randomly choose a planet for you to visit? (Y or N)
Huh?
Sorry, I didn't get that.
Y
Ok! Traveling to...
```

Learning Outcomes

- Recognize that objects may only respond to methods defined as part of their own interface or parent interface hierarchy.
- Discover the Swift method implementation syntax.
- Analyze how readability and expressiveness may be side effects of proper abstraction of code.

Vocabulary

class	method call	method definition
implement	`func`	parameter list
object		

Materials

- **SpaceAdventure Lesson 5** Xcode project

Opening

How can we get the `SpaceAdventure` object to know how to handle the `start` method call?

Agenda

- Discuss how the `SpaceAdventure` object does not know how to handle the `start` method call.
- Add an empty implementation of the `start` method to the `SpaceAdventure` class.

```
class SpaceAdventure {

   func start() {
   }

}
```

- Explain the method implementation syntax including the `func` keyword, method name, and empty parameter list.
- Return to **main.swift**, and observe how the Xcode error notices disappear.
- Discuss how the object now knows how to handle the `start` method call, but that it would not do much in response, because the method definition is empty.
- Cut and paste the existing code from **main.swift** into the body of the `SpaceAdventure` `start` method implementation.

```
func start() {
   let numberOfPlanets = 8
   let diameterOfEarth = 24859.82 // In miles, from pole to pole.
   print("Welcome to our solar system!")
...
```

- Run the program (⌘**R**), and interact with the console (⇧⌘**C**) to demonstrate that the existing functionality remains intact.
- Discuss how **main.swift** now only creates a `SpaceAdventure` object, and tells the `SpaceAdventure` object to `start`.

```
let adventure = SpaceAdventure()
adventure.start()
```

- Discuss whether or not **main.swift** has become more concise, readable and expressive.

Closing

Although **main.swift** is now about space adventures, take a look at the `start` method implementation. How might we make this code more readable and expressive?

Modifications and Extensions

- Our space adventure will include planetary systems and planets. Determine how one might model these concepts in the program, and how these concepts should be provided to our `SpaceAdventure` object. How might you model the solar system as code?
- Redefine the `SpaceAdventure` class as a Swift structure, and determine if structures can also support method implementations.

Resources

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

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

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