SpaceAdventure
Lesson 7

Description

Introduce a PlanetarySystem class into the project, and use an instance when displaying the introduction.

```
Welcome to the Solar System!
There are 8 planets to explore.
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.
Shall I randomly choose a planet for you to visit? (Y or N)
Y
Ok! Traveling to...
```

Learning Outcomes

- Practice implementing a class and discover the Swift property declaration syntax.
- Describe how initializers assign the initial values to object properties during instantiation.
- Observe property initialization through inline assignment.
- Practice instantiating objects and accessing object properties.

Vocabulary

Project Navigator	class definition	property
constant	`let`	type annotation
instantiation	initializer	parameter
`self`		

Materials

- **SpaceAdventure Lesson 7** Xcode project
- **Initialization** presentation

Opening

How do you think we can model planets and a planetary system for our `SpaceAdventure` to use?

Agenda

- Discuss the need to model a collection of planets, using a `PlanetarySystem` class.
- Add a new Swift file (⌘N) called **PlanetarySystem.swift** to the project.
- Using the Project Navigator (⌘1), select **PlanetarySystem.swift** and implement a basic `PlanetarySystem` class definition.

```
class PlanetarySystem {

}
```

- Add a property declaration to the `PlanetarySystem` class to represent the name of the planetary system.

```
class PlanetarySystem {

   let name: String

}
```

- Explain the property declaration syntax, emphasizing the type annotation and the use of `let` to indicate that the `name` of a `PlanetarySystem` object, once initialized, will not change.
- Discuss the error that Xcode displays, and discuss what value the `name` of a `PlanetarySystem` object would be when instantiated, given the existing class definition.
- Explain how Swift requires that all constant properties be assigned values during instantiation, within the implementation of an initializer.
- Add a parameterized initializer to the `PlanetarySystem` class.

```
init(name: String) {
   self.name = name
}
```

- Present the concepts of initializers and initialization.
- Explain how the `PlanetarySystem` initializer expects a `String`, called `name`, and assigns the value of the `name` parameter to the `name` property, using `self` to disambiguate the two.
- Discuss how the `SpaceAdventure` should consist of a `PlanetarySystem` to travel within, and the need to add a `PlanetarySystem` property to the `SpaceAdventure` class.
- Add the new `PlanetarySystem` property to the `SpaceAdventure` class.

```
class SpaceAdventure {

   let planetarySystem = PlanetarySystem(name: "Solar System")
...
```

- Explain the syntax of instantiating a `PlanetarySystem` by invoking the parameterized initializer, the named parameter syntax, and how assigning a default value to a property may be used instead of using property assignment in an initializer.
- Discuss the existing implementation of the `SpaceAdventure` `start` method.
- Update the implementation of `displayIntroduction`, removing some previous demonstration code, and using the `PlanetarySystem` `name` to display the introductory message.

```
private func displayIntroduction() {
   let numberOfPlanets = 8
   print("Welcome to the \(planetarySystem.name)!")
   print("There are \(numberOfPlanets) planets to explore.")
}
```

- Discuss how `name` is a property of a `PlanetarySystem` object, and how `planetarySystem` is a property of a `SpaceAdventure` object.
- Run the program (⌘R), and observe how the console (⇧⌘C) output reflects the name of the planetary system.

Closing

Instead of an independent `numberOfPlanets` variable, where do you think the number of planets should come from?

Modifications and Extensions

- Replace the inline initialization of the `planetarySystem` property with an initializer that assigns the property its initial value. Investigate whether one approach is better than the other.

• Add a `numberOfPlanets` property and a convenience initializer to the `PlanetarySystem` class.

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

Project Navigator Help: Adding a New File https://developer.apple.com/library/ios/recipes/xcode_help-structure_navigator/articles/Adding_a_New_File.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

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

The Swift Programming Language: The Self Property https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Methods.html#//apple_ref/doc/uid/TP40014097-CH15-ID238