SpaceAdventure
Lesson 14

Description

Decrease the coupling between the `SpaceAdventure` class and the `PlanetarySystem`. Extract the `PlanetarySystem` creation into **main.swift**.

```
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)
N
Name the planet you would like to visit.
Neptune
Traveling to Neptune...
Arrived at Neptune. A very cold planet, furthest from the sun.
```

Learning Outcomes

- Recognize potential design constraints in existing code.
- Practice refactoring, to improve existing code without changing functionality.
- Practice writing property declarations, initializers and instantiating objects.
- Distinguish mutability and immutability with `var` and `let`.

Vocabulary

initializer	property declaration	type annotation
parameter	decoupling	mutability

Materials

- **SpaceAdventure Lesson 14** Xcode project

Opening

When the kind of planetary system changes, why does the code in `SpaceAdventure` have to change? How close of a relationship would you say the `SpaceAdventure` class has with the specific `PlanetarySystem` object it uses?

Agenda

- Discuss how the `SpaceAdventure` initializer includes a significant amount of planet data (names and descriptions) and explicitly relies on creating one kind of `PlanetarySystem`.
- Discuss how one might improve the design of the `SpaceAdventure` class, by providing its initializer with a particular `PlanetarySystem` to explore; and by extracting the planet data from the existing initializer.
- Update the `SpaceAdventure` initializer in three steps. First, extract the existing code within the `SpaceAdventure` initializer, by moving it to the top of **main.swift**.

```
import Foundation

// TODO: Reduce repetitive code
let mercury = Planet(name: "Mercury", description: "A very hot
   planet, closest to the sun.")
...
planetarySystem.planets.append(mercury)
...
let adventure = SpaceAdventure()
adventure.start()
```

- Modify the `SpaceAdventure` `planetarySystem` property declaration by removing the assignment of the default `PlanetarySystem` object.

```
class SpaceAdventure {

   let planetarySystem: PlanetarySystem
   ...
```

- Explain why the `planetarySystem` property now requires a type annotation, as no value is explicitly assigned to it yet.
- Update the `SpaceAdventure` initializer to accept a `PlanetarySystem` parameter, assigning it to the `planetarySystem` property.

```
init(planetarySystem: PlanetarySystem) {
   self.planetarySystem = planetarySystem
}
```

- Explain how the initializer receives a `PlanetarySystem` object, and assigns it to the `planetarySystem` property.
- Update the implementation of **main.swift** to prepare an array of `Planet` objects, create a `PlanetarySystem`, and then pass the `PlanetarySystem` object to the `SpaceAdventure` initializer.

```
// TODO: Reduce repetitive code.
let mercury = Planet(name: "Mercury", description: "A very hot
   planet, closest to the sun.")
...
let systemName = "Solar System"
var planets = [Planet]()

planets.append(mercury)
...
planets.append(neptune)

let solarSystem = PlanetarySystem(name: systemName, planets: planets)
let adventure = SpaceAdventure(planetarySystem: solarSystem)
```

- Run the program (⌘**R**) to verify that the functionality remains the same.
- Discuss how the `SpaceAdventure` class is now decoupled from the "Solar System" `PlanetarySystem`, and how any `PlanetarySystem` can be passed to the `SpaceAdventure` initializer.
- Discuss how the `PlanetarySystem` `planets` array property should no longer be mutable, since a `PlanetarySystem` initializer should be provided a complete `Planet` array during initialization.
- Update the `PlanetarySystem` `planets` property declaration, replacing `var` with `let`.

```
class PlanetarySystem {
   let name: String
   let planets: [Planet]
   ...
```

- Discuss how the `Planet` data no longer remains buried within the `SpaceAdventure` class, and allude to how it may be extracted even further, to exist outside the code entirely.
- Run the program (⌘**R**) to verify that the functionality remains the same.

Closing

What about that `TODO` reminding us to reduce the repetitive code? Is there any kind of pattern you see in the code? How do you think we might be able to improve this?

Modifications and Extensions

- Create multiple `PlanetarySystem` and `SpaceAdventure` objects, and modify the program to allow the user to embark on multiple adventures; or to choose which planetary system to travel to.
- Investigate the concept of dependency injection, and determine how it applies to the new `SpaceAdventure` class.
- Investigate how to store the planetary system name and planet data in an external property list (or "plist") file. Update the program such that the data is loaded externally, rather than existing as literal `String` values within the code.

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

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

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

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

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

The Swift Programming Language: Mutability of Collections https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/CollectionTypes.html#//apple_ref/doc/uid/TP40014097-CH8-ID106