# SpaceAdventure

# Lesson 13

## Description

Implement the ability to travel to a random planet.

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)

Υ

Traveling to Neptune...

Arrived at Neptune. A very cold planet, furthest from the sun.

## **Learning Outcomes**

- Relate program design, runtime safety, and user experience to software quality.
- Discover and apply the Swift Array is Empty property.
- Distinguish a computed property from a stored property, and discover the computed property syntax.
- Recognize how Swift optionals can represent the presence or absence of a value.
- Discover and apply the optional binding syntax with if let.
- Practice writing if statements to control program flow.

# Vocabulary

runtime error	encapsulation	array index
computed property	if statement	optional
nil	optional binding	if let

#### **Materials**

SpaceAdventure Lesson 13 Xcode project

## **Opening**

How can we make our code more robust against runtime errors?

#### Agenda

- Run the program (寒R), enter a name, choose Y, witness the program crash, and observe the "Array index out of range" runtime error displayed in the console (公衆c). Stop the program with a keyboard shortcut (衆.) or the stop button in the Xcode Toolbar.
- Discuss three possible improvements to the program: improving encapsulation, by making the PlanetarySystem class responsible for providing a random planet; improving runtime safety, by preventing the use of an invalid array index; and improving user experience, by only prompting the user to travel to a planet when there is at least one Planet object in the planets array.
- Using the Xcode Documentation and API Reference (企業0), investigate the Swift Array is Empty property.
- Add a randomPlanet computed property to the PlanetarySystem class.

```
class PlanetarySystem {
    ...
    var randomPlanet: Planet? {
        if planets.isEmpty {
            return nil
        } else {
            let index = Int(arc4random_uniform(UInt32(planets.count)))
            return planets[index]
        }
    }
}
```

- Explain the computed property syntax, and how the property returns an optional type whose value, when unwrapped, will either be a Planet object or nil.
- Explain the concept of Swift optionals.
- In the SpaceAdventure determineDestination method, replace the existing naive implementation in the first branch of the if statement with an implementation that uses optional binding.

```
if decision == "Y" {
   if let planet = planetarySystem.randomPlanet {
      visit(planet.name)
   } else {
      println("Sorry, but there are no planets in this system.")
   }
} else if decision == "N" {
```

- Explain the mechanics of optional binding with if let.
- Run the program ( $\Re R$ ), enter a name, choose Y, and observe the console ( $\triangle \Re C$ ) output stating that "there are no planets in this system."
- Discuss how the user experience of the program can be improved by only prompting for a planet to visit when the planets array is not empty.
- Update the implementation of start, to check for a non-empty planets array before prompting for the adventure and calling determineDestination.

```
func start() {
    displayIntroduction()
    greetAdventurer()
    if (!planetarySystem.planets.isEmpty) {
        println("Let's go on an adventure!")
        determineDestination()
    }
}
```

- Run the program ( $\Re R$ ), enter a name, and observe the console ( $\triangle \Re C$ ) to see that the user is not prompted for a destination.
- In the SpaceAdventure initializer, restore the addition of each Planet object to the planets array by uncommenting (\(\mathbb{H}/\)) the relevant lines of code.

```
init() {
    ...
    planetarySystem.planets.append(mercury)
    ...
    planetarySystem.planets.append(neptune)
}
```

• Run the program(%R) multiple times, visit a random planet again, and observe the different planets visited.

## Closing

Our codebase has grown, and the SpaceAdventure initializer has a bit of a "code smell." Although it works ok, can you think of a ways we can improve the initializer?

#### **Modifications And Extensions**

- Within the randomPlanet property definition, combine the determination of the index and the subscripting of the planets array into one long statement. Make a decision about the readability of each approach, and determine which approach you feel is better.
- Replace the randomPlanet computed property with a method definition that returns an optional. Determine which implementation you feel is better, and explain why.
- Replace the if let conditional binding with a forced unwrapping of the randomPlanet property. Try running the program with both an empty and non-empty array of planets. Explain why you see an error in one case, and the difference between forced unwrapping and conditional binding.

#### Resources

The Swift Programming Language: About Swift https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/

The Swift Programming Language: A Swift Tour https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/GuidedTour.html

The Swift Programming Language: The Basics https://developer.apple.com/library/prerelease/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: Computed Properties https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/Properties.html#//apple\_ref/doc/uid/TP40014097-CH14-ID259

The Swift Programming Language: Conditional Statements https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/ControlFlow.html#//apple\_ref/doc/uid/TP40014097-CH9-ID127

The Swift Programming Language: Subscripts https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/Subscripts.html

The Swift Programming Language: Optionals https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/TheBasics.html#//apple\_ref/doc/uid/TP40014097-CH5-ID330

BSD Library Functions Manual: ARC4RANDOM(3) https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man3/arc4random\_uniform.3.html

The Swift Programming Language: Integers https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/TheBasics.html#//apple\_ref/doc/uid/TP40014097-CH5-ID317

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