GENERAL ASSEMBLY

# INTERMEDIATE SWIFT

**Wellington Moreno**
Lead iOS Instructor, General Assembly

LESSON

## LEARNING OBJECTIVES

+ **Describe & Use** the Singleton Pattern
+ **Create & Use** enums
+ **Create & Use** protocols and delegates

## APOLITICO

LESSON

## APOLITICO

is a Political News Subscription service. Subscribe to the latest news tailored to your political preference. From Democrat to Anarchist, we have something for everyone.

LESSON

OUR OVERALL GOAL IS TO MODEL A SYSTEM THAT REPRESENTS APOLITICO.

LESSON

## FEATURES

+ Subscribe to a particular party
+ Unsubscribe from the service
+ Get the latest stories
+ Receive updates for new stories

## PLAYGROUNDS: CREATE THE CLASS

CODE

**5 mins**

### DIRECTIONS

Create the basic Apolitico class with the following features:

```
+  initializers()
+  subscribe()
+  unsubscribe()
+  getStories()
+  subscribers: [String]
```

LESSON

## SINGLETONS

LESSON

## SINGLETON

A Singleton is a class or struct that can ever only have **one instance**.

DEMO

## FOLLOW-ALONG

Creating a Singleton

## PLAYGROUNDS: MAKE APOLITICO A SINGLETON

CODE

5 mins

### DIRECTIONS

Update the Apolitico class to make it a Singleton

LESSON

## ENUMS

LESSON

## ENUM

An enumeration defines a **common type for a group of related values**, enabling you to work in a type-safe way within your code.

## ENUMS

```
enum Engine {
  case Gas
  case Diesel
  case Hybrid
  case Electric
}
```

## ENUMS

```
let engine = Engine.Hybrid
```

## ENUMS

```
let engine: Engine = .Hybrid
```

## PRACTICE: CREATE THE ENUMS

CODE

5 mins

### DIRECTIONS

Create an enum for the Political Party

+ Republican
+ Democrat
+ Socialist
+ Anarchist
+ *Other*

LESSON

## PROTOCOLS & DELEGATES

LESSON

## PROTOCOL

A Protocol is not a class, but it represents **the idea of a class.**

The idea can then be implemented by a class, structure, or enumeration.

## PROTOCOLS

```
protocol Bike {
  func pedal()
  func brake()
  func turn(direction: Direction)
}
```

## PROTOCOLS

```
class RoadBike: Bike {
  func pedal() {
    print(“I can go pretty fast”)
  }
}
```

# PROTOCOLS

```
class MountainBike: Bike {
  func pedal() {
    print(“Work hard to move me.”)
  }
}
```

## PROTOCOLS

```
var bike: Bike = RoadBike()
bike.pedal()

Bike = MountainBike()
bike.pedal()
```

LESSON

## DELEGATE

A Delegate is a **listener** that receives notifications when interesting things happens

Delegates are created as a protocol.

DEMO

## FOLLOW-ALONG

Creating a Delegate protocol

## PRACTICE: COMPLETE THE LOOK

CODE

15 mins

### DIRECTIONS

Use Delegates and Protocols to complete the App:

+ Update the Subscribe method to take a delegate
+ Update the Unsubscribe method to take a delegate
+ Implement the Delegate protocol with a class that prints the news articles
+ ***Bonus: Implement a second Delegate that replaces curse words that prints the most popular word in each news article***

CONCLUSION

## TAKEAWAYS

## CONCLUSION

Enums are useful to represent things that have a known predefined set of options

## CONCLUSION

Protocols allow you to separate an idea from the details

## CONCLUSION

Delegates allow you to create clean code designs

LESSON

## Q & A

## THANKS!

## WELLINGTON MORENO

+ GitHub/Slack: **@SirWellington**
+ Twitter: **@SirWellingtonZ**
+ Email: **wellington.moreno@ga.co**

## FOR INSTRUCTOR PURPOSES ONLY

## MATERIALS

+ Lab
+ Lesson Plan