

MOBILE DEVELOPMENT OOP, GESTURES, AND ANIMATION

Tedi Konda

Executive Director, Technology, Unison

OBJECT ORIENTED PROGRAMMING

INHERITANCE DEMO: ANIMALS

OBJECT ORIENTED PROGRAMMING

INITIALIZATION

- ▶ Classes have variables which must equal a value at the time the object is initiated
 - ▶ **Important!** This means that **every** instance variable must either be optional or be assigned a default value during initialization
- ▶ Classes can specify custom initializers that take parameters

OBJECT ORIENTED PROGRAMMING

INITIALIZATION DEMO - ANIMAL & UIVIEWS

OBJECT ORIENTED PROGRAMMING

OO CONCEPTS

▸ Polymorphism

- A method that takes a class (e.g. Animal) can also accept any of its subclasses
- Example:
 - Animal is a class, and Dog is a subclass of Animal
 - `walkAnimal(animal: Animal) {}` is a function that walks an animal
 - Because `walkAnimal()` can accept an Animal or a Dog, because dog **is an** Animal

OBJECT ORIENTED PROGRAMMING

PEER PROGRAMMING - A GAME

- ▶ Work in a playground
- ▶ One person should make three classes, 'Player', 'GoodPlayer' and 'BadPlayer'
 - ▶ Player has an 'attack' method, which returns a tuple (message: String, damage: Int). Message is the message that the player says during the attack, and damage is the amount of damage it does
 - ▶ Both good players and bad players have some (≥ 2) possible attacks. Good and bad players have different possible attacks, they are performed randomly when attack is called
 - ▶ Players also have a health integer (default to 100), and an isAlive method (a player is alive if their health is above 0)
- ▶ The other person creates a 'Match' class, which takes two players during initialization
 - ▶ It also has a 'playGame()' method, which pits each player against each other, alternating taking turns until one of the players is no longer alive. At the end of the match, print out the winner
- ▶ Put one GoodPlayer against a BadPlayer, look at the printed results!
- ▶ Bonus: Give players names, print those out before they match

OBJECT ORIENTED PROGRAMMING

STRUCTS AND PROTOCOLS

OBJECT ORIENTED PROGRAMMING

STRUCTS

OBJECT ORIENTED PROGRAMMING

WHAT IS A STRUCT?

- ▶ A struct is, like a class, a logical grouping of state and methods that encapsulate an entity
 - ▶ e.g. a rectangle, an integer, an array
- ▶ A struct has variables and methods
- ▶ There can be many instances of structs, each of which has methods and state

OBJECT ORIENTED PROGRAMMING

**WHAT'S THE DIFFERENCE BETWEEN
A STRUCT AND A CLASS?**

OBJECT ORIENTED PROGRAMMING

WHAT'S THE DIFFERENCE BETWEEN A STRUCT AND A CLASS?

- ▶ Instances of a struct are **values**, which are copied as they are passed around
- ▶ Instances of a class are **references**, which are not copied as they're passed around

OBJECT ORIENTED PROGRAMMING

VALUE & REFERENCE - DEMO

OBJECT ORIENTED PROGRAMMING

PROTOCOLS

OBJECT ORIENTED PROGRAMMING

PROTOCOLS

- ▶ A group of methods that a class has, encapsulated into its own entity
- ▶ Methods can be required or optional
- ▶ Classes can 'meet' as many interfaces as they'd like

OBJECT ORIENTED PROGRAMMING

DEMO - PROTOCOLS

OBJECT ORIENTED PROGRAMMING

PAIR PROGRAMMING

- ▶ Create an app that has a view controller with a table view
- ▶ Make that view controller both the delegate and data source for the table view
- ▶ Create an array of ten Players (good or bad) when the view comes into view. Print out those players in the table view.