A Swift Kickstart

Daniel H Steinberg
@dimsumthinking

Copyright Dim Sum Thinking, Inc 2014

Classes

Create a Class

```
class CocoaConfAttendee {

}

var attendee = CocoaConfAttendee()
```

Subclass from UIKit

```
import UIKit

class CocoaConfAttendee: UIButton {
}

let attendee = CocoaConfAttendee()
```

Base Class

```
class CocoaConfAttendee {
}

let attendee = CocoaConfAttendee()
```

Properties

```
class CocoaConfAttendee {
    let name = "Daniel"
    let hometown = "Cleveland"
}

let attendee = CocoaConfAttendee()
```

Properties

```
class CocoaConfAttendee {
    let name = "Daniel"
    let hometown = "Cleveland"
}

let attendee = CocoaConfAttendee()
attendee.name
attendee.hometown
```

init

```
class CocoaConfAttendee {
    let name, hometown: String
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
}

let daniel = CocoaConfAttendee(name: "Daniel",
                               hometown: "Cleveland")
daniel.name
daniel.hometown
```

convenience init

```
class CocoaConfAttendee {
    let name, hometown: String
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    convenience init(name: String) {
        self.init(name: name, hometown: "Cleveland")
    }
}

let daniel = CocoaConfAttendee(name: "Daniel")
daniel.name
daniel.hometown

let kim = CocoaConfAttendee(name: "Kim",
                            hometown: "Shaker Heights")
kim.name
kim.hometown
```

convenience init

```
class CocoaConfAttendee {
    let name, hometown: String
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    convenience init(name: String) {
        self.init(name: name, hometown: "Cleveland")
    }
}

let daniel = CocoaConfAttendee(name: "Daniel")
daniel.name
daniel.hometown

let kim = CocoaConfAttendee(name: "Kim",
                            hometown: "Shaker Heights")
kim.name
kim.hometown
```

Methods (and self)

```
class CocoaConfAttendee {
    let name, hometown: String
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    convenience init(name: String) {
        self.init(name: name, hometown: "Cleveland")
    }
    func nameBadge() -> String {
        return "Hello, I'm \(name) from \(hometown)"
    }
}

let daniel = CocoaConfAttendee(name: "Daniel")
daniel.nameBadge()
```

Optional hometown

```
class CocoaConfAttendee {
    let name: String
    let hometown: String?
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    init(name: String) {
        self.name = name
    }
    func nameBadge() -> String {
        return "Hello, I'm \(name) from \(hometown)"
    }
}
```

if let

```
class CocoaConfAttendee {
    let name: String
    let hometown: String?
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    init(name: String) {
        self.name = name
    }
    func nameBadge() -> String {
        if let validHometown = hometown {
            return "Hello, I'm \(name) from \(hometown)"
        } else {
            return "Hello, I'm \(name) from here and there"
        }
    }
}
```

nil coalescing operator

```
class CocoaConfAttendee {
    let name: String
    let hometown: String?
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    init(name: String) {
        self.name = name
    }
    func nameBadge() -> String {
        let validHometown = hometown ?? "here and there"
        return "Hello, I'm \(name) from \(validHometown)"

    }
}
```

Computed Property

```
class CocoaConfAttendee {
    let name: String
    let hometown: String?
    var isFromSomewhere:Bool {
        get {
            return hometown != nil
        }
    }
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    init(name: String) {
        self.name = name
    }
    func nameBadge() -> String {
        let validHometown = hometown ?? "here and there"
        return "Hello, I'm \(name) from \(validHometown)"

    }
}
```

Computed Property

```
class CocoaConfAttendee {
    let name: String
    let hometown: String?
    var isFromSomewhere:Bool {
        return hometown != nil
    }
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    init(name: String) {
        self.name = name
    }
    func nameBadge() -> String {
        let validHometown = hometown ?? "here and there"
        return "Hello, I'm \(name) from \(validHometown)"

    }
}
```

Computed Property

```
class CocoaConfAttendee {
    let name: String
    let hometown: String?
    var isFromSomewhere:Bool {
        return hometown != nil
    }
    init(name: String, hometown: String) {
        self.name = name
        self.hometown = hometown
    }
    init(name: String) {
        self.name = name
    }
    func nameBadge() -> String {
        var validHometown: String
        if isFromSomewhere {
            validHometown = " from \(hometown!)"
        } else {
            validHometown = ""
        }
        return "Hello, I'm \(name)"  + validHometown

    }
}
```

Subclass

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    let tutorial: String
}
```

Subclass init

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    let tutorial: String
    init(name: String, hometown: String, tutorial: String) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
}
```

Override

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    let tutorial: String
    init(name: String, hometown: String, tutorial: String) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial)"
    }
}

let anabelle = CocoaConfTutorialAttendee(name: "Anabelle",
                    hometown: "Detroit", tutorial: "AV")
anabelle.nameBadge()
```

Can't change tutorial

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    let tutorial: String
    init(name: String, hometown: String, tutorial: String) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial)"
    }
}

let anabelle = CocoaConfTutorialAttendee(name: "Anabelle",
                    hometown: "Detroit", tutorial: "AV")
anabelle.nameBadge()
anabelle.tutorial = "Swift"
```

let => var

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    var tutorial: String
    init(name: String, hometown: String, tutorial: String) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial)"
    }
}

let anabelle = CocoaConfTutorialAttendee(name: "Anabelle",
                    hometown: "Detroit", tutorial: "AV")
anabelle.nameBadge()
anabelle.tutorial = "Swift"
anabelle.nameBadge()
```

Parameter Names

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    var tutorial: String
    init(name: String, hometown: String, tutorial: String) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial)"
    }
    func justForShow(name: String, hometown: String,
                     tutorial: String) {
        // doesn't do anything
    }
}

let anabelle = CocoaConfTutorialAttendee(name: "Anabelle",
hometown: "Detroit", tutorial: "AV")
anabelle.justForShow("some name", hometown: "some town",
                                  tutorial: "some tutorial")
```

Force

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    var tutorial: String
    init(name: String, hometown: String, tutorial: String) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial)"
    }
    func justForShow(#name: String, hometown: String,
                     tutorial: String) {
        // doesn't do anything
    }
}

let anabelle = CocoaConfTutorialAttendee(name: "Anabelle",
hometown: "Detroit", tutorial: "AV")
anabelle.justForShow(name: "some name", hometown: "some town",
                     tutorial: "some tutorial")
```

Suppress

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    var tutorial: String
    init(name: String, hometown: String, tutorial: String) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial)"
    }
    func justForShow(name: String, _ hometown: String,
                     _ tutorial: String) {
        // doesn't do anything
    }
}

let anabelle = CocoaConfTutorialAttendee(name: "Anabelle",
hometown: "Detroit", tutorial: "AV")
anabelle.justForShow("some name", "some town", "some tutorial")
```

Why?

```
class CocoaConfTutorialAttendee: CocoaConfAttendee {
    var tutorial: String
    init(name: String, hometown: String, tutorial: String) {
        self.tutorial = tutorial
        super.init(name: name, hometown: hometown)
    }
    override func nameBadge() -> String {
        return super.nameBadge() + ", I'm taking \(tutorial)"
    }
    func justForShow(name: String, hometown: String,
                     tutorial: String) {
        // doesn't do anything
    }
}

let anabelle = CocoaConfTutorialAttendee(name: "Anabelle",
hometown: "Detroit", tutorial: "AV")
anabelle.justForShow("some name", hometown: "some town",
                                  tutorial: "some tutorial")
```

Try this

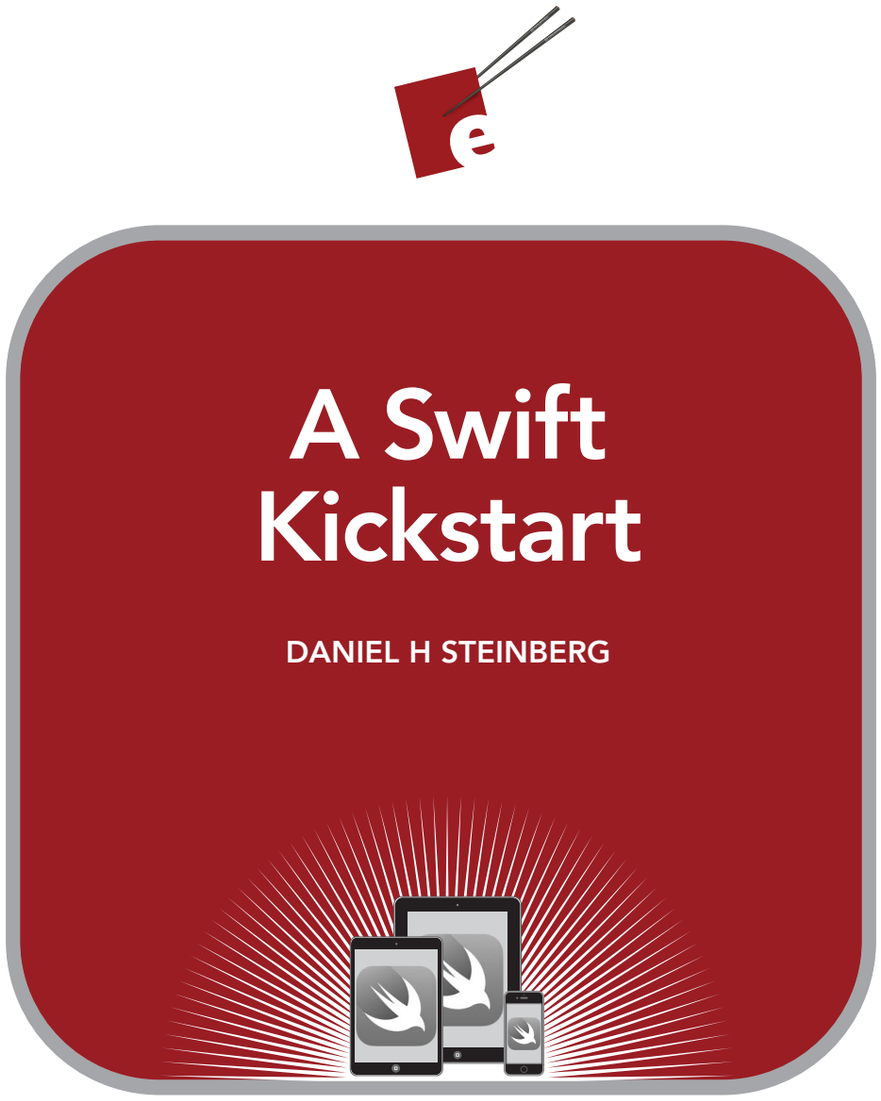
- Create a class named Beverage that contains a property of type Int named amount and a computed property of type Bool named isEmpty

- Create a subclass of Beverage named HotBeverage that includes a method named sip() that reduces the amount by one and prints a message when the Beverage is empty

- Create an instance of HotBeverage and test that it works.

Try this

```
class Beverage {
    var amount: Int
    var isEmpty: Bool {
        return amount <= 0
    }
    init (amount: Int) {
        self.amount = amount
    }
}
class HotBeverage: Beverage {
    func sip() {
        amount--
        if isEmpty {
            println("You're done")
        }
    }
}
let coffee = HotBeverage(amount: 8)
for i in 1...8 {
    coffee.sip()
}
```



https://itunes.apple.com/us/book/a-swift-kickstart/id891801923?mt=11&uo=4&at=11l56E