Chapter 10/11

Classes

Classes

- class instances are **reference types**
- **Computed Properties**
  - value derived based on some form of calculation or logic at which the property is stored/retrieved
- **Protocols**
  - a set of rules (**standard**) that define the minimum requirements which as class must meet
  - defines the properties/methods that a class must contain in order to be in conformance
  - ***Purpose***:
    - Polymorphism
    - Extensible Code

```
protocol Announceable {
  var name: String { get set }
  var speakerLabel: String { get }

  func announce(message: String)
}

class Update: Announceable {
    // Implementation of required properties
    var name: String
    var speakerLabel: String {
        return "[\(name)]"
    }

    init(name: String) {
        self.name = name
```

```
    }

    // Implementation of required method
    func announce(message: String) {
        print("\(speakerLabel) \(message)")
    }
}

var road = Update(name: "Road update")
road.announce(message: "There is construction at Yorba Linda Ave.")

// Polymorphism
func say(something: Announceable) {
    something.announce(message: "Heyo")
}
say(something: road)
```

```
extension Double {
    var squared: Double {
        return self * self
    }
}

extension String {
    func emojify(with emoji: String) -> String {
        return "\(emoji) \(self) \(emoji)"
    }
}

class BankAccount {
    var name: String
    var balance: Float
    var fees: Float?

    var actualBalance: Float? { // get-only computed property
        if let fees = fees {
              return balance - fees
        }
        return nil
    }

    init(name: String, balance: Float = 0) {
        self.name = name
        self.balance = balance
        self.fees = nil
    }

    deinit {
        print("Deleted \(name)'s Account")
    }

    func displayBalance(censor: (String) -> String) {
        print("\(censor(name))'s current balance = $\(balance)")
        print("Fees = \(fees ?? 0)")
```

```
    }

    class func maxBalance() -> Float {
        return 10000.0
    }
}

class SavingsAccount: BankAccount {
    var interestRate: Float = 0.0

    init(name: String, balance: Float, rate: Float) {
        interestRate = rate
        super.init(name: name, balance: balance)
    }

    override func displayBalance(censor: (String) -> String) {
        super.displayBalance(censor: censor)
        let hidden_name = censor(name)
        print("\(hidden_name)'s interest rate = $\(interestRate)")

    }

    func calculateInerest() -> Float {
        return interestRate * balance
    }
}

print("Max-Balance =", BankAccount.maxBalance())

var person1: BankAccount = BankAccount(name: "Jason", balance: 200.5)
var person2: BankAccount = BankAccount(name: "Joanne")
person1.fees = 100

print("\(person1.name).actualBalance = \(person1.actualBalance ?? 0.0)")
print("\(person2.name).actualBalance = \(person2.actualBalance ?? 0.0)")

person1.displayBalance(censor: {(value) -> String in String(repeating: "*", count: value.count)})
person2.displayBalance(censor: {(value) -> String in String(repeating: "*", count: value.count)})

var savings1 = SavingsAccount(name: "Jason", balance: 600.00, rate: 0.07)

print("\(savings1.name) interest rate = \(savings1.interestRate)")
savings1.displayBalance(censor: {(value) -> String in String(repeating: "*", count: value.count)})

print("3.0.squared =", 3.0.squared)
```