

```
Adopting Multiple Protocols

class Vehical: Movable, Goods {
  var speed: Int = 0
  var price: Float
  var model: String
  required init(model: String, price: Float) {
    self.model = model
    self.price = price
  }
  func move() {
    print("Moving")
  }
}

DEPAULUNIVERSITY
```

```
Adopted Protocols in Subclasses

• Protocol requirements apply to the subclasses as well.

• Implementation of property and method requirements can be inherited

• Initializer requirements must be implemented by each

class Car: Vehical {
    required init(model: String, price: Float) {
        super.init(model: model, price: price)
        speed = 35
    }
    override func move() {
        print("Driving")
    }
}
```

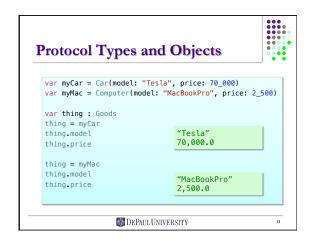
```
Super Types vs. Super Classes

class Computer: Goods {
  var price: Float
  var model: String
  required init(model: String, price: Float) {
    self.model = model
    self.price = price
  }
}

• Both Vehicle and Computer are root classes.

• They don't share a common superclass. No code reuse.

• They adopt a common protocol, i.e., share a common super type.
```



Using Protocols for Delegation Protocols define the interface of the work to be done Delegates are the objects that do the work by implementing the methods declared in the protocols Using protocols, a class can effectively delegate the implementation and work to other classes Widely used in Cocoa Touch A work-around of single inheritance.

```
The Clock Protocol

protocol Clock {
    var hour: Int { get }
    var minute: Int { get }
}

The get property requirements can be satisfied by
Stored properties, or
Constants, or
Computed properties
```

```
An Implementation of
the Clock Protocol

class MyClock: Clock {
  var hour: Int {
    let calendar = NSCalendar.currentCalendar()
    let components = calendar.components(
        .CalendarUnitHour, fromDate: NSDate())
    return components.hour
  }
  var minute: Int {
    let calendar = NSCalendar.currentCalendar()
    let components = calendar.components(
        .CalendarUnitMinute, fromDate: NSDate())
    return components.minute
  }
}

DEPAULUNIVERSITY
```

```
An Clock Implementation with

Delegation

class AppleWatch: Computer, Clock {
    var clock = MyClock()

    required init(model: String, price: Float) {
        super.init(model: model, price: price)
    }

    var hour : Int {
        return clock.hour
    }

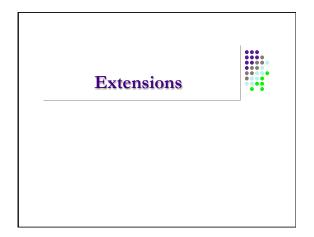
    var minute : Int {
        return clock.minute
    }
}

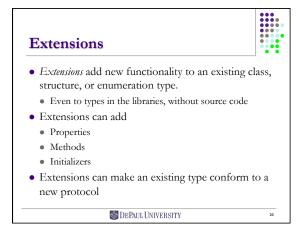
DEPAULUNIVERSITY
```

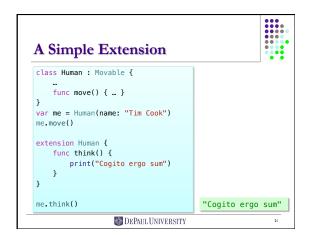
```
An Clock Implementation with Delegation

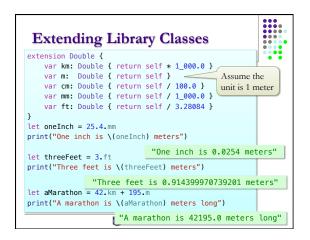
var clock = MyClock()
clock.hour
clock minute

var myWatch = AppleWatch(model: "Gold", price: 17_000)
myWatch.hour
myWatch.minute
```









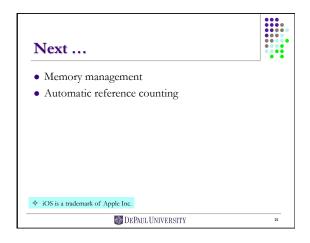
```
Extending Library Classes

extension Int {
    mutating func square() {
        self = self * self
    }
}

var n = 9
n.square()
n

B1
```

```
Extending Library Classes
extension Int {
    func repetitions(task: () -> ()) {
        for _ in 0...self {
    task()
                                         A closure
                                              6 times
5.repetitions({
                                              "Hello!"
    print("Hello!")
                                              "Hello!"
})
5.repetitions {
                                              "Hello!"
                            Trailing closure
    print("Hello!")
                                              "Hello!"
                  DEPAUL UNIVERSITY
```



© Xiaoping Jia, 2016 5