Software Paradigms, Architectures & Patterns
Did We Get It All Wrong?

Jon Reid   @qcoding
Object Oriented Programming
Is OOP Dead?

Alan Kay
Smalltalk-71

Dan Ingalls
Adele Goldberg
Smalltalk-80
Is OOP Dead?

I’m sorry that I long ago coined the term “objects” for this topic because it gets many people to focus on the lesser idea.

The big idea is “messaging”.
Is OOP Dead?

Brad Cox
Tom Love
Objective-C

Beginnings: 1981
Commercialized: 1986
Is OOP Dead?

2016 - 1986 = 30
2016 - 1971 = 45
3-Tier Architecture

- User Interface Layer
- Domain Logic Layer
- Data Access Layer
- Your Cloud Services
Data Access Layer

Domain Logic Layer

User Interface Layer

User

View

Controller

Model

Your Cloud Services

Your Cloud Services
iOS architecture, where MVC stands for Massive View Controller
Apple Sample Code

Copyright (C) 2013 Apple Inc. All Rights Reserved.

*/

#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>

@interface ForwardViewController : UITableViewController

@end

Violates “Single Responsibility Principle”
Single Responsibility Principle
@interface ShareItemViewController: UITableViewController
    <UITableViewDelegate,
    UITableViewDataSource,
    UITextViewDelegate,
    ViewControllerDelegate,
    MFMailComposeViewControllerDelegate,
    TwitterRequestDelegate,
    BitlyRequestDelegate>
“If you can think of more than one motive for changing a class, then that class has more than one responsibility.”

• Mail body changes
• Tweet content changes
• Switch to different URL shortener
Routing
Routing
Routing

Settings → General → About
Dependency Inversion Principle
Presenting Coordinators

NSSpain 2015
Model

View

Presenter
Dependency Inversion Principle

Model

View

Presenter

View Protocol

weak

Controller

Views
Invicta Watch 14541...

US $63.99
US $795.00
91% Off

VIEW ITEM
Invicta Watch 0071...

US $94.99
US $595.00
SOLD OUT

VIEW ITEM
Invicta Watch 14541...

US $63.99  
US $795.00  
91% Off

VIEW ITEM

SOLD OUT

@end
#import "EBayItem.h"

Goal: Eliminate knowledge of model

@implementation SaleItemGridCellView

- (void)setItem:(EBayItem *)item
{
    [super setItem:item];
    if (item.savingsRate > 0)
    {
        self.strikeThroughPriceLabel.text = item.listPrice.stringValue;
        self.strikeThroughPriceLabel.hidden = NO;
    }
    else
    {
        self.strikeThroughPriceLabel.hidden = YES;
    }
}
#import "EBayItem.h"

@implementation SaleItemGridView

- (void)setItem:(EBayItem *)item
{
    [super setItem:item];
    if (item.savingsRate > 0)
    {
        self.strikeThroughPriceLabel.text = item.listPrice.stringValue;
        self.strikeThroughPriceLabel.hidden = NO;
    }
    else
    {
        self.strikeThroughPriceLabel.hidden = YES;
    }
}

- (void)setItem:(EBayItem *)item
{
    [super setItem:item];
    [self.presenter presentStrikeThroughPriceForItem:item];
}
#import "EBayItem.h"

@implementation SaleItemGridCellView

- (void)setItem:(EBayItem *)item
{
    [super setItem:item];
    if (item.savingsRate > 0)
    {
        self.strikeThroughPriceLabel.text = item.listPrice.stringValue;
        self.strikeThroughPriceLabel.hidden = NO;
    }
    else
    {
        self.strikeThroughPriceLabel.hidden = YES;
    }
}

Over in the Presenter...

- (void)presentStrikeThroughPriceForItem:(EBayItem *)item
{
    if (item.savingsRate > 0)
        [self.view showStrikeThroughPrice:item.listPrice.stringValue];
    else
        [self.view hideStrikeThroughPrice];
}
View Controller

- (void)showStrikeThroughPrice:(NSString *)text {
    self.strikeThroughPriceLabel.text = text;
    self.strikeThroughPriceLabel.hidden = NO;
}

- (void)hideStrikeThroughPrice {
    self.strikeThroughPriceLabel.hidden = YES;
}

“Tell, Don’t Ask”
Break Free from MVC

- Extract responsibilities to non-MVC classes
- Extract navigation flow to Presenting Coordinators (see “Application Controller” in *Patterns of Enterprise Application Architecture*)
- Selectively implement MVP or alternatives
- Everything can be done with refactoring
The Model-View-Controller (MVC) design pattern assigns objects in an application one of three roles: model, view, or controller." — Apple
MVC Is Not an Application Architecture

It’s a GUI Architecture

Jon Reid   @qcoding
User

User Interface Layer

Domain Logic Layer

Data Access Layer

Your Cloud Services

Controller

View

Model

User action

Update

Notify

Update
Here lies Buried
the Body of M-

of her Age.
“Favor objection composition over class inheritance.”
Dependency Inversion Principle

My Model → Persistent Model → My Parse Model

Parse Object
Ports & Adapters
Domain Logic

User Interface

Data Access
Data Access

Domain Logic

User Interface

Fake Data Access
Is OOP Dead?
Not If We Apply Its Principles

Jon Reid   @qcoding
Program to Interfaces, Not Implementations
Inheritance vs. Composition
SOLID Principles
Tell, Don’t Ask
Law of Demeter
Command/Query Separation
Clean Architecture
4 Rules of Simple Design
Cohesion & Coupling
Vital across All Programming Paradigms

Jon Reid  @qcoding
Slides & notes:
qualitycoding.org/mce3