SWIFT BEST PRACTICES

COORDINATORS – DIFFICULTY: HARD

INTRODUCTION

The coordinator pattern was introduced to iOS by Soroush Khanlou as a way of simplifying navigation in our apps. Rather than having our view controllers know about, create, configure, and present other view controllers, they instead communicate using a separate object called a coordinator.

So, view controller A tells its coordinator “show the buy screen”, and the coordinator is responsible for deciding what that means. It might take into account A/B test group, whether the user is on an iPad or an iPhone, or other factors.

THE PROBLEM

It's common to see this sort of code in view controllers:

```
navigationController?.pushViewController(someOtherVC, animated: true)
```

That shows a view controller telling its parent what to do, and also having to create, configure, and display an entirely different view controller. This creates hard coupling between our view controllers, forcing a specific app flow.

THE SOLUTION

We can start by defining a coordinator class like this:

```
class MainCoordinator {
    var navigationController: UINavigationController

    init(navigationController: UINavigationController) {
        self.navigationController = navigationController
    }

    func start() {
        let storyboard = UIStoryboard(name: "Main", bundle:
Bundle.main)
        let vc = storyboard.instantiateViewController(withIdentifier:
"SomeIdentifier") as! SomeViewController
        vc.coordinator = self
        navigationController.pushViewController(vc, animated: true)
    }
}
```

We can then assign that coordinator as a property to any view controller that needs to control app flow, like this:

```
weak var coordinator: MainCoordinator?
```

Larger projects are likely to want to use a protocol rather than a concrete type.

We can now make our view controllers call methods on that coordinator as appropriate, such as **coordinator?.show(project)**.

There is one slight wrinkle, which is that storyboards are designed to bootstrap the application, when really we want our main coordinator to do that. To fix this, you need to:

1. Remove the navigation controller from Main.storyboard, making sure there is no initial view controller set.
2. Edit the project settings so that the Main Interface property is empty – it says "Main" by default, which was made it use Main.storyboard.
3. Add this property to the **AppDelegate** class: **var coordinator: MainCoordinator?** – that will store our root coordinator.
4. Finally, add this code to the didFinishLaunchingWithOptions method in the app delegate:

```
let navController = UINavigationController()
coordinator = MainCoordinator(navigationController: navController)
coordinator?.start()

window = UIWindow(frame: UIScreen.main.bounds)
window?.rootViewController = navController
window?.makeKeyAndVisible()
```

Those changes put our coordinator at the center of our app launch, rather than letting storyboards take over.

THE CHALLENGE

Can you upgrade our app to use coordinators? This means following the instructions above so that app launch happens through a main coordinator, then looking for any places where navigation happens and moving that code to the coordinator instead.

For example, when **didSelectRowAt** is triggered in **ViewController**, the only code you should have will find whichever project the user selected, then pass that to a method on its coordinator.