Delegate Pattern in Swift

Schemat delegate pattern

PROTOKÓŁ

metody które muszą być w każdym obiekcie który dziedziczy

KLASA „PODSTAWOWA”

-atrybuty delegata

-Metody funkcji, korzystające z metod delegata

Delegat – Klasa która dziedziczy po protokole

-wszystkie metody narzucone przez protokół

- Jeżeli klasa podstawowa zażąda, delegat ma jej dać to na tacy

Analogia do polityki zagranicznej

Prezydent rządzi krajem, ale nie może być w kilku miejscach jednocześnie. Jak chce coś załatwić za granicą, musi tam mieć swojego delegata który wszystko za niego zrobi. Delegaci ci noszą zazwyczaj tytuł ambasadora i mają załatwić to czego głowa państwa nie może zrobić osobiście. No ale żeby się za bardzo nie rozhasali, obowiązuje ich protokół dyplomatyczny, który określa ich zachowania i możliwości. Protokół pisze się raz na ruski rok, albo nawet kilkadziesiąt ruskich lat – każdy go zna i jest go świadomy jak wygląda. Każdy prezydent ma odpowiednie zadania dla ambasadora – nawet jeżeli jest wakat, to wiadomo jakie to zadania będą bo je określa protokół. Więc jak mianuje ambasadora (tworzy obiekt) to wymaga żeby wykonywał pewne założenia (metody z protokołu).

Protokół

```
4
5  protocol DiplomaticProtocol{
6      func GreetAmbassador() -> String
7      func DeclareWar() -> String
8      func LendMoney() -> Double
9  }
```

Klasa

```
10  class President{
11      var ambassador : DiplomaticProtocol?
12      func sayHello(){
13          if let amb = ambassador{
14              print(amb.GreetAmbassador())
15          } else {
16              print("Ambassador not present")
17          }
18      }
19
20      func startWar(){
21          if let amb = ambassador{
22              print(amb.DeclareWar())
23          } else {
24              print("We have no army bruh")
25          }
26      }
27
28      func lendMoney(){
29          if let amb = ambassador{
30              print("We'll lend you \(amb.LendMoney()) dollars")
31          } else {
32              print("We're broke")
33          }
34      }
35  }
```

Delegat

```
37 class Ambassador : [redacted], DiplomaticProtocol {
38     internal func GreetAmbassador() -> String{
39         return "Siema"
40     }
41     internal func DeclareWar() -> String{
42         return "Wojna"
43     }
44     internal func LendMoney() -> Double{
45         return 14562.89
46     }
47 }
48
49
50 var ambass : Ambassador!
51 var pres : President!
52 pres.ambassador = ambass
53 pres.sayHello()
```

UITableView

Protokół (schowany w definicji)

```
555     open var hasActiveDrag: Bool { get }
556
557
558     // YES if table view is currently tracking a drop session.
559     @available(iOS 11.0, *)
560     open var hasActiveDrop: Bool { get }
561 }
562
563 extension UITableView : UISpringLoadedInteractionSupporting {
564 }
565
566 //_____________________________________________________________________________________________________
567 // this protocol represents the data model object. as such, it supplies no information about appearance (including the cells)
568
569 public protocol UITableViewDataSource : NSObjectProtocol {
570
571
572     @available(iOS 2.0, *)
573     public func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
574
575
576     // Row display. Implementers should *always* try to reuse cells by setting each cell's reuseIdentifier and querying for available reusable cells with dequeueReusableCellWithIdentifier:
577     // Cell gets various attributes set automatically based on table (separators) and data source (accessory views, editing controls)
578
579     @available(iOS 2.0, *)
580     public func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
581
582
583     @available(iOS 2.0, *)
584     optional public func numberOfSections(in tableView: UITableView) -> Int // Default is 1 if not implemented
585
586
587     @available(iOS 2.0, *)
588     optional public func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? // fixed font style. use custom view (UILabel) if you want something different
589
590     @available(iOS 2.0, *)
591     optional public func tableView(_ tableView: UITableView, titleForFooterInSection section: Int) -> String?
592
593
594     // Editing
595
596     // Individual rows can opt out of having the -editing property set for them. If not implemented, all rows are assumed to be editable.
597     @available(iOS 2.0, *)
598     optional public func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Bool
599
600
```

Delegat (View Controller)

```
55  class ViewController : UIViewController, UITableViewDataSource{
56      @IBOutlet weak var tableView: UITableView! //Outlet table view którego wstawiamy w storyboardzie
57
58      override func viewDidLoad() {
59          super.viewDidLoad()
60
61          tableView.dataSource = self
62      }
63
64      //metody narzucone przez protokół UITableViewDataSource
65      func tableView(_ tableView : UITableView, numberOfRowsInSection section: Int) -> Int{
66          return 1
67      }
68      func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
69          let cell = UITableViewCell(style: .default, reuseIdentifier: "cell")
70          cell.textLabel?.text = "Siema"
71          return cell
72      }
73  }
74
```