Lec 3

2018/2/12

Cocoa Design Pattern

- MVC
- Target – Action
- Delegate
- (Notification) <- Anti_Pattern

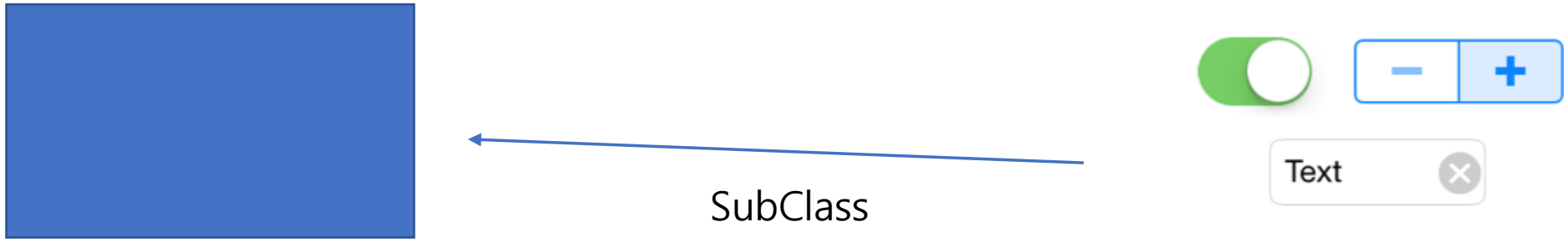
MVC

- https://github.com/duliodenis/cs193p-Winter-2017/blob/master/slides/Lecture-2-Slides.pdf
- ViewController: Controller (+ Model) (Not View)
- Model: Data/Logical Manager
- View: View (UIView Subclass) (@IBOutlet Connection)

- Key: ViewController는 단일 화면의 논리적 단위.

Target-action

- Control 객체는 사전에 지정된 Target과 Action쌍에 대해서 매칭되는 사용자의 이벤트가 발생하면 전달한다.



UIView

(ViewLogic + User Interaction)

UIControl

구체화된 (ViewLogic + User Interaction)

Delegate(Datasource)

- TableView (reclyerView)
- ScrollView
- Etc …

- View Class에 로직을 분리하기 위한 도구
- 뷰 디스플레이 로직을 Query 하거나, 여러 종류의 Event를 전달하기 위해 사용 (No Subclass)

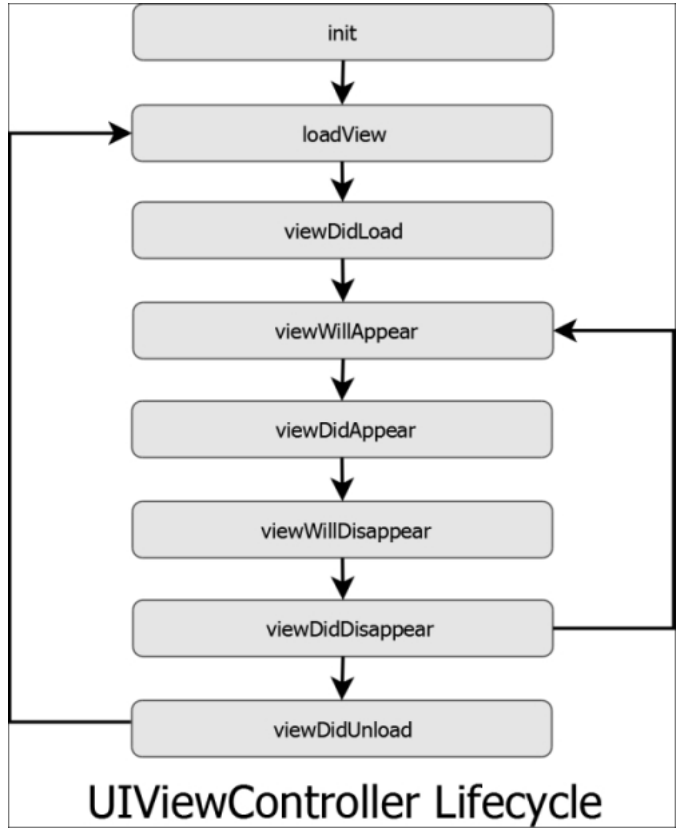
ViewController Life Cycle

- ViewDidLoad 초기화 이후 (Xib/Code initialize)
- 일반적으로 StoryBoard/Xib 로 초기화함
- [중요]StoryBoard/Xib <- Runtime Connection

```
class MyClass {     Class 'MyClass' has no initializers
    var aButton: UIButton
}

class MyClass {
    var aButton: UIButton = UIButton()
}

class MyClass {
    var aButton: UIButton!
}
```



VC 초기화 Mapping 순서

- File’s Owner/Custom Class: (XML 초기화 책임을 가지는 객체)
- StoryBoard 지정
- StoryBoard Identifier 지정
- 해당 뷰의 Custom Class/Filer owner 확인
- 해당 클래스의 init(WithCoder:) 메소드 호출 이후 IBOutlet/IBAction 자동 연결
- 초기화/연결이 완료되면 ViewDidLoad 호출

IBOutlet Mapping 과정

- Storyboard와 owner 객체를 연결하면,
- Outlet name <- 을 키로 연결이 생성됨.
- Storyboard에 하나/ owner 객체에 하나

```
<connections>
    <outlet property="aButton" destination="R9r-4x-1Xz" id="9qh-Dc-
        XUx"/>
</connections>
```

```
@IBOutlet weak var aButton: UIButton!
```

- 주의! runtime crash!!!!
  - Outlet name이 다르거나
  - Owner 객체가 다르거나
  - Outlet class가 다르면

Autolayout

- 이전기술: AutoResizing (springs and struts)
  - Super-Sub View 관계에서만 정의된다.

- 어떤 계층 관계끼리도 정의됨.
- 다양한 요소에 대해서 정의됨(너비/ 간격/ 비율 등)
- Linear하게만 정의됨. (Y = aX + b, a: multiplier, b: constant)
- 이산 시간안에 해를 보장함.

- Key:
  - View는 하나의 축에 대해서 두가지가 정해져야함.
  - 전체 Layout은 하나의 축에대해서 한가지만 열려 있어야함.

Auto layout: (Not View, But layout)