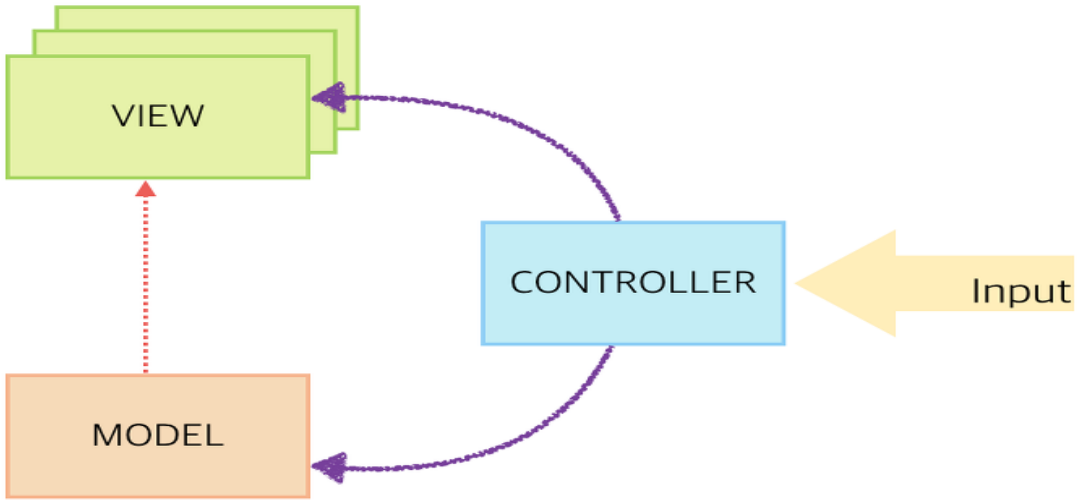
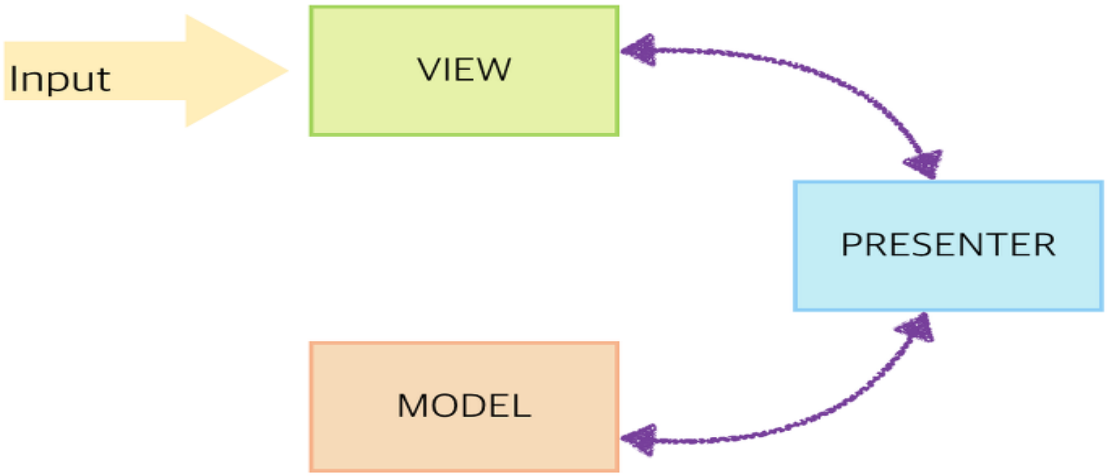
-**MVC** : (Model – View – Controller)

MVC 란 Model ,View, Controller 로 이루어진 구조 입니다.
프로그램을 구성하는 요소들을 Model, View, Controller
세 가지 로 나누어 각각 역할을 부여하고, 각자의 역할(동작) 에만 충실하도록 설계 함으로써 ,결합도(Coupling)을 줄이고 유지보수를 좀 더편하게 만드는 것이 목표인 아키텍쳐 패턴입니다.
그래서 View 는 Controller 가 하는일을 절대로 알수가 없습니다.



-**MVP** :(Model - View - Presenter)

MVC 에서 파생된 아키텍쳐 패턴으로 MVC 에서 Controller 를 Presenter 로 변경하여 프로그램을 구성하는 요소들을 Model, View, Presenter 세 가지 로 나누었다. MVC 에서 발생한 Model 과 View 간의 의존도를 없애기 위해 등장 MVC 모델에서 Controller 가 사라지고 Presenter 가 등장했다.
Presenter 는 Model 과 View 의 상호 작용을 담당한다.
Client 에서 들어오는 입력은 View 가 받아서 처리하며 ,이벤트가 들어왔을 때 View 는 이벤트를 Presenter 에 전달한다. Presenter 는 이벤트를 바탕으로 Model 을 조작하고, 그 결과를 다시 View 에 바인딩하면, View 의 각 요소들이 업데이트 되는 구조이다.



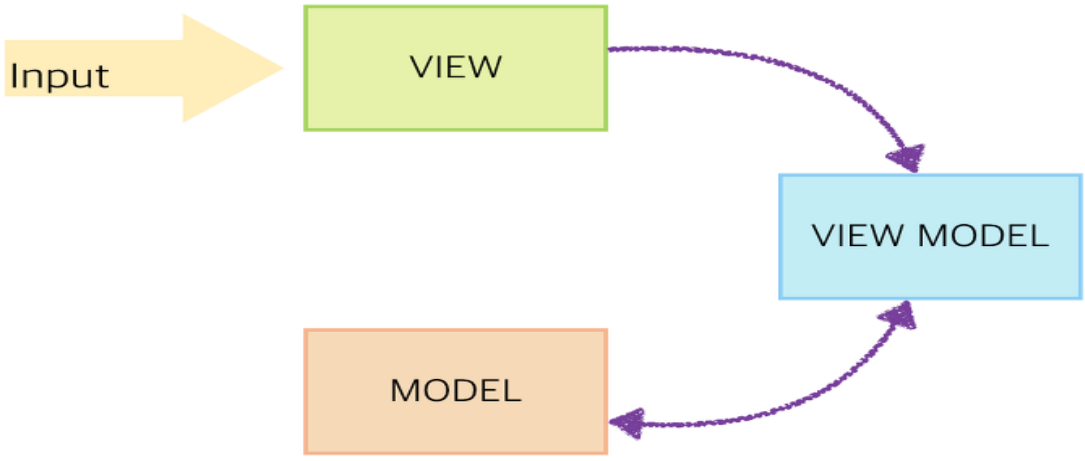
-**MVVM** :(Model - View - ViewModel)

마찬가지로 MVC 모델에서 파생되었습니다.Model 과 View 사이의 의존성 뿐 만 아니라 View 와 Controller 간의 의존성도 고려하여 ,각 Layer 가 완전히 독립적으로 작성되고 테스트 될 수 있도록 설계된 아키텍쳐 패턴

MVP 와 매우 비슷하지만,MVP 에서 Presenter 가 View 와 의존 관계에 있었다면 ViewModel 은 View 와 독립적으로 이루어져 있다는 것이 큰 차이점 이다.

MVVM 에서는 MVP 와 마찬가지로 View 가 입력을 받아 처리하고

ViewModel 은 Presentation logic 을 처리하여 View 에 데이터를 전달하는 역할을 담당한다.



-내가 가장 이상적인 패턴 ?
제가 가장 쓰기 이상적인 것은 MVP 가 저에게 가장 잘 부합할꺼 같습니다.
MVC 는 단순히 Model 를 조작하기만 할뿐이지만 MVP 는 Presenter 가 중간역활을 해서 model 과 view 간의 업데이트를 실시간으로 해줌으로써 좀더 view 에 보이는 업데이트시점이 빨라지고 Model 에 있는 비지니스 로직을 빠르게 실행해서 View 에 보이게 하는 것입니다.

실제로 애플이 사용하는 MVC 패턴 은 ??
현재 애플이 사용하고 있는 MVC 는 델리게이션 , 데이터소스 등이 있습니다.
델리게이션이란 여러개의 작업 수행을 다른 객체에 위임 , 여러 메소드를 델리게이션 함으로써 좀더 개발자에게 좀 더 다양한 방법을 하게 해줍니다.
현재 objective-c 에서 델리게이션이 정말 중요하고 없어서는 안될 패턴이라 생각이 듭니다. 사용자가 여러 문법을 통해 사용할 것을 단 한 메소드를 통해 사용을 할 수 있겠금 해주기 때문에 너무나 편한 거 같습니다 !