View

3D Model

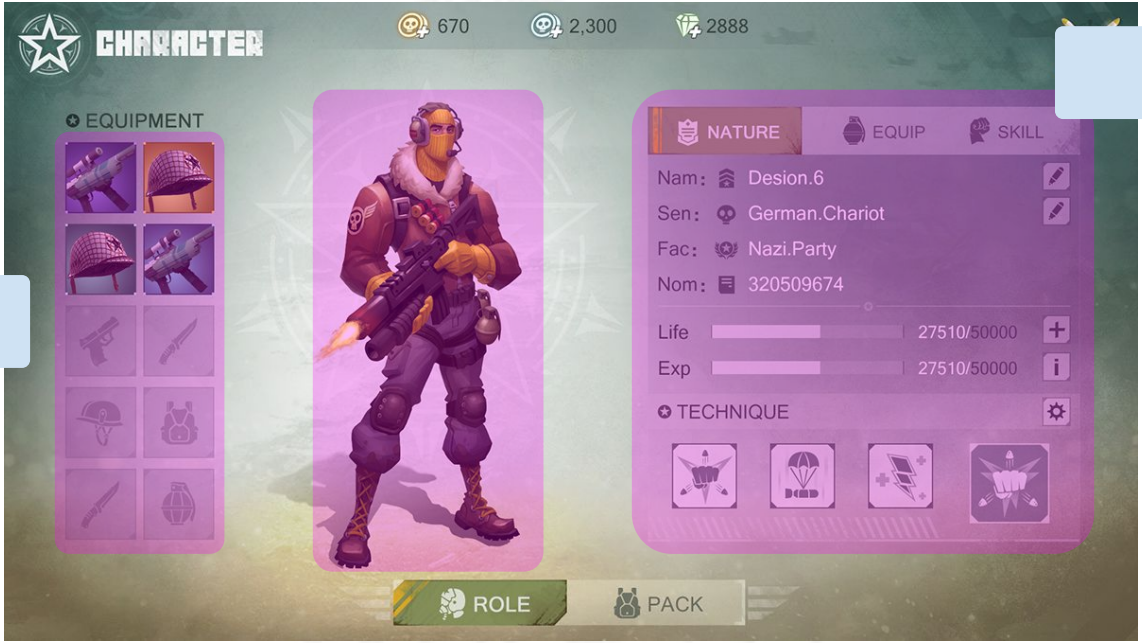
3D Animation

Button

Icon

ScrollView

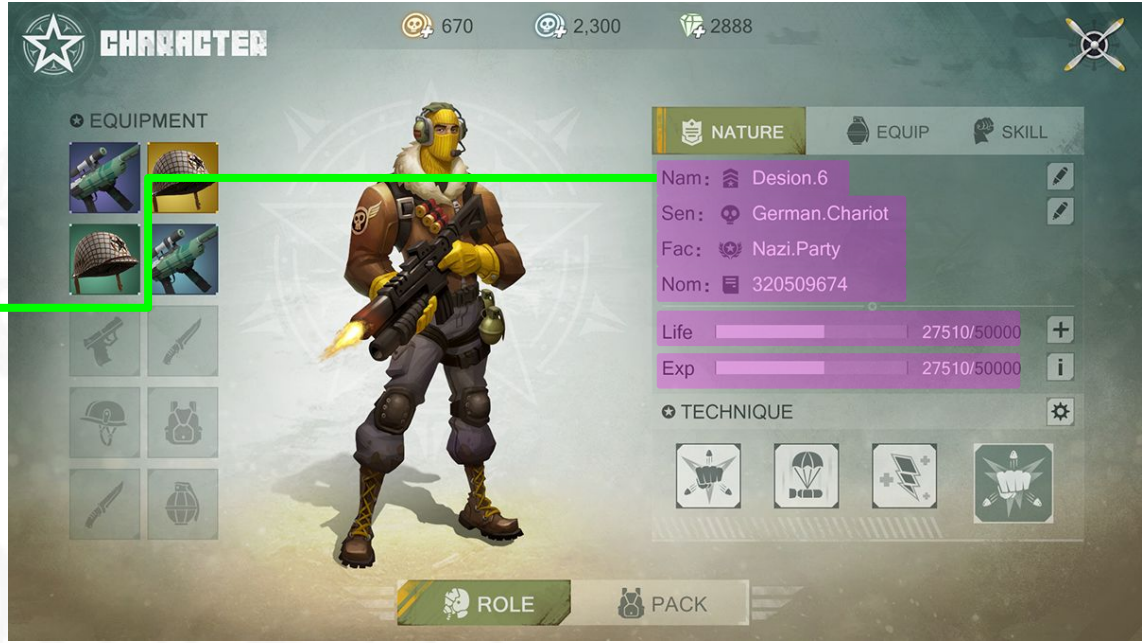
Image



- [사용자]에게 보여지는 부분 (UI, 비주얼 요소들 전부)

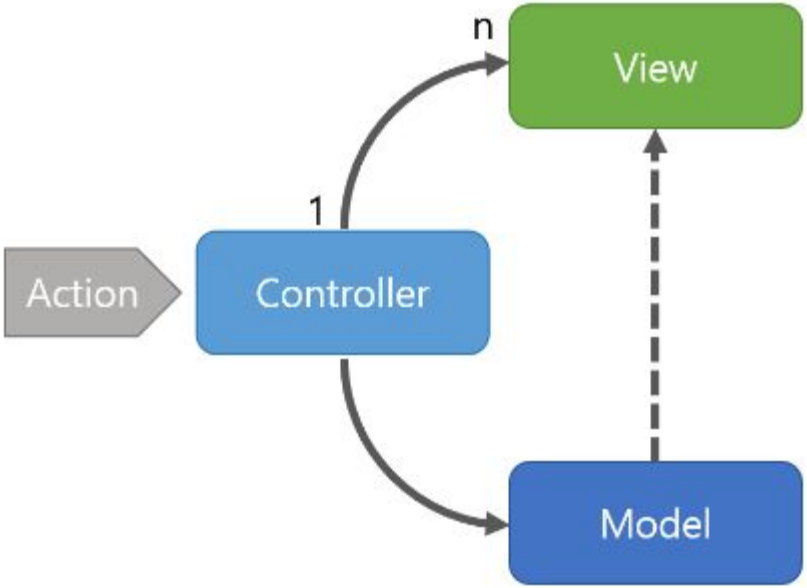
Model

Character
- Name
- Sen
- Fac
- Nom
- Life
- Exp



- 사용되는 [데이터]와 [데이터]를 처리하는 부분

MVC



[동작 순서]

- [Action] -> [Controller] 들어옴
- 액션 처리 후 [Controller] -> [Model] 업데이트
- [Controller] -> [View] 선택
- [View] -> [Model]을 이용해 출력

Controller

Equip()

- [사용자]의 [입력]을 받고 처리하는 부분

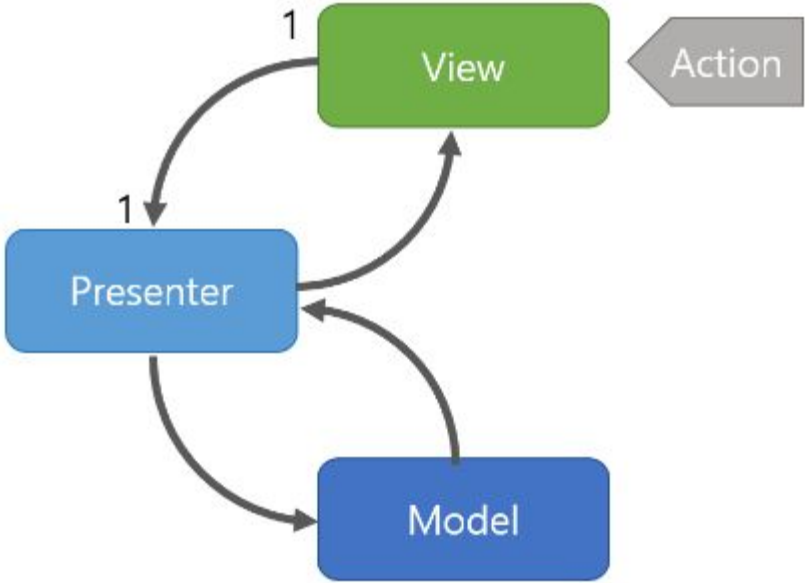
MVC 특징

- 1:n 구조 - Controller는 여러개의 View를 [선택]할 수 있음
- Controller는 View를 [선택]할 뿐 직접 업데이트 하지 않음 (View는 컨트롤러를 알지 못함)

- 장점
  - 널리 사용되며, 가장 단순함

- 단점
  - View와 Model 사이의 의존성이 높음
  - 의존성이 높다 = 규모가 커질수록 유지보수가 어려워짐

MVP



[동작 순서]

- [Action] -> 뷰를 통해 들어옴
- [View] -> [Presenter] 데이터 요청
- [Presenter] -> [Model] 데이터 요청
- [Model] -> [Presenter] 데이터 응답
- [Presenter] -> [View] 데이터 응답
- [View]는 응답받은 데이터를 화면에 출력

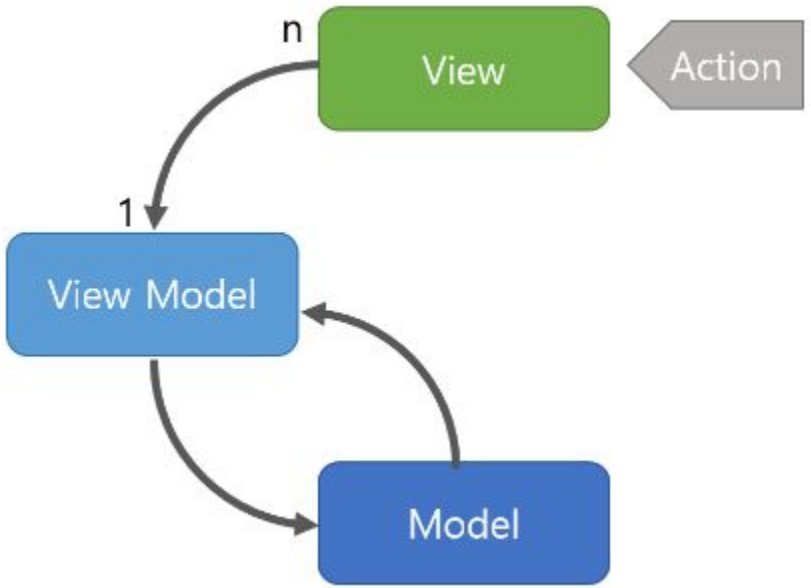
MVP 특징

- **Presenter는 [View]와 [Model]의 인스턴스를 가지고 있어, 둘을 연결**
- **Presenter와 View는 1:1의 관계**

- 장점
  - **View와 Model이 의존성이 없음 (Presenter을 통해서만 데이터 전달)**

- 단점
  - 대신 **View와 Presenter의 의존성이 높아짐**

MVVM



[ViewModel]

- **View**를 표현하기 위해 만든
- **View**를 위한 **Model**
- **View**를 나타내주기 위한 **Model**이자 **View**를 위한 데이터 처리를 진행

[동작순서]

- [Action] -> [View] 들어옴
- **Command** 패턴으로 [View] -> [ViewModel]에 액션 전달
- **[ViewModel]**은 **[Model]**에게 데이터 요청
- **[Model]**은 **[ViewModel]**에게 데이터 응답
- **[ViewModel]**은 응답 데이터를 가공 및 저장
- **[View]**는 바인딩된 뷰모델의 데이터를 출력

MVVM 특징

- **[Command] 패턴과 [Data Binding] 두가지 패턴이 사용되어 구현**
- **View와 ViewModel 사이의 의존성이 없음**
- **ViewModel과 View는 1:n이 가능함**

- 장점
  - 각 구간이 독립적이어 모듈화가 쉬움 (의존성 약하기 때문)

- 단점
  - 설계가 쉽지 않음

Command 패턴이란?

- 실행될 기능을 [캡슐화]
- [행위] 패턴중의 하나
- 기능 실행을 요구하는 [호출자]와 기능을 실제로 수행하는 [수신자]로 나눠짐

- **C#**의 **Extension** 메서드 문법을 사용해 조금 더 용이하게 기능의 캡슐화를 할 수 있음

C# Extension 문법

- [기존 클래스]에 새로운 메서드를 추가하는 것처럼 사용할 수 있게 해주는 기능

- 정적 클래스와 정정 메서드여야 함 **(static class & static method)**
- 첫번째 매개변수로 **this** 키워드를 사용 **(확장 메소드의 호출을 요구하는 객체의 타입)**
- 기존 타입의 변경 없이 기능을 추가 / 제거 가능
- **Namespace** 권장 **(확장 메서드 사용범위 제한을 위해)**
- 정적 메소드로써 호출 가능

MVVM

N : 1

뷰(Main UI)

뷰(캐릭터 UI)

뷰(캐릭터 HUD)

뷰모델

모델

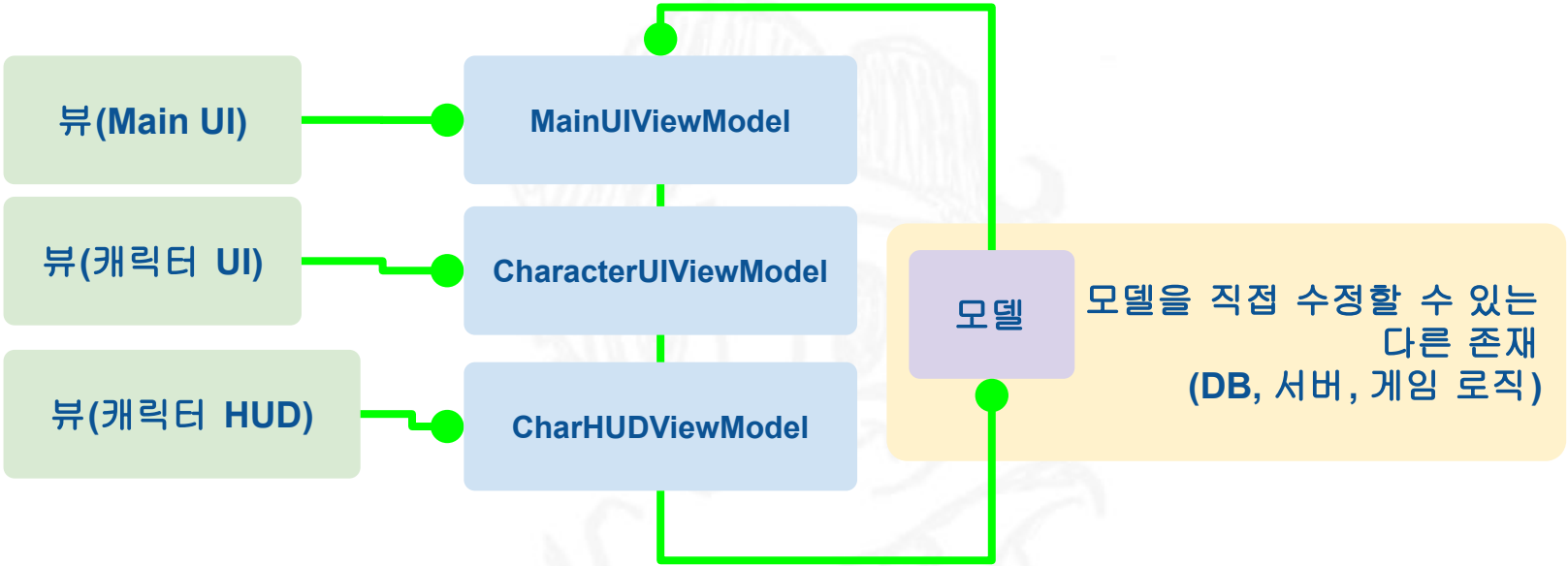
모델을 직접 수정할 수 있는
다른 존재
(DB, 서버, 게임 로직)

UserLevel

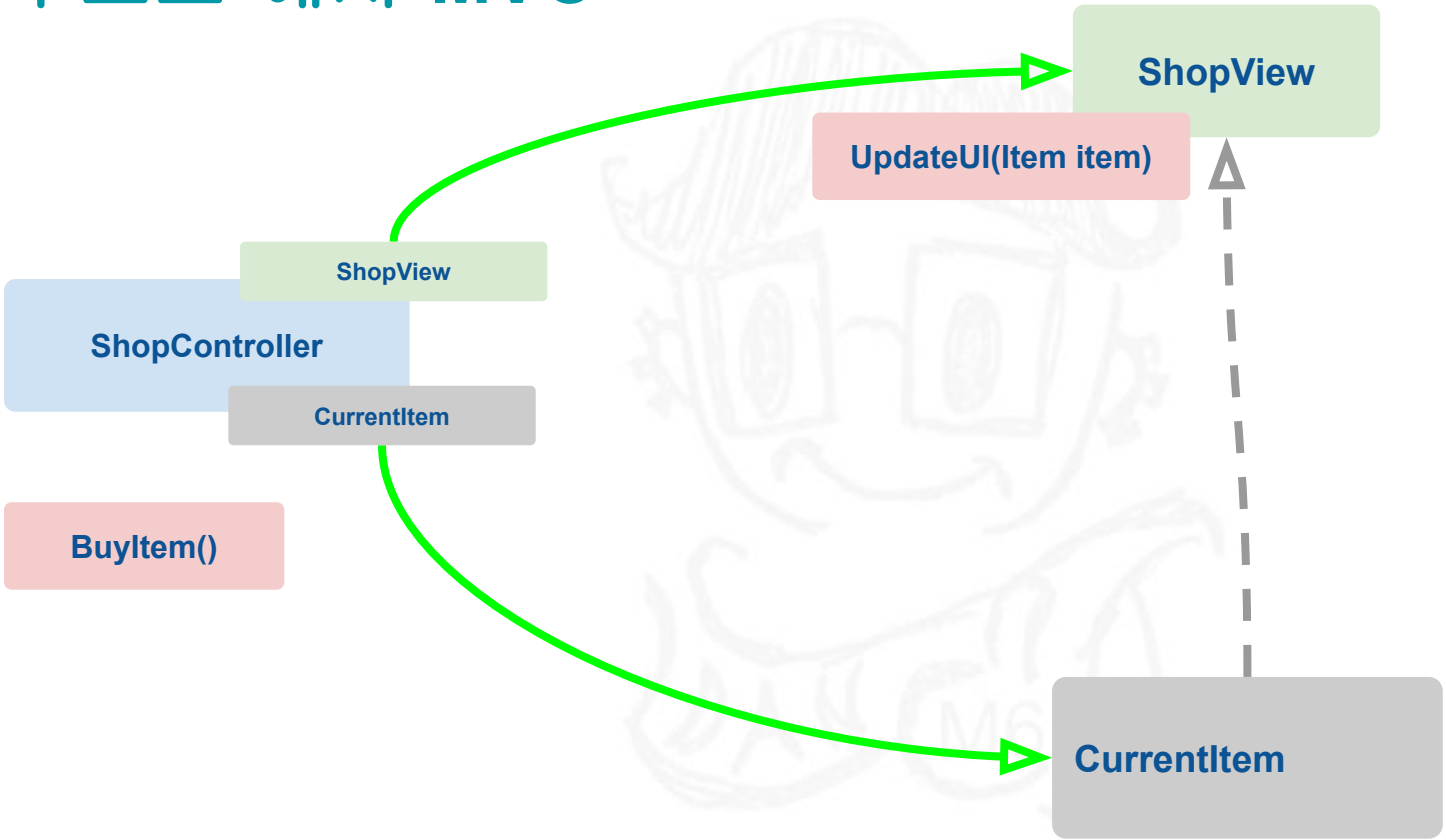
UserLevel

게임 로직

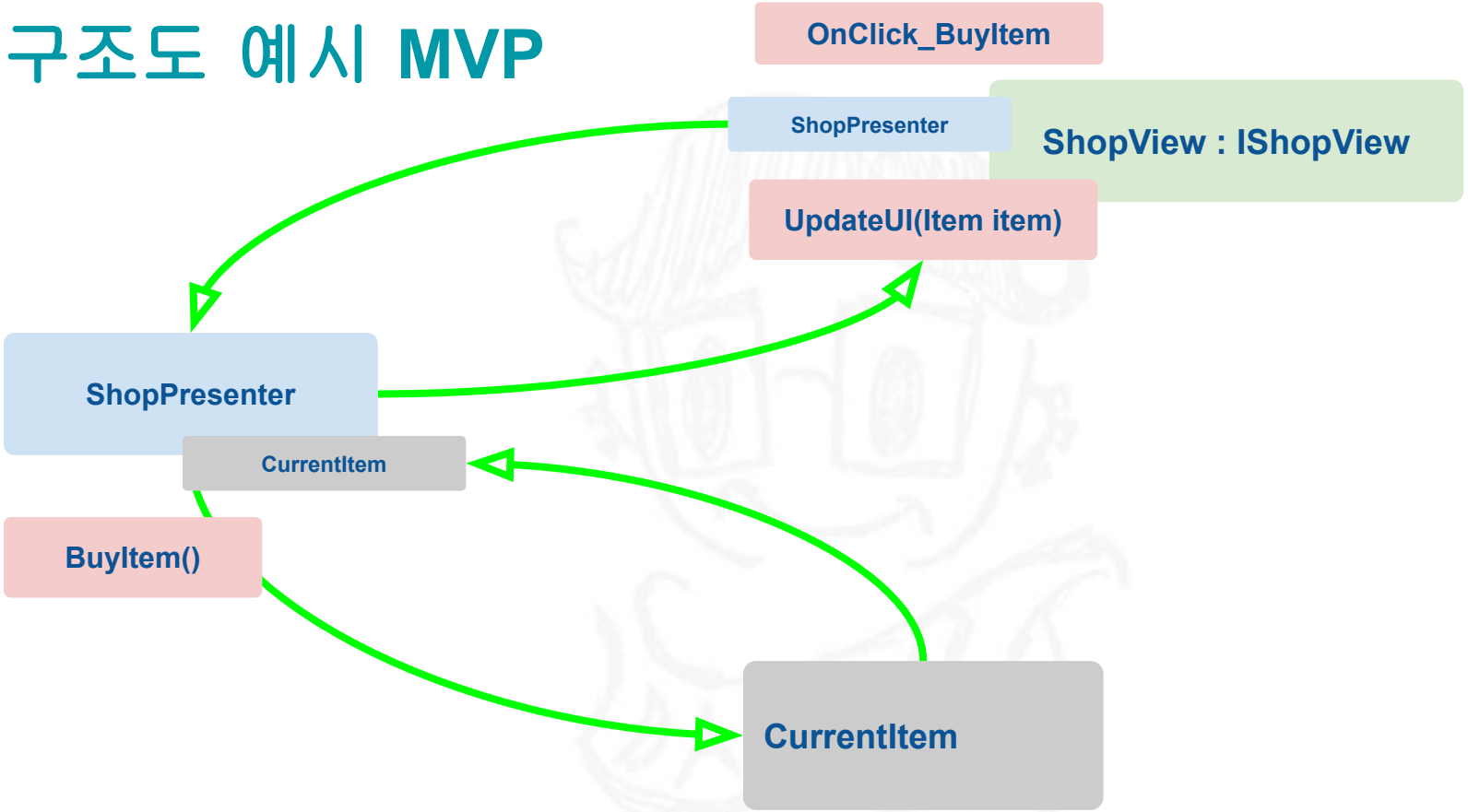
MVVM


뷰(Main UI)
뷰(캐릭터 UI)
뷰(캐릭터 HUD)
MainUIViewModel
CharacterUIViewModel
CharHUDViewModel
모델
모델을 직접 수정할 수 있는
다른 존재
(DB, 서버, 게임 로직)

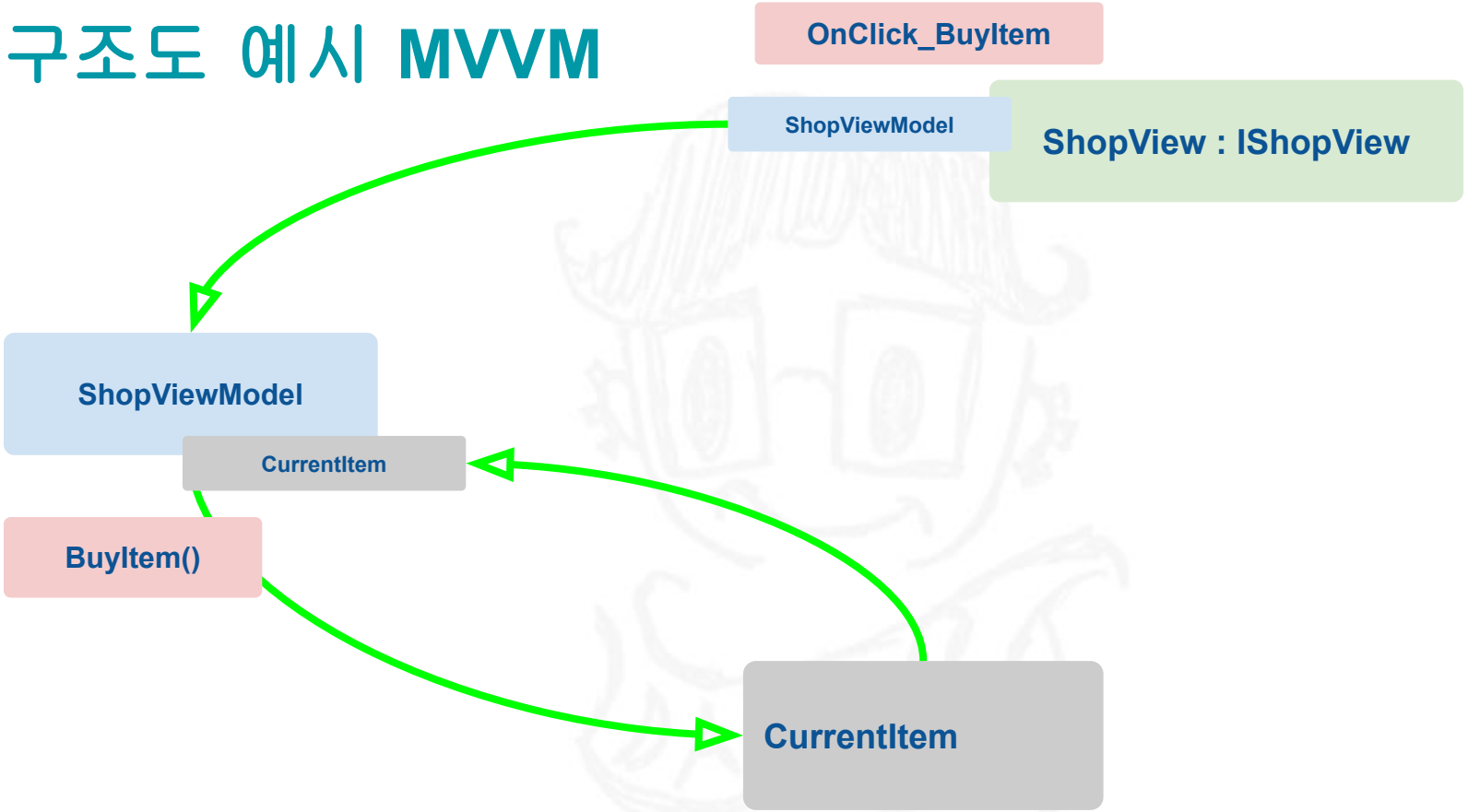
구조도 예시 MVC


ShopView
UpdateUI(Item item)
ShopView
ShopController
CurrentItem
BuyItem()
CurrentItem

구조도 예시 MVP


OnClick_BuyItem
ShopPresenter
ShopView : IShopView
UpdateUI(Item item)
ShopPresenter
CurrentItem
BuyItem()
CurrentItem

구조도 예시 MVVM


OnClick_BuyItem
ShopViewModel
ShopView : IShopView
ShopViewModel
CurrentItem
BuyItem()
CurrentItem

구조도 예시 MVVM Extension