

객체지향에서  
계속 까먹는 부분

Drake

# 목차

- 객체지향이란?
- 피드백
- 문제의 코드
- 수정후
- 이후의 과제

# 객체지향이란?

- 객체를 만들어서 객체들끼리 데이터를 주고 받는것
- 캡슐화가 기본
  - 결과 데이터를 전부 주면 안됨. 필요한 부분만 가공해서 리턴
- 데이터 응집도는 높은게 좋다
  - 다른 객체의 데이터가 들어왔는데 내 데이터랑 연관이 없다?
  - 연관있는 데이터들 끼리 모여 있을때 응집도 ↑

# 피드백



godrm an hour ago

Contributor

GameBoard 객체가 하는 makeSlot() 메소드와 데이터가 응집도가 없네요.

객체를 만드는 것은 메소드로 행동을 표현하는 것도 있지만, 관련 데이터를 객체 내부에 은닉해서 캡슐화하는 목적도 있습니다.

캡슐화한다는 것은 객체가 책임져야 하는 핵심 데이터가 내부에 있어야 한다는 겁니다.

makeSlots() 메소드는 게임보드 객체라기 보다는 그냥 slotlist를 만드는 함수 하나일 뿐이죠.

- Deck 객체와의 관계는 어떨까요? GameBoard 내부에 있어도 되지 않을까요?
- step4() 함수 코드가 긴 이유는 Deck 동작을 직접하고 있고, GameBoard가 추상화가 덜 되서 GameBoard에 있어야 할 코드가 여기 있기 때문입니다.

추상화가 되면 하위 모듈이 좀 더 복잡하고, 그것을 호출하는 상위 모듈은 더 간결해지는 게 보통입니다.

이 부분을 한 번더 개선해보세요. 이전 미션들에서 배웠던 설계/구조가 전혀 반영되지 않았습니다.

. 이전 미션들에서 배웠던 설계/구조가 전혀 반영되지 않았습니다.

# 문제의 코드 - 캡슐화

```
// 게임 모드를 선택한다
let gameMode = gameInputView.selectGameMode()
// 플레이어 수를 선택한다
let playerNumber = gameInputView.selectPlayerNumber()
// (플레이어수 + 딜러 1) * 게임모드 카드수 만큼 카드를 뽑는다
guard let cards = deck.removeCards((playerNumber+1)*gameMode.rawValue) else {
    outputView.noMoreCardMessage()
    return ()
}
// slot 배열을 만든다. 인덱스 0 이 딜러, 이후 인덱스가 플레이어
guard let slotList = gameBoard.makeSlots(gameMode: gameMode, playerNumber: playerNumber, cards: cards) else {
    outputView.noMoreCardMessage()
    return ()
}
```

- makeSlots 에서 결과값인 [Slot] 을 통째로 메인함수에서 사용

# 문제의 코드 - 데이터 응집도

```
/// 게임종류, 인원수와 카드배열을 받아서 딜러, 플레이어의 슬롯배열을 리턴
func makeSlots(gameMode: GameMode, playerNumber: Int, cards: [Card]) -> [Slot]? {
    // 들어온 카드들을 덱으로 만든다
    var deck = Deck(cardList: cards)
    // slot 배열을 만든다. 인덱스 0 이 딜러, 이후 인덱스가 플레이어
    var slotList : [Slot] = []
    // 플레이어수 + 1 만큼 반복
    for _ in 0...playerNumber {
        // 게임 종류별로 필요한 만큼 카드를 뽑는다
        guard let pickedCards = deck.removeCards(gameMode.rawValue) else {
            // 카드가 다 떨어지면 게임을 종료한다
            return nil
        }
        // 뽑은 카드를 슬롯 리스트에 넣는다
        slotList.append(Slot(pickedCards))
    }
    return slotList
}
```

- 내부의 데이터와 교류 없이 처리만 하고 나감



```
/// 게임모드, 인원 을 받아서 게임결과를 문자형 배열로 리턴
mutating func startCardGame(gameMode: GameMode, playerNumber: Int) -> [String]? {
    // 덱을 리셋하고 섞는다
    deck.reset()
    deck.shuffle()
    // 슬롯 배열을 만든다. 카드가 다 떨어지면 nil 리턴
    guard let slots = makeSlots(gameMode: gameMode, playerNumber: playerNumber) else {
        return nil
    }
    return getInfo(slots: slots)
}
```

- makeSlots 함수가 [Slot] 이 아닌 [String] 을 리턴
- 캡슐화 성공

# 수정 후

```
/// 덱 선언
```

```
private var deck = Deck()
```


```
/// 게임모드, 인원 을 받아서 게임결과를 문자형 배열로 리턴
```

```
mutating func startCardGame(gameMode: GameMode, playerNumber: Int) -> [String]? {  
    // 덱을 리셋하고 섞는다  
    deck.reset()  
    deck.shuffle()  
    // 슬롯 배열을 만든다. 카드가 다 떨어지면 nil 리턴  
    guard let slots = makeSlots(gameMode: gameMode, playerNumber: playerNumber) else {  
        return nil  
    }  
    return getInfo(slots: slots)  
}
```

- Deck 을 GameBoard 객체 내부에서 private 선언
- 게임모드, 인원수를 외부에서 입력받아서 [Slot] 을 내부에서 생성
- 결과값으로 [Slot]이 아닌 [String] 출력
- 데이터 응집도 상승, 캡슐화 처리



# 이후의 과제

- 현재 결과를 [String] 으로 리턴해서 OutputView 가 받음
- 인덱스 0 이 딜러, 나머지가 플레이어
-  **godrm** 38 minutes ago Contributor  
OutputView가 [0]번째를 딜러용으로 출력할꺼라는 걸 알고 있어야 하는데 데이터 구조나 코드 자체로 실수를 덜하도록 유도할 방법이 있을까요?
- 방법 고민중
  - 딕셔너리, 딜러유저 포함해서 리턴하기, 리턴용 객체 생성 등등