CES1017 Programming Fundamentals

# Week 10: OOP 사례학습

## - 블랙잭 카드놀이 소프트웨어 설계 및 구현

Instructor: Eunil Park (pa1324@hanyang.ac.kr)


HANYANG UNIVERSITY

# Contents

HANYANG UNIVERSITY

## Today's Schedule

1. 소프트웨어 창작
2. 블랙잭 카드놀이
3. MVC 아키텍쳐 기반 소프트웨어 설계 및 구현

# 01. 소프트웨어 창작

HANYANG UNIVERSITY

## 소프트웨어 창작

- 끊임 없는 훈련이 필요. 시간과 노력이 필요함
- 제작에 앞서 **설계**가 필요
  - 소설, 그림, 건축 등과 마찬가지
  - 예: 건물의 설계도인 청사진 완성

    ➔ 건물시공자가 설계도에 따라 건물을 지음
- 소프트웨어도 마찬가지임!
  - 코딩을 하기 전에 설계부터 해야 함
  - 설계가 소프트웨어의 품질을 좌우함
  - 설계도인 소프트웨어 아키텍쳐를 결정해야 함 (예: MVC)

# 02. 블랙잭 카드놀이

HANYANG UNIVERSITY

## Blackjack 규칙

- 딜러가 카드 두 장을 손님과 자신에게 각각 한 장씩 교대로 나누어 준다.
- 딜러의 첫째 카드는 보여주지 않는다.
- 카드의 합이 딜러보다 먼저 21이 되거나, 딜러보다 21에 가깝게 되면 이기고, 카드를 더 받았는데 21 을 초과(bust)하면 진다.
- 먼저 받은 카드 두 장의 합이 21에 못 미치면 원하는 만큼 21이 넘지 않는 한도 내에서 한 장씩 더 요청할 수 있다.
- 딜러는 카드의 합이 16 이하이면 무조건 한 장을 더 받아야 하고, 17 이상이면 더 이상 받을 수 없다.
- 딜러의 카드와 합이 같으면 비긴다.
- A(에이스) 카드는 1 또는 11로 취급할 수 있고, 10, J, Q, K는 모두 10으로 계산한다.
- 처음 받은 카드 두 장이 A와 10, J, Q, K 중의 하나로 합이 21이 되면 블랙잭 (blackjack)으로 무조건 이긴다.

# 02. 블랙잭 카드놀이

HANYANG UNIVERSITY

## 소프트웨어 요구사항 (1/3)

- 프로그램이 시작하면 Welcome to SMaSH Casino! 라는 메시지를 프린트 한다.
- 카드는 1벌(52장)을 잘 섞어서 사용한다. 다 쓰면 1벌 전체를 새로 만들어 섞어서 사용한다.
- 카드는 처음 2장씩 나누어 주는데, 손님, 딜러, 손님, 딜러 순으로 나누어주고, 딜러의 카드는 한 장만 보여준다.
- 펼쳐진 카드는 출력할 때 Spade.J 와 같은 방식으로 프린트한다. 보여주지 않는 카드는 xxxxx.xx로 프린트 한다.
- 처음 2장씩 나누어 준 후, 다음과 같이 출력 창에 프린트해야 한다. 딜러의 이름은 Dealer이고, 손님의 이름이 Pooh이면,

  “Dealer : Heart.7 XXX”

  “Pooh : Spade.A Diamond.8”
- 손님은 점수가 21 미만인 경우 카드를 추가로 요청할 수 있다. 딜러는 실행창에 다음과 같이 추가 카드를 원하는지 여부를 물어봐야 하며 손님은 영문의 o(예) 또는 x(아니오)로 의사를 표시한다. 대문자인 O와 X도 소문자와 같이 처리할 수 있어야 한다.

  “Hit? (o/x)”

# 02. 블랙잭 카드놀이

HANYANG UNIVERSITY

## 소프트웨어 요구사항 (2/3)

- 추가로 카드를 받으면 다음과 같이 받은 카드를 모두 프린트 해줘야 한다.
  “Pooh : Spade.A Diamond.8 Heart.2”
- 딜러는 카드의 합이 16 이하이면 카드를 1장 무조건 받아야 하며, 16을 넘으면 더 이상 받을 수 없다.
- A(에이스)는 1 또는 11 중 하나를 유리한 쪽으로 선택할 수 있어야 한다.
- 손님이 이기면 Pooh wins.를 프린트하고 다음 라운드로 넘어간다. 그런데 블랙잭으로 이기면 Blackjack! Pooh wins.를 프린트하고, 딜러의 버스트로 이기면 Dealer busts!를 프린트한다.
- 손님이 지면 Pooh loses.를 프린트하는데, 손님의 버스트로 지면 다음과 같이 프린트한다.
  “Pooh busts!”
- 비기면 “We draw.”를 프린트한다.
- 손님이 블랙잭으로 이긴 경우를 제외하고 라운드의 종료와 함께 딜러의 카드를 모두 보여준다.

# 02. 블랙잭 카드놀이

HANYANG UNIVERSITY

## 소프트웨어 요구사항 (3/3)

- 점수 칩의 개수를 나타내며 0에서 시작하여 매 라운드마다 손님이 이기면 1 증가하고, 지면 1 감소한다. 블랙잭으로 이기면 2 증가한다. 딜러는 블랙잭으로 이겨도 보너스가 없다. 점수는 매 라운드마다 다음과 같이 표시한다.

  “Pooh has 6 Chips.”
- 매 라운드 마다 계속할지 물어봐야 하며 손님은 o(예) 또는 x(아니오)로 의사를 표시한다.
- “Playmore, Pooh? (o/x)”
- 매 라운드 사이의 구분을 위해서 새 게임은 == new round == 로 시작을 표시한다.
- 게임을 마치면 다음과 같은 메시지를 프린트한다.

  “Bye, Pooh!”

# 02. 블랙잭 카드놀이

HANYANG UNIVERSITY

## 실행 사례 (1/2)

```
Welcome to SMaSH Casino!
How many players? (1-4) 1
Guest #1, Enter your name : Pooh
== new round ==
Smavi : Clover.5 XXX
---
Pooh : Heart.9 Diamond.J
Pooh: Hit?(o/x) x
---
Smavi : Clover.5 Heart.10 Clover.J
Bust!
Pooh : Heart.9 Diamond.J
Pooh has 1 chips.
Play more? (o/x) o
== new round ==
Smavi : Clover.8 XXX
---
Pooh : Heart.J Spade.3
Pooh: Hit?(o/x) o
Pooh : Heart.J Spade.3 Clover.4
Pooh: Hit?(o/x) x
---
Smavi : Clover.8 Heart.8 Heart.K
Bust!
Pooh : Heart.J Spade.3 Clover.4
Pooh has 2 chips.
Play more? (o/x) o
== new round ==
Smavi : Heart.6 XXX
---
Pooh : Clover.Q Diamond.5
Pooh: Hit?(o/x) o
Pooh : Clover.Q Diamond.5 Heart.7
Bust!
---
Smavi : Heart.6 Diamond.4 Spade.10
Pooh : Clover.Q Diamond.5 Heart.7
Pooh has 1 chips.
Play more? (o/x) o
== new round ==
Smavi : Clover.2 XXX
---
```

# 02. 블랙잭 카드놀이

HANYANG UNIVERSITY

## 실행 사례 (2/2)

```
Welcome to SMaSH Casino!
How many players? (1-4) 1
Guest #1, Enter your name : Pooh
== new round ==
Smavi : Clover.2 XXX
---
Pooh : Spade.J Spade.6
Pooh: Hit?(o/x) x
---
Smavi : Clover.2 Diamond.Q Diamond.5
Pooh : Spade.J Spade.6
Pooh has -1 chips.
Play more? (o/x) o
== new round ==
Smavi : Heart.7 XXX
---
Pooh : Heart.A Spade.4
Pooh: Hit?(o/x) o
Pooh : Heart.A Spade.4 Diamond.K
Pooh: Hit?(o/x) o
Pooh : Heart.A Spade.4 Diamond.K Heart.6
Blackjack!
---
Smavi : Heart.7 Diamond.A
Pooh : Heart.A Spade.4 Diamond.K Heart.6
Pooh has 1 chips.
Play more? (o/x) o
== new round ==
Smavi : Spade.7 XXX
---
Pooh : Diamond.10 Clover.J
Pooh: Hit?(o/x) x
---
Smavi : Spade.7 Clover.K
Pooh : Diamond.10 Clover.J
Pooh has 2 chips.
Play more? (o/x) o
== new round ==
Smavi : Diamond.2 XXX
---
Pooh : Clover.6 Heart.J
Pooh: Hit?(o/x) o
Pooh : Clover.6 Heart.J Heart.K
Bust!
---
Smavi : Diamond.2 Clover.10 Diamond.8
Pooh : Clover.6 Heart.J Heart.K
Pooh has 1 chips.
Play more? (o/x) o
== new round ==
Smavi : Heart.Q XXX
---
Pooh : Clover.9 Spade.K
Pooh: Hit?(o/x) x
---
Smavi : Heart.Q Diamond.7
Pooh : Clover.9 Spade.K
Pooh has 1 chips.
Play more? (o/x) o
== new round ==
Smavi : Heart.3 XXX
---
Pooh : Spade.A Clover.Q
Blackjack!
---
Smavi : Heart.3 Heart.10 Heart.2 Diamond.
Bust!
Pooh : Spade.A Clover.Q
Pooh has 3 chips.
Play more? (o/x) x
Bye, everyone!
>>>
```

# 02. 블랙잭 카드놀이

HANYANG UNIVERSITY

## 알고리즘

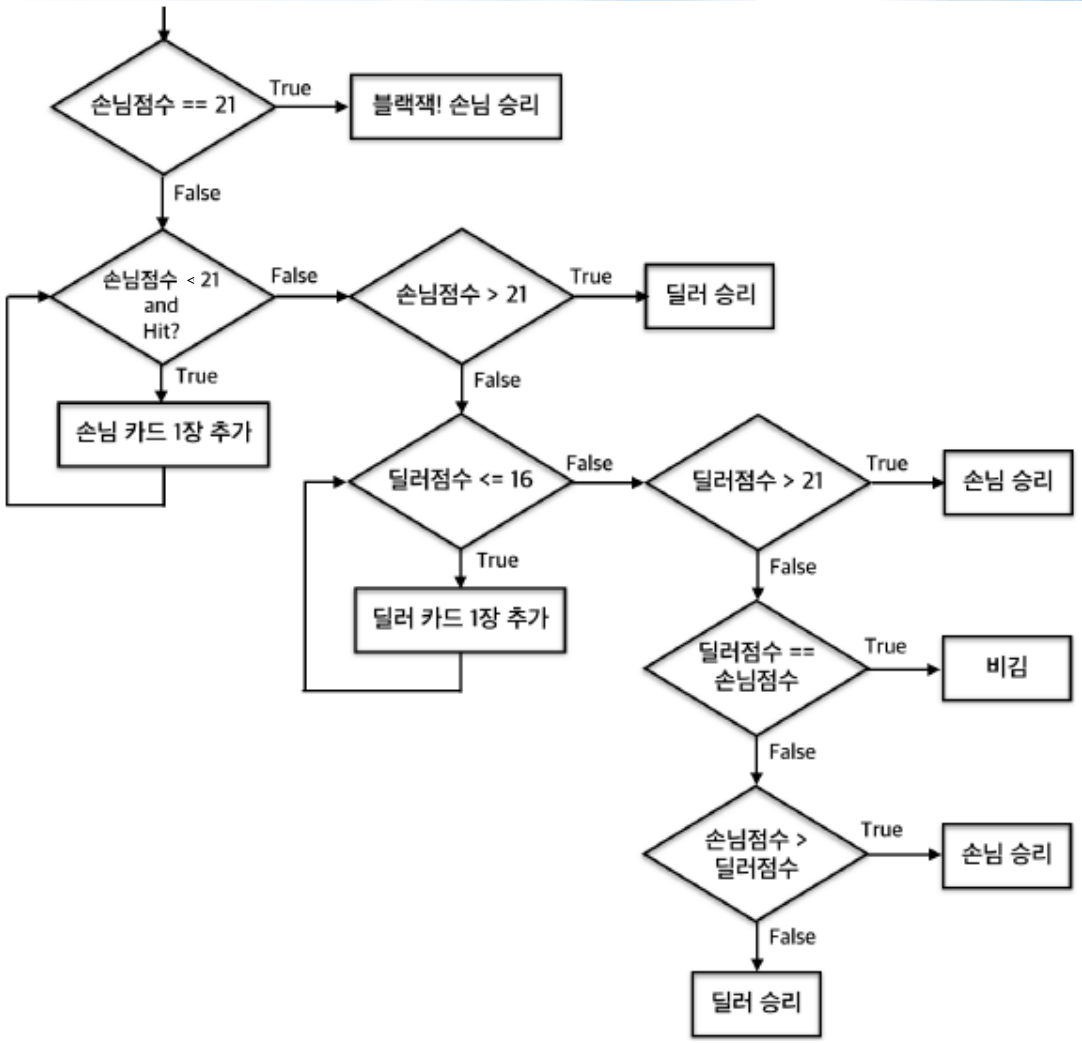
손님이 원하는 한, 다음 라운드를 반복한다.

1. 손님, 딜러, 손님, 딜러 순으로 카드를 총 2장씩 배분한다.
2. 손님의 카드는 모두 보여주고, 딜러의 카드는 한장만 보여준다.
3. 손님과 딜러의 카드 두 장의 정수를 각각 계산한다.
4. 다음 장의 흐름 차트에 따라서 게임을 진행한다
5. 더 할지 손님에게 물어봐서 그만하길 원하면 끝내고, 더하길 원하면 1번으로 돌아가서 계속한다.

# 02. 블랙잭 카드놀이

HANYANG UNIVERSITY

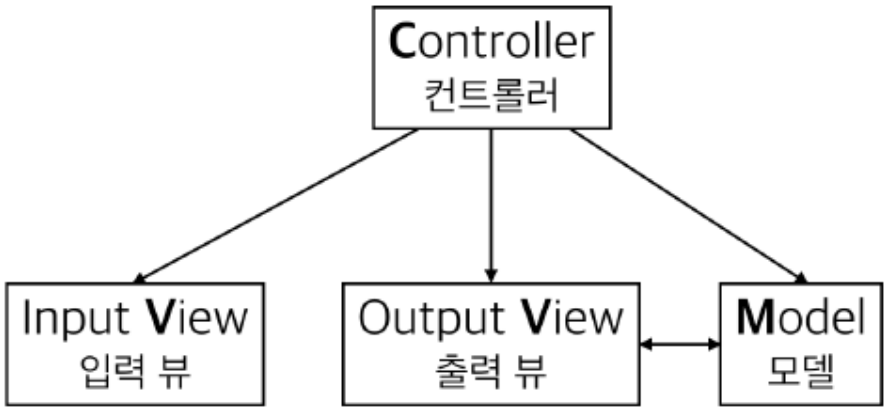
## 흐름챠트(Flow Chart)



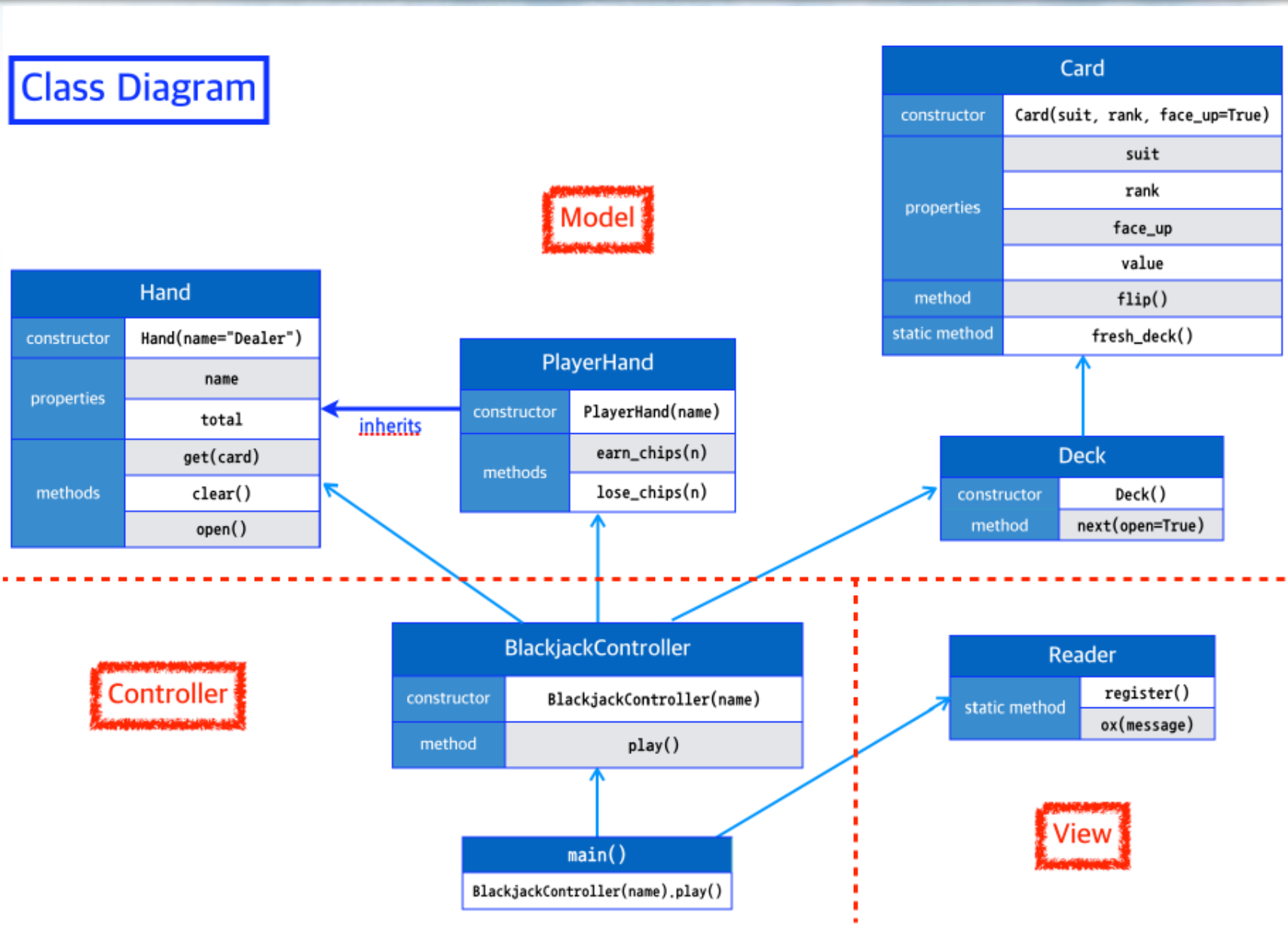
HANYANG UNIVERSITY

## MVC 아키텍쳐

- MVC (Model – View – Controller) 아키텍쳐
  - 컨트롤러(Controller)는 프로그램의 흐름(실행순서)을 제어한다.
  - 입/출력 뷰(View)는 외부와의 의사소통을 담당한다.
  - 모델(Model)은 문제를 모델링하고 해답을 계산하는 역할을 한다.




- 컨트롤러는 입력 뷰에 메시지를 보내 데이터를 요청한다.
- 컨트롤러는 그 데이터를 가지고 모델에 계산을 요청한다.
- 컨트롤러는 출력 뷰에게 계산결과를 보여주라고 요청한다.
- 경우에 따라서 모델이 직접 출력 뷰에 계산결과를 보여주라고 요청할 수도 있고, 출력 뷰가 능동적으로 모델에서 계산 결과를 가져다 보여줄 수도 있다.


Class Diagram
Model
Card
constructor
Card(suit, rank, face_up=True)
properties
suit
rank
face_up
value
method
flip()
static method
fresh_deck()
Hand
constructor
Hand(name="Dealer")
properties
name
total
methods
get(card)
clear()
open()
inherits
PlayerHand
constructor
PlayerHand(name)
methods
earn_chips(n)
lose_chips(n)
Deck
constructor
Deck()
method
next(open=True)
Controller
BlackjackController
constructor
BlackjackController(name)
method
play()
Reader
static method
register()
ox(message)
main()
BlackjackController(name).play()
View

HANYANG UNIVERSITY

## Card

| Card | | | |
|---|---|---|---|
| class | attributes | self.__suits | ("Diamond", "Heart", "Spade", "Clover") |
| | | self.__ranks | ("A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K") |
| | static method | fresh_deck() | returns a brand-new deck of shuffled cards with all face down |
| object | properties | suit | its suit value in Card.__suits |
| | | rank | its rank value in Card.__ranks |
| | | face_up | its face_up value : True or False |
| | | value | its face value (according to blackjack rule) |
| | constructor | Card(suit, rank, face_up=True) | creates a playing card object<br>arguments:<br>suit -- must be in Card.__suits<br>rank -- must be in Card.__ranks<br>face_up -- True or False (default True) |
| | method | flip() | flips itself |

## Card

```
def __init__(self, suit, rank, face_up=True):
    if suit in Card.__suits and rank in Card.__ranks:
        self.__suit = suit
        self.__rank = rank
        self.__face_up = face_up
    else:
        print("Error: Not a valid card")
    self.__value = Card.__ranks.index(self.__rank) + 1
    if self.__value > 10:
        self.__value = 10
```

"J", "Q", "K"는 모두 10점
"A"는 경우에 따라서 1점 또는 11점 선택 가능하지만 일단 1점으로 둠

HANYANG UNIVERSITY

## Deck

카드 한벌(deck)을 형상화하여 관리하는 객체

| Deck | | | |
|---|---|---|---|
| object | atrribute | self.__deck | Deck object |
| | constructor | Deck() | creates a deck object consisting of 52 cards shuffled |
| | method | next(open=True) | removes a card from deck and returns the card with its face up if open == True, or with its face down if open == False |

```
def __init__(self):
    self.__deck = Card.fresh_deck()
    print("<< A brand-new deck of card! >>")
```

잘 섞인 카드 한벌을 리스트로 갖고 있음

```
def next(self, open=True):
    if self.__deck == []:
        self.__deck = Card.fresh_deck()
        print("<< A brand-new deck of card! >>")
    card = self.__deck.pop()
    if open :
        card.flip()
    return card
```

next() -> 앞면
next(False) -> 뒷면

카드를 다 쓰고 없으면, Card.fresh_deck() 메소드를 호출하여 카드 한 벌을 새로 받아옴

## Hand

각 게임 참여자가 받아서 갖고 있는 게임 형상화

| Hand | | | |
|---|---|---|---|
| object | constructor | Hand(name="Dealer") | creates player/dealer's empty hand<br>argument: name -- player's name in string (default: |
| | attributes | self.__name | its name : either player's name or 'Dealer' |
| | | self.__hand | list of Card objects in its hand |
| | properties | name | its name : either player's name or 'Dealer' |
| | | total | the total value of its hand |
| | methods | get(card) | gets a card from deck and puts the card into its hand |
| | | clear() | empties its hand |
| | | open() | turns all of its hand's cards' faces up |

```
def __init__(self, name="Dealer"):
    self.__name = name
    self.__hand = []
```

## Hand

```
@property
def name(self):
    return self.__name

@property
def total(self):
    point = 0
    number_of_ace = 0
    for card in self.__hand:
        if card.rank == 'A':
            point += 11
            number_of_ace += 1
        else:
            point += card.value
    while point > 21 and number_of_ace > 0:
        point -= 10
        number_of_ace -= 1
    return point
```

점수 처리 및
"A" 카드 처리

```
def get(self, card):
   self.__hand.append(card)

def clear(self):
   self.__hand = []

def open(self):
   for card in self.__hand:
        if not card.face_up:
            card.flip()
```

> card 객체를 자신의 카드 리스트에 추가

> card 리스트를 비움

> 자신의 모든 카드를 앞면으로 뒤집음

HANYANG UNIVERSITY

## Player Hand

- Hand 클래스는 손님과 딜러가 공통으로 사용할 수 있지만, 손님의 경우 자신이 따거나 잃은 칩의 개수를 기록해두고 알려주는 기능이 추가로 있어야 함
- Hand 클래스가 갖고 있는 속성과 메소드는 그대로 유지한 채, 새 속성과 메소드를 추가하고 싶으면 **상속(inheritance)** 기능을 활용하면 됨
- 손님전용 클래스의 추가해야 할 속성과 메소드가 다음과 같음

| PlayerHand | | | |
|---|---|---|---|
| object | constructor | PlayerHand(name) | creates player's empty hand<br>with the capability of counting chips it owns<br>argument: name -- player' name in string |
| | attribute | self.__chips | the number of chips it has (initial value = 0) |
| | method | earn_chips(n) | increases the number of chips by n |
| | | lose_chips(n) | decreases the number of chips by n |

HANYANG UNIVERSITY

## Player Hand

```
class PlayerHand(Hand):
    def __init__(self, name):
        super().__init__(name)
        self.__chips = 0

    def earn_chips(self, n):
        self.__chips += n
        print("Your have", self.__chips, "chips.")

    def lose_chips(self, n):
        self.__chips -= n
        print("Your have", self.__chips, "chips.")
```

**상속받을 클래스는 클래스 선언 부분에서 파라미터로 상속받는 클래스 이름을 명시**

- 새로 선언하는 PlayerHand 클래스가 Hand 클래스를 상속받으므로 class PlayerHand(Hand) 라고 기술한 것임
- PlayerHand 클래스를 **하위클래스(subclass)**라 하고, Hand 클래스를 **상위클래스(superclass)**라고 함

**self.__chips 속성변수의 값을 인수만큼 줄이거나 늘리는 역할을 함**

**상속**

- 이미 완성되어 있는 클래스를 재사용하고 추가할 속성이나 메소드만 추가하면 됨
- 객체지향 프로그래밍에서 가장 강력한 기능 중의 하나

HANYANG UNIVERSITY

## Reader

뷰 역할을 하는 클래스

| Reader | | |
|---|---|---|
| static method | register() | gets player's name and returns it (string) |
| | ox(message) | returns True if player inputs 'o' or 'O',<br>False if player inputs 'x' or 'X' |

```
class Reader:
    @staticmethod
    def register():
        return input("Enter your name : ")

    @staticmethod
    def ox(message):
        response = input(message).lower()
        while not (response == 'o' or response == 'x'):
            response = input(message).lower()
        return response == 'o'
```

클래스 고유의 기능을 정의하는 클래스 소속 메소드

HANYANG UNIVERSITY

## BlackjackController

컨트롤러 역할을 하는 클래스

| BlackjackController | | | |
|---|---|---|---|
| object | constructor | BlackjackController(name) | creates player/dealer's empty hand and a deck of cards<br>argument: name -- player' name in string<br>(default: 'Dealer') |
| | attributes | self.__player | PlayerHand object |
| | | self.__dealer | Dealer object |
| | | self.__deck | Deck object |
| | method | play() | plays a round of blackjack game |

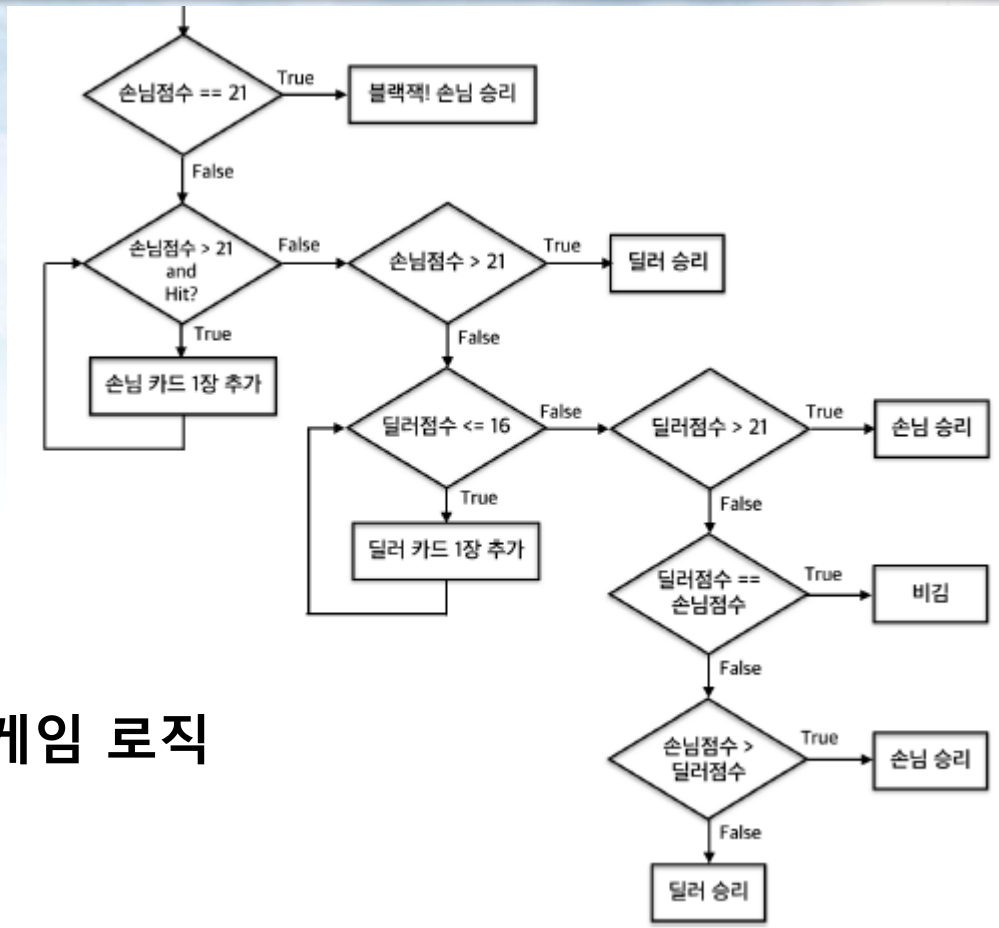
```
def __init__(self, name):
    self.__player = PlayerHand(name)
    self.__dealer = Hand()
    self.__ deck   = Deck()
```

# 03. MVC 아키텍처 기반 소프트웨어 설계 및 구현

HANYANG UNIVERSITY

```
def play(self):
    print("== new game ==")
    player = self.__player
    dealer = self.__dealer
    deck = self.__deck
    player.get(deck.next())
    dealer.get(deck.next())
    player.get(deck.next())
    dealer.get(deck.next(open=False))
    print("Dealer :", dealer)
    print(player.name, ":", player)
    if player.total == 21:
        print("Blackjack!", player.name, "wins.")
        player.earn_chips(2)
    else:
        while player.total < 21 and \
              Reader.ox(player.name + ": Hit?(o/x) "):
            player.get(deck.next())
            print(player.name, ":", player)
        if player.total > 21:
            print(player.name, "busts!")
            player.lose_chips(1)
        else:
            while dealer.total <= 16:
                dealer.get(deck.next())
            if dealer.total > 21:
                print("Dealer busts!")
                player.earn_chips(1)
            elif dealer.total == player.total:
                print("We draw.")
            elif dealer.total > player.total:
                print(player.name, "loses.")
                player.lose_chips(1)
            else:
                print(player.name, "wins.")
                player.earn_chips(1)
        dealer.open()
        print("Dealer :", dealer)
    player.clear()
    dealer.clear()
```

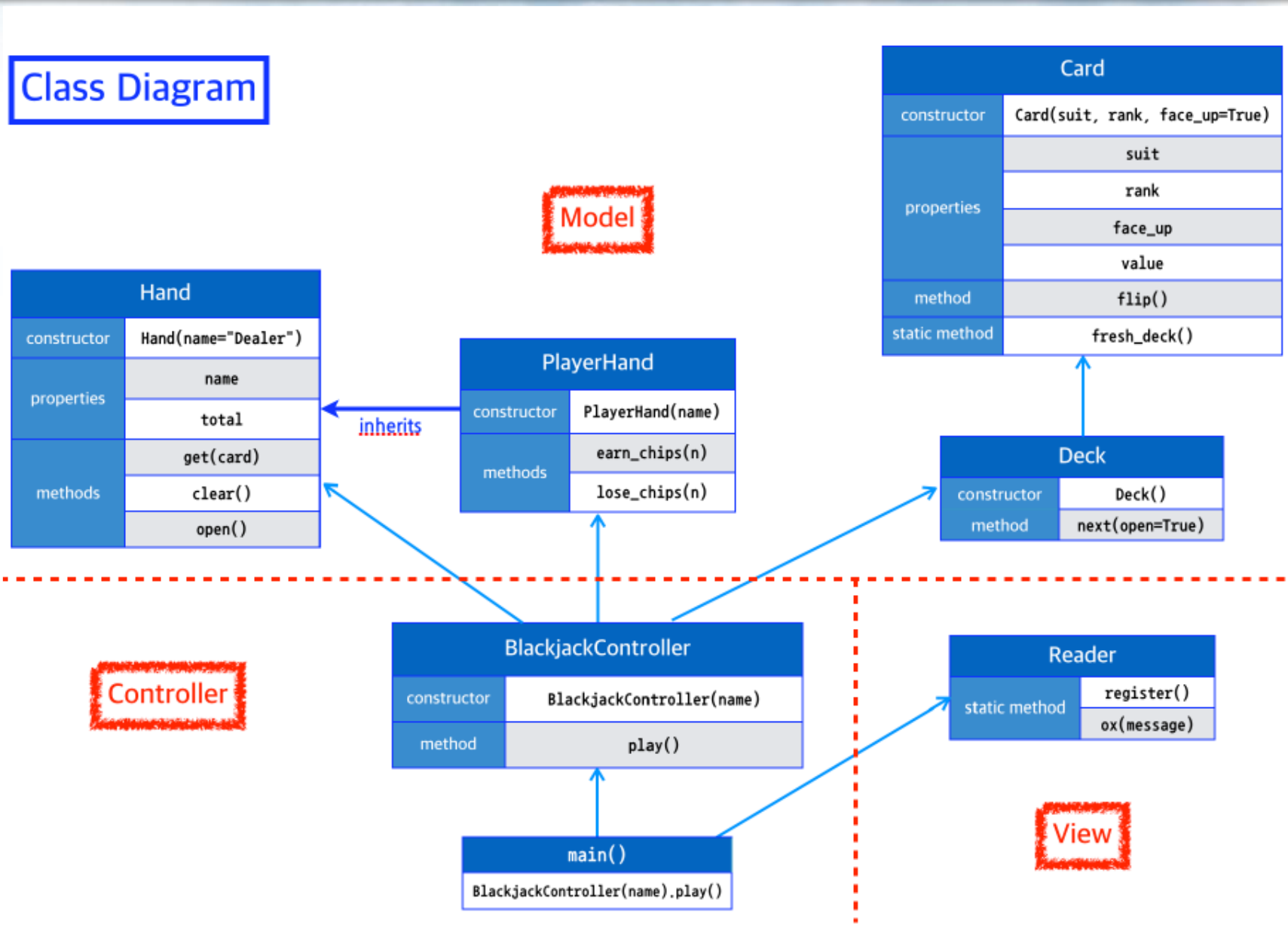
게임 로직



## 메인 함수

| main() |
| --- |
| BlackjackController(name).play() |

```
def main():
   print("Welcome to SMaSH Casino!")
    name = Reader.register()
    game = BlackjackController(name)
    while True:
        game.play()
        if not Reader.ox("Play more, " + name + "? (o/x) "):
            break
    print("Bye, " + name + "!")
```


Class Diagram
Model
Card
constructor
Card(suit, rank, face_up=True)
properties
suit
rank
face_up
value
method
flip()
static method
fresh_deck()
Hand
constructor
Hand(name="Dealer")
properties
name
total
methods
get(card)
clear()
open()
inherits
PlayerHand
constructor
PlayerHand(name)
methods
earn_chips(n)
lose_chips(n)
Deck
constructor
Deck()
method
next(open=True)
Controller
BlackjackController
constructor
BlackjackController(name)
method
play()
Reader
static method
register()
ox(message)
main()
BlackjackController(name).play()
View

# Today's Lessons!

HANYANG UNIVERSITY

## Summary

1. 소프트웨어 창작
2. 블랙잭 카드놀이
3. MVC 아키텍쳐 기반 소프트웨어 설계 및 구현

# Thanks

Week 10: OOP 사례학습 - 블랙잭 카드놀이 소프트웨어 설계 및 구현

Instructor: Eunil Park (pa1324@hanyang.ac.kr)