CS101 – 객체 (object)로 블랙잭 카드 게임 만들기 (2/2)

Lecture 22

School of Computing
KAIST

학습 목표:

- 블랙잭 카드 게임에 필요한 자료구조들을 객체를 사용해서 만들 수 있다.
- 블랙잭 카드 게임에 필요한 사용자 인터페이스 프로그래밍을 할 수 있다.

블랙잭

블랙잭 게임은 총 52장으로 이루어진 플레잉 카드를 사용합니다.
각각의 카드는 무늬와 끗수를 가지고 있습니다.
(무늬: 클럽, 스페이드, 하트, 다이아몬드)
(끗수: 2, 3, ... , 10, J, Q, K, A)


```
class Card(object):
    """A Blackjack card."""
    pass

card = Card()
card.face = "Ace"
card.suit = "Spades"
card.value = 11
```

멤버 함수

카드의 값은 끗수에 의해서 결정되기 때문에, 카드의 값을 의미하는 속성이 따로 있을 필요는 없습니다.
대신 값을 계산해주는 멤버 함수 `value()`를 만들어서 사용합시다.

```
class Card(object):
  """A Blackjack card."""
  def value(self): # method of Card
    if type(self.face) == int:
      return self.face
    elif self.face == "Ace":
      return 11
    else:
      return 10
```

`self` 는 객체의 멤버 함수에서 객체 자신을 가리킵니다.

멤버 함수

Card 객체를 만들어서 사용해 봅시다.

```
>>> card1 = Card()
>>> card1.face = "Ace"
>>> card1.suit = "Spades"
>>> card2 = Card()
>>> card2.face = 2
>>> card2.suit = "Clubs"
>>> card_string(card1)
'an Ace of Spades'
>>> card1.value()
11
>>> card_string(card2)
a 2 of Clubs
>>> card2.value()
2
```

Card 객체를 만드는 과정이 너무 비직관적이네요.
Card(8, "Clubs") 처럼 좀 더 근사하게 객체를 만들고 싶습니다.

card_string() 함수도 Card 객체의 멤버 함수로 만들고 싶어요.

생성자

객체는 __init__이라는 특별한 멤버 함수를 가지고 있습니다.
이 멤버 함수는 **생성자(Constructor)** 라고 불립니다.
객체가 생성될 때 생성자는 자동으로 호출됩니다.

```
FACES = list(range(2, 11)) +
        ['Jack', 'Queen', 'King', 'Ace']
SUITS = ['Clubs', 'Diamonds', 'Hearts', 'Spades']

Class Card(object):
  """A Blackjack card."""
  def __init__(self, face, suit):
    assert face in FACES and suit in SUITS
    self.face = face
    self.suit = suit
```

생성자

이제 Card 객체를 훨씬 간단하게 만들 수 있습니다.

```
hand = [Card("Ace", "Spades"), Card(8, "Diamonds"),
          Card("Jack", "Hearts"), Card(10, "Clubs")]
```

card_string(card) 함수를 Card 객체의 멤버 함수로 만들어 봅시다.

```
class Card(object):
  """A Blackjack card."""
  """Already defined  __init__ and value methods"""
  def string(self):
    article = "a "
    if self.face in [8, "Ace"]:
      article = "an "
      return (article + str(self.face) +  " of " + self.suit)
```

이제 다음처럼 카드 내용을 출력할 수 있습니다.

```
>>> for card in hand:
...     print(card.string(), "has value", card.value())
```

문자열 변환

card_string()함수를 사용하지 않고도 카드 내용을 문자열로 바꾸는 방법이 있습니다.
str(card) 는 card의 특별한 멤버 함수 __str__ 를 호출합니다.

```
class Card(object):
    """A Blackjack card."""
    """Already defined ___init__ and value methods"""
    def __str__(self):
        article = "a "
        if self.face in [8, "Ace"]:
            article = "an "
        return (article + str(self.face) + " of"
                    + self.suit)
```

이제 다음처럼 카드 내용을 출력할 수 있습니다.

```
>>> for card in hand:
...     print(card, "has value", card.value())
```

print 함수는 인자 card 를 __str__를 사용해서 자동으로 문자열 형태로 바꿉니다.

카드 덱

블랙잭 게임에는 52장의 모든 카드를 섞어서 만든 카드 덱(deck)이 필요합니다.
카드 덱을 의미하는 객체를 만들어봅시다.
이 객체에는 덱에서 카드를 한 장 뽑는 역할의 멤버 함수가 필요합니다.

```
class Deck(object):
  """A deck of cards."""
  def __init__(self):
    "Create a deck of 52 cards and shuffle them."
    self.cards = []
    for suit in SUITS:
      for face in FACES:
        self.cards.append(Card(face, suit))
    random.shuffle(self.cards)

  def draw(self):
    """Draw the top card from the deck."""
    return self.cards.pop()
```

카드 덱

```
num_players = 3
num_cards = 5
deck = Deck()
hands = [] # A list of lists (one for each player)
for j in range(num_players):
  hands.append([])
for i in range(num_cards):
  for j in range(num_players):
    card = deck.draw()
    hands[j].append(card)
    print("Player", j+1, "draws", card)

for j in range(num_players):
  print ("Player %d's hand (value %d):" %
         (j+1, hand_value(hands[j])))
  for card in hands[j]:
    print (" ", card)
```

블랙잭

이제 블랙잭 게임을 만들어 봅시다.

```
You are dealt a 6 of Hearts
Dealer is dealt a hidden card
You are dealt a 3 of Spades
Dealer is dealt a 9 of Hearts
Your total is 9
Would you like another card? (y/n) y
You are dealt an Ace of Clubs
Your total is 20
Would you like another card? (y/n) n
The dealer's hidden card was a 10 of Spades
The dealer's total is 19
Your total is 20
The dealer's total is 19
You win!
```

부등식

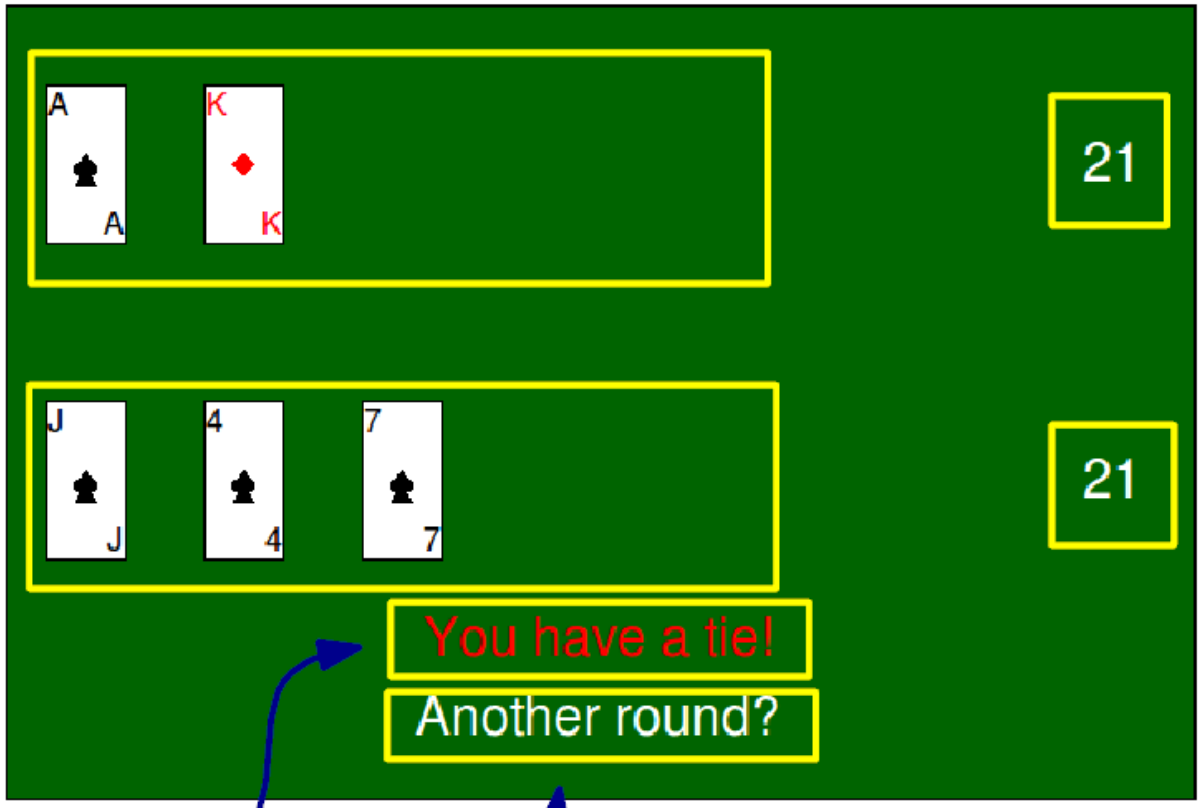
객체에서 비교 연산자 (==, !=, < 등)를 사용하면 예상과 다른 결과가 나올 수 있습니다.

```
>>> Card(8, "Diamonds") == Card(9, "Diamonds")
False
>>> Card(8, "Diamonds") == Card(8, "Diamonds")
False
```

사용자의 생각대로 비교 연산자를 통해 객체를 비교하기 위해서는 다음과 같이 정의를 해야 합니다.

```
class Card(object):
  """A Blackjack card."""
  """Already defined other methods"""
  def __eq__(self, rhs):
    return (self.face == rhs.face and
            self.suit == rhs.suit)
  def __ne__(self, rhs):
    return not self == rhs
```

블랙잭 게임 인터페이스



플레이어의 손패

플레이어의 점수

딜러의 손패

딜러의 점수

메시지

사용자의 Y/N 입력을 기다리는 메시지

Table 객체

Table 객체는 블랙잭 게임 판을 의미하는 객체입니다.
이 객체는 다음과 같은 멤버 함수들을 가지고 있습니다.

- `clear()` : 게임 초기화
- `close()` : 창을 닫고 게임을 종료
- `set_score(which, text)`
  : 0,1 중 하나의 값을 가진 `which`에 해당하는 대상의 점수를 설정
- `set_message(text)`
- `ask(prompt)`: 사용자의 입력 (y, n)을 받아서 **`True`** 나 **`False`** 를 반환

Table 객체는 `dealer`, `player`라는 두 개의 속성을 가지고 있습니다.
이 속성들은 각각 플레이어의 손패를 의미하는 Hand 객체입니다.

Hand 객체는 다음과 같은 멤버 함수들을 가지고 있습니다.

- `clear()`
- `add(card, hidden = False)`
- `show()`: 모든 숨겨진 카드를 공개
- `value()`: 손에 있는 카드들의 값의 합을 반환

멤버 함수

Table.ask(prompt) 함수는 사용자의 입력을 기다려야 합니다.

```
def ask(self, prompt):
  self.question.setMessage(prompt)
  while True:
    e = self.canvas.wait()
    d = e.getDescription()
    if d == "canvas close":
      sys.exit(1)
    if d == "keyboard":
      key = e.getKey()
      if key == 'y':
        return True
      if key == 'n':
        return False
```

사용자 인터페이스 프로그래밍

그래픽 유저 인터페이스(GUI)를 사용하는 프로그램은 여러 종류의 **이벤트**를 기반으로 작동합니다. 이런 프로그램은 이벤트를 기다리고, 호출된 이벤트에 따라 일을 실행합니다.

다음과 같은 이벤트가 있을 수 있습니다.

- 키 입력
- 윈도우 창의 최소화/최대화/종료
- 마우스 버튼 클릭
- 마우스 커서가 창 안에 들어옴/나감

이벤트 기반 프로그래밍은 순차적으로 실행되기만 하는 프로그램을 개발하는 것이 아닌, 여러 이벤트들에 의한 함수 호출이 필요한 프로그램을 개발하는 것을 의미합니다.

정리 및 예습

본 강의 학습목표:

- 블랙잭 카드 게임에 필요한 자료구조들을 객체를 사용해서 만들 수 있다.
- 블랙잭 카드 게임에 필요한 사용자 인터페이스 프로그래밍을 할 수 있다.

다음 강의 학습 목표:

- 지금까지 강의에서 사용한 객체의 상태와 동작을 이해할 수 있다.
- 닭과 여러 병아리들이 움직이는 애니메이션을 객체를 사용해 만들 수 있다.