

숙제8 – Yahtzee 게임 파이썬 설명

플레이어 명수(1~10)와 플레이어 이름 입력 후 설정완료 버튼

플레이어	명수	
플레이어1	이름	김영식
플레이어2	이름	이재영
플레이어3	이름	
플레이어4	이름	
플레이어5	이름	
플레이어6	이름	
플레이어7	이름	
플레이어8	이름	
플레이어9	이름	
플레이어10	이름	
Yahtzee 플레이어 설정 완료		

플레이어1 차례: Roll Dice 버튼 누르기

tk

Roll Dice	Category	김영식	이재영
?	Ones		
?	Twos		
?	Threes		
?	Fours		
?	Fives		
	Sixes		
	Upper Scores		
	Upper Bonus(35)		
	Three of a kind		
	Four of a kind		
	Full House(25)		
	Small Straight(30)		
	Large Straight(40)		
	Yahtzee(50)		
	Chance		
	Lower Scores		
	Total		

김영식차례: Roll Dice 버튼을 누르세요

보관할 주사위 선택후 Roll Again

tk

Roll Again

2

6

4

3

2

Category	김영식	이재영
Ones		
Twos		
Threes		
Fours		
Fives		
Sixes		
Upper Scores		
Upper Bonus(35)		
Three of a kind		
Four of a kind		
Full House(25)		
Small Straight(30)		
Large Straight(40)		
Yahtzee(50)		
Chance		
Lower Scores		
Total		

보관할 주사위 선택 후 Roll Again

한번 더 보관할 주사위 선택후 Roll Again

tk

Roll Again	Category	김영식	이재영
6	Ones		
6	Twos		
4	Threes		
3	Fours		
2	Fives		
	Sixes		
	Upper Scores		
	Upper Bonus(35)		
	Three of a kind		
	Four of a kind		
	Full House(25)		
	Small Straight(30)		
	Large Straight(40)		
	Yahtzee(50)		
	Chance		
	Lower Scores		
	Total		

보관할 주사위 선택 후 Roll Again

플레이어1은 3번 Roll 후 13개 카테고리 중에 남아있는 것 중에 하나(small straight) 선택

tk

Roll Again

1
1
4
3
2

Category	김영식	이재영
Ones		
Twos		
Threes		
Fours		
Fives		
Sixes		
Upper Scores		
Upper Bonus(35)		
Three of a kind		
Four of a kind		
Full House(25)		
Small Straight(30)		
Large Straight(40)		
Yahtzee(50)		
Chance		
Lower Scores		
Total		

카테고리를 선택하세요

플레이어 2 차례 : Roll Dice 버튼 누르기

tk

Roll Dice	Category	김영식	이재영
?	Ones		
?	Twos		
?	Threes		
?	Fours		
?	Fives		
?	Sixes		
	Upper Scores		
	Upper Bonus(35)		
	Three of a kind		
	Four of a kind		
	Full House(25)		
	Small Straight(30)	30	
	Large Straight(40)		
	Yahtzee(50)		
	Chance		
	Lower Scores		
	Total		

이재영차례: Roll Dice 버튼을 누르세요

class Player

```
class Player:
    UPPER = 6 # upper category 6개
    LOWER = 7 # lower category 7개
    def __init__(self, name):
        self.name = name
        self.scores=[0 for i in range(self.UPPER+self.LOWER)] #13개category점수
        #13개 category 사용여부
        self.used=[False for i in range(self.UPPER+self.LOWER)]

    def setScore(self, score, index):
    def setAtUsed(self, index):
    def getUpperScore(self):
    def getLowerScore(self):
    def getUsed(self):
    def getTotalScore(self):
    def toString(self):
        return self.name
    def allLowerUsed(self): #lower category 7개 모두 사용되었는가 ?
    def allUpperUsed(self): #upper category 6개 모두 사용되었는가 ?
        #UpperScores, UpperBonus 계산에 활용
        for i in range(self.UPPER):
            if (self.used[i] == False):
                return False
        return True
```


class Dice

```
import random  
class Dice:
```

```
    def rollDie(self):  
        self.roll = random.randint(1,6)    #[1:6] 랜덤 정수  
    def getRoll(self):  
        return self.roll
```

class YahtzeeBoard

```
from tkinter import *
from tkinter import font
import tkinter.messagebox
from player import *
from dice import *
from configuration import *
```

```
class YahtzeeBoard:
```

```
    UPERTOTAL = 6           #UpperScore 범주 인덱스
```

```
    UPPERBONUS = 7          #UpperBonus 범주 인덱스
```

```
    LOWERTOTAL = 15         #LowerScore 범주 인덱스
```

```
    TOTAL = 16              #Total 범주 인덱스
```

```
    dice = []               #Dice 객체 리스트
```

```
    diceButtons = []        #diceButton 리스트
```

```
    fields = []             #각 플레이어 점수판 2차원 리스트
```

```
                            #열 플레이어, 0열=플레이어1, 1열=플레이어2, ...
```

```
                            #17행 점수 = 카테고리13행 + upperScore + upperBonus + LowerScore + Total
```

```
    players = []            #player 객체 리스트
```

```
    numPlayers = 0
```

```
    player = 0              # 플레이어 순서를 제어
```

```
    round = 0               # 13라운드를 제어
```

```
    roll = 0                # 각 라운드마다 3번 굴리기 roll을 할 수 있음
```

```
def __init__(self):
    self.InitPlayers()
```

class YahtzeeBoard

```
class YahtzeeBoard:
```

```
    . . .
def InitPlayers(self): #player window 생성하고 최대 10명까지 플레이어 설정
    self.pwindow = Tk()
    self.TempFont = font.Font(size=16, weight='bold', family='Consolas')
    self.label = []
    self.entry = []
    self.label.append(Label(self.pwindow, text="플레이어 명수", font=self.TempFont))
    self.label[0].grid(row=0, column=0)

    for i in range(1,11):
        self.label.append(Label(self.pwindow, text="플레이어"+str(i)+" 이름",
                                font=self.TempFont))
        self.label[i].grid(row=i, column=0)

    for i in range(11):
        self.entry.append(Entry(self.pwindow, font=self.TempFont))
        self.entry[i].grid(row=i, column=1)
    Button(self.pwindow, text="Yahtzee 플레이어 설정 완료",
           font=self.TempFont, command=self.playerNames).grid(row=11, column=0)
    self.pwindow.mainloop()

def playerNames(self): #플레이어 설정 완료 버튼 누르면 실행되는 함수
    self.numPlayers = int(self.entry[0].get())
    for i in range(1, self.numPlayers+1):
        self.players.append(Player(str(self.entry[i].get())))
    self.pwindow.destroy()
    self.initInterface() #Yahtzee 보드판 플레이어 명수 만큼 생성
```

class YahtzeeBoard

```
class YahtzeeBoard:
```

```
    . . .  
def initInterface(self): #Yahtzee 보드 윈도우 생성  
    self.window = Tk( " Yahtzee Game " )  
    self.window.geometry( " 1600x800 " )  
    self.TempFont = font.Font(size=16, weight= ' bold ' , family= ' Consolas ' )  
  
    for i in range(5): #Dice 객체 5개 생성  
        self.dice.append(Dice())  
  
    self.rollDice = Button(self.window, text= " Roll Dice " , font=self.TempFont,  
        command=self.rollDiceListener)//Roll Dice 버튼  
    self.rollDice.grid(row=0, column=0)  
  
    for i in range(5): #dice 버튼 5개 생성  
        self.diceButtons.append(Button(self.window, text= "? " ,  
            font=self.TempFont, width=8, command=lambda row=i: self.diceListener(row)))  
        #각각의 dice 버튼에 대한 이벤트 처리 diceListener 연결  
        #람다 함수를 이용하여 diceListener 매개변수 설정하면 하나의 Listener로 해결  
        self.diceButtons[i].grid(row=i + 1, column=0)  
    . . .
```

class YahtzeeBoard

```
class YahtzeeBoard:
```

```
    ...
    def initInterface(self): #Yahtzee 보드 윈도우 생성
        ...
        for i in range(self.TOTAL + 2): # i행 : 점수
            Label(self.window, text=Configuration.configs[i], font=self.TempFont).grid(row=i, column=1)
            for j in range(self.numPlayers): # j열 : 플레이어
                if (i == 0): # 플레이어 이름 표시
                    Label(self.window, text=self.players[j].toString(), font=self.TempFont).grid(
                        row=i, column=2 + j)
                else:
                    if (j==0): #각 행마다 한번씩 리스트 추가, 다중 플레이어 지원
                        self.fields.append(list())
                    #i-1행에 플레이어 개수 만큼 버튼 추가하고 이벤트 Listener 설정, 매개변수 설정
                    self.fields[i-1].append(Button(self.window, text="", font=self.TempFont, width=8,
                        command=lambda row=i-1: self.categoryListener(row)))
                    self.fields[i-1][j].grid(row=i, column=2 + j)
                    # 누를 필요없는 버튼은 disable 시킴
                    if (j != self.player or (i-1) == self.UPPERTOTAL or (i-1) == self.UPPERBONUS
                        or (i-1) == self.LOWERTOTAL or (i-1) == self.TOTAL):
                        self.fields[i-1][j]['state'] = 'disabled'
                        self.fields[i-1][j]['bg'] = 'light gray'
                    #상태 메시지 출력
                    self.bottomLabel=Label(self.window, text=self.players[self.player].toString()+
                        "차례: Roll Dice 버튼을 누르세요", width=35, font=self.TempFont)
                    self.bottomLabel.grid(row=self.TOTAL + 2, column=0)
                    self.window.mainloop()
```

class YahtzeeBoard

```
class YahtzeeBoard:
```

```
    . . .  
    def rollDiceListener(self): #rollDiceListener  
        for i in range(5):  
            if (self.diceButtons[i]['state']!='disabled'):  
                self.dice[i].rollDie()  
                self.diceButtons[i].configure(text=str(self.dice[i].getRoll()))  
  
            if (self.roll == 0 or self.roll ==1):  
                self.roll += 1  
                self.rollDice.configure(text="Roll Again")  
                self.bottomLabel.configure(text="보관할 주사위 선택 후 Roll Again")  
            elif (self.roll==2):  
                self.bottomLabel.configure(text="카테고리를 선택하세요")  
                self.rollDice['state'] = 'disabled'  
                self.rollDice['bg'] = 'light gray'  
  
    def diceListener(self, row): #DiceListener  
        self.diceButtons[row]['state'] = 'disabled'  
        self.diceButtons[row]['bg'] = 'light gray'
```

class YahtzeeBoard

```
def categoryListener(self, row): #categoryListener
```

```
    score = Configuration.score(row, self.dice) #점수 계산
```

```
    index = row
```

```
    if (row>7):
```

```
        index = row-2
```

```
    #선택한 카테고리 점수 적고 disable 시킴
```

```
    self.players[self.player].setScore(score, index)
```

```
    self.players[self.player].setAtUsed(index)
```

```
    self.fields[row][self.player].configure(text=str(score))
```

```
    self.fields[row][self.player]['state'] = 'disabled'
```

```
    self.fields[row][self.player]['bg'] = 'light gray'
```

```
    # UPPER category가 전부 사용되었으면 UpperScore, UpperBonus 계산
```

```
    if (self.players[self.player].allUpperUsed()):
```

```
        self.fields[self.UPPERTOTAL][self.player].configure(text =  
            str(self.players[self.player].getUpperScore()))
```

```
        if (self.players[self.player].getUpperScore() > 63):
```

```
            self.fields[self.UPPERBONUS][self.player].configure(text="35") #UPPERBONUS=7
```

```
        else:
```

```
            self.fields[self.UPPERBONUS][self.player].configure(text="0") #UPPERBONUS=7
```

class YahtzeeBoard

```
def categoryListener(self, row): #categoryListener
```

```
    . . .
```

```
    # LOWER category 전부 사용되었으면 LowerScore 계산
```

```
    if (self.players[self.player].allLowerUsed()):
```

```
        . . .
```

```
    # UPPER category와 LOWER category가 전부 사용되었으면 TOTAL 계산
```

```
    if (self.players[self.player].allUpperUsed() and self.players[self.player].allLowerUsed()):
```

```
        . . .
```

```
    #다음 플레이어로 넘어가고 선택할 수 없는 카테고리들은 disable 시킴
```

```
    self.player = (self.player + 1) % self.numPlayers
```

```
    for i in range(self.TOTAL+1):
```

```
        for j in range(self.numPlayers):
```

```
            . . .
```


class YahtzeeBoard

```
def categoryListener(self,row): #categoryListener
```

```
...
```

```
# 라운드 증가 시키고 종료 검사
```

```
if (self.player == 0):
```

```
    self.round += 1
```

```
if (self.round == 13):
```

```
    ...
```

```
#다시 Roll Dice 과 diceButtons 버튼 활성화, bottomLabel 초기화
```

```
...
```

class Configuration

```
from dice import *
```

```
class Configuration:
```

```
    configs = ["Category", "Ones", "Twos", "Threes", "Fours", "Fives", "Sixes",  
              "Upper Scores", "Upper Bonus(35)", "Three of a kind", "Four of a kind", "Full House(25)",  
              "Small Straight(30)", "Large Straight(40)", "Yahtzee(50)", "Chance", "Lower Scores", "Total"]
```

```
def getConfigs():    # 정적 메소드: 객체생성 없이 사용 가능  
    return Configuration.configs
```

```
def score(row, d):    # 정적 메소드: 객체생성 없이 사용 가능  
    # row에 따라 주사위 점수를 계산 반환. 예를 들어, row가 0이면 "Ones"가 채점되어야 함을  
    # 의미합니다. row가 2이면, "Threes"가 득점되어야 함을 의미합니다. row가 득점 (scored)하지  
    # 않아야 하는 버튼 (즉, UpperScore, UpperBonus, LowerScore, Total 등)을 나타내는 경우  
    # -1을 반환합니다.
```

```
    if (row>=0 and row<=6):  
        return Configuration.scoreUpper(d, row+1)  
    elif (row==8):  
        . . .
```

class Configuration

```
class Configuration:
```

```
    . . .
```

```
def scoreUpper(d, num): # 정적 메소드: 객체생성 없이 사용 가능
    #Upper Section 구성 (Ones, Twos, Threes, ...)에 대해 주사위 점수를 매깁니다. 예를 들어,
    # num이 10이면 "Ones"구성의 주사위 점수를 반환합니다.
```

```
    . . .
```

```
def scoreThreeOfAKind(d):
```

```
    . . .
```

```
def scoreFourOfAKind(d):
```

```
    . . .
```

```
def scoreFullHouse(d):
```

```
    . . .
```

```
def scoreSmallStraight(d):
```

```
    #1 2 3 4 혹은 2 3 4 5 혹은 3 4 5 6 검사
```

```
    #1 2 2 3 4, 1 2 3 4 6, 1 3 4 5 6, 2 3 4 4 5
```

```
    . . .
```

```
def scoreLargeStraight(d):
```

```
    # 1 2 3 4 5 혹은 2 3 4 5 6 검사
```

```
    . . .
```

```
def scoreYahtzee(d):
```

```
    . . .
```

```
def sumDie(d):
```

```
    . . .
```