SW Project2
AD Project Report

-목차-

20191643 이아연
20191644 이연주

Ⅰ. 요구사항 수집 및 분석

-(1) Player

- 기능적 요구사항
  - Main 클래스에서 받은 Player의 데이터를 표현할 수 있어야 한다.
  - Player 이름을 통해 Player를 선택한 후, 현재 창을 닫고 Main 클래스를 띄워야 한다.
  - Player 닉네임을 정해서 새로운 Player를 생성한 후, 현재 창을 닫고 Main 클래스를 띄워야 한다.
  - Close 버튼을 누르면 현재 창을 닫아야 한다.

- 게임 로직의 기능적 요구사항
  - Main에서 전달받은 자료를 통해 Player의 정보를 문자열로 정리해야 한다.
    - “key” +“ ” + “value” 의 형식으로 계속 문자열을 추가해야 한다.
    - Player 리스트(Text Edit창)를 Player리스트의 문자열을 setText해줘야 한다.
  - Player를 선택할 때 없는 Player를 선택해선 안된다.
  - Player를 생성할 때 빈 문자열으로 생성하거나 빈문자열이 들어와선 안된다.

- 인터페이스 요구사항
  - ‘Choose Player’라는 Title을 정해준다.
  - Text Edit을 통해 Player 리스트를 보여준다.
  - Player를 선택하거나, 생성하는 LineEdit을 만들어 입력할 수 있도록 한다.
  - ‘Choose/ New Player’와 같은 버튼을 통해 LineEdit 의 텍스트로 게임을 진행하도록 한다.

- 비기능적 요구사항
  - Python과 PyQt5를 이용한다.

-(2) Main

- 기능적 요구사항
  - 현재 본인의 코인이 정확하게 표기되어야 한다.
  - 코인을 걸게 되면 건만큼 차감하여 표기해야 한다.
  - 각 버튼을 클릭할 시 각각 새로운 창이 뜨도록 해야 한다.
    - GameRule을 띄우는 창
    - 게임을 시작하는 창 : 코인을 거는 창을 열고 그 코인으로 게임을 진행한다.
    - 광고를 시청하는 창
  - Player 닉네임을 정해서 새로운 Player를 생성한 후, 현재 창을 닫고 Main 클래스를 띄워야 한다.
  - Close 버튼을 누르면 현재 창을 닫아야 한다.

- 게임 로직의 기능적 요구사항
  - Main 클래스에서 코인을 거는 클래스의 Coin을 참조할 수 있어야 한다.
  - 각각 버튼에 따른 callback함수를 구현하고 새로운 클래스 파일과 연결시켜야 한다.
    - GameRule의 callback : GameRule에 대해 설명하는 창을 띄운다.
    - 게임을 시작하는 버튼의 callback:
      1) 코인을 건 양만큼 값을 Main클래스에 반환할 수 있어야 한다.
      2) 코인을 건 양과 보유한 코인을 비교하여 처리할 수 있어야 한다.
      3) 게임창으로 넘어갈 수 있어야 한다.
    - 광고를 시청하는 버튼의 callback:
      1) 유튜브의 동영상을 가져올 수 있어야 한다.
      2) 동영상을 재생하고 중단할 수 없이 끝까지 재생되도록 한다.
      3) 일정한 코인을 얻을 수 있도록 해야한다.
  - Player를 선택할 때 없는 Player를 선택해선 안된다.
  - Player를 생성할 때 빈 문자열으로 생성하거나 빈문자열이 들어와선 안된다.

- 인터페이스 요구사항
  - 'Main'이라는 Title을 정해준다.
  - 새로운 창을 여는 버튼을 각각 생성해 준다.
  - 코인이미지와 코인이 얼마나 남았는지 알려주는 Label을 표시해준다.
  - 게임제목을 보여주도록 디자인한다.(Blackjack Game)

- 비기능적 요구사항
  - Python과 PyQt5를 이용한다.
  - 각 버튼을 눌렀을 때 실행되는 클래스들을 import한다.
  - Pytube, OpenCV를 이용하여 동영상을 다운받고 띄울 수 있도록 한다.

-(3) Game

- 기능적 요구사항
  - 첫 번째 카드를 제외한 모든 카드는 1 ~ 9중에 랜덤으로 선택되어야 한다.
  - 첫 번째 카드는 1 ~ 11 중에 랜덤으로 선택되어야 한다.
  - 플레이어와 딜러의 카드의 합을 저장해두어야 한다.
  - 딜러의 첫 번째 카드는 뒤집혀져 (숫자를 알 수 없게) 있어야 한다.
  - hit 버튼을 누르면 플레이어가 카드를 하나 내야한다.
  - stand 버튼을 누르면 딜러가 카드를 내야한다.
  - stand 버튼을 누르면 더 이상 hit 버튼을 누를 수 없어야 한다.
  - 딜러가 카드를 다 내게 되면 게임이 끝나야 하며, 더 이상 stand버튼을 누르지 못한다.

- 게임 로직의 기능적 요구사항
  - 게임의 결과의 요소들을 게임 결과의 창을 보여주는 클래스에 넘겨야 한다.
  - 딜러의 카드를 낼 때 조건이 다르다.
    1) 합이 16이하라면 무조건 카드를 더 내야 한다.
    2) 합이 17이상이라면 무조건 카드를 더 내지 말아야 한다.
  - 게임이 끝나는 조건의 분기를 나눠야 한다.
    1) 딜러나 플레이어 중 한 사람이라도 21을 넘으면 게임이 끝나야 한다.
    2) 딜러나 플레이어 중 한 사람이라도 21이 된다면 게임이 끝나야 한다.
    3) 딜러와 플레이어가 모두 21미만이라면 더 21에 가까운 사람이 승리한다.

- 인터페이스 요구사항
  - ‘Blackjack Game'이라는 Title을 정해준다.
  - 배경색은 카지노의 테이블과 비슷한 초록색으로 한다.
  - 카드는 일정한 간격으로 둔다.
  - 카드합을 표시해주어야 한다.
  - 배팅한 코인을 표시해주어야 한다,

- 비기능적 요구사항
  - Python과 PyQt5를 이용한다.
  - random을 이용한다.

-(4) Result

- 기능적 요구사항
  - 배팅한 코인을 받아서 돌려주는 코인을 변경해야 한다.
    - 플레이어가 졌을 때
      - 이미 코인을 배팅하면서 차감된 상태
      - 배팅한 코인의 -0.5배를 돌려주어야 한다.
    - 플레이어가 이겼을 때
      - 배팅한 코인의 +1.5배를 돌려주어야 한다.
    - 비겼을 때
      - 이미 코인을 배팅하면서 차감된 상태
      - 배팅한 코인 그대로 돌려주어야 한다.
  - exit 기능을 구현해준다.

- 기능적 로직의 요구사항
  - Game클래스로부터 승리(혹은 패배) 결과와 배팅한 코인을 받아온다.
  - 각 승리(혹은 패배) 조건에 따른 다양한 결과를 낸다.
  - exit을 하면 Result 뿐만이 아니라 Game의 창도 꺼져야 한다.

- 인터페이스 요구사항
  - 'Result'라는 Title을 정해준다.
  - 승리(혹은 패배)의 종류에 따른 다른 텍스트를 보여준다.
  - 차감된 코인의 결과를 보여준다.

- 비기능적 요구사항
  - Python과 PyQt5를 이용한다.

Ⅱ. 소프트웨어 구조 설계

모듈	클래스	역할	메서드	입력	출력	기능
player.py	Player (QWidget)	Player의 정보를 보여주고 Main에 전달한다.	__init__	dic	-	1)인자값 할당 2)initUI()호출
			initUI	-	-	1)“dic”을 이용하여 TextEdit에 다중문자열 추가 2)위젯들을 추가하고 레이아웃 정리
			end_window	-	-	현재 윈도우 창 닫기
main.py	Main (QWidget)	새로운 창을 열도록 버튼을 생성하고 코인 비교한다.	__init__	-	-	1)Player파일을 열고(self.p),정보를 담을 딕셔너리(self.playDic)를 생성한다. 2)자신의 coin값을 변수(self.text)에 할당하여준다. 3)Player를 선택하는 함수(who)호출 3) Player이름을 담을 self.nowPlay라는 변수를 생성한다.
			who	-	-	1)Player파일을 split하여 딕셔너리에 담아준다, (key:player name value:coin) 2) Player 클래스를 호출해준다. 3)각 Player 클래스의 버튼을 클릭했을 경우를 처리해준다.
			choose Player	-	-	1)Player를 선택하였을 때 처리를 해주는 함수 2)Player의 딕셔너리에 값이 없으면 에러를 처리한다. 3)Player의 딕셔너리에 값이 있으면 해당 key의 value값을 coin값으로

						할당하여 준다,
			newPlayer	-	-	1)새로운 Player를 생성할 때 처리하는함수 2)Player명이 비거나 공백이 있을경우의 에러처리 3)이미 Player 리스트에 있는 이름일 경우의 에러처리 4)coin을 디폴트값(=100)으로 지정하여주고, self.nowPlay에 Player 닉네임을 할당하여준다.
			initUI	-	-	1)PyQt를 이용하여 레이아웃 설정 2)각 버튼에 대한 callback함수와 connect
			howtoplay	-	-	GameRule에 대한 창(pixmap)을 연다.
			next	-	-	1)코인을 거는 창을 연다. 2)코인을 거는 창에서 'Click'버튼이 눌렸을 때의 처리를 해준다.(callback함수와 연결)
			next_game	-	-	1)코인을 차감하기 위한 button 'Click'에 대한 callback함수 2)값을 입력하지 않거나 숫자가 아닌경우의 에러처리 3)현재 가지고있는 코인이 건 코인보다 적을 때의 에러 처리 4)코인을 건 만큼 자신의 코인을 차감하고 화면에 보이는 코인(self.cLabel)을 수정 5)'BlackjackGame'화면으로 넘어가기 위한 객체 생성 6)'BlackjackGame'에서 'Exit'버튼이 눌렸을 때의 처리를 해준다.

						(callback함수와 연결)
			exitClicked	-	-	1)'BlackjackGame'결과에 따른 코인을 처리해 주기 위한 'Exit'버튼에 대한 callback함수 2)getter메소드를 이용하여 'BlackjackGame'에서 처리된 코인을 가져와 현재 코인에 처리한다. 3)화면에 보이는 코인(self.cLabel)을 수정한다. 4)'BlackjackGame'창을 종료한다.
			coin	-	-	1)광고를 통한 코인을 얻기 위해 'CoinMain()' 객체를 생성한다 2)광고가 종료된 후 plus할 coin을 가져와 현재 코인을 수정한다. 3)화면에 보이는 코인(self.cLabel)을 수정한다.
			end	-	-	1)파일을 "w"상태로 열어 입력하여 저장한다. 2)playDic에 저장된 내용을 문자열로 정리하여 파일에 쓴다. 3)프로그램의 eventloop을 종료한다.
nextMain.py	NextMain (QWidget)	현재 보유한 코인을 보여주며, 코인을 얼마나 걸 것인지 입력하게 한다.	__init__	text	-	1)보유한 코인을 입력받아 화면에 띄운다. 2)경고문을 띄울 라벨(self.warn)을 생성한다. 3)initUI()를 호출한다.
			initUI	-	-	1)PyQt5를 이용하여 레이아웃을 설정한다. 2)코인을 입력받는 창(QLineEdit :self.coinLine)을 통해 걸 코인을 입력받는다.
			end_game	-	-	main화면에서 callback호출을 하면 창을 닫는다.
coinMain.py	CoinMain		__init__	-	-	downloadV()를

						호출한다.
	(Qwidget)	광고를 다운받고 그 광고를 화면에 띄운후 코인을 얻도록 한다.	downloadV	-	-	1)pytube모듈을 이용하여 유튜브링크를 통해 동영상을 다운받는다. (화질, 동영상크기, 확장자 선택) 2)디렉토리를 설정한 후 video_true호출
			video_true	-	-	1)opencv를 이용하여 영상을 frame을 연속적으로 보여주어 화면에 띄운다. 2)영상이 끝나면 창을 닫는다.
how toplay.py	HowToPlay (Qwidget)	GameRule에 관련한 창을 띄운다.	__init__	-	-	initUI()를 호출한다.
			initUI	-	-	1)PyQt5를 이용하여 레이아웃을 설정하여준다. 2)GameRule에 대해서는 Pixmap을 이용하여 편집한 사진을 띄운다.
			end_window	-	-	main에서 받은 'Close'버튼을 클릭했을 때에 대한 callback을 받으면 창을 종료한다.
blackjack Game.py	BlackjackGame (Qwidget)	Blackjack게임의 기능과 인터페이스를 구현하고 결과를 GameFinished 클래스에 보내준다.	__init__	coin	-	1) 윈도우 창의 인터페이스 구현 2) 카드 사이의 간격을 self.interval에 저장한다. 3) 플레이어와 딜러의 전 카드의 x좌표를 self.u_previous, self.d_previous라는 변수에 저장해준다. 4) 딜러와 플레이어의 카드 사전(카드 사진의 이름, 카드값), 카드값 리스트, 현재까지 나온 카드값들을 담을 리스트, 현재까지 카드값의 합을 0으로 하는 self.u_point, self.d_point변수를 만든다.

						5) 인자값(배팅받은 코인)을 변수에 할당해준다. 6) 배팅된 코인값을 보여준다. 7) 딜러의 첫 번째 카드와 두 번째 카드를 랜덤으로 뽑는다. 9) 딜러의 두 번째 카드의 값을 self.d_point에 더해주고, 리스트에도 추가해준다. 8) 딜러 카드의 합을 self.dealer_sum에 저장한다. 9) QPixmap을 통해 뽑은 카드의 이미지를 보여준다. 10) showPoint를 호출한다. 11) self.d_previous에 self.interval을 더해서 갱신해준다. 12) startGame을 호출한다. 13) self.gamefinish라는 GameFinised 클래스의 객체를 만든다.
			startGame	-	-	1) 플레이어의 첫 번째 카드를 보여준다. 2) 플레이어의 두 번째 카드를 보여준다. 3) hit 버튼과 stand 버튼을 구현해준다. 4) hit 버튼이 눌렸을 때는 PushHit함수에, stand 버튼이 눌렸을 때는 PlayDealer함수에 연결되도록 해준다.
			pushHit	-	-	1) 플레이어의 카드를 랜덤으로 뽑는다. 2) 카드값을 현재까지 나온 카드값 리스트에 추가하고, self.u_point에

						더해준다. 3) QPixmap을 사용해서 카드의 이미지를 보여준다. 4) self.u_prevoius에 self.interval을 더해서 갱신시켜준다. 5) 플레이어 카드의 합을 self.user_sum에 저장한다. 6) self.user_sum이 21과 같거나 그보다 많으면 hit을 더 이상 누르지 못하게 한다. 7) showPoint를 호출한다. 8) userfinised를 호출한다.
			playDealer	-	-	1) hit 버튼을 누르지 못하게 한다. 2) 딜러의 첫 번째 카드를 랜덤으로 뽑아주고 이미지를 보여준다. 3) self.d_point와 self.dealer_sum갱신 4) 첫 번째 카드를 보여준 후 showPoint와 dealerFinished를 호출 5) self.dealer_sum이 16보다 작을때동안 카드를 계속 뽑으며 위와 동일한 변수들을 계속해서 갱신한다. 6) self.d_previous에 self.interval을 더해서 갱신해준다. 7) while문이 끝나면 dealerFinished 호출
			showPoint	-	-	플레이어와 딜러의 카드값의 합을 보여준다.
			userFinished	-	-	1) self.gamefinish를 GameFinised에 플레이어의 승리(혹은 패배) 조건의 내용과,

						배팅한 코인을 인자로 한 객체로 갱신시켜준다. - self.user_sum이 21보다 클 경우 - self.user_sum이 21과 같을 경우
			dealer Finished	-	-	1) 더 이상 stand 버튼이 눌리지 않게 한다. 2) self.gamefinish를 GameFinised에 딜러의 승리(혹은 패배) 조건의 내용과, 배팅한 코인을 인자로 한 객체로 갱신시켜준다. - self.dealer_sum과 self.user_sum이 같을 경우 - self.dealer_sum이 21보다 클 경우 - self.dealer_sum이 21과 같을 경우 - self.dealer_sum이 (21 미만) self.user_sum보다 클 경우 - self.dealer_sum이 (21 미만) self.user_sum보다 작을 경우
			exit	-	-	해당 윈도우를 지운다.
	Game Finished (Qwidget)		__init__	-	-	1) 윈도우 창의 인터페이스 구현 2) self.coin과 self.result_coin 변수를 0을 초기값으로 생성 3) 결과값들을 보여줄 객체 생성 4) exit 버튼 생성 5) 레이아웃을 만들고 각 객체를 레이아웃에 추가 6) self.result를 초기값을 ‘’로 변수 생성
			setResultCoin	result, coin	-	1) 인자값들을 self.result와

						self.coin에 할당 2) gameFinished 호출
			gameFinished	-	-	1) self.result_coin을 0으로 초기화 2) 각 self.result에 따라 윈도우 창을 보여줌 - 플레이어 승리시) self.result_coin에 self.coin * 1.5를 더함 - 플레이어 패배시) self.result_coin에 self.coin * 0.5를 뺌
			get Result_Coin	-	self.result_coin	self.result_coin을 반환
			exit	-	-	해당 위도우를 지운다.

Ⅲ. 구현(소스코드)

-player.py

```
from PyQt5.QtWidgets import QWidget
from PyQt5.QtWidgets import QGridLayout
from PyQt5.QtWidgets import QPushButton, QTextEdit, QLineEdit, QLabel
from PyQt5.QtGui import QPalette, QColor
```

먼저 PyQt5의 QtWidgets에서 레이아웃에 필요한 'QWidget, QgridLayout, QpushButton, QTextEdit, QLineEdit, QLabel'을 호출하였다.
그리고 PyQt5의 QtGui에서 'QPalette, QColor'를 호출하였다. 이 모듈을 통해 GUI의 배경을 검정색으로 채우도록 하였다.

```
class Player(QWidget):
    def __init__(self, dic):
        super().__init__()
        self.play = dic

        pal = QPalette()
        pal.setColor(QPalette.Background, QColor(0, 0, 0))
        self.setPalette(pal)

        self.initUI()
```

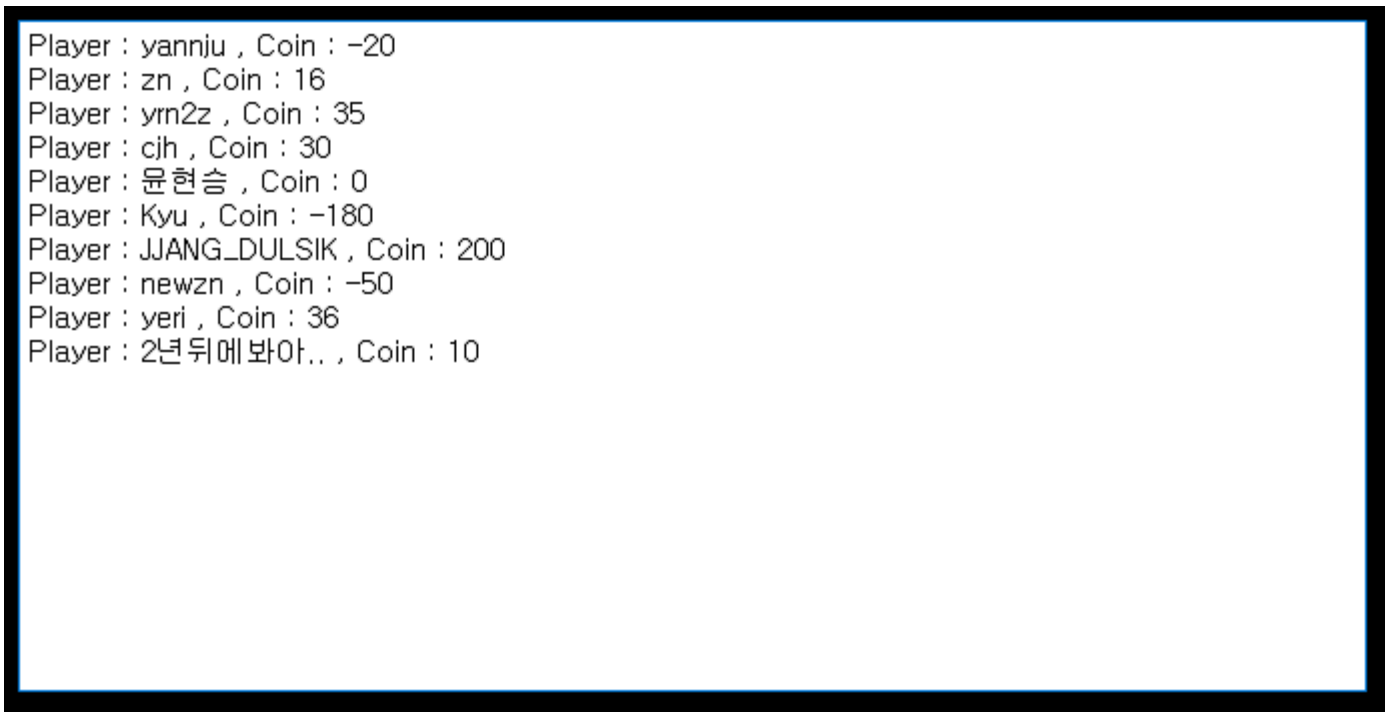
클래스의 생성자부분에는 인자로 받은 dic을 self.play라는 변수에 할당하여주었다.
dic은 Main클래스에서 파일('player.txt')을 읽어들여 Player들의 정보를(key에는 Player명을, value에는 coin을) 딕셔너리로 만든 것으로, self.play는 결국 Player들의 정보를 가진 딕셔너리를 의미한다.
그 후, 배경을 검정색으로 채워주기 위해 pal이라는 변수에 QPalette()를 할당하여 준 후, 배경을 (0, 0, 0)으로 set해 주었다.

```
    def initUI(self):
        self.list = QTextEdit()

        if len(self.play) != 0:
            self.playTxt = ''''''
            for x in self.play:
                self.playTxt += "Player : " + x + " , Coin : " + str(self.play[x]) + "\n"
            self.list.setText(self.playTxt)

        else:
            pass
```

호출받은 initUI()에선, Player들의 리스트를 화면에 표시하도록 하기 위한 레이아웃을 설정하여주었다.
먼저 self.list라는 변수에 QTextEdit()을 만들어 주었다.
딕셔너리가 비어있지 않다면 self.playTxt라는 변수에 지정한 다중문자열(‘‘‘ ’’’)을 사용하여, 딕셔너리의 key값과 value값을 표기하도록 하였다.
for loop를 마치고 나면 self.playTxt를 QTextEdit의 텍스트로 지정하였으며 실제로 보면 다음과 같은 것을 확인할 수 있다.



다중문자열로 여러 문자가 들어가였으며, 형식을 잘 맞추어서 들어간 것을 확인할 수 있다.

```
29          self.list.setReadOnly(True)
30
31          self.choose = QLineEdit()
32          self.chBtn = QPushButton("choose")
33
34          self.newPlay = QLineEdit()
35          self.newStart = QPushButton("new Play")
36
37          self.warn = QLabel("")
38
39          self.endBtn = QPushButton("CLOSE")
```

QTextEdit인 self.list는 오직 읽기만 가능하도록 설정하여, 화면에서 입력할 수 없도록 하였다.
그 후 화면에 보여질 버튼들과 라벨들을 만들어 주었다.
Player 리스트에 있는 Player를 입력하고 선택하는 QLineEdit, QPushButton과
새로운 Player를 생성하기 위해 Player명을 입력하는 QLineEdit과 QpushButton을 생성하였다.

self.warn은 여러 에러사항이 발생하였을 때 경고문을 띄워주는 라벨로, 초기값은 빈문자열을 주었다.
현재까지의 에러사항은 다음과 같다.
- 존재하지 않는 Player를 선택하였을 때
- 새로운 Player를 생성할 때 빈문자열로 생성하였을 때
- 새로운 Player가 이미 존재하는 Player와 이름이 같을 때
- Player의 이름에 공백이 포함되는 경우 파일을 저장하거나 불러올 때 에러가 생기기 때문에, 공백이 있을 때

위와 같은 경우에 따른 에러처리는 Main클래스에서 처리하였다.
마지막으로 창을 닫는 버튼 또한 생성하여 주었다.

```
41          self.grid = QGridLayout()
42
43          self.grid.addWidget(self.list, 0, 0, 13, 2)
44          self.grid.addWidget(self.warn, 13, 0, 1, 2)
45          self.grid.addWidget(self.choose, 14, 0, 1, 1)
46          self.grid.addWidget(self.chBtn, 14, 1, 1, 1)
47          self.grid.addWidget(self.newPlay, 15, 0, 1, 1)
48          self.grid.addWidget(self.newStart, 15, 1, 1, 1)
49          self.grid.addWidget(self.endBtn, 16, 0, 1, 2)
50
51          self.setLayout(self.grid)
52          self.setWindowTitle("Choose Player!")
53          self.setGeometry(700, 250, 600, 500)
54
55      def end_window(self):
56          self.deleteLater()
```

QGridLayout을 이용하여, 위의 모든 위젯들을 추가하여 주었고, 위치와 타이틀을 지정하여 주었다.
end_window 메소드는 Main클래스에서 callback호출을 받게되면 창을 닫도록 하였다.

-main.py

```
import sys

from PyQt5.QtWidgets import QApplication, QWidget
from PyQt5.QtWidgets import QGridLayout, QHBoxLayout
from PyQt5.QtWidgets import QPushButton, QLabel
from PyQt5.QtGui import QPixmap, QPalette, QColor
from PyQt5.QtCore import *

from player import Player
from howtoplay import HowToPlay
from nextMain import NextMain
from coinMain import CoinMain
from bustGame import BustGame, GameFinished
```

Main클래스에서 호출한 모듈들은 위와 같다.
EventLoop을 위한 'sys, QApplication' 과
레이아웃을 위한 'Qwidget, QGridLayout, QGHBoxLayout, QPushButton, QLabel, QPixmap, QPalette, QColor'을 호출해 주었다.
또한 프로그램을 버튼을 통해 종료하도록 하기 위해 QtCore도 호출해 주었다.

아래의 'Player, HowToPlay, NextMain, CoinMain, BlackjackGame, GameFinished'는 게임과 연결된 파일들이다.

```
class Main(QWidget):
    def __init__(self):
        super().__init__()
        self.text = 0
        self.p = open("player.txt", "r")
        self.s = self.p.readlines()
        self.playerDic = dict()

        self.nowPlay = ""
        self.who() #player선택

        pal = QPalette()
        pal.setColor(QPalette.Background, QColor(0, 0, 0))
        self.setPalette(pal)
```

생성자는 위와 같다. 먼저 코인을 담을 변수를 생성하고 파일을 열어 한줄씩 읽어들이도록 하였다.

읽어들인 파일의 정보를 담을 딕셔너리를 생성하였다.
self.nowPlay라는 변수에 현재 Player명을 담도록 하기 위해서 생성하여 주었다.
그 후 Player를 선택하는 함수를 호출하여 주었고, Main화면 또한 검정색을 배경을 하도록 처리하였다.

```
32      def who(self):
33          for player_str in self.s:
34              self.l = player_str.split()
35              self.playerDic[self.l[0]] = int(self.l[1])
36
37          self.playWindow = Player(self.playerDic)
38          self.playWindow.show()
39
40          self.playWindow.chBtn.clicked.connect(self.choosePlayer)
41          self.playWindow.newStart.clicked.connect(self.newPlayer)
42          self.playWindow.endBtn.clicked.connect(self.end)
```

플레이어를 선택하기 위해 호출이 되면, 먼저 파일을 한줄씩 읽어들인 문자열에 대한 for문을 돌린다.
저장된 Player의 정보가 'Player명 Coin'와 같은 형식으로 되어있으므로, 공백을 기준으로 하여 나누어 준 리스트를 통해 딕셔너리에 추가하여준다.

그 후 Player를 선택하는 창을 띄우도록 하며, 버튼을 클릭하였을 때의 callback함수도 지정하여 주었다.

```
45      def choosePlayer(self):
46          if self.playWindow.choose.text() not in self.playerDic:
47              self.playWindow.warn.setStyleSheet('color:red')
48              self.playWindow.warn.setText("플레이어를 찾을 수 없습니다!")
49              self.playWindow.choose.setText("")
50
51          else:
52              self.nowPlay = self.playWindow.choose.text()
53              self.text = int(self.playerDic[self.playWindow.choose.text()])
54              self.playWindow.end_window()
55              self.initUI()
```

Player를 선택하였을 경우는 위와 같다. 먼저 입력받은 Player명이 playerDic에 없는 경우는 존재하지 않는 Player라는 말과 같으므로 위에서 만들어준 self.warn이라는 라벨에 경고가 뜨도록 처리하였고, 입력창을 비워주었다.
만약 존재하는 경우에는 self.nowPlay에 그 Player명을 할당하여주고, coin도 보유한 coin으로

시작할 수 있도록 현재 보유한 코인을 처리하여주었다.
그다음 Player선택창을 닫고 initUI()를 호출하여준다.

```
57      def newPlayer(self):
58          if self.playWindow.newPlay.text() == "" or " " in self.playWindow.newPlay.text():
59              self.playWindow.warn.setStyleSheet('color:red')
60              self.playWindow.warn.setText("플레이어 이름을 비우거나 공백을 포함시킬 수 없습니다!")
61              self.playWindow.newPlay.setText("")
62          elif self.playWindow.newPlay.text() in self.playerDic:
63              self.playWindow.warn.setStyleSheet('color:red')
64              self.playWindow.warn.setText("이미 존재하는 플레이어 입니다!")
65              self.playWindow.newPlay.setText("")
66          else:
67              self.playerDic[self.playWindow.newPlay.text()] = 100
68              self.nowPlay = self.playWindow.newPlay.text()
69              self.text = 100
70              self.playWindow.end_window()
71              self.initUI()
72
```

Player를 새로 생성할 때의 코드는 위와 같다.
만약 Player명을 입력하는 칸이 비었거나, 공백을 포함한다면 self.warn에 경고문을 띄운다.
존재하는 Player명을 설정한 경우에도 경고문을 띄우게 되며,
위와 같은 상황이 아닌 경우에는 Coin을 디폴트값(=100)으로 설정하여 주고 self.nowPlay를 새로 생성한 Player명으로 설정하여 준다.
그다음 Player설정창을 닫고 initUI()를 호출한다.

```
75      def initUI(self):
76          #Button
77          self.howtoplayButton = QPushButton('How to Play the Game?')
78          self.startButton = QPushButton('Start Game!')
79          self.pluscoinButton = QPushButton('see adv to get COIN!')
80          self.endButton = QPushButton('End Game')
```

레이아웃을 지정하는 메소드로, 메인화면에 띄울 버튼들을 생성하여 주었다.
- GameRule에 관련한 버튼
- 게임을 시작하는 창을 여는 버튼
- 광고를 통해 coin을 얻는 버튼
- 게임을 종료하고 현재 상태를 저장하는 버튼

```
82          #gamename image
83          self.gnimgLabel = QLabel()
84          self.setMinimumHeight(95)
85          self.setMinimumWidth(200)
86          pm = QPixmap('gamename.png')
87          pm = pm.scaledToHeight(95)
88          self.gnimgLabel.setPixmap(pm)
```

먼저 게임타이틀을 띄워주기 위해 Pixmap을 이용하여 사진의 크기와, 사진을 지정하여 레이아웃에 띄워주었다.

```
90          #coin image
91          self.cimgLabel = QLabel()
92          self.setMinimumHeight(95)
93          self.setMinimumWidth(95)
94          pixmap = QPixmap('coin.png')
95          pixmap = pixmap.scaledToHeight(75)
96          self.cimgLabel.setPixmap(pixmap)
97
98          #coin text
99          self.cLabel = QLabel()
100         font = self.cLabel.font()
101         font.setPointSize(font.pointSize() + 10)
102         self.cLabel.setFont(font)
103         self.cLabel.setStyleSheet('color:white')
104         self.cLabel.setText("X {} ".format(self.text))
```

코인또한 이미지와 현재 보유한 코인을 나타내기 위해 위와 같이 표현하였다.
코인에 관련한 이미지를 띄우고, Label을 통해 현재 보유한코인(self.text)를 보여주도록 하였다.

```
106         #main : grid / coint : coinBox/ button : buttonGrid
107         self.grid = QGridLayout()
108         self.coinBox = QHBoxLayout()
109
110         self.coinBox.addWidget(self.gnimgLabel)
111         self.coinBox.addStretch(1)
112         self.coinBox.addWidget(self.cimgLabel)
113         self.coinBox.addWidget(self.cLabel)
114
115         self.buttonGrid = QGridLayout()
116
117         self.grid.addLayout(self.coinBox, 0, 1)
118         self.grid.addLayout(self.buttonGrid, 1, 0, 4, 0)
119
120         self.buttonGrid.addWidget(self.howtoplayButton, 0, 0, 2, 2)
121         self.buttonGrid.addWidget(self.startButton, 1, 0, 2, 2)
122         self.buttonGrid.addWidget(self.pluscoinButton, 2, 0, 2, 2)
123         self.buttonGrid.addWidget(self.endButton, 3, 0, 2, 2)
```

GridLayout을 통해 coin을 보여주는 위젯들과 버튼들의 위치를 지정해 주었다.

coin을 보여주는 label들과 이미지, 게임타이틀은 coinBox라는 HBoxLayout에,
Main화면에 버튼들은 ButtonGrid라는 GridLayout에 넣어주고 위치를 지정해 준 다음,
두 레이아웃을 grid에 넣어주었다.

```
125         #button을 눌렀을 때
126         self.howtoplayButton.clicked.connect(self.howtoplay)
127         self.startButton.clicked.connect(self.next)
128         self.pluscoinButton.clicked.connect(self.coin)
129         self.endButton.clicked.connect(self.end)
130
131         self.setLayout(self.grid)
132         self.setGeometry(500, 150, 900, 700)
133         self.setWindowTitle('Main')
134         self.show()
```

그 다음 각각 버튼을 클릭하였을 때의 callback함수를 지정하여 주고, 타이틀과, 위치를 지정하여 준 후 창을 띄우도록 하였다.

```
136 #게임설명창
137     def howtoplay(self):
138         self.playWindow = HowToPlay()
139         self.playWindow.show()
140
141 #다음게임화면으로 넘어가기 위한 새로운창
142     def next(self):
143         self.newWindow = NextMain(self.text)
144         self.newWindow.show()
145         self.newWindow.coinBtn.clicked.connect(self.next_game)
```

게임설명을 위한 버튼을 클릭하였을 때 새로운 창을 띄워주기 위해 HowToPlay()라는 클래스를 생성하게 된다.
게임창으로 넘어가기 위한 버튼 또한 같으며, 먼저 코인을 걸도록 하기 때문에, 코인을 건 후 버튼을 클릭하면 callback함수로 넘어가도록 하였다.

```
147 #코인을 차감하기 위한 button click callback함수
148     def next_game(self):
149         if self.newWindow.coinLine.text() == "":
150             self.newWindow.warn.setStyleSheet('color:red')
151             self.newWindow.warn.setText("값을 입력해 주세요!")
152             self.newWindow.coinLine.clear()
153         elif int(self.text) < int(self.newWindow.coinLine.text()):
154             self.newWindow.warn.setStyleSheet('color:red')
155             self.newWindow.warn.setText("코인이 부족합니다!")
156             self.newWindow.coinLine.clear()
```

먼저, 코인을 차감하기 위한 callback함수를 받게 되면, 여러 에러사항을 처리한 후 게임을 넘어가도록 하였다.
- 빈칸이 입력되는 경우
- 숫자가 아닌 다른 문자(ex.한글, 영어, 연산자..)가 들어온 경우
- 코인이 부족한 경우
- 코인을 음수로 걸 경우

와 같은 경우 경고문과 함께 입력창을 비우도록 처리하였다.

```
157         elif not self.newWindow.coinLine.text().isdigit():
158             self.newWindow.warn.setStyleSheet('color:red')
159             self.newWindow.warn.setText("숫자만 입력해 주세요!")
160             self.newWindow.coinLine.clear()
161         elif int(self.newWindow.coinLine.text()) < 0 :
162             self.newWindow.warn.setStyleSheet('color:red')
163             self.newWindow.warn.setText("코인이 부족합니다!")
164             self.newWindow.coinLine.clear()
165         else:
166             self.text -= int(self.newWindow.coinLine.text())
167             self.newWindow.end_game()
168             self.cLabel.setStyleSheet('color:white')
169             self.cLabel.setText("X {} ".format(self.text))
170 
171             #게임창으로 넘어가기
172             self.gameWindow = BustGame(self.newWindow.coinLine.text())
```

모든 에러사항에 걸리지 않는 경우 현재 보유한 코인에서 건 코인을 차감하고 그 창을 닫도록 한다.
그 후 메인에 표시된 현재 보유한 코인을 수정하고 BlackjackGame창을 건 코인을 인자로 전달하여 열도록 하였다.

```
174         #exit버튼 눌렸을때 (게임결과창에서)
175             self.gameWindow.gamefinish.exitGame.clicked.connect(self.exitClicked)
176 
177     def exitClicked(self):
178         self.text += int(self.gameWindow.gamefinish.getResult_Coin())
179         self.cLabel.setText("X {} ".format(self.text))
180         self.gameWindow.gamefinish.exit()
181         self.gameWindow.exit()
```

(현재 175번줄은 위 사진의 else문에 걸려있음)
BlackjackGame창에서 Exit버튼이 클릭되었을 때의 callback함수를 호출한다.

호출된 exitClicked메소드는 BlackjackGame에서 처리된(승패에 따른 코인처리)코인을 더하고, 화면에 표기한다.
그다음 BlackjackGame창을 모두 닫는다.

```
183     #광고 시청후 coin++
184         def coin(self):
185             self.otherWindow = CoinMain()
186             self.text += self.otherWindow.c
187             self.cLabel.setStyleSheet('color:white')
188             self.cLabel.setText("X {} ".format(self.text))
```

위는 코인이 부족하여 광고를 시청하는 버튼을 클릭하였을 때의 메소드이다.
먼저 CoinMain()클래스를 호출하고, 광고를 모두 시청한 경우 CoinMain()에서 정한 코인만큼 얻도록 하였다.
그 후 보유한 코인이 수정되었으므로 화면에 수정하여 표기한다.

```
190     #플레이어와 코인을 파일에 저장한 후 게임을 종료
191         def end(self):
192             self.p = open("player.txt", "w")
193             for each in self.playerDic:
194                 if each == self.nowPlay:
195                     self.p.write(self.nowPlay + " " + str(self.text) + "\n")
196                 else:
197                     self.p.write(each + " " + str(self.playerDic[each]) + "\n")
198             QCoreApplication.quit()
```

게임을 종료하기 위해 게임종료 버튼을 눌렀을 때의 메소드이다.
먼저 Player들의 정보를 저장해야 하므로 파일을 “w”상태로 연다.
self.playDic딕셔너리를 시퀀스로 하여 for loop을 도는데, 현재의 플레이어인 경우에는 Coin이 변경되었으므로 현재 보유한 코인으로 수정하여 적도록 하였다.
for loop이 종료되면 EventLoop을 종료하게 처리하였다.

-howtoplay.py

```
1   from PyQt5.QtWidgets import QLabel, QPushButton
2   from PyQt5.QtWidgets import QGridLayout
3   from PyQt5.QtGui import QPixmap, QPalette, QColor
4   from PyQt5.QtWidgets import QWidget
```

레이아웃을 위해 호출한 ‘QLabel, QPushButton, QGridLayout, QWidget’과, 편집된 이미지를 통해 GameRule을 띄우기 위해 ‘QPixmap’을 호출하였다.
마지막으로 배경을 검정색으로 하기 위해 ‘QPalette, QColor’ 또한 호출해 주었다.

```
6   class HowToPlay(QWidget):
7
8       def __init__(self):
9           super().__init__()
10
11          pal = QPalette()
12          pal.setColor(QPalette.Background, QColor(0, 0, 0))
13          self.setPalette(pal)
14
15          self.initUI()
```

생성자에는 검은 배경을 위한 코드와, initUI()를 호출하도록 하였다.

```
17      def initUI(self):
18          self.grid = QGridLayout()
19
20          self.playLabel = QLabel()
21          self.setMinimumHeight(404)
22          self.setMinimumWidth(650)
23          pixmap = QPixmap('gamerule.png')
24          pixmap = pixmap.scaledToHeight(404)
25          self.playLabel.setPixmap(pixmap)
26
27          self.close_Btn = QPushButton("CLOSE")
28
29          self.close_Btn.clicked.connect(self.end_window)
30
31          self.grid.addWidget(self.playLabel, 0, 0, 8, 2)
32          self.grid.addWidget(self.close_Btn, 8, 0, 1, 2)
33
34          self.setLayout(self.grid)
35          self.setWindowTitle('how To play Game?')
36          self.setGeometry(620, 170, 680, 650)
37          self.show()
```

레이아웃은 위와같이 설정하여 주었다.
GameRule이미지를 위한 Label과, Close버튼을 생성하여 준 후 Close버튼이 클릭하였을 때의 callback 또한 지정해 주었다.
GridLayout에 위젯들을 추가하여 주었으며 타이틀과 위치를 설정한 후 창을 띄워 주었다.

```
39      def end_window(self):
40          self.deleteLater()
```

종료 callback을 받았을 때 창을 종료하도록 하는 메소드이다.

-nextMain.py

```
1   from PyQt5.QtWidgets import *
2
3   class NextMain(QWidget):
4       def __init__(self, text):
5           super().__init__()
6
7           self.resultLabel = QLabel("현재 보유 코인 : {} coin ".format(text))
8           self.resultLabel.setStyleSheet('color:blue')
9           self.warn = QLabel("")
10          self.initUI()
```

레이아웃을 위해 'QtWidgets'을 호출하여 주었다.

```
12      def initUI(self):
13          self.coinLabel = QLabel('몇 코인을 거시겠습니까?')
14          self.coinLine = QLineEdit()
15          self.coinBtn = QPushButton('CLICK!')
16
17
18          vbox = QVBoxLayout()
19
20          vWidgetBox = QHBoxLayout()
21          vWidgetBoxT = QHBoxLayout()
22          vWidgetBox.addWidget(self.coinLabel)
23          vWidgetBoxT.addWidget(self.coinLine)
24          vWidgetBoxT.addWidget(self.coinBtn)
25
26          vbox.addWidget(self.resultLabel)
27          vbox.addLayout(vWidgetBox)
28          vbox.addLayout(vWidgetBoxT)
29          vbox.addWidget(self.warn)
30
31          self.setLayout(vbox)
32          self.setWindowTitle('howCoin?')
33          self.setGeometry(850, 300, 150, 150)
```

생성자에는 현재 보유한 코인을 띄우기 위한 QLabel과, 경고문을 위한 QLabel을 생성하여 주었다.
그 후 initUI()를 호출하였다.

```
35      def end_game(self):
36          self.deleteLater()
```

위의 메소드에서 띄워진 창에 대한 레이아웃은 위와같이 설정하여 주었다.
main클래스에서 Click버튼을 눌렀을 때의 callback을 받게 되면 창을 닫도록 하는 메소드이다.

-coinMain.py

```
1   from PyQt5.QtWidgets import *
2   import pytube
3   import cv2
```

광고를 저장하기 위한 'pytube'와, 화면에 동영상을 띄우기 위해 'cv2'를 호출하였다.

```
5   class CoinMain(QWidget):
6       def __init__(self):
7           super().__init__()
8           self.downloadV()
9
10      def downloadV(self):
11          pl = pytube.YouTube("https://www.youtube.com/watch?v=kALFxY9EvZM&list=PL&index=2&t=0s")
12          video = pl.streams.all()
13
14          parent_dir = "C:\\Users\\dusyn\\PycharmProjects\\SchooL\\adproject_main"
15          video[0].download(parent_dir)
16
17          self.video_true()
```

먼저 downloadV 메소드를 호출하고 pl이라는 변수에 pytube를 통해 링크를 연결하여 할당해 준다.
video는 pl의 화질, 저장할 확장자등의 정보가 담겨있으며, 원하는 디렉토리에 원하는 정보로 다운로드 되도록 해 주었다.
그 후 비디오를 띄우도록 self.video_true()를 호출해 주었다.

```
19      def video_true(self):
20          cap = cv2.VideoCapture('[국민대학교] 미래교육을 Kreate하다.mp4')
21          while True:
22              ret, frame = cap.read()
23              if ret:
24                  cv2.imshow("get CoiN!", frame)
25                  if cv2.waitKey(3) > 0:
26                      break
27              else:
28                  cap.release()
29                  cv2.destroyAllWindows()
30                  self.c = 20
31                  break
```

먼저 캡쳐할 동영상의 이름을 지정해 주고, 읽어들여 frame이 끝나기 전까지 화면을 띄워준다.
캡쳐가 끝나면 화면을 종료하고, 20코인을 주도록 할당하여준다.

-blackjackgame.py

```
from PyQt5.QtWidgets import QWidget, QLabel, QApplication, QPushButton, QVBoxLayout, QHBoxLayout, QLineEdit
from PyQt5.QtGui import QPixmap, QPalette, QColor, QFont
from PyQt5.QtCore import Qt
import random
```

필요한 모듈을 import하였다.

1. BlackjackGame class

- __init__(self, coin)

```
class BlackjackGame(QWidget):

    def __init__(self, coin):
        super().__init__()
        self.setWindowTitle("BlackJack Game")
        self.setGeometry(340, 150, 900, 400)
        self.setFixedSize(900, 400)
        pal = QPalette()
        pal.setColor(QPalette.Background, QColor(0, 100, 0))
        self.setPalette(pal)
        self.interval = 100
        self.d_previous = 50
        self.u_previous = 50
        self.d_point = 0
        self.u_point = 0
        self.dealer_point = QLabel(self)
        self.user_point = QLabel(self)
        self.dealer = []
        self.user = []
```

1. 윈도우 창의 이름, 크기, 위치, 배경을 구현하고 카드가 일정하게 놓이게 하기 위해 절대 위치를 사용할 것이므로 크기변경을 불가능하게 하였다.
2. 카드사이의 간격 변수를 만들었다.
3. 딜러와 플레이어의 첫 번째 카드의 x좌표 변수를 만들어 두었다.
4. 딜러와 플레이어의 카드 값의 합(포인트)을 0으로 만들어두었다.
5. 딜러와 플레이어의 나온 카드 숫자를 저장할 리스트를 생성하였다.

```
        self.coin = int(coin)
        # coin
        self.coin_st = QLineEdit(self)
        self.coin_st.setPlaceholderText(str(int(self.coin)))
        self.coin_st.setGeometry(800, 30, 60, 30)
        self.coin_st.setReadOnly(True)
        self.coin_st.setAlignment(Qt.AlignRight)
```

6. 인자로 받은 코인을 저장
7. 코인의 값을 보여주는 QLable 생성

```
        # dealer
        self.dealerCardDic = {str(x) + "_r.jpg" : x for x in range(1,12)}
        self.dealerCardList = list(self.dealerCardDic[str(x) + "_r.jpg"] for x in range(1, 10))
        self.first_dealerCardList = list(self.dealerCardDic.keys())
```

8. 딜러의 카드이름을 key로, 카드 값을 values로 하는 사전 생성
9. 딜러의 카드 값들만 담은 리스트 생성

```
        # user
        self.userCardDic = {str(x) + ".jpg" : x for x in range(1,12)}
        self.userCardList = list(self.userCardDic[str(x) + ".jpg"] for x in range(1, 10))
        self.first_userCardList = list(self.userCardDic.keys())

        # dealer's first init
        self.dealer_lbl = QLabel(self)
        self.dealer_card = QPixmap("card_reverse.jpg")
        self.dealer_lbl.setGeometry(self.d_previous, 70, 60, 100)
        self.dealer_lbl.setPixmap(self.dealer_card)

        # dealer's second card
        d_lbl2 = QLabel(self)
        d_card2 = random.sample(self.dealerCardList, 1)[0]
        self.d_point += d_card2
        self.dealer.append(d_card2)
        d_card2 = QPixmap(str(d_card2) + "_r.jpg")
        d_lbl2.setGeometry(self.d_previous + self.interval, 70, 60, 100)
        self.d_previous = self.d_previous + self.interval
        d_lbl2.setPixmap(d_card2)

        self.startGame()
        self.gamefinish = GameFinished()
```

10. 플레이어의 카드이름을 key로, 카드 값을 value로 하는 사전 생성
11. 플레이어의 카드 값만을 담은 리스트 생성
12. 딜러의 첫 번째 카드를 뒤집은 상태의 이미지를 보여준다.
13. 딜러의 두 번째 카드를 랜덤하게 뽑고 이미지를 보여준다.
14. 딜러의 두 번째 카드의 값을 리스트에 추가하고 포인트에 더해준다.
15. 딜러의 전 카드 위치를 간격을 추가한 값으로 변경한다.
16. startGame() 호출
17. GameFinished() 클래스 객체 생성

- StartGame(self)

```
def startGame(self):

    # user's first Card
    user_lbl = QLabel(self)
    user_card = random.sample(self.first_userCardList, 1)[0]
    self.u_point += self.userCardDic[user_card]
    self.user.append(self.userCardDic[user_card])
    user_card = QPixmap(user_card)
    user_lbl.setGeometry(self.u_previous, 230, 60, 100)
    user_lbl.setPixmap(user_card)
    self.showPoint()

    # user's second Card
    user_lbl2 = QLabel(self)
    user_card2 = random.sample(self.userCardList, 1)[0]
    self.u_point += user_card2
    self.user.append(user_card2)
    user_card2 = QPixmap(str(user_card2) + ".jpg")
    user_lbl2.setGeometry(self.u_previous + self.interval, 230, 60, 100)
    user_lbl2.setPixmap(user_card2)
    self.u_previous = self.u_previous + self.interval
    self.user_sum = sum(self.user)
    self.showPoint()

    # hit, stand button
    self.hitButton = QPushButton(self)
    self.hitButton.setText("Hit!")
    self.hitButton.setGeometry(800, 160, 50, 30)
    self.hitButton.clicked.connect(self.pushHit)

    self.standButton = QPushButton(self)
    self.standButton.setText("Stand")
    self.standButton.setGeometry(800, 210, 50,30)
    self.standButton.clicked.connect(self.playDealer)
```

1. 플레이어의 첫 번째 카드를 랜덤하게 뽑고 이미지를 보여준다.

2. 앞서 말한 딜러와 동일하게 각 플레이어 관련 객체와 변수를 변경해준다.
3. showPoint() 호출
4. hit 버튼과 stand 버튼의 텍스트, 위치, 크기를 지정해준다.
5. hit 버튼이 눌리면 pushHit()에 연결한다.
6. stand 버튼이 눌리면 playDealer()에 연결한다.

- pushHit(self)

```
def pushHit(self):

    # user puts card
    u_lbl = QLabel(self)
    u_card = random.sample(self.userCardList, 1)[0]
    self.user.append(u_card)
    self.u_point += u_card
    u_card = QPixmap(str(u_card) + ".jpg")
    u_lbl.setGeometry(self.u_previous + self.interval, 230, 60, 100)
    u_lbl.setPixmap(u_card)
    self.u_previous = self.u_previous + self.interval
    self.user_sum = sum(self.user)
    if self.user_sum > 21:
        self.hitButton.setDisabled(True)

    u_lbl.show()
    self.showPoint()
    self.userFinished()
```

1. 플레이어의 카드를 랜덤으로 뽑아서 이미지를 보여준다.
2. 위와 동일하게 플레이어 관련 객체와 변수를 변경시켜준다.
3. 플레이어 카드값들의 합을 저장해준다.
4. 플레이어 카드 값의 합이 21 이상일 경우, hit 버튼의 기능을 중지시킨다.
5. showPoint()호출
6. userFinished()호출

- playDealer(self)

```
def playDealer(self):

    self.hitButton.setDisabled(True)

    self.dealer_sum = sum(self.dealer)
    self.dealer_card = random.sample(self.first_dealerCardList, 1)[0]
    self.dealer.append(self.dealerCardDic[self.dealer_card])
    self.d_point += (self.dealerCardDic[self.dealer_card])
    self.dealer_card = QPixmap(self.dealer_card)

    self.dealer_lbl.setPixmap(self.dealer_card)
    self.showPoint()
    self.dealerFinished()
```

1. hit버튼의 기능을 중지시킨다.
2. 딜러의 첫 번째 카드를 랜덤으로 뽑아서 이미지를 보여준다.
3. 동일하게 딜러 관련 객체와 변수를 변경시켜준다.
4. showPoint() 호출
5. dealerFinished() 호출

```
    # dealer puts card
    while self.dealer_sum <= 16:

        d_lbl = QLabel(self)
        d_card = random.sample(self.dealerCardList, 1)[0]
        self.dealer.append(d_card)
        self.d_point += d_card
        d_card = QPixmap(str(d_card) + "_r.jpg")
        d_lbl.setPixmap(d_card)
        d_lbl.setGeometry(self.d_previous + self.interval, 70, 60, 100)
        self.d_previous = self.d_previous + self.interval
        self.dealer_sum = sum(self.dealer)
        d_lbl.show()
        self.showPoint()
```

7. 딜러 카드값들의 합이 16이하일 동안 딜러의 카드를 뽑음
8. dealerFinised() 호출

- dealerFinished(self)

```
def dealerFinished(self):
    self.standButton.setDisabled(True)

    if self.user_sum == self.dealer_sum:
        self.gamefinish.setResultCoin("push", self.coin)
        self.hitButton.setDisabled(True)

    elif self.dealer_sum > 21:
        self.gamefinish.setResultCoin("d_bust", self.coin)
        self.hitButton.setDisabled(True)

    elif self.dealer_sum == 21:
        self.gamefinish.setResultCoin("d_blackjack", self.coin)
        self.hitButton.setDisabled(True)

    elif self.dealer_sum > self.user_sum:
        self.gamefinish.setResultCoin("d_win", self.coin)
        self.hitButton.setDisabled(True)

    elif self.dealer_sum < self.user_sum:
        self.gamefinish.setResultCoin("u_win", self.coin)
        self.hitButton.setDisabled(True)
```

각 조건에 맞는 if문마다 객체의 인자값을 다르게 해서 self.gamefinish를 변경시켜준다. 순서가 매우 중요한데, 위 순서대로 하지 않으면 21이 넘어도 딜러의 합이 플레이어의 합보다 크다며 딜러가 승리하는 오류가 발생한다.

-exit(self)

```
def exit(self):
    self.deleteLater()
```

해당 윈도우를 지운다.

2. GameFinised(self, result, coin) class

- __init__(self)

1. 윈도우의 기본 인터페이스를 구현

```
class GameFinished(QWidget):

    def __init__(self):

        super().__init__()
        self.setWindowTitle("Result")
        self.setGeometry(340, 600, 900, 400)
        self.setFixedSize(900, 400)
        pal = QPalette()
        pal.setColor(QPalette.Background, QColor(105, 55, 105))
        self.setPalette(pal)
        self.coin = 0
        self.result_coin = 0

        hbox1 = QHBoxLayout()
        self.hbox2 = QHBoxLayout()
        hbox4 = QHBoxLayout()

        vbox = QVBoxLayout()

        self.result_lbl = QLabel(self)
        self.exitGame = QPushButton()
        self.exitGame.setText("Exit")
        self.coin_state = QLabel(self)

        hbox1.addWidget(self.result_lbl)
        self.hbox2.addWidget(self.coin_state)
        hbox4.addWidget(self.exitGame)
        vbox.addLayout(hbox1)
        vbox.addLayout(self.hbox2)
        vbox.addLayout(hbox4)
        self.setLayout(vbox)

        self.result = ""
```

2. self.result값을 ‘’로 초기화한다.

- setResultCoin(self, result, coin)

```
def setResultCoin(self, result, coin):
    self.result = result
    self.coin = coin
    self.gameFinished(self.result)
```

1. result, coin인자를 할당한다.
2. gameFinished(self.result) 호출

- gameFinished(self.result)

```
def gameFinished(self, result):
    self.result_coin = 0

    if result == "u_bust":
        self.result_lbl.setText("You're Bust!")
        self.result_lbl.setFont(QFont("times", 50))
        self.result_lbl.setStyleSheet("Color : black")
        self.coin_state.setText("-" + str(int(self.coin * 1.5)))
        self.coin_state.setFont(QFont("times", 30))
        self.coin_state.setStyleSheet("Color : black")
        self.result_coin -= int(self.coin * 0.5)
        self.show()

    elif result == "u_blackjack":
        self.result_lbl.setText("You hit BlackJack!")
        self.result_lbl.setFont(QFont("times", 50))
        self.result_lbl.setStyleSheet("Color : black")
        self.coin_state.setText("+" + str(int(self.coin * 1.5)))
        self.coin_state.setFont(QFont("times", 30))
        self.coin_state.setStyleSheet("Color : black")
        self.result_coin += int(self.coin * 1.5)
        self.show()

    elif result == "push":
        self.result_lbl.setText("Push! Draw!")
        self.result_lbl.setFont(QFont("times", 50))
        self.result_lbl.setStyleSheet("Color : black")
        self.coin_state.setText("+" + str(int(self.coin * 1.5)))
        self.coin_state.setFont(QFont("times", 30))
        self.coin_state.setStyleSheet("Color : black")
        self.coin_state.setText(str(int(self.coin)))
        self.result_coin = self.coin
        self.show()



    elif result == "u_win":
        self.result_lbl.setText("You are closer to 21!")
        self.result_lbl.setFont(QFont("times", 50))
        self.result_lbl.setStyleSheet("Color : black")
        self.coin_state.setText("+" + str(int(self.coin * 1.5)))
        self.coin_state.setFont(QFont("times", 30))
        self.coin_state.setStyleSheet("Color : black")
        self.result_coin += int(self.coin * 1.5)
        self.show()
```

각 조건에 맞게 텍스트를 출력하고 self.result_coin을 변경해준다.

```
elif result == "d_bust":
    self.result_lbl.setText("Dealer is Bust!")
    self.result_lbl.setFont(QFont("times", 50))
    self.result_lbl.setStyleSheet("Color : black")
    self.coin_state.setText("+" + str(int(self.coin * 1.5)))
    self.coin_state.setFont(QFont("times", 30))
    self.coin_state.setStyleSheet("Color : black")
    self.result_coin += int(self.coin * 1.5)
    self.show()

elif result == "d_blackjack":
    self.result_lbl.setText("Dealer hits BlackJack!")
    self.result_lbl.setFont(QFont("times", 50))
    self.result_lbl.setStyleSheet("Color : black")
    self.coin_state.setText("-" + str(int(self.coin * 1.5)))
    self.coin_state.setFont(QFont("times", 30))
    self.coin_state.setStyleSheet("Color : black")
    self.result_coin -= int(self.coin * 0.5)
    self.show()

elif result == "d_win":
    self.result_lbl.setText("Dealer is closer to 21!")
    self.result_lbl.setFont(QFont("times", 50))
    self.result_lbl.setStyleSheet("Color : black")
    self.coin_state.setText("-" + str(int(self.coin * 1.5)))
    self.coin_state.setFont(QFont("times", 30))
    self.coin_state.setStyleSheet("Color : black")
    self.result_coin -= int(self.coin * 0.5)
    self.show()
```

플레이어 승리시 미리 차감된 수를 고려해서 1.5배 한 수를 더해주고, 패배 시에는 0.5배를 한 수를 빼준다.

- getResult_coin(self)

```
def getResult_Coin(self):
    return self.result_coin

def exit(self):
    self.deleteLater()
```

self.result_coin을 반환해준다.

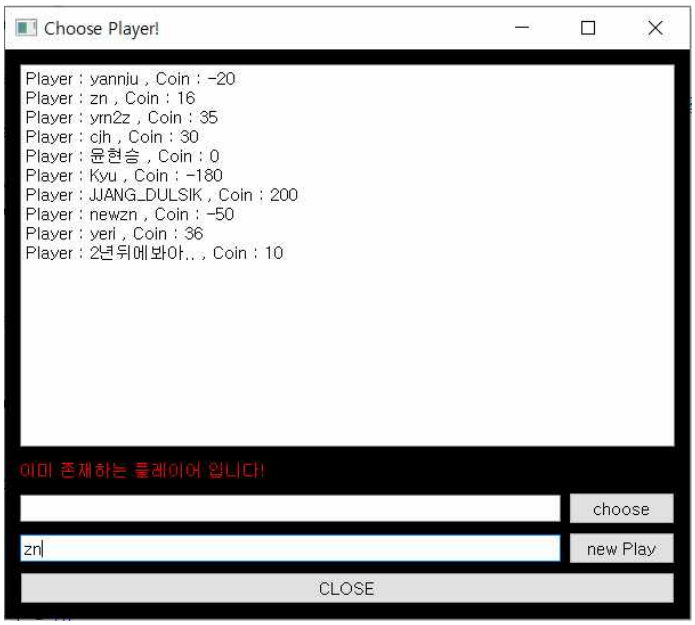
- exit(self)

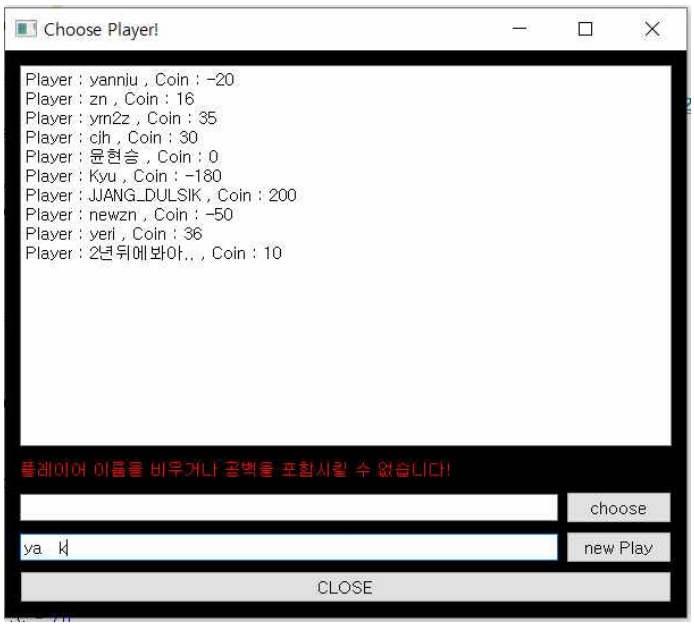
```
def exit(self):
    self.deleteLater()
```

해당 윈도우를 지운다.

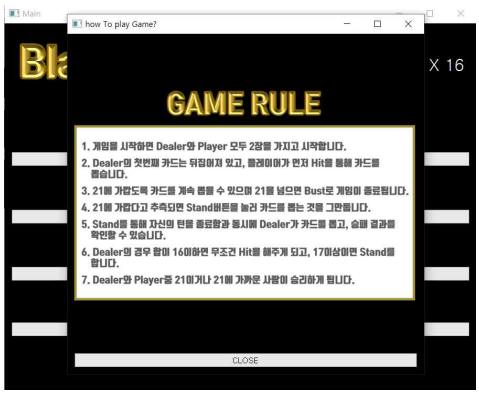
Ⅳ. 결과물

- 각 플레이어 입력에 관한 오류에 대한 처리를 해주었다.

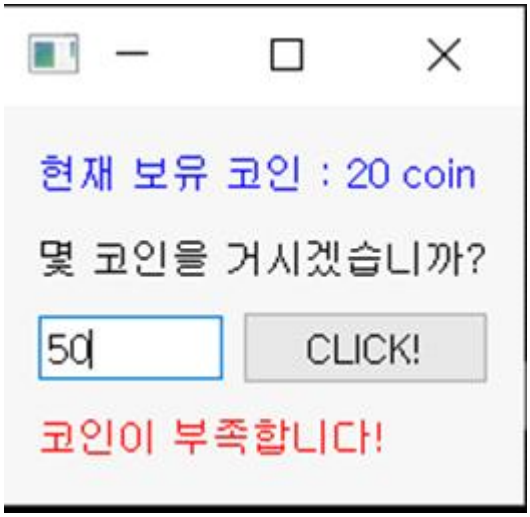




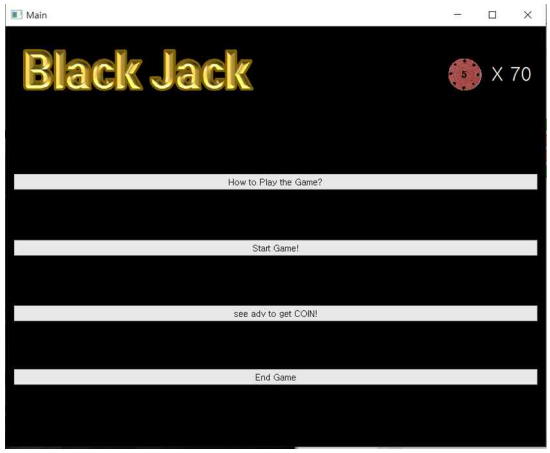
- how to play를 누르면 게임설명 창이 나온다.



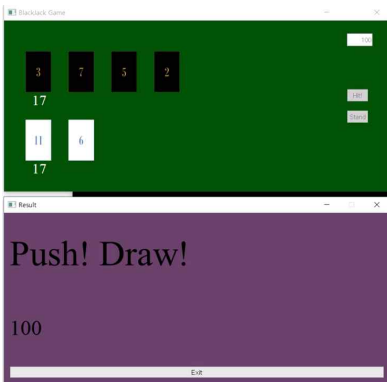
- 코인이 부족할 경우 에러를 처리해주었다.

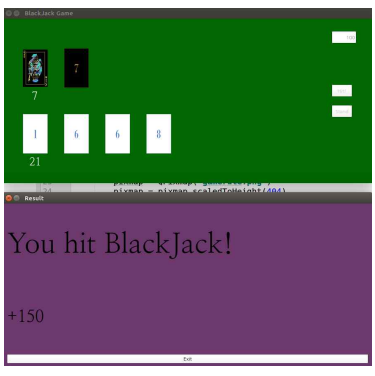


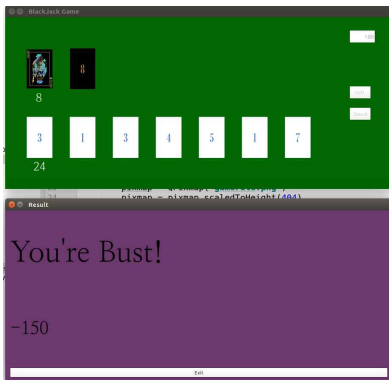
- 광고를 볼 수 있고 광고를 본 후 코인이 추가되었다. (50 -> 70)

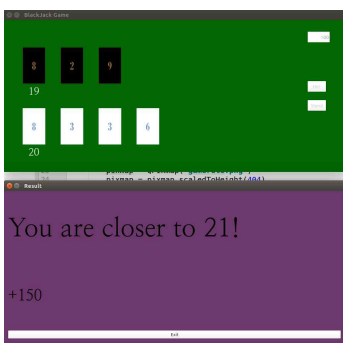


- 각 조건별 결과에 따른 다른 창을 보여준다.










BlackJack Game
6
9
6
21
2
2
7
7
18
Result
Dealer hits BlackJack!
-150
Exit