CP2_201902748_전해준_12주차

과제 목표

- 자바의 기능 중 GUI를 이용하여 BlackJack 게임을 구현할 수 있다.
- JFrame, JPanel, JButton과 ActionListener를 이용할 수 있다.

Card Class

[전체 코드]

```
3  public class Card {
4      private String suit;
5      private String rank;
6
7      // 특정 모양과 숫자를 가지는 카드를 만드는 생성자
8      public Card(String suit, String rank) {
9          this.suit = suit;
10         this.rank = rank;
11     }
12
13     // 카드의 모양을 반환
14     public String getSuit() {
15         return suit;
16     }
17
18     // 카드의 숫자를 반환
19     public String getRank() {
20         return rank;
21     }
22
23     // 카드를 출력할 때 이용
24     public String toString() {
25         return getSuit() + "_" + getRank();
26     }
27 }
28
```

과제 해결 과정

private로 선언되어 카드 모양을 뜻하는 String suit와 카드 숫자를 뜻하는 String rank를 선언하고 생성자에서 값을 전달받아 초기화한다.

private로 선언된 변수이기에 Getter 메소드 getSuit()와 getRank()를 생성한다.

마지막으로 “(모양)_(숫자)” 꼴로 카드가 출력되도록 하는 toString() 메소드를 생성한다.

CardDeck Class

[전체 코드]

```java
import java.util.ArrayList;
import java.util.Collections;

public class CardDeck {
    private String suits[] = {"Spade", "Heart", "Diamond", "Club"};
    private String ranks[] = {"A", "2", "3", "4", "5", "6", "7", "8", "9" ,"10", "K","Q", "J"};
    private ArrayList<Card> deck;

    // 카드 52장을 가지고 있는 카드덱을 만드는 생성자
    public CardDeck() {
        deck = new ArrayList<Card>();
        for(int i=0; i<4; i++) {
            for(int j=0; j<13; j++) {
                Card card = new Card(suits[i],ranks[j]);
                deck.add(card);
            }
        }
    }

    // 제일 앞에 있는 카드를 뽑아서 반환
    public Card popCard() {
        deck.remove(0);
        return deck.get(0);
    }

    // 덱에 남은 카드 수를 반환
    public int getSize() {
        return deck.size();
    }

    // 덱을 셔플함
    public void shuffle() {
        Collections.shuffle(deck);
    }
}
```

과제 해결 과정

모두 private이고 카드 모양이 담긴 String suits[] 배열과 카드 숫자가 담긴 String ranks[] 배열 그리고 카드덱을 구성할 ArrayList<Card> deck이 선언되어있다. 이 중 deck은 생성자에서 초기화되고 이중 for문을 이용해 Card의 모양과 숫자를 조합하여 카드덱을 구성한다.

popCard 메소드를 보면 제일 앞의 카드를 뽑는다. 하지만 덱을 생성 후 한번 셔플한 뒤 패를 분배하므로 인덱스가 0인 카드를 제거한 후 분배하여도 플레이어 입장에서는 인덱스가 1인 카드가 제일 앞에 있는 카드가 되어 지장이 없어진다. 따라서 remove 후 return 한다.

카드덱에 남은 카드 수 즉, deck의 크기를 반환하는 getSize() 메소드와 Collections의 특성을 이용한 shuffle 즉, 카드덱을 셔플하는 shuffle() 메소드를 추가하여 마무리한다.

Player Class

[전체 코드]

```java
import java.util.ArrayList;
import java.util.Iterator;

public class Player {
    private ArrayList<Card> hand;

    // 본인의 패(hand)를 가지는 플레이어 생성자
    public Player() {
        hand = new ArrayList<Card>();
    }

    // 플레이어의 패에 카드를 추가
    public void hit(Card card) {
        hand.add(card);
    }

    // 플레이어의 패 점수를 계산하여 반환
    public int getScore() {
        int score = 0;
        for(int i=0; i<hand.size(); i++) {
            switch(hand.get(i).getRank()) {
            case "A":
                if(score <= 10) {
                    score += 11; break;
                } else {
                    score += 1; break;
                }
            case "2": score += 2; break;
            case "3": score += 3; break;
            case "4": score += 4; break;
            case "5": score += 5; break;
            case "6": score += 6; break;
            case "7": score += 7; break;
            case "8": score += 8; break;
            case "9": score += 9; break;
            case "10": case "K" : case "Q" : case "J": score += 10; break;
            }
        }
        return score;
    }

    // 패의 점수가 21이 넘은 경우 true, 아닌 경우 false
    public boolean isBust() {
        if(getScore() > 21) return true;
        return false;
    }

    // 패의 점수가 21이 된 경우 true, 아닌 경우 false
    public boolean isBlackJack() {
        if(getScore() == 21) return true;
        return false;
    }
}
```

CP2_201902748_전해준_12주차

과제 해결 과정

사용자의 손에 든 카드를 저장할 ArrayList<Card> hand를 private로 선언하고 생성자에서 초기화한다.
hit(Card card) 메소드는 플레이어의 패에 카드를 추가한다.
getScore() 메소드는 hand의 크기만큼 반복하는 for문 속에서 카드의 숫자와 제시된 조건에 따라 switch문으로 비교하여 점수를 계산 후 int score를 return하는 메소드이다.
isBust()는 사용자의 패의 점수가 21점을 넘었는가를 판단하는 boolean 메소드이고 isBlackJack()은 사용자의 패의 점수가 딱 21점이 되었는가를 판단하는 boolean 메소드이다.

Dealer Class

[전체 코드]

```
3  import java.util.ArrayList;
4
5  public class Dealer extends Player {
6      private ArrayList<Card> hand;
7
8      public Dealer() {
9          hand = new ArrayList<Card>();
10     }
11 }
12
```

과제 해결 과정

게임을 이끄는 딜러를 다루는 Dealer 클래스는 위 Player 클래스를 부모 클래스로 하여 extends한다.
딜러의 패를 저장할 ArrayList<Card> hand를 private로 선언하고 생성자에서 초기화한다.

ImagePanel Class

[전체 코드]

```
3⊕import java.awt.BorderLayout;
13
14 public class ImagePanel extends JPanel {
15     private Image img;
16
17     public ImagePanel() { }
18
19⊖    public ImagePanel(String img) {
20         // 이미지 경로를 넣으면 이미지 Panel을 생성해주는 생성자
21         this(new ImageIcon(img).getImage());
22     }
23
24⊖    public ImagePanel(Image img) {
25         this.img = img;
26         Dimension size = new Dimension(img.getWidth(null), img.getHeight(null));
27         setPreferredSize(size);
28         setMinimumSize(size);
29         setMaximumSize(size);
30         setSize(size);
31         setLayout(null);
32     }
33
34⊖    public void setImage(String img) {
35         // 이미지를 변경해주는 메소드
36         this.img = new ImageIcon(img).getImage();
37         repaint();
38     }
39
40⊖    public void paintComponent(Graphics g) {
41         g.drawImage(img, 0, 0, null);
42     }
43 }
```

과제 해결 과정

JPanel을 extends한 ImagePanel이다. 이 Panel은 카드 그림을 나타내는 Panel이 되고 private Image 변수 img를 선언한다.

생성자에 아무런 값을 전달하지 않으면 빈칸으로 두고 이미지 경로를 전달 시 Panel을 생성하는 두 가지 생성자를 만든다.

public ImagePanel(Image img)에서는 img를 전달받고 크기를 화면에 맞추어서 자동으로 조정되도록 설정해준다.

setImage(String img)는 이미지 경로를 전달 받고 그 이미지로 repaint()하여 바꾸어주는 메소드이다.

painComponent(Graphic g)는 g를 그리는 메소드이다.

PlayerPanel Class

[전체 코드]

```
3 import java.awt.BorderLayout;
10
11 public class PlayerPanel extends JPanel {
12     private ImagePanel imagepanel;
13     private ImagePanel[] panels;
14     private String[] Address;
15     private int number;
16
17     public PlayerPanel() {
18         setSize(400, 70);
19         setLayout(new FlowLayout(FlowLayout.LEFT, 0, 0));
20         Address = new String[8];
21         panels = new ImagePanel[8];
22         imagepanel = new ImagePanel();
23         for(int i=0; i<8; i++) {
24             panels[i] = imagepanel;
25         }
26         number = 0;
27         setVisible(true);
28     }
29
30     public void ShowCard(String address) {
31         Address[number] = address;
32         panels[number] = new ImagePanel("./images/cards/" + address + ".png");
33         for(int i=0; i<8; i++) {
34             add(panels[i]);
35         }
36         number++;
37     }
38 }
```

과제 해결 과정

JPanel을 extends한 PlayerPanel이다. Panel은 사용자 카드 패를 나타내는 Panel이 된다. 카드를 표현할 private ImagePanel imagepanel 변수, imagepanel들의 배열이 되는 private ImagePanel[] panels 변수, 패에 있는 카드 정보들을 기억하는 private String[] Address 변수 그리고 손에 있는 카드 수를 세는 private int number 변수가 있다.

PlayerPanel()에서 크기를 400*70으로 설정하고 카드가 왼쪽부터 차곡차곡 그려지도록 FlowLayout을 LEFT로 설정한다. 패에 담는 최대 카드 수를 8장으로 하여 Address, panels, imagepanel을 초기화하고 panels에는 imagepanel들을 for문으로 담아둔다. number 또한 0으로 초기화하고 setVisible(true)로 하여 출력하도록 한다.

ShowCard(String address)에서는 카드 숫자를 address로 받고 Address[number]에 이를 저장한다. 또한 panels[number]에는 address를 이용한 img 경로를 전달하여 새로운 ImagePanel을 저장한다. for문으로 PlayerPanel에 panels[]를 추가하여 카드를 보여주고 number++을 하여 패가 한 장 늘었음을 표시한다.

DealerPanel Class

[전체 코드]

```
3 import java.awt.BorderLayout;
10
11 public class DealerPanel extends JPanel {
12     private ImagePanel imagepanel;
13     private ImagePanel[] panels;
14     private String[] Address;
15     private int number;
16
17     public DealerPanel() {
18         setSize(400, 70);
19         setLayout(new FlowLayout(FlowLayout.LEFT, 0, 0));
20         Address = new String[8];
21         panels = new ImagePanel[8];
22         imagepanel = new ImagePanel();
23         for(int i=0; i<8; i++) {
24             panels[i] = imagepanel;
25         }
26         number = 0;
27         setVisible(true);
28     }
29
30     public void ShowCard(String address) {
31         Address[number] = address;
32         panels[number] = new ImagePanel("./images/cards/" + address + ".png");
33         for(int i=0; i<8; i++) {
34             add(panels[i]);
35         }
36         number++;
37     }
38
39     public void HideCard(String address) {
40         Address[number] = address;
41         panels[number] = new ImagePanel("./images/cards/back.png");
42         for(int i=0; i<8; i++) {
43             add(panels[i]);
44         }
45         number++;
46     }
47
48     public void ShowAll() {
49         for(int i=0; i<8; i++) {
50             if(Address[i] != null) panels[i].setImage("./images/cards/" + Address[i] + ".png");
51         }
52     }
53 }
```

과제 해결 과정

JPanel을 extends한 DealerPanel이다. Panel은 딜러의 카드 패를 나타내는 Panel이 된다.
ShowCard(String address) 메소드까지는 위의 PlayerPanel과 이름을 제외하고 일치한다.
HideCard(String address) 메소드는 딜러의 패는 첫 장을 제외하고 뒷면을 보여주어야 하므로 생긴 메소드이다. ShowCard와 유사하나 img를 무조건 뒷면으로 전달하여 첫장을 제외한 나머지 카드의 뒷면만 보여주도록 기능한다.
ShowAll() 메소드는 게임이 종료되는 경우 딜러의 패를 모두 보여주도록 하는 메소드이다. for문을 이용하여 Address 배열에 카드 정보가 저장되어 있다면 그 카드 정보와 같은 img로 교체하는 동작을 한다. 즉 뒷면이었던 카드를 보여주는 메소드이다.

BlackJackFrame Class

[전체 코드 118줄 중 62줄까지]

```
 3⊕import java.awt.BorderLayout;
16
17 public class BlackJackFrame extends JFrame {
18     private static Dealer dealer;
19     private static Player player;
20     private static CardDeck deck;
21     private DealerPanel dealerpanel;
22     private PlayerPanel playerpanel;
23     private JPanel UIpanel;
24     private JLabel text;
25     private JButton hit;
26     private JButton stand;
27
28⊖    public BlackJackFrame() {
29         setSize(450, 320);
30         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31         setTitle("BlackJack");
32         setLayout(new GridLayout(3,1));
33
34         dealer = new Dealer();
35         player = new Player();
36         deck = new CardDeck();
37         dealerpanel = new DealerPanel();
38         Border dealerborder = BorderFactory.createTitledBorder("Dealer");
39         dealerpanel.setBorder(dealerborder);
40         playerpanel = new PlayerPanel();
41         Border playerborder = BorderFactory.createTitledBorder("Player");
42         playerpanel.setBorder(playerborder);
43         UIpanel = new JPanel();
44         UIpanel.setSize(400, 70);
45
46         deck.shuffle();
47         for(int i=0; i<2; i++) {
48             Object tmp = deck.popCard();
49             String address = tmp.toString();
50             playerpanel.ShowCard(address);
51             player.hit((Card) tmp);
52         }
53         int i = 0;
54         while(dealer.getScore() < 16) {
55             Object tmp = deck.popCard();
56             String address = tmp.toString();
57             if(i==0) {dealerpanel.ShowCard(address);}
58             else {dealerpanel.HideCard(address);}
59             dealer.hit((Card) tmp);
60             i++;
61         }
62
```

과제 해결 과정

JFrame을 extends한 BlackJackFrame이다. 이 Frame은 전체 프로그램의 바탕이 된다.

Dealer, Player, CardDeck은 공통적으로 이용되므로 private static으로 각각 dealer, player, deck으로 선언한다. 그 외에 이용되는 DealerPanel, PlayerPanel, JPanel, JLabel, JButton 두 개를 private로 각각 dealerpanel, playerpanel, UIpanel, text, hit, stand로 선언한다.

450*320으로 크기를 설정하고 CLOSE를 누르면 프로그램이 꺼지도록 설정하며 제목을 BlackJack으로 설정한다. 제시된 조건에 따라 GridLayout(3,1)도 설정한다.

dealer, player, deck을 초기화한다. dealerpanel과 playerpanel도 초기화하는데 이 둘은 경계선을 제목과 함께 넣어주는 조건이 있으므로 Border와 BorderFactory를 이용하여 각각 Dealer, Player 제목을 붙여 초기화한다. 마지막으로 안내문구와 버튼 두 개가 들어갈 UIPanel을 400*70 크기로 초기화한다.

우선 카드 덱을 셔플하고 for문을 이용하여 사용자에게 카드 두 장을 건넨다. 이때 ShowCard(address)와 hit(Card card)가 이용된다. 그 후 딜러 카드를 뽑는데 딜러 패의 점수 총합이 16이상이 될 때까지 뽑아야 하므로 while문을 이용한다. 이 때 딜러의 첫 장만 공개하고 나머지는 숨긴다는 점에 유의하여 마지막에 int i = 0을 i++ 해주는 것으로 구분한다.

[전체 코드 118줄 중 63줄 ~ 79줄]

```
63          text = new JLabel("Dealer:??, Player:" + player.getScore());
64          if(player.isBlackJack()) {
65              text.setText("Dealer:" + dealer.getScore() + ", Player:" + player.getScore() + "  BlackJack!!");
66              dealerpanel.ShowAll();
67          }
68          text.setSize(400, 70);
69          hit = new JButton("Hit");
70          hit.addActionListener(new HitAction());
71          stand = new JButton("Stand");
72          stand.addActionListener(new StandAction());
73          UIpanel.add(text); UIpanel.add(hit); UIpanel.add(stand);
74
75          this.add(dealerpanel); this.add(playerpanel); this.add(UIpanel);
76
77          setVisible(true);
78      }
79
```

과제 해결 과정

현재 상황을 표현하는 JLabel text를 안내와 함께 초기화한다. 만약 시작부터 사용자가 BlackJack인 경우 바로 점수와 패를 공개하여 끝을 낸다. text 크기를 400*70으로 한다.

hit와 stand를 각각 Hit, Stand 이름을 붙이고 addActionListener까지 붙여 초기화한다.

UIpanel에 JLable text, JButton hit, JButton stand를 add하고 전체 Frame에 dealerpanel, playerpanel, UIpanel을 add한다.

setVisible(true)로 보이도록 설정한다.

[전체 코드 118줄 중 80줄 ~ 끝]

```
80    class HitAction implements ActionListener {
81        @Override
82        public void actionPerformed(ActionEvent e) {
83            if(e.getSource() == hit) {
84                try{
85                    Object tmp = deck.popCard();
86                    String address = tmp.toString();
87                    playerpanel.ShowCard(address);
88                    player.hit((Card) tmp);
89                    text.setText("Dealer:??,Player:" + player.getScore());
90                    if(player.isBlackJack()) {
91                        text.setText("Dealer:" + dealer.getScore() + ", Player:" + player.getScore() + "  BlackJack!!");
92                        dealerpanel.ShowAll();
93                    }
94                    if(player.isBust()) {
95                        text.setText("Dealer:" + dealer.getScore() + ", Player:" + player.getScore() + "  Bust하셨습니다..");
96                        dealerpanel.ShowAll();
97                    }
98                } catch (Exception x) {
99                    text.setText("더 이상 진행할 수 없습니다.");
100               }
101           }
102       }
103   }
104
105   class StandAction implements ActionListener {
106       @Override
107       public void actionPerformed(ActionEvent e) {
108           if(e.getSource() == stand) {
109               if(Math.abs(dealer.getScore()-21) <= Math.abs(player.getScore()-21)) {
110                   text.setText("Dealer:" + dealer.getScore() + ", Player:" + player.getScore() + " 딜러의 승리");
111               } else {
112                   text.setText("Dealer:" + dealer.getScore() + ", Player:" + player.getScore() + " 플레이어의 승리");
113               }
114               dealerpanel.ShowAll();
115           }
116       }
117   }
118 }
```

과제 해결 과정

hit과 관련된 ActionListener HitAction이다. actionPerformed(ActionEvent e)에서 e.getSource가 hit과 같을 때 사용자가 카드 한 장을 더 받는다. 카드 한 장을 받고 text를 바뀐 점수와 함께 출력한다. 만약 카드 한 장을 더 받고 사용자가 BlackJack이 된 경우 text를 점수와 “BlackJack!!”이라는 안내로 수정하고 모든 패를 공개한다. 또는 Bust 된 경우, text를 점수와 “Bust하셨습니다..”라는 안내로 수정하고 모든 패를 공개한다. 게임이 끝나도 hit 버튼을 눌러 의미없는 진행을 하다가 패가 8장이 넘어가면 “더 이상 진행할 수 없습니다.”라는 text로 설정한다.

stand와 관련된 ActionListener StandAction이다. actionPerformed(ActionEvent e)에서 e.getSource가 stand와 같을 때 사용자가 게임을 중단하고 결과를 확인한다. 딜러와 사용자의 점수에서 21을 뺀 값을 Math.abs()로 절댓값 비교하여 승패를 결정한다. 만약 사용자의 오차가 딜러의 오차보다 크거나 같으면 “딜러의 승리”라는 text가 추가되고 작게 되면 “플레이어의 승리”라는 text가 추가된다. 마지막에는 모든 패를 공개한다.

Main Class

[전체 코드]

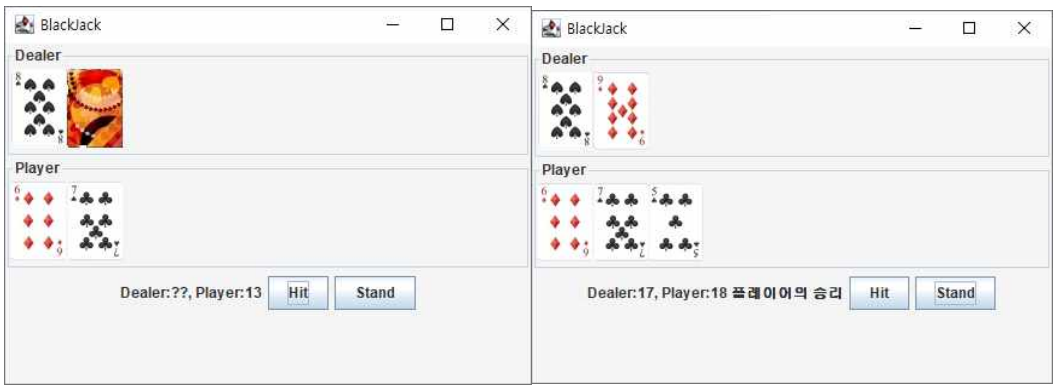
```
3  public class Main {
4      public static void main(String[] args) {
5          BlackJackFrame frame = new BlackJackFrame();
6          frame.setVisible(true);
7          frame.setLocationRelativeTo(null);
8      }
9  }
10
```

과제 해결 과정

제작된 모든 것을 실행시키는 Main 클래스이다. BlackJackFrame변수 frame을 생성&초기화하고 frame을 setVisible(true)로 하여 보이도록 설정한다. setLocationRelativeTo(null)은 프로그램을 실행할 때 모니터의 정중앙에서 실행되도록 설정하는 것이다.

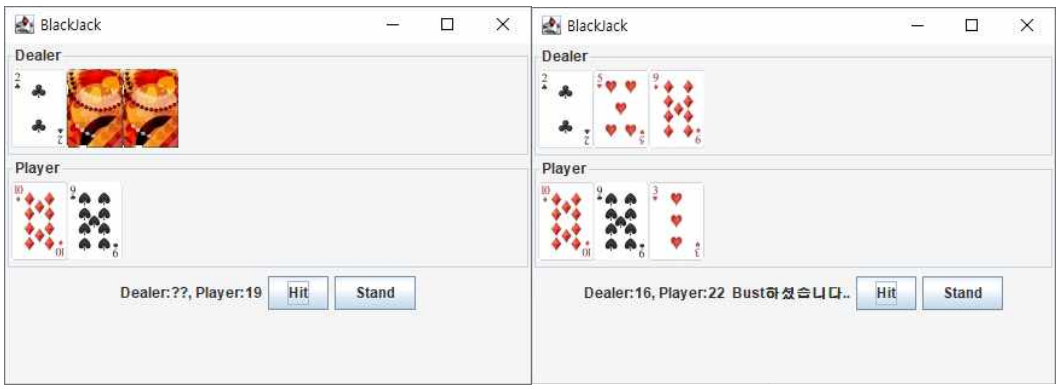
<실행 결과>

Game 1.



<실행 결과>

Game 2.



Game 3. (시작하자마자 BlackJack)

Game 4. (의미없는 진행 결과)

(※ 기재된 실행 결과 외에 다양한 실행 결과가 발생 할 수 있다.)