BLACK_JACK

▶ 구현한 기능들
▶ 게임 진행 방식
▶ 클래스 명세표와 구조도
▶ 예외 처리

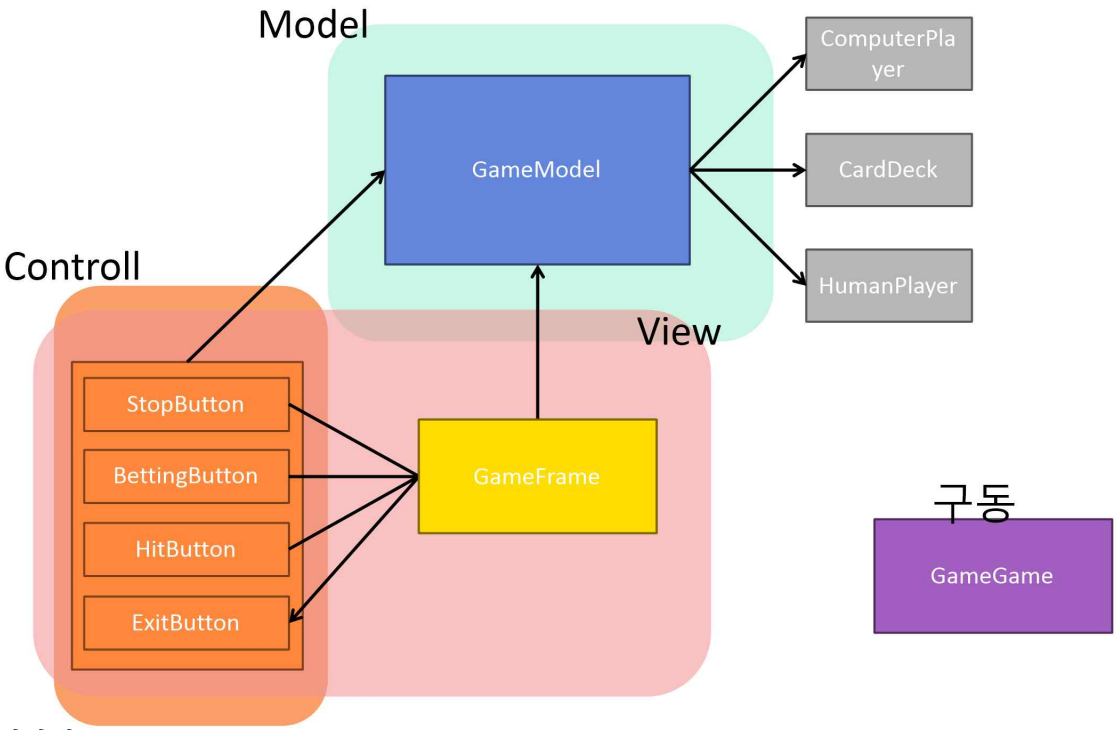
1. 구현한 기능들

수업 시간 때 다루었던 Card Game을 확장하였다. Card 클래스, CardDeck 클래스와 CardPlayer 클래스의 receiveCard 메소드는 기존의 것을 그대로 사용하였다.
베팅기능 : 사용자가 카드를 받기 전에 자신이 원하는 만큼의 돈을 베팅할 수 있으며 게임 결과에 따라 얻을 수도 잃을 수도 있다.
블라인드 기능 : 사용자가 카드를 더 받을지 그만 받을지 결정을 어렵게 하도록 딜러가 받은 처음 2개의 카드 정보를 가렸다가 게임이 끝나면 알려준다.
행운의 숫자 : 게임을 시작하기 전에 사용자에게 행운의 숫자를 입력받는다. 게임 중에 행운의 숫자와 같은 카드가 나오면 그 카드를 사용자의 결정에 따라 카드를 새로 바꿀 수 있다.
(기타) **GUI** : 베팅, 카드 받기(hit), 카드 그만 받기, 게임 종료 기능을 버튼을 통해 실행할 수 있도록 세팅하였다.

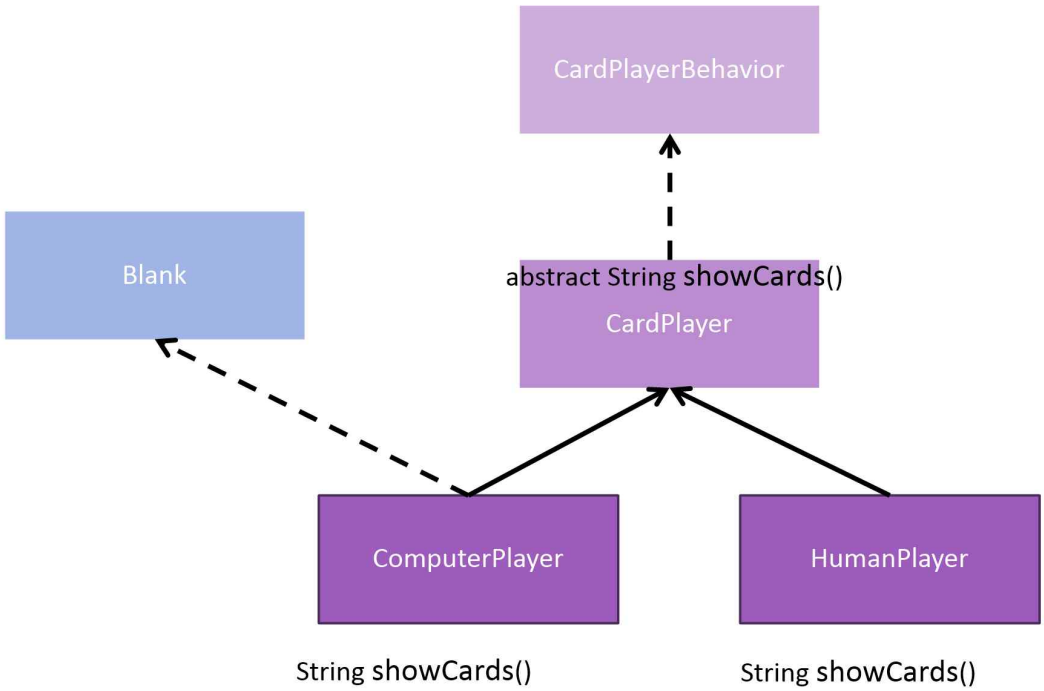
2. 게임 진행 방식

1. 게임 시작 전 행운의 숫자(1~13)를 입력한다.
2. 만 원을 가지고 게임이 시작되며 Betting 버튼을 통해 원하는 금액을 베팅한다.
3. Hit 버튼을 눌러 카드를 받는다. 더 받고 싶다면 Hit 버튼을, 그만 받고 싶다면 Stop 버튼을 누른다. (받은 카드가 5장이 되면 더 받을 수 없다.)
4. 행운의 숫자와 같은 숫자의 카드를 가지고 있다면 그 카드를 버리고 다시 뽑을 수 있다.
5. 딜러의 점수와 자신의 점수를 비교한다. 만약 상대가 21점을 넘었거나, 자신의 점수가 더 높다면 승리이다. 점수가 같거나 둘 다 21점을 넘었다면 비긴다. (숫자 11, 12, 13의 점수는 10점이다.)
6. 이겼다면 베팅 금액의 2배로 돌려받고(점수가 21점이라면 3배), 졌다면 돌려받지 못한다.
7. 게임을 끝내고 결과를 보고 싶다면 exit 버튼을 누른다.

3. 클래스 명세표와 구조도



연관관계



상속 및 인터페이스

abstract class CardPlayer (implements CardPlayerBehavior)

생성자	
public CardPlayer(int max_cards, int money)	가질 수 있는 카드 수와 초기 돈을 입력 받아서 객체 생성
속성	
protected Card[] my_hand protected int card_count protected int my_money protected int score protected int max_cards	-현재 가지고 있는 카드들 -가지고 있는 카드 수 -현재 나의 돈 -게임 중 나의 점수 -최대로 가질 수 있는 카드 수
메소드	
public void updatescore() public void receiveCard(Card c) public abstract String showCards() public void reset_hand()	-현재 점수를 계산, score에 저장 -입력받은 카드를 my_hand에 저장 - -가지고 있는 카드와 현재 점수를 초기화

class ComputerPlayer extneds CardPlayer

생성자	
public ComputerPlayer(int max_cards, int money)	상위 클래스CardPlayer 생성
속성	
boolean is_hit boolean is_win private String blankinfo	-딜러가 카드를 더 받을 것인지 확인 -딜러가 이겼는지 확인 -딜러의 가려져 있는 카드의 정보
메소드	
public String showCards() public String showBlank() public void resetBlank()	-2번째 카드까지는 정보를 보여주지 않음. 나머지 카드들의 suit와 count를 반환 -blankinfo를 알려주는 문자열 반환 -blankinfo를 초기화

class HumanPlayer extends CardPlayer

생성자	
public HumanPlayer(int max_cards, int money)	상위 클래스 CardPlayer 객체 생성
속성	
boolean is_bet boolean is_hit boolean is_stop boolean is_win	-베팅 버튼의 활성화 확인 -hit 버튼의 활성화 확인 -스탑 버튼의 활성화 확인 -사용자가 이겼는지 확인

메소드	
public String showCards()	사용자가 가지고 있는 카드의 suit와 count 정보를 반환

class GameModel(Model)

생성자	
public GameModel(HumanPlayer p, ComputerPlayer d, CardDeck cd)	사용자 객체, 딜러 객체, 카드 덱을 받아와서 게임 모델 생성
속성	
private HumanPlayer player	-사용자
private ComputerPlayer dealer	-딜러
private String message	-출력 뷰에 전달할 메시지
private CardDeck cd	-사용할 카드 덱
메소드	
public String get_message()	-message를 반환
public String get_money()	-사용자의 돈 정보를 문자열로 반환
public String get_betmoney()	-사용자가 베팅한 돈 정보를 문자열로 반환
public String get_playerinfo()	-사용자의 카드 정보를 반환
public String get_dealerinfo()	-딜러의 카드 정보를 반환
----------	----------
public void betting()	-베팅 버튼을 눌렀을 때 사용자에게 베팅 금액을 입력받고 베팅 버튼 비활성, hit 버튼 활성
public void hit()	-히트 버튼을 눌렀을 때 스탑 버튼 활성. 사용자와 딜러가 카드를 받음. 받을 손이 없거나 딜러의 점수가 17점을 넘어갔다면 버튼 비활성
public void stop()	-행운의 숫자 기능 실행 후 딜러와 사용자의 점수 비교, 승자와 점수, blank 정보를 message에 남기고 사용자에게 알맞은 돈을 돌려줌
----------	----------
public void choose_lucky()	-게임 시작 전 행운의 숫자를 입력 받음 -카드를 새로 뽑을 것인지 물어봄. 바꾼다면 true, 아니라면 false 반환
public boolean lucky_change(card c)	-행운의 카드를 가지고 있고, 다시 뽑기로 했다면 다시 뽑고 score 업데이트
public boolean Lucky()	
public void reset()	-사용자와 딜러의 손, 카드 덱, 승리 정보, 초기화
public void exit()	-게임 종료 버튼을 눌렀을 때 게임 결과를 출력해 주고 게임 종료. 가지고 있는 금액에 따

	라 문구 다르게 출력

class BetingButton(Controller, viewer) (extends JButon implements ActionListener)

생성자	
public BettingButton(GameFrame f, GameModel m)	“Betting”이라는 버튼 생성. 제어 신호를 보낼 모델과 출력정보를 전달할 프레임 저장
속성	
private GameFrame f private GameModel m	출력 뷰 게임 모델
메소드	
public void actionPerformed(ActionEvent e)	모델의 betting() 실행 후 메시지를 프레임에 보내주고 프레임 업데이트

class HitButton(Controller, viewer) (extends JButon implements ActionListener)

생성자	
public HitButton(GameFrame f, GameModel m)	“Hit”이라는 버튼 생성. 제어 신호를 보낼 모델과 출력 정보를 전달할 프레임 저장
속성	
private GameFrame f private GameModel m	출력 뷰 게임 모델
메소드	
public void actionPerformed(Action Event e)	모델의 hit() 실행 후 메시지를 프레임에 보내주고 프레임 업데이트

class StopButton(Controller, viewer) (extends JButon implements ActionListener)

생성자	
StopButton(GameFrame f, GameModel m)	“Stop”이라는 버튼 생성, 제어 신호를 보낼 모델과 출력 정보를 보낼 프레임 저장
속성	
GameFrame f GameModel m	출력 뷰 게임 모델
메소드	
public void actionPerformed(Action Event e)	모델의 stop 실행 후 메시지를 프레임에 보내주고 프레임 업데이트, 모델의 reset() 호출

class ExitButton(Controller, viewer) (extends JButon implements ActionListener)

생성자	
public ExitButton(GameModel m)	“exit”라는 버튼 생성 제어 신호를 보낼 모델 저장
속성	
private GameModel m	게임 모델
메소드	
public void actionPerformed(Action Event e)	모델의 exit() 호출

class GameFrame(viewer) (extends JFrame)

생성자	
public GameFrame(GameModel m)	게임 모델을 받아서 게임 프레임 생성
속성	
private GameModel m private JLabel bet private JLabel money private JLabel dealerinfo private JLabel playerinfo private JLabel message private JLabel change_message	-게임 모델 -베팅 금액 정보 창 -현재 돈 정보 창 -딜러의 카드 정보 창 -플레이어의 카드 정보 창 -메세지 창 -모델에서 받은 메시지 저장
메소드	
public void set_message(String s) public void update()	-모델에서 받은 메시지로 메시지 창 업데이트 -게임 프레임 업데이트

4. 예외 처리

행운의 숫자 입력 예외 :
choose_lucky() 함수에서 행운의 값을 입력받을 때 입력 값이 없다면 NullPointerException 에러가 발생했었다. 이런 상황일 때, choose_lucky() 메소드를 재호출 하여 제대로 된 입력값이 들어오도록 하였다.
입력값이 문자일 때는 NumberFormatException 에러가 발생하는 것을 확인했다. 이 경우에도 메소드를 재호출 하였다.
입력값이 정수이지만 범위를 벗어났을 때는 범위의 값을 알려주었다.

버튼 남발 예외 :
게임 실행 중에 사용자가 아무 버튼이나 누른다면 게임이 엉망이 되어버린다. HumanPlayer 속성 변수에 버튼 활성화와 관련된 변수들을 추가해 활용하였다. 이 변수들은 GameModel 클래스가 다루도록 하였다.

베팅 버튼 입력 예외 :
사용자가 입력한 베팅 금액이 문자일 때 NumberFormatException 이 발생하여서 catch를 사용해 메시지 창에 입력을 확인해 보라는 문구를 출력하였다. 입력값이 정수이나 범위를 벗어났을 때는 NumberFormatException을 발생시켜서 예외처리가 되도록 하였다.

히트 버튼 입력 예외 :
사용자나 딜러가 최대로 받을 수 있는 카드 수 이상을 받았을 때, 히트 버튼을 비활성화 하였다. 딜러는 점수가 17점을 넘어가면 사용자가 히트 버튼을 눌러도 카드를 받지 않도록 설계하였다. 활성화가 되지 않은 상태에서 눌렀을 때는 먼저 베팅 버튼을 누르라는 문구를 띄어주었다.

행운의 숫자 확인 입력 예외 :
카드를 새로 뽑을 건지 물어보는 입력차에 입력이 없거나 문자이거나 숫자 등, Y나 N이 아니면 계속 입력창을 다시 띄웠다.(게임 진행상 매우 중요한 입력이기 때문)
돈을 모두 잃었을 때 : 종료 버튼 이외의 버튼이 입력을 받지 않게 하였다.