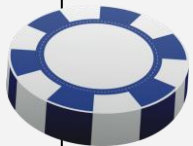

바카라 Baccarat

<코딩보다 도박을 잘함>
김다현, 김준태, 안태승, 장은호





목 차



게임 규칙
순서도
클래스 다이어그램
기능 소개
소스코드
프로그램 시연



게임 규칙

게임 규칙

1. 플레이어와 뱅커 중 어느 쪽이 이길지 매회 예상을 하고 배팅을 한다.
2. Ace는 1, King·Queen·Jack 10이며, 십의 자리 숫자는 생각하지 않는다.
3. 타이: 플레이어와 뱅커의 수 합이 서로 같다에 배팅 하는 것이다.
맞으면 배팅금액의 9배로 돌려준다.
4. 페어: 최초 2장의 카드가 같은 숫자가 나온다에 배팅 하는 것이다
맞으면 배팅 금액의 12배로 돌려준다.

게임 규칙

카드를 받는 규칙

1. 우선 플레이어와 뱅커 모두 2장의 카드를 받는다.
2. 플레이어 또는 뱅커 둘 중의 하나라도, 2장의 합이 8 또는 9가 되면 이를 내추럴(Natural)이라 부르며, 양쪽 모두 추가로 카드를 받지 못한다.

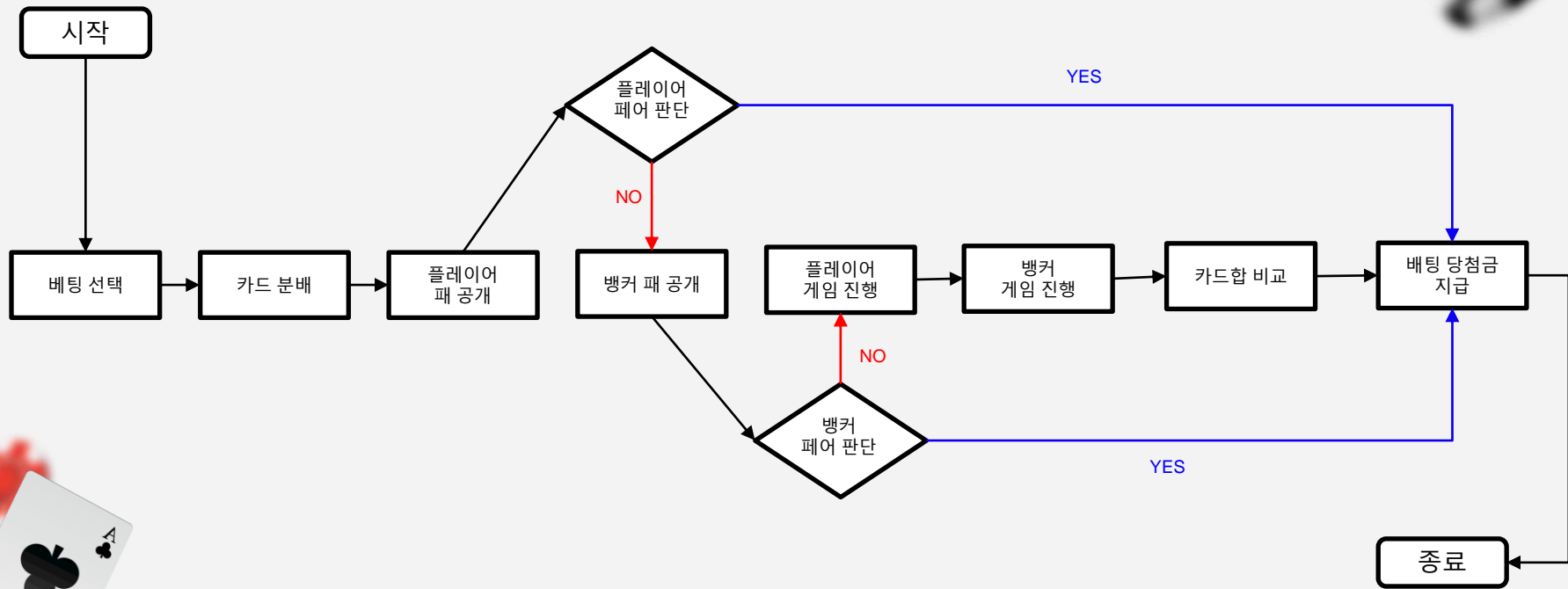
즉, 이 상태에서 바로 해당 라운드가 종료된다.
양쪽 모두 2장만으로 계산하여, 더 높은 쪽이 승리한다.

3. 양쪽 모두 내추럴이 아닐 경우, 플레이어가 받은 2장의 수의 합을 봐서 다음과 같은 조건에 따라 추가 카드를 받거나 안 받거나 한다.

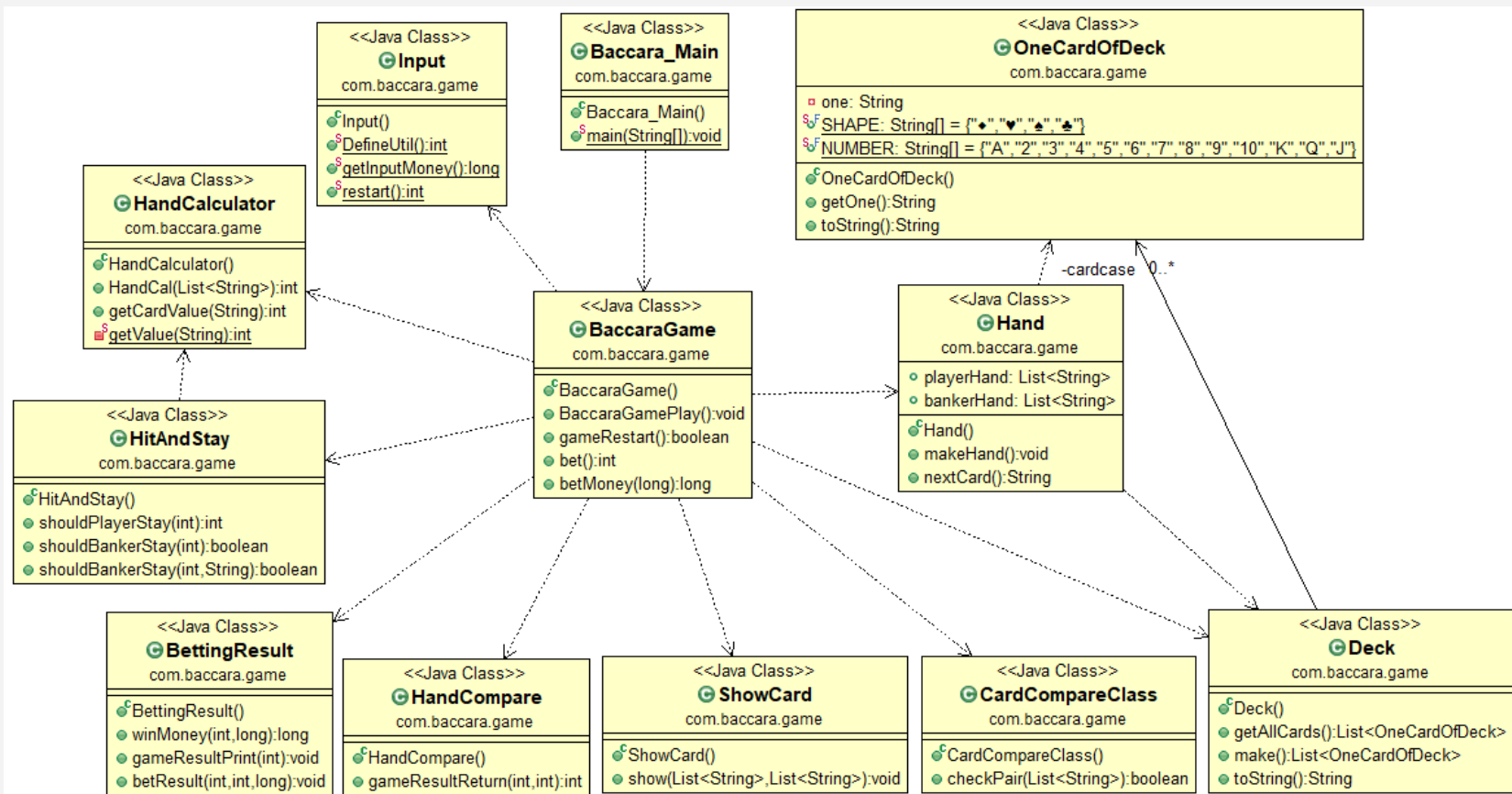
플레이어 수의 합	결정권
6 또는 7	스탠드(Stand) : 카드를 받지 않음
0~5	카드를 추가로 1장 받는다.

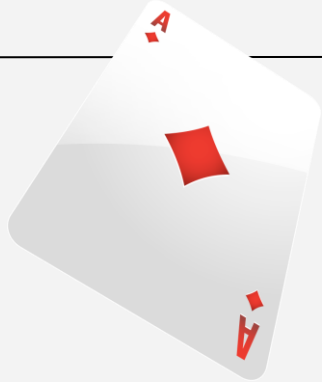
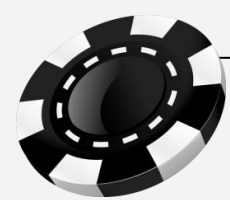
		플레이어의 3번째 카드									
		0	1	2	3	4	5	6	7	8	9
뱅크의 카드 합	7	스탠드									
	6	스탠드					드로우			스탠드	
	5	스탠드				드로우					스탠드
	4	스탠드		드로우						스탠드	
	3	드로우								스탠드	드로우
	2	드로우									
	1	플레이어의 3번째 카드에 관계 없이 드로우									
	0	드로우									

순서도



Class Diagram





기능 소개



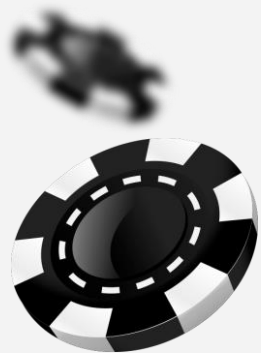
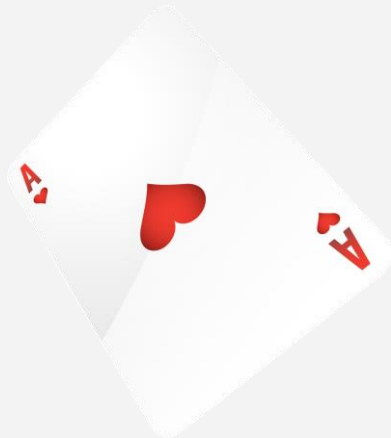
기능 소개

입력 기능

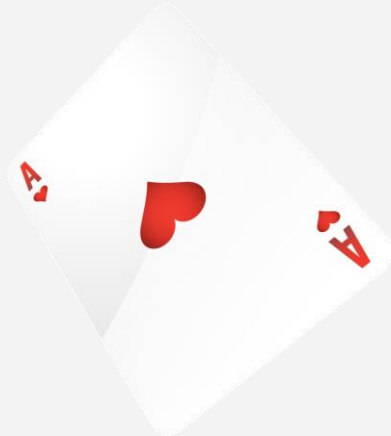
- 게임시작 - 베팅 입력
- 플레이어, 뱅커, 타이, 페어 종료 중 택 1
- 베팅 금액 입력, 입력 범위 제한

원하는 곳에 배팅을 선택해 주십시오
종료하시기를 원하시면 '5'를 눌러주세요
1. Player , 2. Banker 3. TIE 4. PAIR
배팅을 선택해주시시오
1
배팅 금액을 입력해주세요
50000
당신의 배팅금은 : 50000

원하는 곳에 배팅을 선택해 주십시오
1. Player , 2. Banker 3. TIE 4. PAIR
배팅을 선택해주시시오
1
배팅 금액을 입력해주세요
300000000
당신의 배팅금은 : 300000000
소지금을 넘는 배팅은 불가능합니다.
배팅 금액을 입력해주세요



-



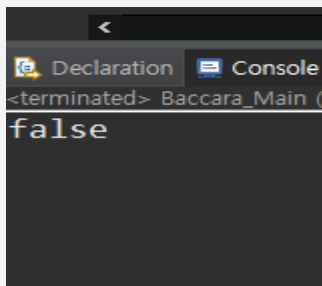
[47, +5, 46, V9, +7, 45, 45, 49, V1, 46, V3, +0, +6, 40, 41, V10, +3, V6, +10, 49, 4K, V8, V0, 4A, +8, 4K, VK, 48,

기능 소개

만들어진 패의 룰에 따른 판단

- 출력한 카드들의 패어 판단
- 플레이어의 합에 따른 플레이어 추가 행동
- 플레이어의 합에 따른 뱅커의 추가 행동

Ex) 패어 판단



```
public static void main(String[] args) {  
    List<String> test = new ArrayList<>();  
    test.add("♥7");  
    test.add("♥8");  
    CardCompareClass cc = new CardCompareClass();  
    System.out.println(cc.checkPair(test));  
}
```

```
-----플레이어 카드 오픈-----  
Player: ♥7 *  
Banker: * *  
-----뱅커 카드 오픈-----  
Player: ♥7 *  
Banker: ♥7 *  
-----플레이어 카드 오픈-----  
Player: ♥7 ♥7  
Banker: ♥7 *  
-----뱅커 카드 오픈-----  
Player: ♥7 ♥7  
Banker: ♥7 ♥7  
페어 승리  
페어 입니다.  
베팅에 성공 하셨습니다.  
현재 소지금: 210000원  
게임을 다시 시작하시겠습니까?  
[1]네 [2]아니요
```

기능 소개

출력 기능

- 출력된 결과에 따른 배팅 반환값 계산

```
*/
public class UserInfo extends BacaraGame{

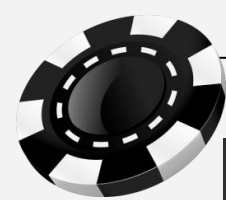
    BacaraGame game = new BacaraGame();
    public int userBetNum = n;
    public double userBetMoney = 500000;
    //      betAmount;

    /**
     * 게임 결과를 int로 받아 user가 배팅한 점수와 비교하여 user가 배팅한 금액을 계산하는 메소드
     * @param gameResult = 게임 결과
     * @return (long)result
     */
    public long winMoney(int gameResult) {
        double result = 0;
        switch (gameResult) {
            case 1:
                result = userBetMoney * 2;
                System.out.println("플레이어 승리");
                break;
            case 2:
                result = userBetMoney * 1.95;
                System.out.println("뱅크 승리");
                break;
            case 3:
                result = userBetMoney * 9;
                System.out.println("타이 승리");
                break;
            case 4:
                result = userBetMoney * 12;
                System.out.println("페어 승리");
                break;
            default:
                break;
        }
        return (long)result;
    }
    // 게임 결과가 뱅크 승리고 가져온 userBetMoney는 500000으로 테스트
    UserInfo userInfo = new UserInfo();
    System.out.println(userInfo.winMoney(2));
}
```

뱅크 승리
975000

소스코드





Input 1 – 사용자로부터 입력을 받는 클래스

```
public class Input {
```

```
    /**
```

```
     * 배팅을 선택하기 위한 숫자를 입력받는 메소드
```

```
     * 1. 플레이어 2. 뱅커 3. 타이 4. 페어
```

```
     */
```

```
    public static int DefineUtil() {
```

```
        @SuppressWarnings("resource")
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        int userInput;
```

```
        while (true) {
```

```
            System.out.println("배팅을 선택해주시오");
```

```
            if (scanner.hasNextInt()) {
```

```
                userInput = scanner.nextInt();
```

```
                if (userInput >= 1 && userInput <= 5) { // 선택지 안의 숫자가 입력될 시 진행  
                    break;
```

```
                }
```

```
            }
```

```
            scanner.nextLine();
```

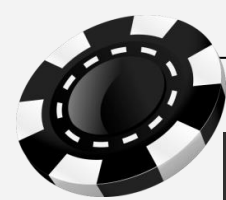
```
            System.out.println("잘못된 입력입니다. 재입력 해주시오."); // 선택지 바깥을 입력했을 경우 재입력
```

```
        }
```

```
        return userInput;
```

```
    }
```





Input 2 – 사용자로부터 입력을 받는 클래스

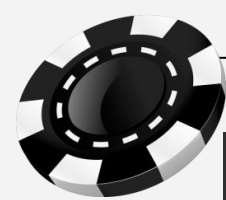


```
public static long getInputMoney() {
    Scanner scanner = new Scanner(System.in);
    long inputMoney = 0;

    try {
        System.out.println("배팅 금액을 입력해주세요");
        inputMoney = scanner.nextLong();
    } catch (InputMismatchException e) {
        System.out.println("정수를 입력해주세요");
    }
    System.out.println("당신의 배팅금은: " + inputMoney);
    return inputMoney;
}

/**
 * 정수 값을 입력받아 게임을 재시작하는 메소드
 */
public static int restart() {
    Scanner scanner = new Scanner(System.in);
    int select = 0;
    try {
        System.out.println("게임을 다시 시작하시겠습니까?");
        System.out.println("    [1] 네    [2] 아니요");
        select = scanner.nextInt();
    } catch (InputMismatchException e) {
        System.out.println("정수를 입력해주세요");
    }
    return select;
}
```





OneCardOfDeck – 정해진 문양과 카드를 조합하는 클래스

```
public class OneCardOfDeck {
    private String one;

    public static final String[] SHAPE = {"♦", "♥", "♠", "♣"};
    public static final String[] NUMBER = {"A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "K", "Q", "J"};

    public OneCardOfDeck() {
        int s = (int) (Math.random() * SHAPE.length);
        int n = (int) (Math.random() * NUMBER.length);
        one = SHAPE[s] + NUMBER[n];
    }

    public String getOne() {
        return one;
    }

    @Override
    public String toString() {
        return String.format("%s", one);
    }
}
```



Deck - 카드를 받아서 덱을 구성(8덱)하는 클래스

/* OneCardOfDeck 클래스에서 카드를 받아 덱을 구성하는 클래스

*/

```
public class Deck {
```

```
    private List<OneCardOfDeck> cardcase; //덱
```

```
    public Deck() {
```

```
        cardcase= new ArrayList<OneCardOfDeck>();
```

```
        make();
```

```
    }
```

```
    public List<OneCardOfDeck> getAllCards() {
```

```
        return cardcase;
```

```
    }
```

```
    public List<OneCardOfDeck> make() { //덱 생성 메소드
```

```
        int cnt=0;
```

```
        while (true) {
```

```
            OneCardOfDeck card = new OneCardOfDeck();
```

```
            if(!cardcase.contains(card)) { //랜덤으로 받은 카드 중복체크
```

```
                cardcase.add(card);
```

```
                cnt++;
```

```
            }
```

```
            if(cnt==OneCardOfDeck.SHAPE.length*OneCardOfDeck.NUMBER.length) {
```

```
                for (int i = 0; i < 3; i++) { //8개의 덱으로 구성된 하나의 덱을 만들어줌
```

```
                    cardcase.addAll(cardcase);
```

```
                }
```

```
                break;
```

```
            }
```

```
        }
```

```
        return cardcase;
```


```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Deck [make()=" + make() + " ]";
```

```
    }
```



Hand – 덱에 있는 카드를 플레이어와 뱅커에게 부여하는 클래스




```
import java.util.ArrayList;

/**
 * 이번 게임에서 사용될 덱을 생성하고 플레이어와 뱅커에게 카드를 뽑아 손패에 넣어주는 클래스
 */
public class Hand {
    public List<String> playerHand; //플레이어의 손패
    public List<String> bankerHand; //뱅커의 손패


    /**
     * 뽑은 카드를 각 손패에 한장씩 번갈아가며 나눠주는 메소드
     */
    public void makeHand() {
        playerHand = new ArrayList<String>();
        bankerHand = new ArrayList<String>();
        playerHand.add(nextCard());
        bankerHand.add(nextCard());
        playerHand.add(nextCard());
        bankerHand.add(nextCard());
    }

    /**
     * 덱을 생성하고 덱에서 한장을 뽑은 뒤, 뽑은 카드는 덱에서 제거하는 메소드
     * @return 덱에서 뽑은 카드
     */
    public String nextCard() {
        Deck dk = new Deck();
        List<OneCardOfDeck> deck = dk.getAllCards(); // 덱 생성
        String nextCard = deck.get(0).toString(); // 카드 한 장 뽑기
        deck.remove(0); // 덱에서 뽑은 카드를 지우기
        return nextCard;
    }
}
} //class
```





HandCalculator – 입력 받은 패의 합을 바카라 스코어로 판단하는 클래스

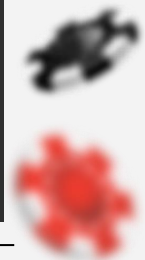



```
/**
 * 입력 받은 손패의 합을 구하는 클래스
 */
public class HandCalculator {

    //입력받은 손패의 합을 길이와 상관없이 계산해주는 메소드
    public int HandCal(List<String> hand) {
        int total = 0;
        for (int i = 0; i < hand.size(); i++) {
            String oneCard= hand.get(i);
            total += getValue(oneCard);
        }
        return total%10;// 카드의 합을 바카라 계산법에 맞게 1의자리숫자만 반환
    }

    //get value의 값을 반환해주는 메소드
    public int getCardValue(String card) {
        return getValue(card);
    }

    //입력받은 카드 한장의 int 값을 반환해주는 메소드
    private static int getValue(String card) {
        String rank = card.substring(1);
        int result=switch (rank) {
            case "A"->1;
            case "K", "Q", "J"->10;
            default->Integer.parseInt(rank);
        };
        return result;
    }
}
```







CardCompareClass – 페어 여부를 판단하는 클래스

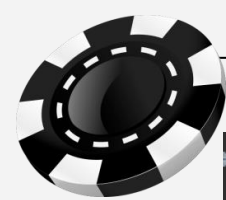


```
import java.util.List;

/**
 * 페어 체크 클래스
 */
public class CardCompareClass {

    /**
     * 패를 받아서 패의 2장의 카드의 수나 문자를 비교해서 페어를 판단하는 메소드
     * @return true 페어
     */
    public boolean checkPair(List<String> hand) {
        boolean isc = false;
        char hand1Num = hand.get(0).charAt(1);
        char hand2Num = hand.get(1).charAt(1);
        // charAt을 사용해 잘려진 인자를 비교해 같다면 true 반환 :하트1:k, :스페이드:k -> k, k
        if (hand1Num == hand2Num) {
            isc = true;
        }
        return isc;
    }
}
```






HandCompare – 바카라게임 승리 판단 클래스



```
/**
 * 플레이어와 뱅커의 패를 비교하는 클래스
 */
public class HandCompare {

    //게임의 결과를 판단하는 메소드
    public int gameResultReturn(int playerHand, int bankerHand ) {
        char result=0;
        if(playerHand>bankerHand) { //플레이어 승리
            result=1;
        }else if(playerHand<bankerHand) { //뱅크 승리
            result= 2;
        }else if(playerHand==bankerHand) { //타이
            result=3;
        }
        return result;
    }
}
```



HitAndStay 1 – 바카라 룰에 따른 히트&스테이를 판단하는 클래스

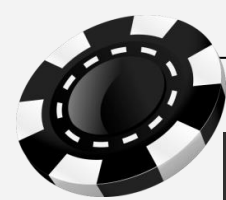


```
/**
 * 플레이어와 뱅커의 추가행동을 결정하는 클래스
 */
public class HitAndStay {

    // 플레이어가 Hit 또는 Stay 판단 메소드
    public int shouldPlayerStay(int playerTotal) { // 토탈 값을 인수로 받아서 비교 연산
        int result = 0;
        if (playerTotal >= 8) { // 네추럴
            result = 1; // 플레이어 또는 뱅커가 8 또는 9인 경우, 더 이상 카드를 뽑지 않음
        } else if (playerTotal <= 5) { // 히트
            result = 2; // 플레이어의 총점이 5 이하인 경우, 카드를 더 뽑아야 함
        } else { // 스탠드
            result = 3; // 그 외의 경우, 플레이어는 카드를 더 뽑지 않음
        }
        return result;
    }

    /**
     * 플레이어가 스탠드 했을 때의 행동 메소드
     * @return true는 뱅커가 카드를 뽑는다
     */
    public boolean shouldBankerStay(int bankerTotal) {
        System.out.println("플레이어 스테이");
        if (bankerTotal <= 5) {
            System.out.println("뱅크 히트");
            return true; // 플레이어가 Stay를 선택하고, 뱅커의 총점이 5 이하인 경우, 카드를 뽑음
        }
        else {
            System.out.println("뱅크 스테이");
            return false;
        }
    }
}
```






HitAndStay 2




```
/**
 * 플레이어가 히트 했을 때의 행동 메소드
 * @return true는 뱅커가 카드를 뽑는다
 */
public boolean shouldBankerStay(int bankerTotal, String playerThirdCard) {
    HandCalculator hc = new HandCalculator();
    int ptc = hc.getCardValue(playerThirdCard);
    //플레이어 히트
    System.out.println("플레이어 히트");
    if (bankerTotal <= 2) {
        System.out.println("뱅크 스탠드");
        return false; // 뱅커의 총점이 2 이하인 경우, 더 이상 카드를 뽑지 않음
    } else if (bankerTotal == 3 && ptc != 8) {
        System.out.println("뱅크 스탠드");
        return false; // 뱅커의 총점이 3이고, 플레이어의 총점이 8이 아닌 경우, 더 이상 카드를 뽑지 않음
    } else if (bankerTotal == 4 && ptc >= 2 && ptc <= 7) {
        System.out.println("뱅크 스탠드");
        return false; // 뱅커의 총점이 4이고, 플레이어의 총점이 2 이상 7 이하인 경우, 더 이상 카드를 뽑지 않음
    } else if (bankerTotal == 5 && ptc >= 4 && ptc <= 7) {
        System.out.println("뱅크 스탠드");
        return false; // 뱅커의 총점이 5이고, 플레이어의 총점이 4 이상 7 이하인 경우, 더 이상 카드를 뽑지 않음
    } else if (bankerTotal == 6 && ptc >= 6 && ptc <= 7) {
        System.out.println("뱅크 스탠드");
        return false; // 뱅커의 총점이 6이고, 플레이어의 총점이 6 이상 7 이하인 경우, 더 이상 카드를 뽑지 않음
    }
    System.out.println("뱅크 히트");
    return true; // 그 외의 경우, 뱅커는 카드를 뽑음
}
```







BettingResult 1 – 선택지에 따른 배팅값을 반환하는 클래스




```
public class BettingResult {
    /**
     * 게임 결과를 int로 받아 user가 베팅한 정수와 비교하여 user가 베팅한 금액을 계산하는 메소드
     *
     * @param gameResult = 게임 결과, vrvr = 유저가 베팅한 금액
     * @return long result
     */
    public long winMoney(int gameResult, long vrvr) { // gameResult 는 1~4중 하나
        long result = 0;
        switch (gameResult) {
            case 1:
                result = vrvr * 2;
                System.out.println("플레이어 승리");
                break;
            case 2:
                result = vrvr * 2;
                System.out.println("뱅크 승리");
                break;
            case 3:
                result = vrvr * 9;
                System.out.println("타이 승리");
                break;
            case 4:
                result = vrvr * 12;
                System.out.println("페어 승리");
                break;
            default:
                break;
        }
        return result;
    }
}
```





BettingResult 2






```
public void gameResultPrint(int GameResult) {
    if (GameResult == 1) {
        System.out.println("-----플레이어가 승리 하였습니다-----");
    } else if (GameResult == 2) {
        System.out.println("-----뱅크가 승리 하였습니다-----");
    } else if (GameResult == 3) {
        System.out.println("-----타이 입니다-----");
    } else if (GameResult == 4) {
        System.out.println("-----페어 입니다-----");
    }
}

/**
 * 게임의 결과, 유저 베팅, 유저 소지금의 최종 결과를 출력하는 메소드
 *
 * @param (int)GameResult
 * @param (int)n
 * @param (long)totalMoney
 */
public void betResult(int GameResult, int n, long totalMoney) {

    if (n == GameResult) {
        System.out.println("▲( < )▶ 베팅에 성공 하셨습니다 ▲( < )▶\n 현재 소지금:" + totalMoney + "원");

    } else {
        System.out.println("({>^<}) 베팅에 실패 하셨습니다.({>^<})\n 현재 소지금:" + totalMoney + "원");
    }
}
```





ShowCard – 유저에게 보이는 화면을 출력하는 클래스



```
public class ShowCard {
    public void show(List<String> playerHand, List<String> bankerHand ) throws InterruptedException {

        String playerCard1 = playerHand.get(0);
        String playerCard2 = playerHand.get(1);
        String bankerCard1 = bankerHand.get(0);
        String bankerCard2 = bankerHand.get(1);

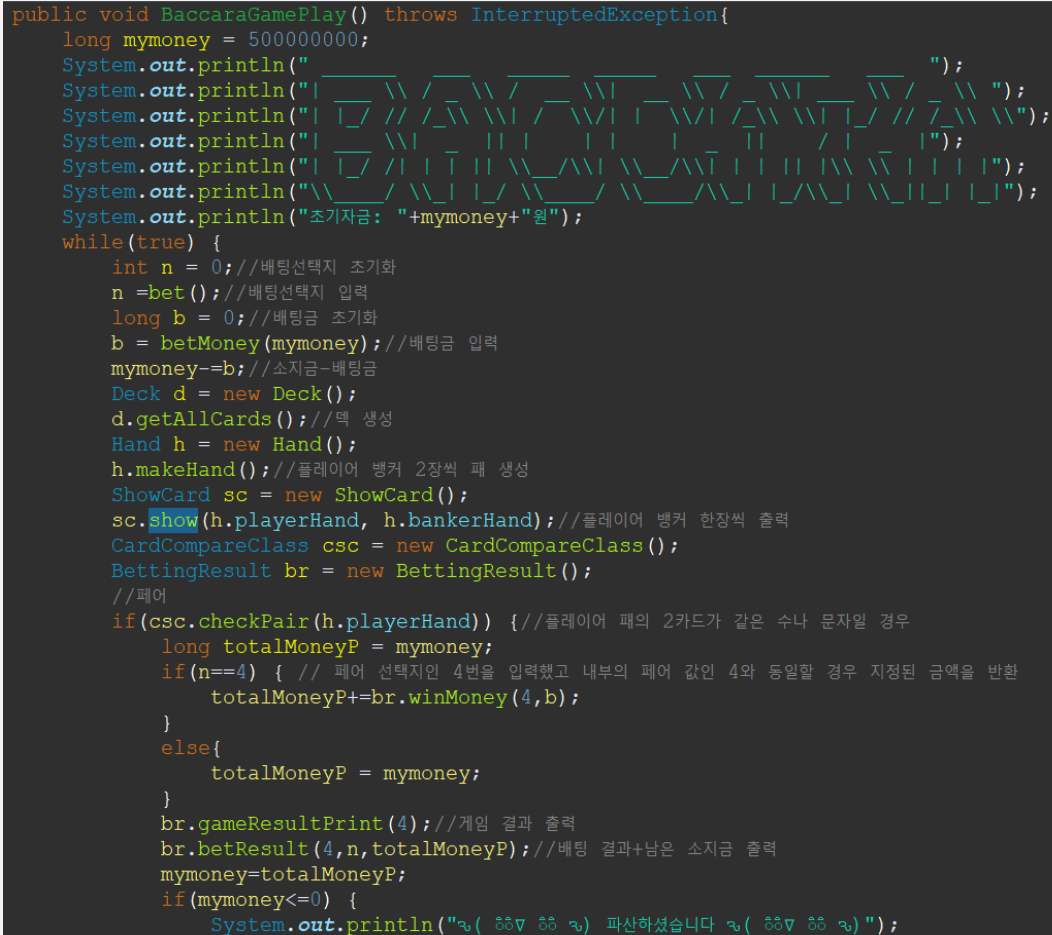
        System.out.println("-----플레이어 카드 오픈-----");
        System.out.println("Player: " + playerCard1 + " " + "*");
        System.out.println("Banker: " + " * " + " " + "*");
        Thread.sleep(2000);
        System.out.println("-----뱅크어 카드 오픈-----");
        System.out.println("Player: " + playerCard1 + " " + "*");
        System.out.println("Banker: " + bankerCard1 + " " + "*");
        Thread.sleep(2000); // sleep을 사용해 카드를 나중에 오픈하는 효과를 보여줌
        System.out.println("-----플레이어 카드 오픈-----");
        System.out.println("Player: " + playerCard1 + " " + playerCard2);
        System.out.println("Banker: " + bankerCard1 + " " + "*");
        Thread.sleep(2000);
        System.out.println("-----뱅크어 카드 오픈-----");
        System.out.println("Player: " + playerCard1 + " " + playerCard2);
        System.out.println("Banker: " + bankerCard1 + " " + bankerCard2);

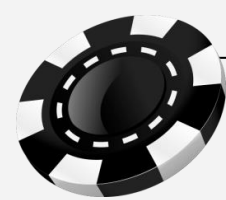
    }
}
```





바카라 게임진행 클래스



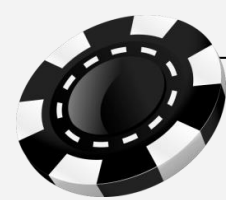


BaccaraGamePlay 2

바카라 게임진행 클래스



```
br.betResult(4,n,totalMoneyP); //배팅 결과+남은 소지금 출력
mymoney=totalMoneyP;
if (mymoney<=0) {
    System.out.println("ㄹ( ㄹㄹ▽ ㄹㄹ ㄹ) 파산하셨습니다 ㄹ( ㄹㄹ▽ ㄹㄹ ㄹ)");
    System.out.println("플레이해주셔서 감사합니다");
    break;
}
if(gameRestart()) { //게임재시작여부
    continue;
}
System.out.println("플레이해주셔서 감사합니다");
break;
}
else if(csc.checkPair(h.bankerHand)) { //페어 선택지인 4번을 입력했고 내부의 1
    long totalMoneyP = mymoney;
    if(n==4) {
        totalMoneyP+=br.winMoney(4,b);
    }
    else{
        totalMoneyP = mymoney;
    }
    br.gameResultPrint(4); //게임 결과 출력
    br.betResult(4,n,totalMoneyP); //배팅 결과+남은 소지금 출력
    mymoney=totalMoneyP;
    if (mymoney<=0) {
        System.out.println("ㄹ( ㄹㄹ▽ ㄹㄹ ㄹ) 파산하셨습니다 ㄹ( ㄹㄹ▽ ㄹㄹ ㄹ)");
        System.out.println("플레이해주셔서 감사합니다");
        break;
    }
    if(gameRestart()) { //게임재시작여부
        continue;
    }
    System.out.println("플레이해주셔서 감사합니다");
    break;
}
}
HandCalculator hc = new HandCalculator();
int playerHandSum = hc.HandCal(h.playerHand); //플레이어 패합을 계산
```



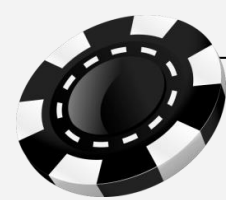
BaccaraGamePlay 3

바카라 게임진행 클래스



```
int playerHandSum = hc.HandCal(h.playerHand); //플레이어 패합을 계산
HitAndStay has = new HitAndStay(); // 플레이어와 뱅커의 패의 합에 따라 hit와 stay를 결정하는 인스턴스
HandCalculator hs = new HandCalculator(); //받은 패의 합을 계산하는 인스턴스
HandCompare hcl = new HandCompare(); // 게임의 승부를 결정짓는 인스턴스
switch (has.shouldPlayerStay(playerHandSum)) { //플레이어 패합에 따라 플레이어 추가행동
case 1:// 네추럴
    int grn = hcl.gameResultReturn(hs.HandCal(h.playerHand), hs.HandCal(h.bankerHand));
    long totalMoney = mymoney;
    if(grn==n) { //사용자가 배팅을 올바르게 맞췄을 때
        totalMoney += br.winMoney(grn,b); //소지금에 배팅성공반환금을 더해준다
    }else {
        totalMoney=mymoney;
    }
    br.gameResultPrint(grn); //게임 결과 출력
    br.betResult(grn,n,totalMoney); //배팅 결과+남은 소지금 출력
    mymoney=totalMoney;
    break;

case 2:// 플레이어가 히트
    String 플레이어다음카드 =h.nextCard();
    h.playerHand.add(플레이어다음카드); // 플레이어 히트일 경우 랜덤한 패 한 장을 추가
    if(has.shouldBankerStay(hs.HandCal(h.bankerHand), 플레이어다음카드)) {
        String 뱅커다음카드 =h.nextCard(); // 플레이어 3번째 카드를 룰에 의해 뱅커도 hit 판단
        h.bankerHand.add(뱅커다음카드);
    }
    Thread.sleep(2000);
    System.out.println("-----최종 카드 패-----");
    System.out.println("Player: " + h.playerHand);
    System.out.println("Banker: " + h.bankerHand);
    long totalMoney1 = mymoney;
    int gr = hcl.gameResultReturn(hs.HandCal(h.playerHand), hs.HandCal(h.bankerHand));
    if(gr==n) { //사용자가 배팅을 올바르게 맞췄을 때
        totalMoney1 +=br.winMoney(gr,b); //소지금에 배팅성공반환금을 더해준다
    }else {
        totalMoney1=mymoney;
    }
    br.gameResultPrint(gr); //게임 결과 출력
```

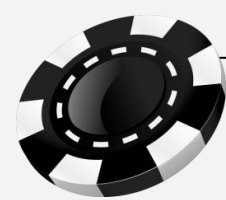


BaccaraGamePlay 4

바카라 게임진행 클래스



```
br.gameResultPrint(gr); //게임 결과 출력
br.betResult(gr,n,totalMoney1); //배팅 결과+남은 소지금 출력
mymoney=totalMoney1;
break;
case 3: // 플레이어가 스탠드
    if(has.shouldBankerStay(hs.HandCal(h.bankerHand))) { // 게임 룰에 의해 첫 패의 합을 보고 stay판단 했을경우
        String 뱅커다음카드 =h.nextCard();
        h.bankerHand.add(뱅커다음카드);
    }
    Thread.sleep(2000);
    System.out.println("-----최종 카드 패-----");
    System.out.println("Player: " + h.playerHand);
    System.out.println("Banker: " + h.bankerHand);
    long totalMoney2 = mymoney;
    int grs = hcl.gameResultReturn(hs.HandCal(h.playerHand), hs.HandCal(h.bankerHand));
    if(grs==n) { //사용자가 배팅을 올바르게 맞췄을 때
        totalMoney2 +=br.winMoney(grs,b); //소지금에 배팅성공반환금을 더해준다
    } else {
        totalMoney2=mymoney;
    }
    br.gameResultPrint(grs); //게임 결과 출력
    br.betResult(grs,n,totalMoney2); //배팅 결과+남은 소지금 출력
    mymoney=totalMoney2;
    break;
}
if(mymoney<=0) {
    System.out.println("ㄱ( ㉹ ㉹ ㄱ) 파산하셨습니다 ㄱ( ㉹ ㉹ ㄱ)");
    System.out.println("플레이해주셔서 감사합니다");
    break;
}
if(gameRestart()) { //게임재시작여부
    continue;
}
System.out.println("플레이해주셔서 감사합니다");
break;
```



BaccaraGamePlay 5

바카라 게임진행 클래스

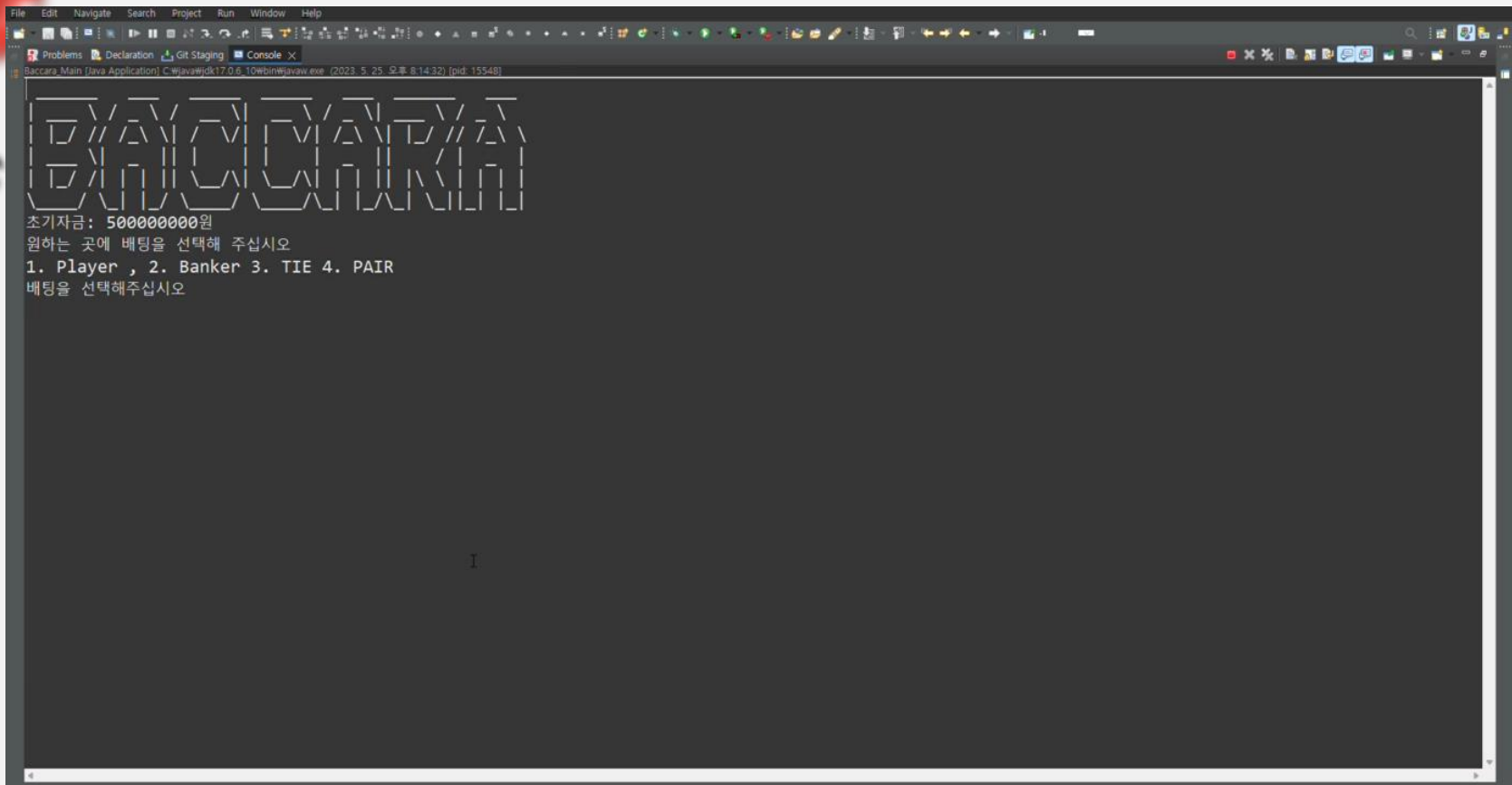


```
}  
public boolean gameRestart () { // 1번 입력시 게임 재시작 2번 입력시 게임 종료  
    boolean a = false;  
    int select = Input.restart();  
    if(select == 1) {  
        a = true;  
    }  
    else if(select == 2) {  
        a = false;  
    }  
    return a;  
}  
  
public int bet () { // 선택지를 입력받는 스캔 메소드  
    int choice = 0;  
    System.out.println("원하는 곳에 배팅을 선택해 주십시오");  
    System.out.println("1. Player , 2. Banker 3. TIE 4. PAIR ");  
    choice = Input.DefineUtil();  
    return choice;  
}  
  
public long betMoney(long mymoney) { //배팅금액을 입력받고 반환하는 메소드  
    long ba = 0;  
    while(true) {  
        ba = Input.getInputMoney();  
        if (ba > mymoney) {  
            System.out.println("소지금을 넘는 배팅은 불가능합니다.");  
            continue;  
        }  
        else {  
            break;  
        }  
    }  
    return ba;  
}  
}
```



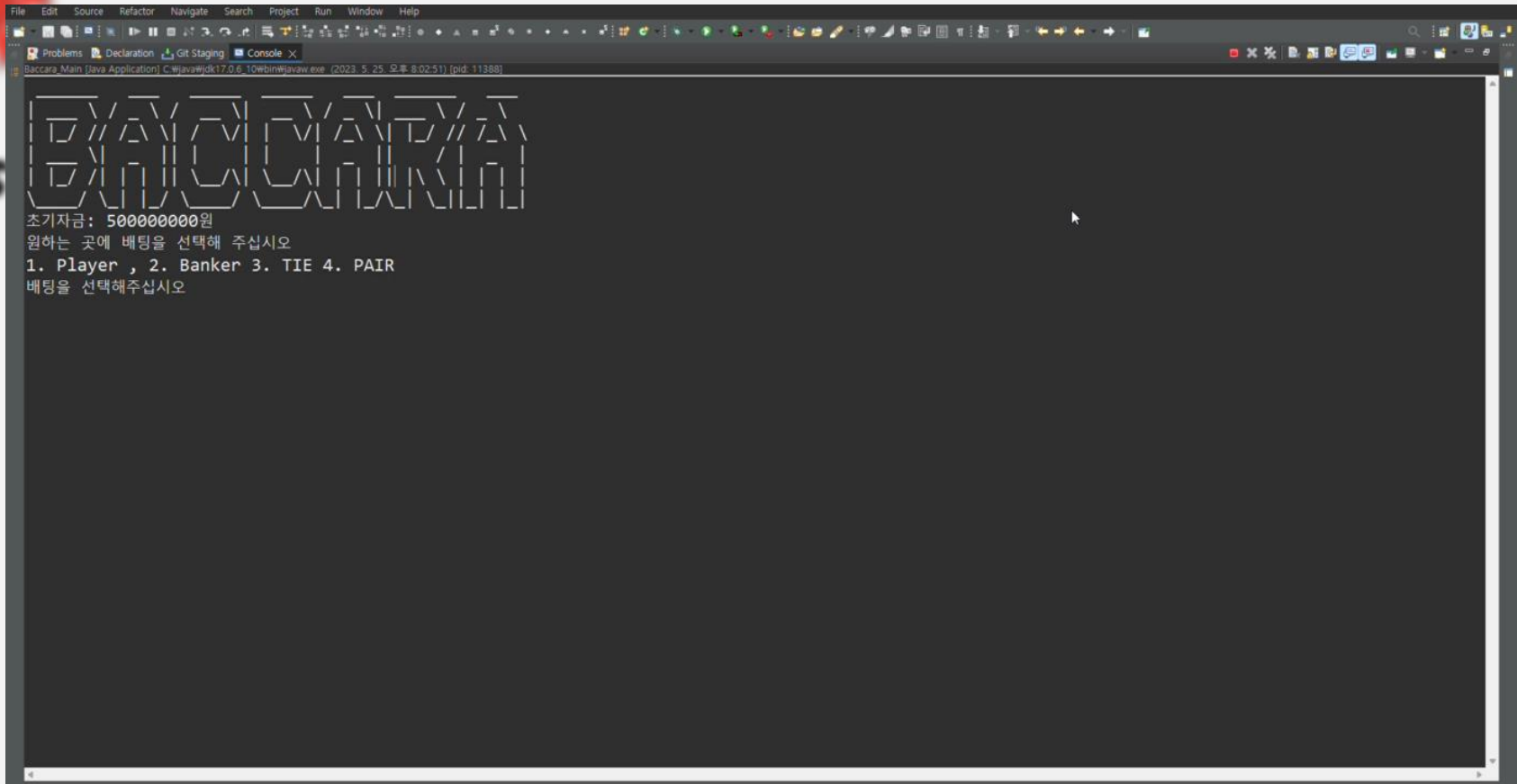
프로그램 시연





```
File Edit Navigate Search Project Run Window Help
Baccara_Main [Java Application] C:\j\java\jdk17.0.6_10\bin\javaw.exe (2023. 5. 25. 오후 8:14:32) [pid: 15548]

BACCARA
초기자금: 500000000원
원하는 곳에 배팅을 선택해 주십시오
1. Player , 2. Banker 3. TIE 4. PAIR
배팅을 선택해 주십시오
```



```
File Edit Source Refactor Navigate Search Project Run Window Help
Baccara_Main [Java Application] C:\j\javawjdk17.0.6\10\bin\javaw.exe (2023. 5. 25. 오후 8:02:51) [pid: 11388]
Baccara_Main [Java Application] C:\j\javawjdk17.0.6\10\bin\javaw.exe (2023. 5. 25. 오후 8:02:51) [pid: 11388]
초기자금: 500000000원
원하는 곳에 배팅을 선택해 주십시오
1. Player , 2. Banker 3. TIE 4. PAIR
배팅을 선택해주십시오
```

Thank you

