REPORT

동국대학교
dongguk university

- 과목명 : 객체지향프로그래밍
- 담당교수 : 엄진영 교수님
- 학과 : 컴퓨터공학과
- 학번 : 2019112130
- 이름 : 조양진

객체지향프로그래밍 11주차 실습과제 보고서

2019112130 조양진

문제 1.

Blackjack

- ✓ 딜러와 Player 2명이 존재한다.
- ✓ 카드는 조커를 포함한 53장이다.
- ✓ 2~10은 숫자 그대로 점수를, K/Q/J는 10점으로, A는 1 또는 11, 조커는 가장 유리한 경우로(1~11) 계산한다.
- ✓ 게임 시작과 동시에 딜러와 게이머는 순차적으로 카드를 하나씩 뽑아 각자 2개의 카드를 소지한다.
- ✓ 딜러부터 게임을 진행하며 딜러는 2카드의 합계 점수가 16점 이하이면 반드시 1장을 추가로 뽑고, 17점 이상이면 Stay 한다.
- ✓ Player는 얼마든지 카드를 추가로 뽑을 수 있다.
- ✓ 양쪽 다 Stay한 경우 딜러와 Player 중 소유한 카드의 합이 21에 가장 가까운 쪽이 승리한다.
- ✓ 단 21을 초과하면 초과한 쪽이 진다.
- ✓ 딜러와 Player의 카드는 전부 오픈되어 있으며 플레이어는 Blackjack의 확률을 계산할 수 있어야 한다.

- 문제 분석

블랙잭 게임을 만들어야 합니다. 우선 CPU대 플레이어로 1인 게임입니다. 또한 카드에는 조커 한 장이 추가되어 있고, 2~10은 숫자 그대로 점수, K,Q,J는 10점, A는 1 또는 11, 조커는 유리한 경우 (1~11)로 계산해야 합니다. 이는 점수 계산 알고리즘을 만들어서 각 상황마다 가장 유리한 상황으로 만들어야 합니다.

점수 계산 알고리즘은 다음과 같습니다.

1. 2~10, K,Q,J는 for문 안의 조건문으로 total 변수 안에 다 더하고, A의 개수, 조커의 유무를 확인합니다
2. A의 개수만큼 반복을 하며 total의 크기에 따라 11 혹은 1로 만들어 더합니다
3. 조커의 유무를 통해 조커가 있다면 가장 유리하게 total에 더합니다

우선 딜러에게 두 장, 플레이어에게 두 장을 먼저 뽑고, Hit, Stay, Analyze의 선택지를 플레이어에게 줍니다.

1. Hit

A. Hit은 다음 카드를 뽑는 것입니다. 이때 만약 딜러의 카드의 합이 17을 넘지 않거나 2장의 합이 16이하라면 추가적으로 카드를 뽑습니다.
B. 만약 플레이어의 카드 총 합이 21을 넘겼다면 hit을 하면 게임이 자동으로 끝납니다.

2. Stay
   A. 게임을 수동 종료하는 것입니다.
   B. 원래 블랙잭이라면 딜러의 카드가 숨겨져있어 중요하겠지만 모든 카드가 공개되어 있어 그렇게 중요하지 않게 되었습니다.
3. Analyze
   A. 게임의 현황을 한눈에 알 수 있게 합니다.
   B. 또한 문제에서 요구한 Blackjack의 확률을 계산할 수 있게끔 확률을 %로 표시합니다. (사실 blackjack이란 단어는 처음 두 장의 합이 21일때만 blackjack이라 하니, 여기선 21점을 만드는 확률로 말을 바꿔서 사용하겠습니다)

- 코드 설명 (week11-Blackjack.java)

static int playerTurn, dealerTurn – player, dealer의 카드를 뽑은 횟수를 저장하는 변수

static int[] cardSet – 게임에서 사용될 카드의 덱 입니다. 초기 데이터는 다음과 같습니다.
{4,4,4,4,4,4,4,4,4,4,4,4,4,1};
이는 순서대로 2,3,4,5,6,7,8,9,10,K,Q,J,A,Joker으로 조커를 제외한 모든 카드가 4장씩 (다이아, 스페이드 클로버, 하트) 있다는 뜻입니다. 만약 카드를 딜러나 플레이어가 뽑으면 저 배열에서 -1씩을 하는 방법으로 덱을 관리할 것입니다.

static boolean endGame - main 메소드에서 사용될 while문의 flag으로 사용됩니다.

static int[] dealerCardSet, playerCardSet – 각각 딜러의 카드패, 플레이어의 카드패입니다. 안에 cardSet배열의 index가 들어갑니다.

static int dealerCardPoint, playerCardPoint – 각각 딜러의 카드패의 점수, 플레이어의 카드패 점수입니다. 이는 카드를 뽑을 때마다 정해지는 것이 아닌, 다른 함수를 통해 딜러와 플레이어에게 가장 유리한 점수 계산 방식으로 정해질 것입니다.

public static void dealerDrawCard(), playerDrawCard() – 각각 딜러의 카드를 뽑고,

플레이어의 카드를 뽑는 메소드입니다. 여기서 중요한 점은 cardSet 배열에서 0이된 index는 다시 선택하지 않도록 recursive하게 구현하는 것입니다. 따라서 똑 같은 카드를 플레이어나 딜러가 다시 뽑지 않을 수 있습니다. ex) 플레이어가 뽑은 joker를 딜러가 다시 뽑는 일이 없어야 합니다

public static void printDealerCard(), printPlayerCard() – 각각 딜러와 플레이어의 카드를 출력하는 메소드입니다. dealerCardSet과 playerCardSet 배열을 사용하여 각 배열 안에 들은 값을 사용합니다. 우선 2~10에 해당하는 카드 (데이터로는 0~8)까지는 데이터에 +2를 해서 정수로 출력하면 됩니다. 데이터가 9, 10, 11, 12, 13이라면 else if 문을 통해 K, Q, J, A, Joker를 따로 출력해주면 됩니다.

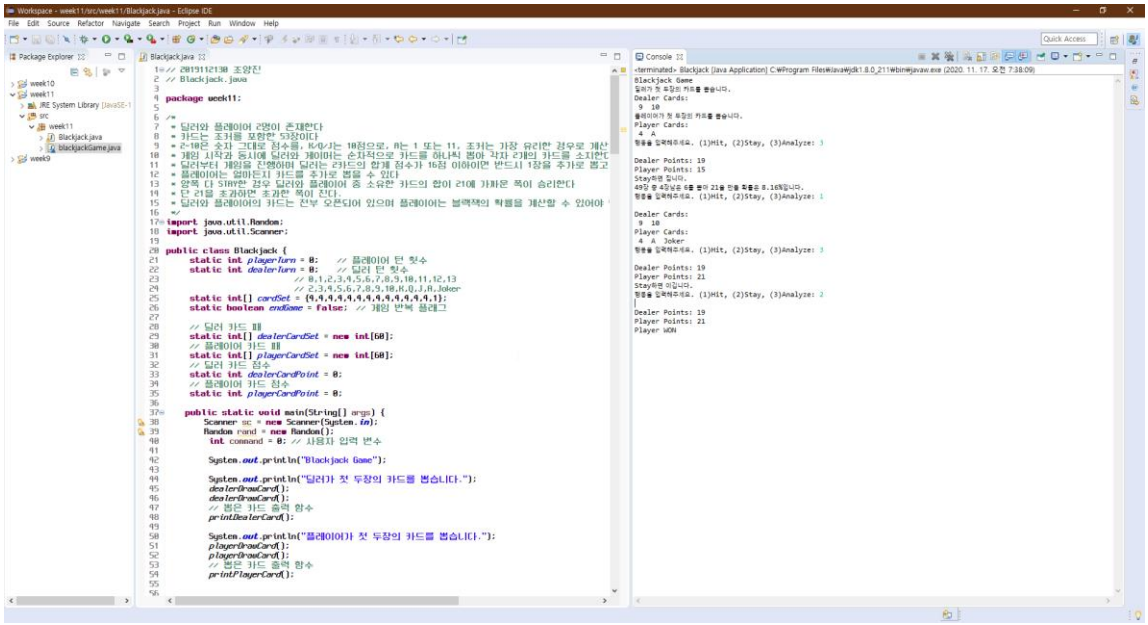
public static void playerAnalyze() – 문제에서 요구한 blackjack을 만들 수 있는 확률 계산 메소드입니다. (정확히는 blackjack이 아니라 점수 21을 만들 수 있는 확률을 구하는 메소드입니다) 우선 21을 만들기 위해 더 카드를 더 뽑아야 하는지 확인하고, 만들 수 있다면 각 케이스마다 덱에 남은 카드의 수를 분모로, 필요한 카드의 수를 분자로 하여 퍼센테이지로 나타냅니다. 또한 현재의 딜러와 플레이어의 점수 상황도 확인할 수 있습니다.

public static int calculate(int[] input, int turnCounter)- 이 과제에서 가장 중요하다고도 할 수 있는 카드 점수 계산 알고리즘입니다. 우선 변수로는 total (반환할 값을 저장합니다), aCounter (A 카드를 몇 개 뽑았는지 저장), isJokerIn (조커를 뽑았는지 유무) 변수를 사용합니다. 첫번째로 2~10, J, Q, K 카드는 조건문으로 바로 total에 더합니다. 겸사겸사 A 카드가 몇 개인지 aCounter 변수에 저장하고 조커를 뽑았다면 isJokerIn 변수를 true로 만듭니다. 두번째로 A의 개수를 저장해놨다가 1번 과정을 통해 나온 total 값과 비교하여 유리한 방식으로 더합니다. 예를 들면 total의 값이 11이상인데 A의 값을 11로 더하면 게임에서 지기 때문에, 1로 바꿔서 더합니다. 세번째로는 조커의 유무에 따라 조커가 있다면 가장 유리한 방식으로 total에 더해줍니다.

이 함수의 인자에는 dealerCardSet, playerCardSet이 int형 배열, dealerTurn, playerTurn이 int형 에 들어와서, 각각 dealerCardPoint와 playerCardPoint 변수에 저장됩니다.

public static void normalEndGame() – 게임을 마무리하고 딜러와 플레이어의 점수를 출력하고, 누가 승리했는지 그리고 어떻게 승리했는지를 출력합니다.

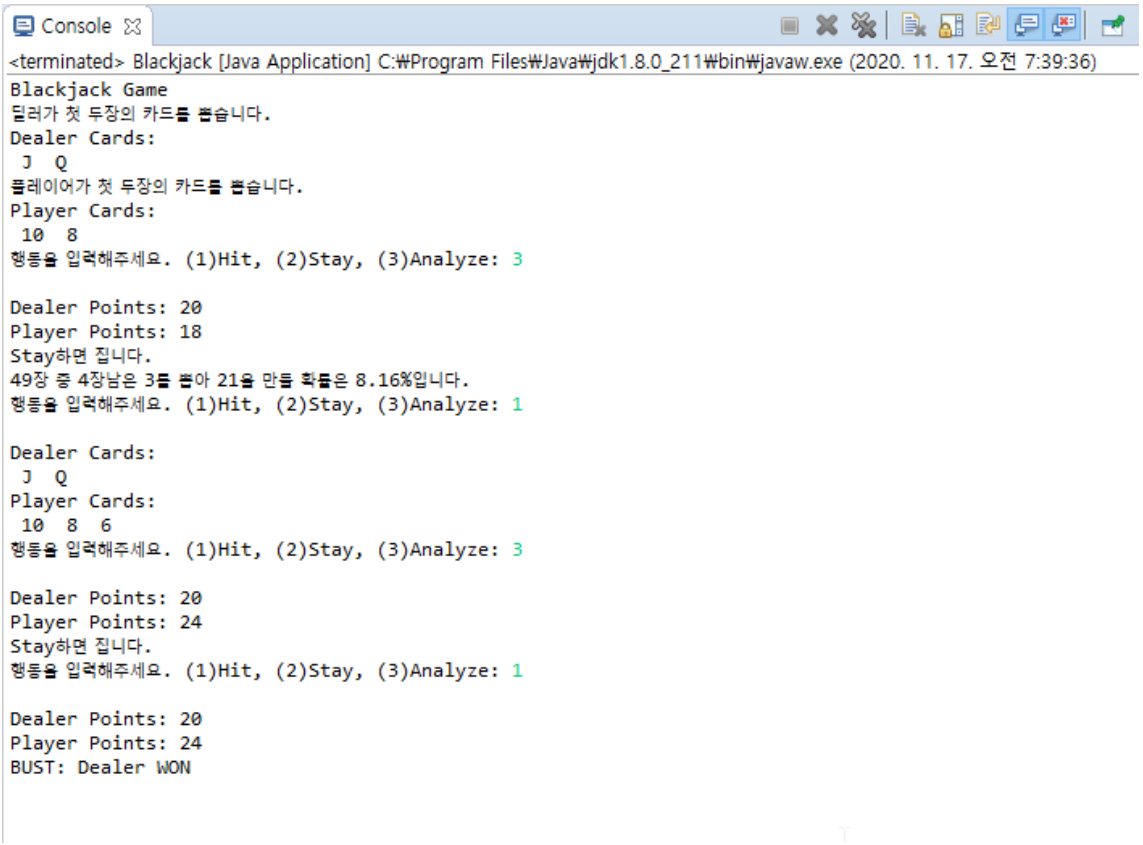
- 결과 및 분석



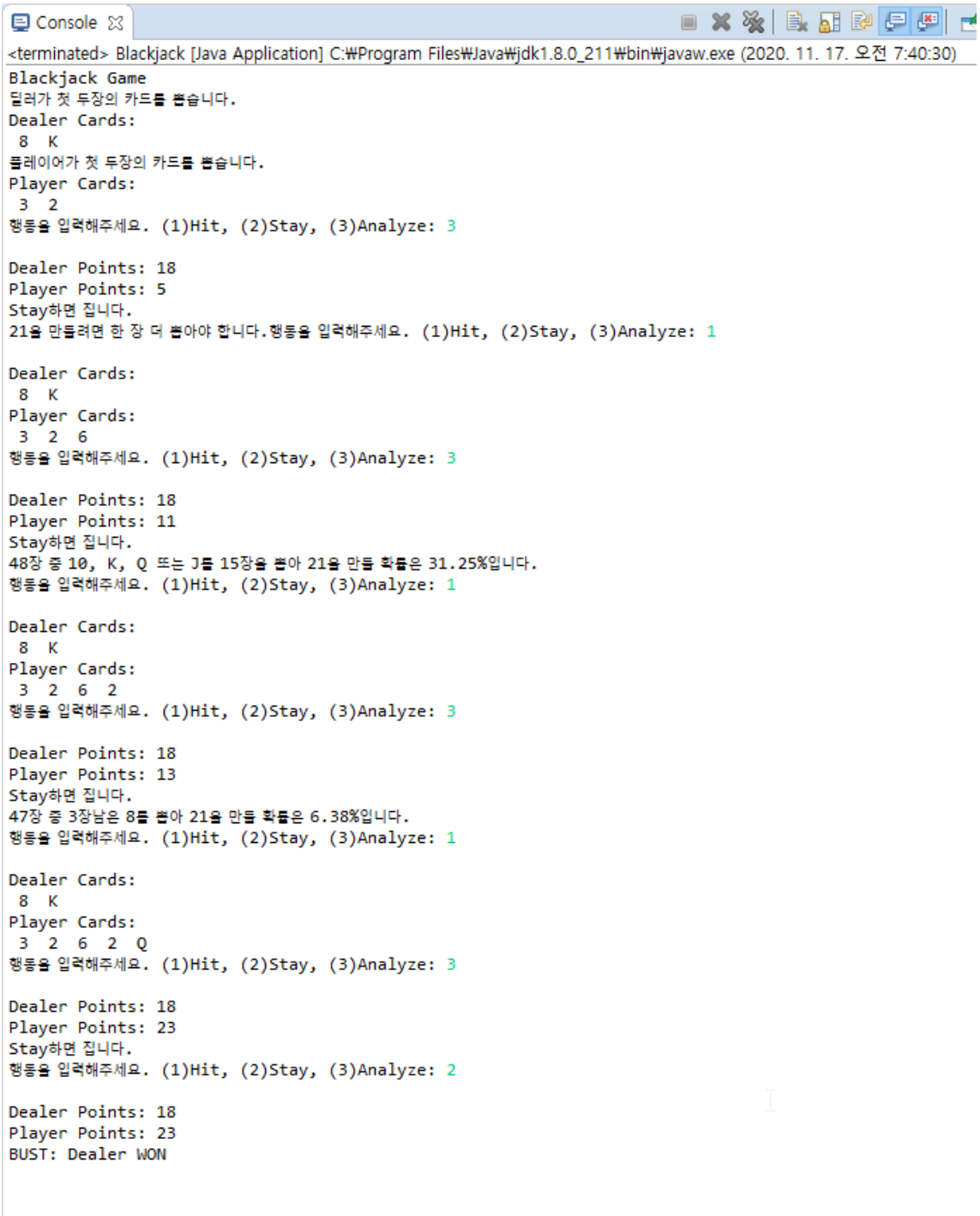
- **eclipse 상에서 실행한 모습**

```
Console
<terminated> Blackjack [Java Application] C:₩Program Files₩Java₩jdk1.8.0_211₩bin₩javaw.exe (2020. 11. 17. 오전 7:38:09)
Blackjack Game
딜러가 첫 두장의 카드를 뽑습니다.
Dealer Cards:
 9  10
플레이어가 첫 두장의 카드를 뽑습니다.
Player Cards:
 4  A
행동을 입력해주세요. (1)Hit, (2)Stay, (3)Analyze: 3

Dealer Points: 19
Player Points: 15
Stay하면 집니다.
49장 중 4장남은 6를 뽑아 21을 만들 확률은 8.16%입니다.
행동을 입력해주세요. (1)Hit, (2)Stay, (3)Analyze: 1

Dealer Cards:
 9  10
Player Cards:
 4  A  Joker
행동을 입력해주세요. (1)Hit, (2)Stay, (3)Analyze: 3

Dealer Points: 19
Player Points: 21
Stay하면 이깁니다.
행동을 입력해주세요. (1)Hit, (2)Stay, (3)Analyze: 2

Dealer Points: 19
Player Points: 21
Player WON
```

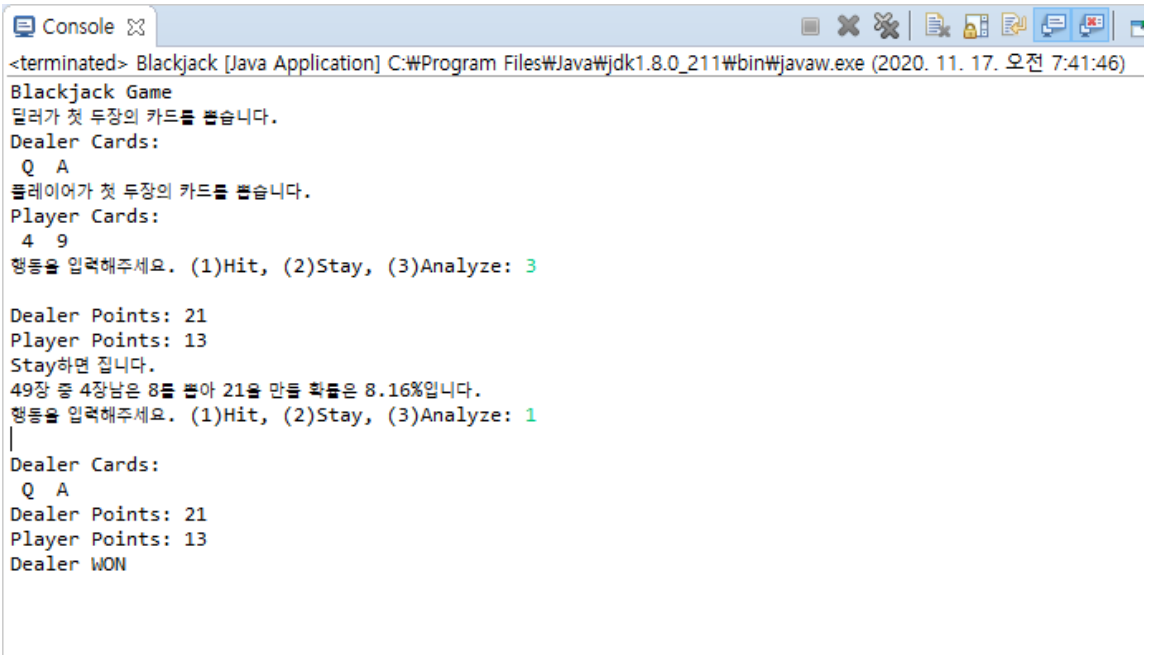
- **A와 Joker가 함께 있어도 가장 유리한 방법으로 점수를 계산합니다**



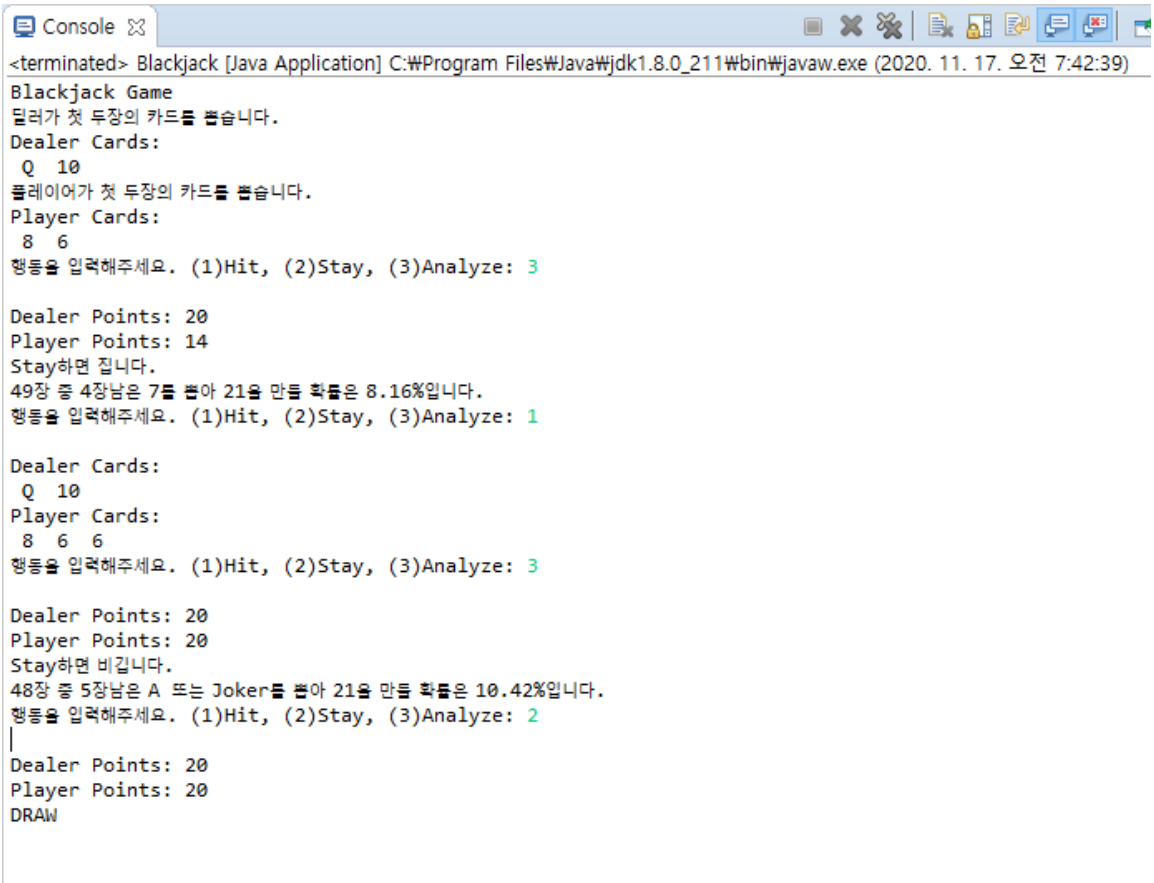
- 플레이어가 21을 넘긴 경우에는 hit, stay 어떤 것을 하더라도 BUST 패 처리하고 게임이 종료되도록 했습니다.



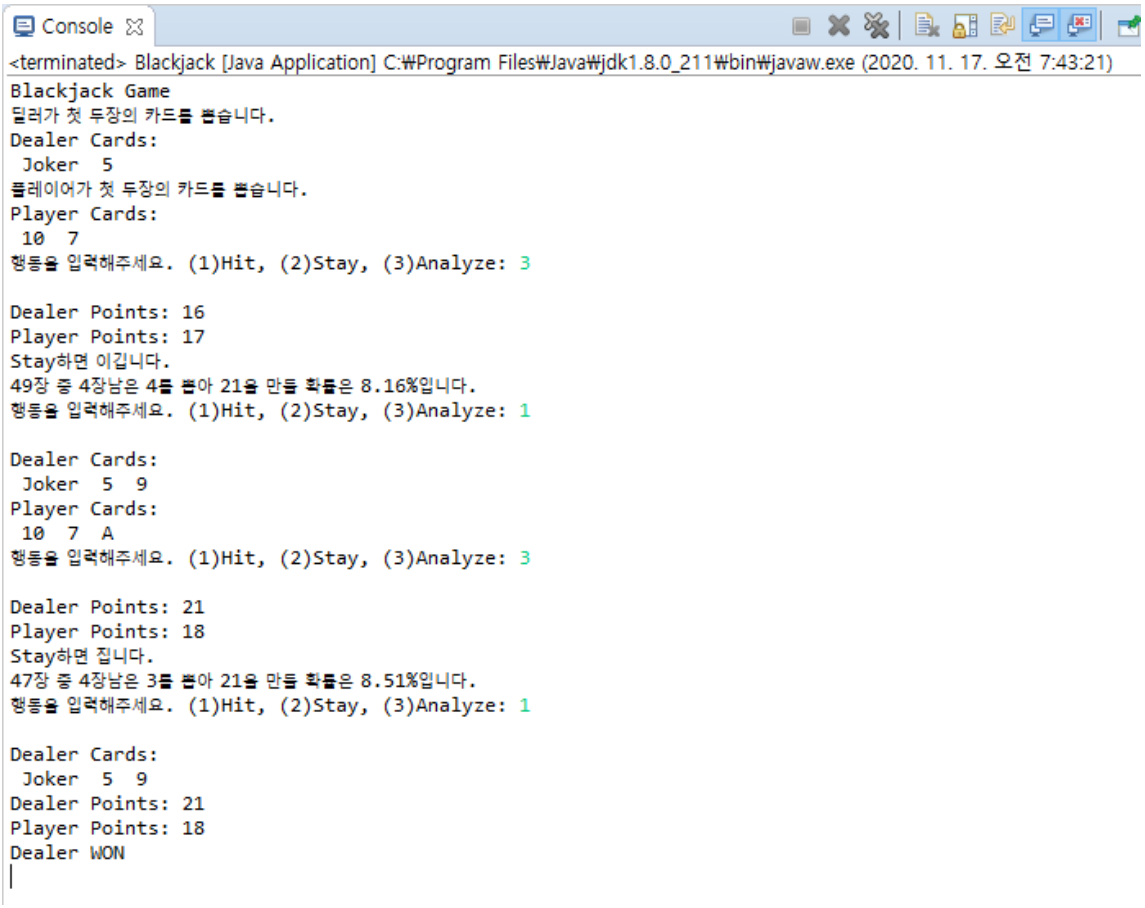
- 문제에서 요구한 blackjack (21 점수)을 만들기 위해서 한장 더 뽑아야 한다는 문구도 확실하게 나오고 있습니다.



- 딜러가 첫 두장에서 21을 만들었습니다. 이런 경우에도 마찬가지로 게임이 종료됩니다.



- 점수가 동일하게 비겼을 때도 잘 표기하고 있습니다



- 조커와 A의 값을 제대로 가장 유리하게 계산하고 있습니다.

- 소감

블랙잭의 룰도 간단하고 복잡한 룰 (스플릿, 등등)없이 본질적인 블랙잭 게임을 만드는 것이라 부담스럽지 않고 재밌었던 것 같습니다. 한가지 아쉬운 점은 제가 객체 지향적인 프로그래밍을 하지 못하고 아직도 절차지향적으로 프로그램을 짜는 것 같아 매우 아쉽습니다. 다음에는 조금 더 객체지향 프로그래밍에서만 사용할 수 있는 클래스, 상속 등의 좋은 기능들을 많이 사용해보고자 합니다.

- 부록 (코드 전문)

```
// 2019112130 조양진
// Blackjack.java

package week11;

/*
 * 딜러와 플레이어 2명이 존재한다
 * 카드는 조커를 포함한 53장이다
 * 2~10은 숫자 그대로 점수를, K/Q/J는 10점으로, A는 1 또는 11, 조커는 가장 유리한 경우로
```

```
계산
 * 게임 시작과 동시에 딜러와 게이머는 순차적으로 카드를 하나씩 뽑아 각자 2개의 카드를 소지한다
 * 딜러부터 게임을 진행하며 딜러는 2카드의 합계 점수가 16점 이하이면 반드시 1장을 추가로 뽑고, 17점 이상이면 STAY한다
 * 플레이어는 얼마든지 카드를 추가로 뽑을 수 있다
 * 양쪽 다 STAY한 경우 딜러와 플레이어 중 소유한 카드의 합이 21에 가까운 쪽이 승리한다
 * 단 21을 초과하면 초과한 쪽이 진다.
 * 딜러와 플레이어의 카드는 전부 오픈되어 있으며 플레이어는 블랙잭의 확률을 계산할 수 있어야 한다.
 */
import java.util.Random;
import java.util.Scanner;

public class Blackjack {
	static int playerTurn = 0;	// 플레이어 턴 횟수
	static int dealerTurn = 0;	// 딜러 턴 횟수
					// 0,1,2,3,4,5,6,7,8,9,10,11,12,13
					// 2,3,4,5,6,7,8,9,10,K,Q,J,A,Joker
	static int[] cardSet = {4,4,4,4,4,4,4,4,4,4,4,4,4,1};
	static boolean endGame = false;	// 게임 반복 플래그

	// 딜러 카드 패
	static int[] dealerCardSet = new int[60];
	// 플레이어 카드 패
	static int[] playerCardSet = new int[60];
	// 딜러 카드 점수
	static int dealerCardPoint = 0;
	// 플레이어 카드 점수
	static int playerCardPoint = 0;

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		Random rand = new Random();
		 int command = 0; // 사용자 입력 변수

		 System.out.println("Blackjack Game");

		 System.out.println("딜러가 첫 두장의 카드를 뽑습니다.");
		 dealerDrawCard();
```

```
dealerDrawCard();
// 뽑은 카드 출력 함수
printDealerCard();

System.out.println("플레이어가 첫 두장의 카드를 뽑습니다.");
playerDrawCard();
playerDrawCard();
// 뽑은 카드 출력 함수
printPlayerCard();



/*
 * // 첫 두장에 게임이 끝난 경우
if(((playerCardPoint == 21) && (playerTurn == 2))&&
                (dealerCardPoint == 21) && (dealerTurn == 2)) {
        System.out.println("BLACKJACK!!!: DRAW");
        endGame = true;
}
else if((playerCardPoint == 21) && (playerTurn == 2)) {
        System.out.println("BLACKJACK!!!: Player WON");
        endGame = true;
}else if((dealerCardPoint == 21) && (dealerTurn == 2)) {
        System.out.println("BLACKJACK!!!: Dealer WON");
        endGame = true;
}
*/

while(!endGame) {
        // 반복되는 입력
        dealerCardPoint = calculate(dealerCardSet, dealerTurn);
        playerCardPoint = calculate(playerCardSet, playerTurn);
        System.out.print("행동을 입력해주세요. (1)Hit, (2)Stay, (3)Analyze: ");
        try {
                command = sc.nextInt();
                System.out.println();

                if(command == 2) {
                        // 게임 종료 함수
                        normalEndGame();
                }
```

```
                else if(command == 1){
                        // 딜러 합 16 체크 - 작으면 딜러 카드 뽑는 함수
                        if(dealerCardPoint <= 16 && dealerTurn==2) {
                                dealerDrawCard();
                        }else if(dealerCardPoint < 17) {
                                dealerDrawCard();
                        }
                        // 플레이어 합 21 오버 체크 - 크면 게임 종료 함수
                        if(playerCardPoint >= 21) {
                                // 게임 종료 함수
                                normalEndGame();
                                break;
                        }
                        // 딜러 드로우
                        printDealerCard();
                        // 플레이어 합 21 오버 체크 - 크면 게임 종료 함수
                        if(dealerCardPoint >= 21) {
                                // 게임 종료 함수
                                normalEndGame();
                                break;
                        }
                        // 플레이어 드로우
                        playerDrawCard();
                        // 뽑은 카드 출력 함수
                        printPlayerCard();

                }
                else if(command == 3) {
                        playerAnalyze();
                }
        }catch (Exception e) {
                System.out.println("잘못된 입력입니다. 다시 입력해주세요.");
                e.printStackTrace();
                throw e;
        }
    }
}

// 카드 뽑는 함수들
public static void dealerDrawCard() {
```

```
		Random rand = new Random();
		// 0~13
		int temp = rand.nextInt(14);
		// cardSet에서 1 빼고 비교
		if((cardSet[temp] == 0)) {
			// 해당 숫자의 카드가 없다면 다시 뽑기
			dealerDrawCard();
		}else {
			cardSet[temp] -= 1;
			dealerCardSet[dealerTurn] = temp;
			dealerTurn++;
		}
	}

	public static void playerDrawCard() {
		Random rand = new Random();
		int temp = rand.nextInt(14);
		// cardSet에서 1 빼고 비교
		if((cardSet[temp] == 0)) {
			// 해당 숫자의 카드가 없다면 다시 뽑기
			playerDrawCard();
		}else {
			cardSet[temp] -= 1;
			playerCardSet[playerTurn] = temp;
			playerTurn++;
		}
	}



	// 카드 출력 함수들
	public static void printDealerCard() {
		System.out.printf("Dealer Cards:₩n");
		for(int i=0; i<dealerTurn; i++) {
			if(dealerCardSet[i] >=0 && dealerCardSet[i] <=8) {
				System.out.printf(" %d ", dealerCardSet[i]+2);
			}
			else if(dealerCardSet[i]==9) {
				System.out.print(" K ");
			}
```

```
                else if(dealerCardSet[i]==10) {
                        System.out.print(" Q ");
                }
                else if(dealerCardSet[i]==11) {
                        System.out.print(" J ");
                }
                else if(dealerCardSet[i]==12) {
                        System.out.print(" A ");
                }
                else if(dealerCardSet[i]==13) {
                        System.out.print(" Joker ");
                }
        }
        System.out.println();
}

public static void printPlayerCard() {
        System.out.printf("Player Cards:₩n");
        for(int i=0; i<playerTurn; i++) {
                if(playerCardSet[i] >=0 && playerCardSet[i] <=8) {
                        System.out.printf(" %d ", playerCardSet[i]+2);
                }
                else if(playerCardSet[i]==9) {
                        System.out.print(" K ");
                }
                else if(playerCardSet[i]==10) {
                        System.out.print(" Q ");
                }
                else if(playerCardSet[i]==11) {
                        System.out.print(" J ");
                }
                else if(playerCardSet[i]==12) {
                        System.out.print(" A ");
                }
                else if(playerCardSet[i]==13) {
                        System.out.print(" Joker ");
                }
        }
        System.out.println();
}
```

```
	// 21을 만들 수 있는 확률 계산 함수
	public static void playerAnalyze() {
		System.out.println("Dealer Points: "+ dealerCardPoint);
		System.out.println("Player Points: "+ playerCardPoint);

		if((playerCardPoint > dealerCardPoint)&&((dealerCardPoint > 21) ||
(playerCardPoint <=21))) {
			System.out.println("Stay하면 이깁니다.");
		}else if(playerCardPoint == dealerCardPoint) {
			System.out.println("Stay하면 비깁니다.");
		}else {
			System.out.println("Stay하면 집니다.");
		}

		// 21점을 만들 수 있는 확률을 계산하는 조건문
		if(playerCardPoint < 21) {
			int temp = 21 - playerCardPoint;
			if(temp > 11) {
				System.out.print("21을 만들려면 한 장 더 뽑아야 합니다.");
			}else {
				int total = 0;
				int winTemp = 0;
				float winCase = 0;
				for(int i=0; i<14; i++) {
					total += cardSet[i];
				}
				if((temp == 11) || (temp == 1)) {
					winTemp = cardSet[12] + cardSet[13];
					winCase = (float)winTemp/total;
					System.out.printf("%d장 중 %d장남은 A 또는 Joker를
뽑아 21을 만들 확률은 %.2f%%입니다.", total, winTemp, winCase*100);
				}else if(temp == 10) {
					winTemp = cardSet[8] + cardSet[9] + cardSet[10] +
cardSet[11];
					winCase = (float)winTemp/total;
					System.out.printf("%d장 중 10, K, Q 또는 J를 %d장을
뽑아 21을 만들 확률은 %.2f%%입니다.", total, winTemp, winCase*100);
				}else if(temp == 9) {
					winTemp = cardSet[7];
```

```
                winCase = (float)winTemp/total;
                System.out.printf("%d장 중 %d장남은 9를 뽑아 21을
만들 확률은 %.2f%%입니다.", total,winTemp, winCase*100);
            }else if(temp == 8) {
                winTemp = cardSet[6];
                winCase = (float)winTemp/total;
                System.out.printf("%d장 중 %d장남은 8를 뽑아 21을
만들 확률은 %.2f%%입니다.", total,winTemp, winCase*100);
            }else if(temp == 7) {
                winTemp = cardSet[5];
                winCase = (float)winTemp/total;
                System.out.printf("%d장 중 %d장남은 7를 뽑아 21을
만들 확률은 %.2f%%입니다.", total,winTemp, winCase*100);
            }else if(temp == 6) {
                winTemp = cardSet[4];
                winCase = (float)winTemp/total;
                System.out.printf("%d장 중 %d장남은 6를 뽑아 21을
만들 확률은 %.2f%%입니다.", total,winTemp, winCase*100);
            }else if(temp == 5) {
                winTemp = cardSet[3];
                winCase = (float)winTemp/total;
                System.out.printf("%d장 중 %d장남은 5를 뽑아 21을
만들 확률은 %.2f%%입니다.", total,winTemp, winCase*100);
            }else if(temp == 4) {
                winTemp = cardSet[2];
                winCase = (float)winTemp/total;
                System.out.printf("%d장 중 %d장남은 4를 뽑아 21을
만들 확률은 %.2f%%입니다.", total,winTemp, winCase*100);
            }else if(temp == 3) {
                winTemp = cardSet[1];
                winCase = (float)winTemp/total;
                System.out.printf("%d장 중 %d장남은 3를 뽑아 21을
만들 확률은 %.2f%%입니다.", total,winTemp, winCase*100);
            }else if(temp == 2) {
                winTemp = cardSet[0];
                winCase = (float)winTemp/total;
                System.out.printf("%d장 중 %d장남은 2를 뽑아 21을
만들 확률은 %.2f%%입니다.", total,winTemp, winCase*100);
            }
            System.out.println();
```

```
                }
        }
}

// 카드 점수 계산 알고리즘 함수
public static int calculate(int[] input, int turnCounter) {
        int total = 0;
        int aCounter = 0;
        boolean isJokerIn = false;
        // 1. 2~10,J,Q,K는 우선 처리
        for(int i=0; i<turnCounter; i++) {
                if(input[i] == 0) {
                        total +=2;
                }else if(input[i]==1){
                        total +=3;
                }else if(input[i]==2){
                        total +=4;
                }else if(input[i]==3){
                        total +=5;
                }else if(input[i]==4){
                        total +=6;
                }else if(input[i]==5){
                        total +=7;
                }else if(input[i]==6){
                        total +=8;
                }else if(input[i]==7){
                        total +=9;
                }else if(input[i]==8){
                        total +=10;
                }else if(input[i]==9){
                        total +=10;
                }else if(input[i]==10){
                        total +=10;
                }else if(input[i]==11){
                        total +=10;
                }else if(input[i]==12) {
                        aCounter++;
                }else if(input[i]==13) {
                        isJokerIn = true;
                }
```

```
		}
		// 2. A의 개수를 저장해놨다가 1번과 비교하여 더하기
		if(aCounter > 0) {
			for(int j=0; j<aCounter; j++) {
				if(total < 11) {
					total += 11;
				}else {
					total += 1;
				}
			}
		}
		// 3. 조커의 유무
		if(isJokerIn) {
			// total값이 1과 11 사이라면 total 21에 맞출 수 있도록 더합니다
			if((21-total)>=1 && (21-total)<=11) {
				total += (21-total);
				// total값이 11보다 작다면 11을 더합니다
			}else if(total<11) {
				total += 11;
				// 나머지 경우는 1을 더합니다
			}else {
				total +=1;
			}
		}
		// total을 반환합니다
		return total;
	}


	// 게임 종료 함수
	public static void normalEndGame() {
		System.out.println("Dealer Points: "+ dealerCardPoint);
		System.out.println("Player Points: "+ playerCardPoint);
		endGame = true;
		if(playerCardPoint > 21) {
			System.out.println("BUST: Dealer WON");
		}else if(dealerCardPoint > 21) {
			System.out.println("BUST: Player WON");
		}else if(playerCardPoint == dealerCardPoint) {
			System.out.println("DRAW");
```

```
                }else if(playerCardPoint-dealerCardPoint>0) {
                        System.out.println("Player WON");
                }else if(playerCardPoint-dealerCardPoint<0) {
                        System.out.println("Dealer WON");
                }
        }

}
```