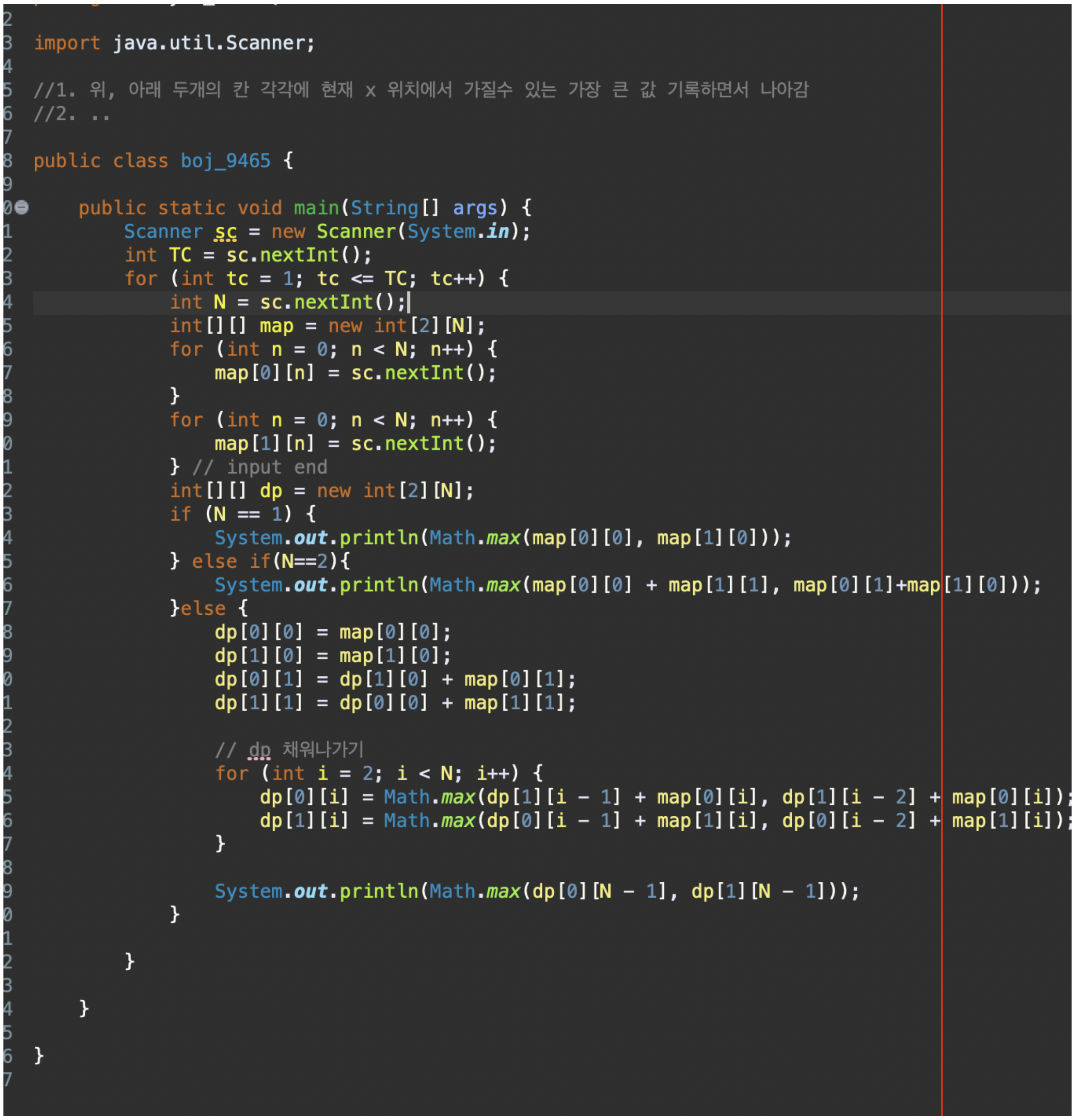
# 9월 3주차_dp 풀이

| | |
|---|---|
| Created | |
| Reviewed | ☐ |
| Type | |
| 기한 | |

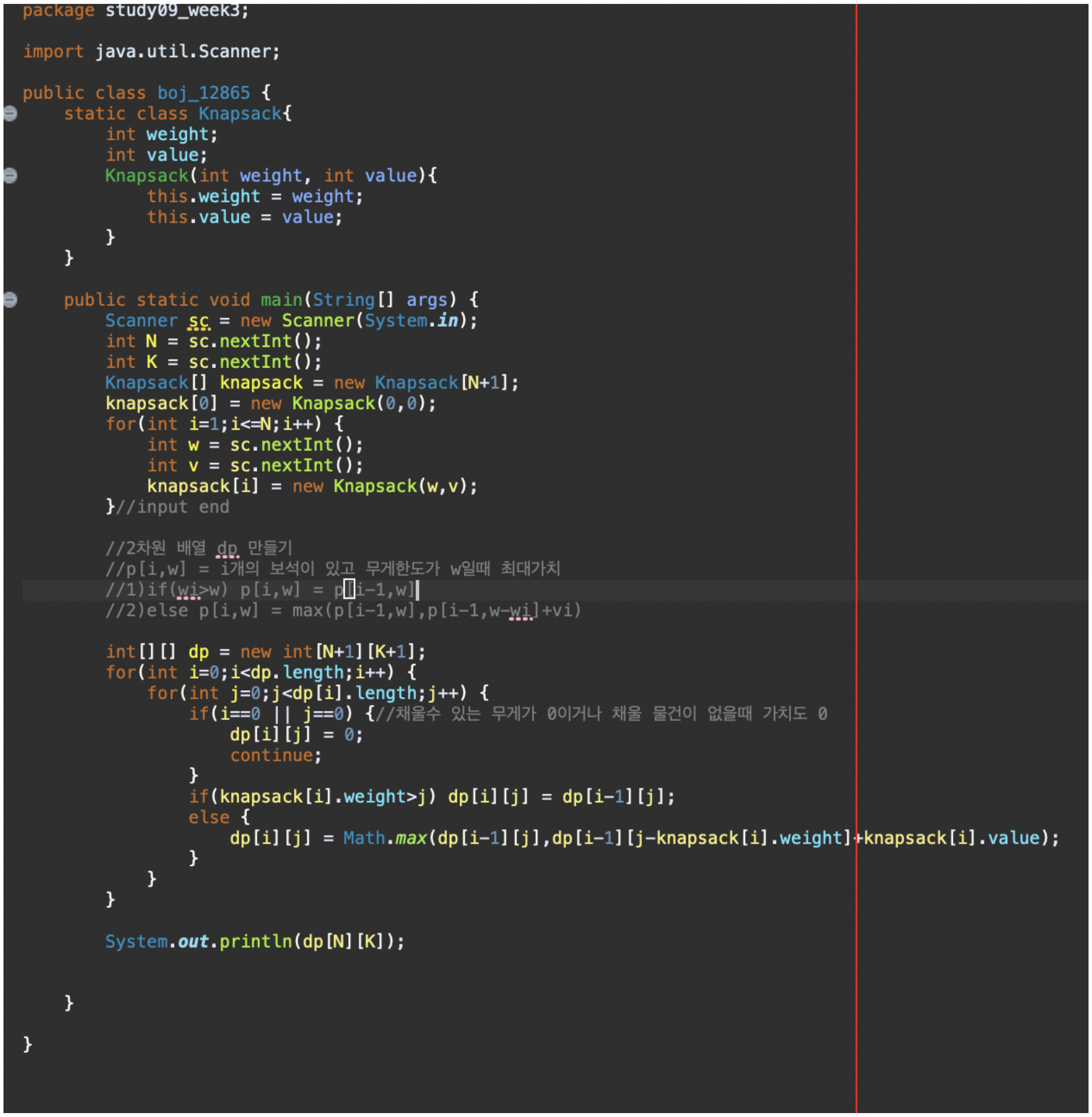
## 송예인

### BOJ 9465



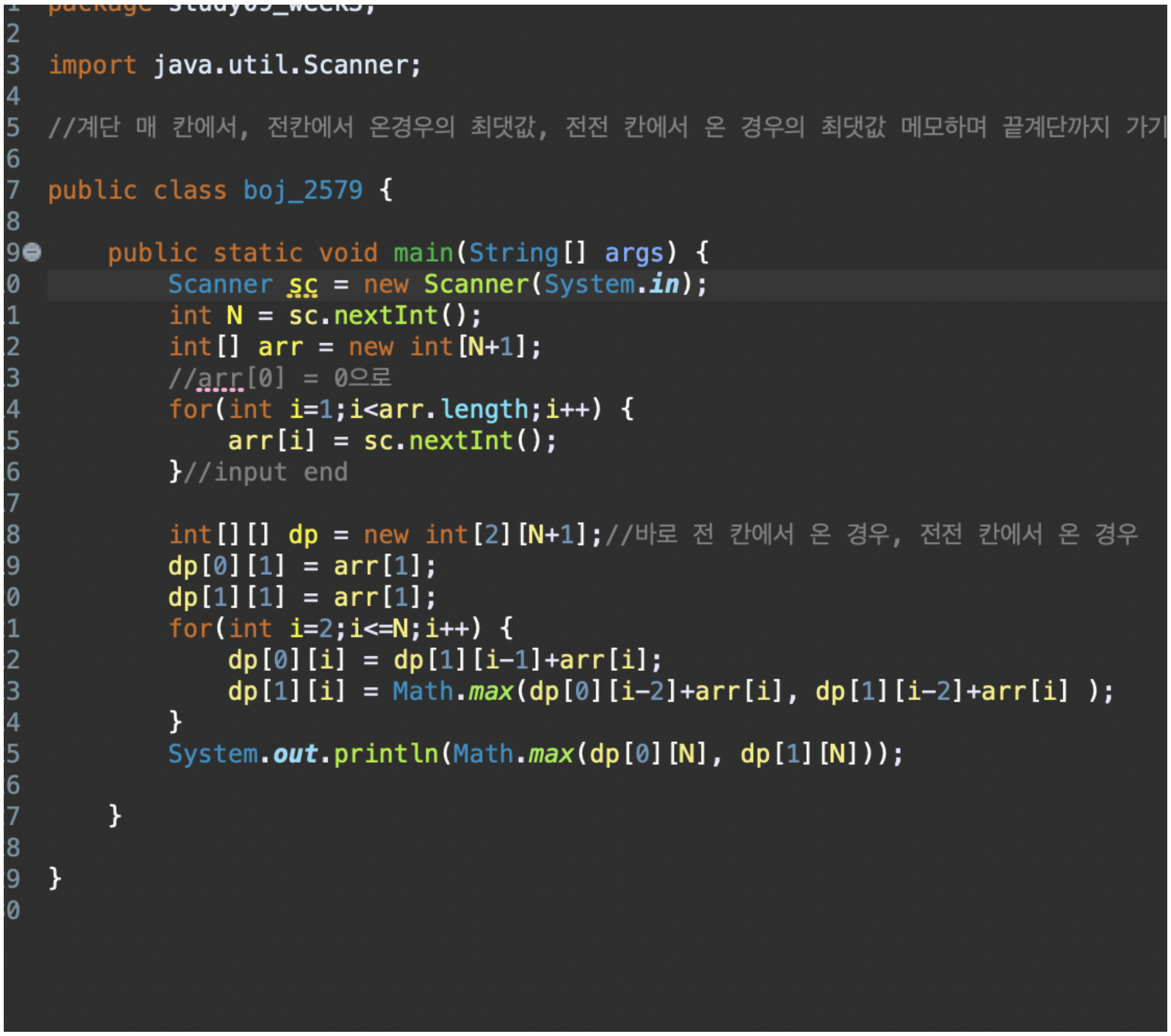
```java
import java.util.Scanner;

//1. 위, 아래 두개의 칸 각각에 현재 x 위치에서 가질수 있는 가장 큰 값 기록하면서 나아감
//2. ..

public class boj_9465 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int TC = sc.nextInt();
        for (int tc = 1; tc <= TC; tc++) {
            int N = sc.nextInt();
            int[][] map = new int[2][N];
            for (int n = 0; n < N; n++) {
                map[0][n] = sc.nextInt();
            }
            for (int n = 0; n < N; n++) {
                map[1][n] = sc.nextInt();
            } // input end
            int[][] dp = new int[2][N];
            if (N == 1) {
                System.out.println(Math.max(map[0][0], map[1][0]));
            } else if(N==2){
                System.out.println(Math.max(map[0][0] + map[1][1], map[0][1]+map[1][0]));
            }else {
                dp[0][0] = map[0][0];
                dp[1][0] = map[1][0];
                dp[0][1] = dp[1][0] + map[0][1];
                dp[1][1] = dp[0][0] + map[1][1];

                // dp 채워나가기
                for (int i = 2; i < N; i++) {
                    dp[0][i] = Math.max(dp[1][i - 1] + map[0][i], dp[1][i - 2] + map[0][i]);
                    dp[1][i] = Math.max(dp[0][i - 1] + map[1][i], dp[0][i - 2] + map[1][i]);
                }

                System.out.println(Math.max(dp[0][N - 1], dp[1][N - 1]));
            }

        }

    }

}
```

## BOJ 12865



```java
package study09_week3;

import java.util.Scanner;

public class boj_12865 {
    static class Knapsack{
        int weight;
        int value;
        Knapsack(int weight, int value){
            this.weight = weight;
            this.value = value;
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int K = sc.nextInt();
        Knapsack[] knapsack = new Knapsack[N+1];
        knapsack[0] = new Knapsack(0,0);
        for(int i=1;i<=N;i++) {
            int w = sc.nextInt();
            int v = sc.nextInt();
            knapsack[i] = new Knapsack(w,v);
        }//input end

        //2차원 배열 dp 만들기
        //p[i,w] = i개의 보석이 있고 무게한도가 w일때 최대가치
        //1)if(wi>w) p[i,w] = p[i-1,w]
        //2)else p[i,w] = max(p[i-1,w],p[i-1,w-wi]+vi)

        int[][] dp = new int[N+1][K+1];
        for(int i=0;i<dp.length;i++) {
            for(int j=0;j<dp[i].length;j++) {
                if(i==0 || j==0) {//채울수 있는 무게가 0이거나 채울 물건이 없을때 가치도 0
                    dp[i][j] = 0;
                    continue;
                }
                if(knapsack[i].weight>j) dp[i][j] = dp[i-1][j];
                else {
                    dp[i][j] = Math.max(dp[i-1][j],dp[i-1][j-knapsack[i].weight]+knapsack[i].value);
                }
            }
        }

        System.out.println(dp[N][K]);


    }

}
```

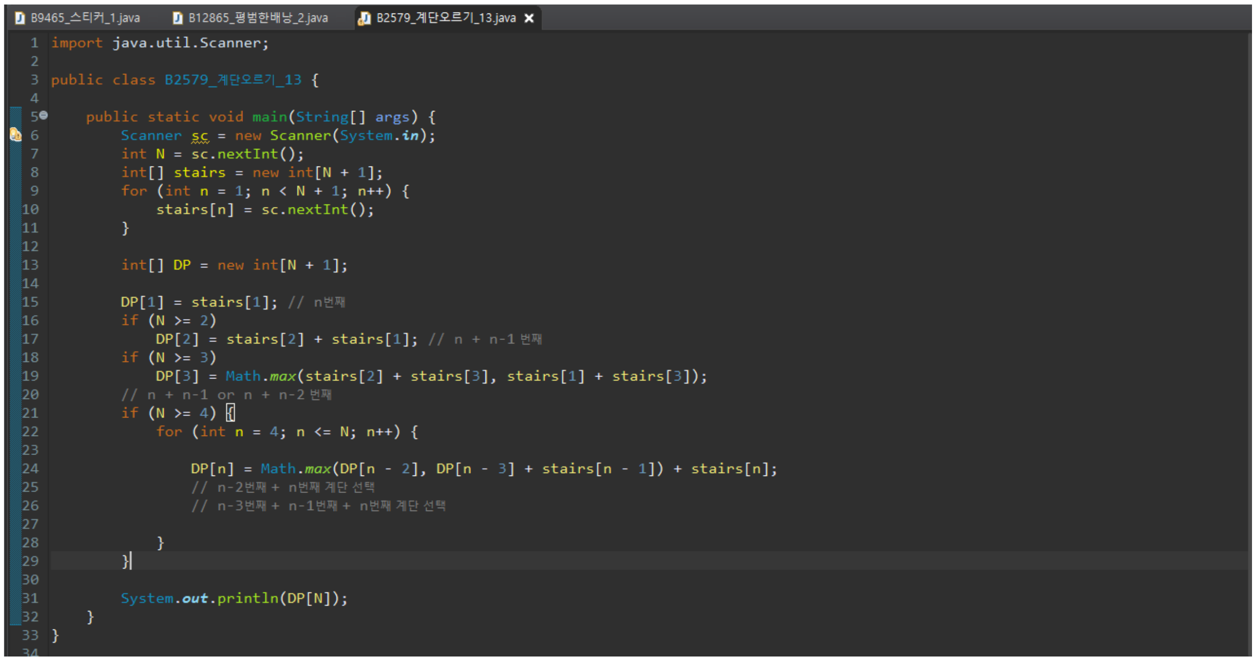
## BOJ 2579



```java
package study05_week3;

import java.util.Scanner;

//계단 매 칸에서, 전칸에서 온경우의 최댓값, 전전 칸에서 온 경우의 최댓값 메모하며 끝계단까지 가기

public class boj_2579 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] arr = new int[N+1];
        //arr[0] = 0으로
        for(int i=1;i<arr.length;i++) {
            arr[i] = sc.nextInt();
        }//input end

        int[][] dp = new int[2][N+1];//바로 전 칸에서 온 경우, 전전 칸에서 온 경우
        dp[0][1] = arr[1];
        dp[1][1] = arr[1];
        for(int i=2;i<=N;i++) {
            dp[0][i] = dp[1][i-1]+arr[i];
            dp[1][i] = Math.max(dp[0][i-2]+arr[i], dp[1][i-2]+arr[i] );
        }
        System.out.println(Math.max(dp[0][N], dp[1][N]));

    }

}
```

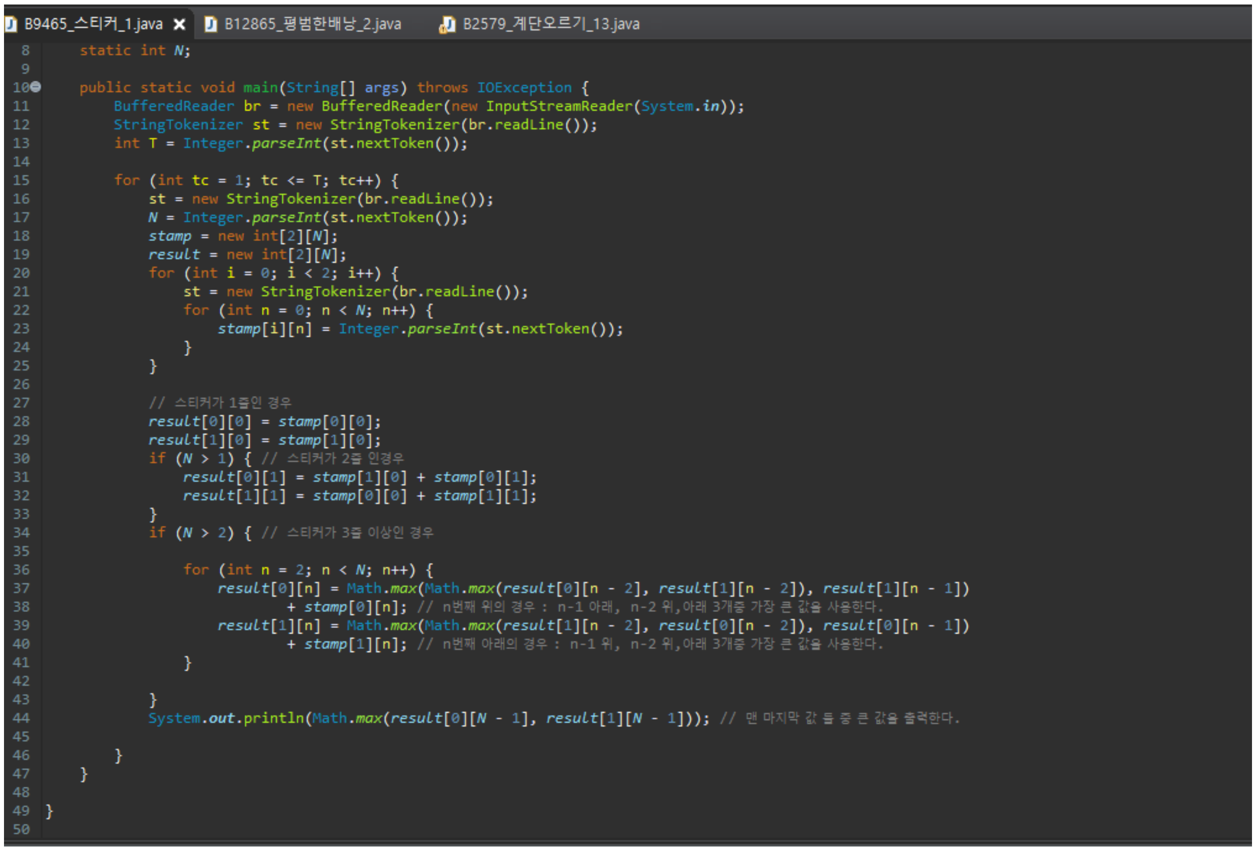
## 김유진

**BOJ 2579**


B9465_스티커_1.java　B12865_평범한배낭_2.java　B2579_계단오르기_13.java

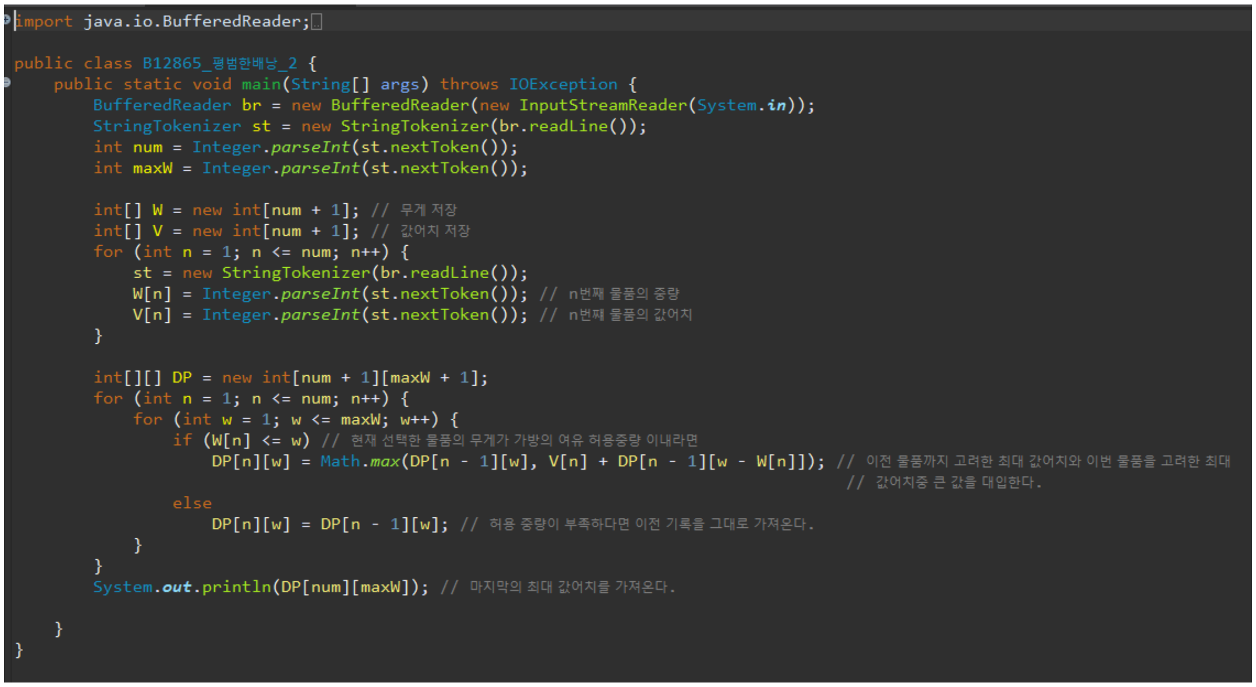
```java
import java.util.Scanner;

public class B2579_계단오르기_13 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] stairs = new int[N + 1];
        for (int n = 1; n < N + 1; n++) {
            stairs[n] = sc.nextInt();
        }

        int[] DP = new int[N + 1];

        DP[1] = stairs[1]; // n번째
        if (N >= 2)
            DP[2] = stairs[2] + stairs[1]; // n + n-1 번째
        if (N >= 3)
            DP[3] = Math.max(stairs[2] + stairs[3], stairs[1] + stairs[3]);
        // n + n-1 or n + n-2 번째
        if (N >= 4) {
            for (int n = 4; n <= N; n++) {

                DP[n] = Math.max(DP[n - 2], DP[n - 3] + stairs[n - 1]) + stairs[n];
                // n-2번째 + n번째 계단 선택
                // n-3번째 + n-1번째 + n번째 계단 선택

            }
        }

        System.out.println(DP[N]);
    }
}
```

## BOJ 9465


B9465_스티커_1.java　B12865_평범한배낭_2.java　B2579_계단오르기_13.java

```java
    static int N;

    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
        int T = Integer.parseInt(st.nextToken());

        for (int tc = 1; tc <= T; tc++) {
            st = new StringTokenizer(br.readLine());
            N = Integer.parseInt(st.nextToken());
            stamp = new int[2][N];
            result = new int[2][N];
            for (int i = 0; i < 2; i++) {
                st = new StringTokenizer(br.readLine());
                for (int n = 0; n < N; n++) {
                    stamp[i][n] = Integer.parseInt(st.nextToken());
                }
            }

            // 스티커가 1줄인 경우
            result[0][0] = stamp[0][0];
            result[1][0] = stamp[1][0];
            if (N > 1) { // 스티커가 2줄 인경우
                result[0][1] = stamp[1][0] + stamp[0][1];
                result[1][1] = stamp[0][0] + stamp[1][1];
            }
            if (N > 2) { // 스티커가 3줄 이상인 경우

                for (int n = 2; n < N; n++) {
                    result[0][n] = Math.max(Math.max(result[0][n - 2], result[1][n - 2]), result[1][n - 1])
                            + stamp[0][n]; // n번째 위의 경우 : n-1 아래, n-2 위,아래 3개중 가장 큰 값을 사용한다.
                    result[1][n] = Math.max(Math.max(result[1][n - 2], result[0][n - 2]), result[0][n - 1])
                            + stamp[1][n]; // n번째 아래의 경우 : n-1 위, n-2 위,아래 3개중 가장 큰 값을 사용한다.
                }

            }
            System.out.println(Math.max(result[0][N - 1], result[1][N - 1])); // 맨 마지막 값 들 중 큰 값을 출력한다.

        }
    }

}
```

## BOJ 12865



```java
import java.io.BufferedReader;

public class B12865_평범한배낭_2 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
        int num = Integer.parseInt(st.nextToken());
        int maxW = Integer.parseInt(st.nextToken());

        int[] W = new int[num + 1]; // 무게 저장
        int[] V = new int[num + 1]; // 값어치 저장
        for (int n = 1; n <= num; n++) {
            st = new StringTokenizer(br.readLine());
            W[n] = Integer.parseInt(st.nextToken()); // n번째 물품의 중량
            V[n] = Integer.parseInt(st.nextToken()); // n번째 물품의 값어치
        }

        int[][] DP = new int[num + 1][maxW + 1];
        for (int n = 1; n <= num; n++) {
            for (int w = 1; w <= maxW; w++) {
                if (W[n] <= w) // 현재 선택한 물품의 무게가 가방의 여유 허용중량 이내라면
                    DP[n][w] = Math.max(DP[n - 1][w], V[n] + DP[n - 1][w - W[n]]); // 이전 물품까지 고려한 최대 값어치와 이번 물품을 고려한 최대
                                                                                    // 값어치중 큰 값을 대입한다.
                else
                    DP[n][w] = DP[n - 1][w]; // 허용 중량이 부족하다면 이전 기록을 그대로 가져온다.
            }
        }
        System.out.println(DP[num][maxW]); // 마지막의 최대 값어치를 가져온다.

    }
}
```

## 오서하

### BOJ 9465

```java
import java.util.Scanner;

public class Baekjoon_9465_스티커 {

  public static void main(String[] args) {
    // TODO Auto-generated method stub

    Scanner sc = new Scanner(System.in);
    int TC = sc.nextInt();

    for (int tc = 1; tc <= TC; tc++) {
      int N = sc.nextInt();

      int[][] map = new int[2][N];
      int[][] di = new int[3][N];

      // 입력 받기
      for (int i = 0; i < 2; i++) {
        for (int j = 0; j < N; j++) {
          map[i][j] = sc.nextInt();
        }
      }

      di[0][0] = map[0][0];
      di[1][0] = map[1][0];
      for (int j = 1, i = 1; j < N; j++, i++) {
        if (i == 2)
          i = 0;
        di[0][j] = di[0][j - 1] + map[i][j];
      }
      for (int j = 1, i = 0; j < N; j++, i++) {
        if (i == 2)
          i = 0;
        di[1][j] = di[1][j - 1] + map[i][j];
      }
      for (int j = 2; j < N; j++) {
        di[2][j] = Math.max(Math.max(di[0][j-2], di[1][j-2]), di[2][j-2])  + Math.max(map[0][j], map[1][j]);
      }

      int ans = Math.max(Math.max(di[0][N-1], di[1][N-1]), di[2][N-1]);

      System.out.println(ans);
    }
  }
}
```

## BOJ 2579

```
import java.util.Scanner;

public class baekjoon_2579_계단오르기 {

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int N = sc.nextInt();

    int[] stair = new int[N];
    int[][] d = new int[2][N];

    for (int i = 0; i < N; i++) {
      stair[i] = sc.nextInt();
    }

    if(N < 2) { //  1이라면
      System.out.println(stair[N-1]);
    }
    else if(N >= 2) {
      d[0][0] = stair[0];
      d[0][1] = stair[1];
      d[1][1] = d[0][0] +  stair[1];

      for(int i = 2 ; i < N ; i++) {
        d[1][i] = d[0][i-1] + stair[i];
        d[0][i] = Math.max(d[0][i-2], d[1][i-2]) + stair[i];
      }
      System.out.println(Math.max(d[0][N-1], d[1][N-1]));
    }
  }
}
```