

TOWER OF HANOI

박경호

INDEX



01 SCHEDULE

Page 3 - 4

02 DESIGN

Page 5 - 7

**FOLDER
STRUCTURE**

03

Page 8 - 9

04 CODE

Page 10 - 18

05 REVIEW

Page 19 - 20

SCHEDULE



SCHEDULE

Day 1

패턴 분석 & JS

하노이 타워 패턴 분석
패턴 JS 코드화 작업

Day 2

디자인 작업

스케치 작업
HTML & CSS 구성
Animation 작업

Day 3

JAVA & PYTHON

JS 코드 기반 JAVA 작업
PYTHON 재귀 함수 구현

Day 4

Project 마무리

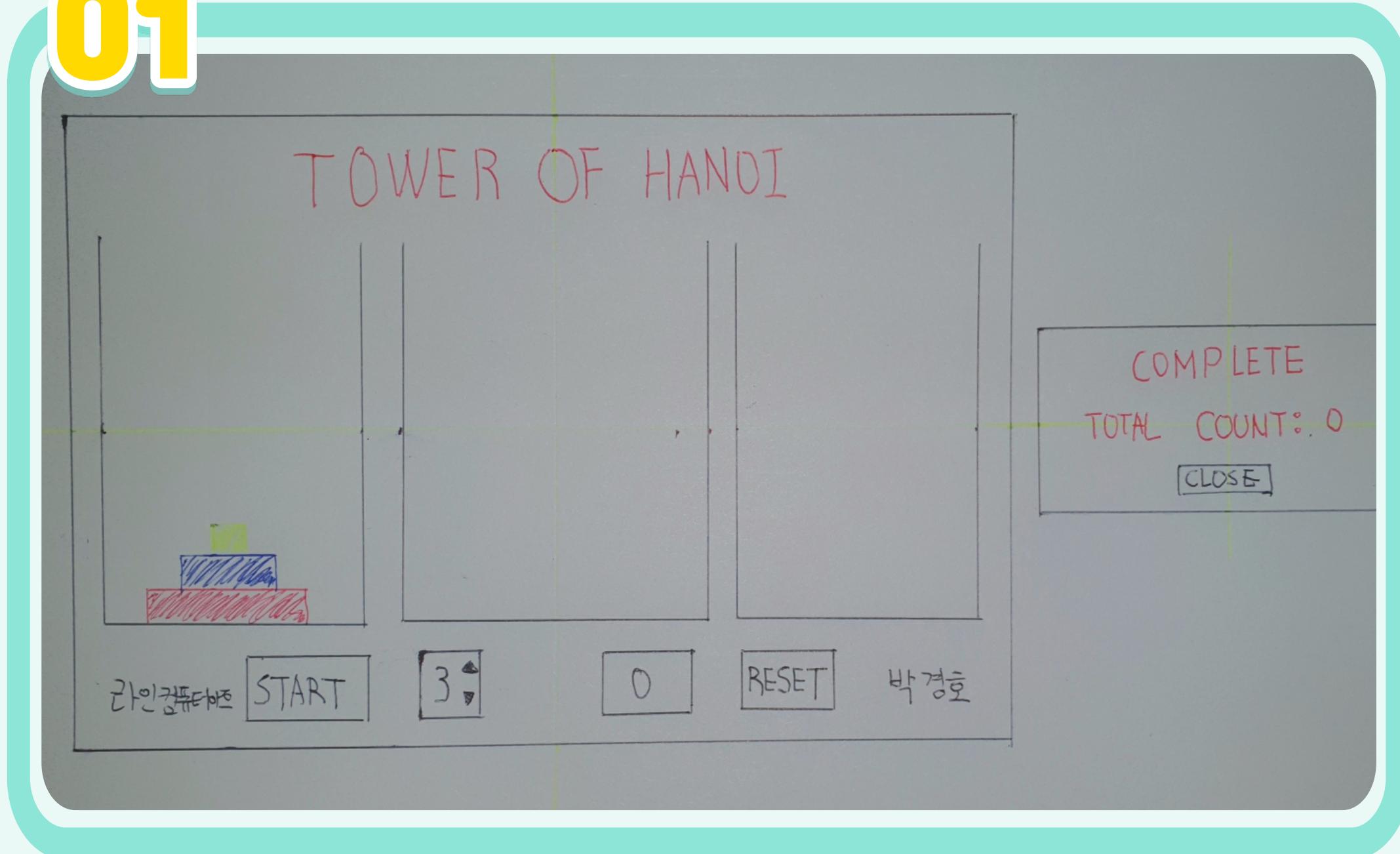
코드 정리
PPT 작업



The word "DESIGN" is written in a bold, yellow, sans-serif font. It features a white outline and a thick, wavy teal shadow at the bottom. Small white starburst shapes are scattered around the letters, particularly on the left and right sides.

DESIGN

01



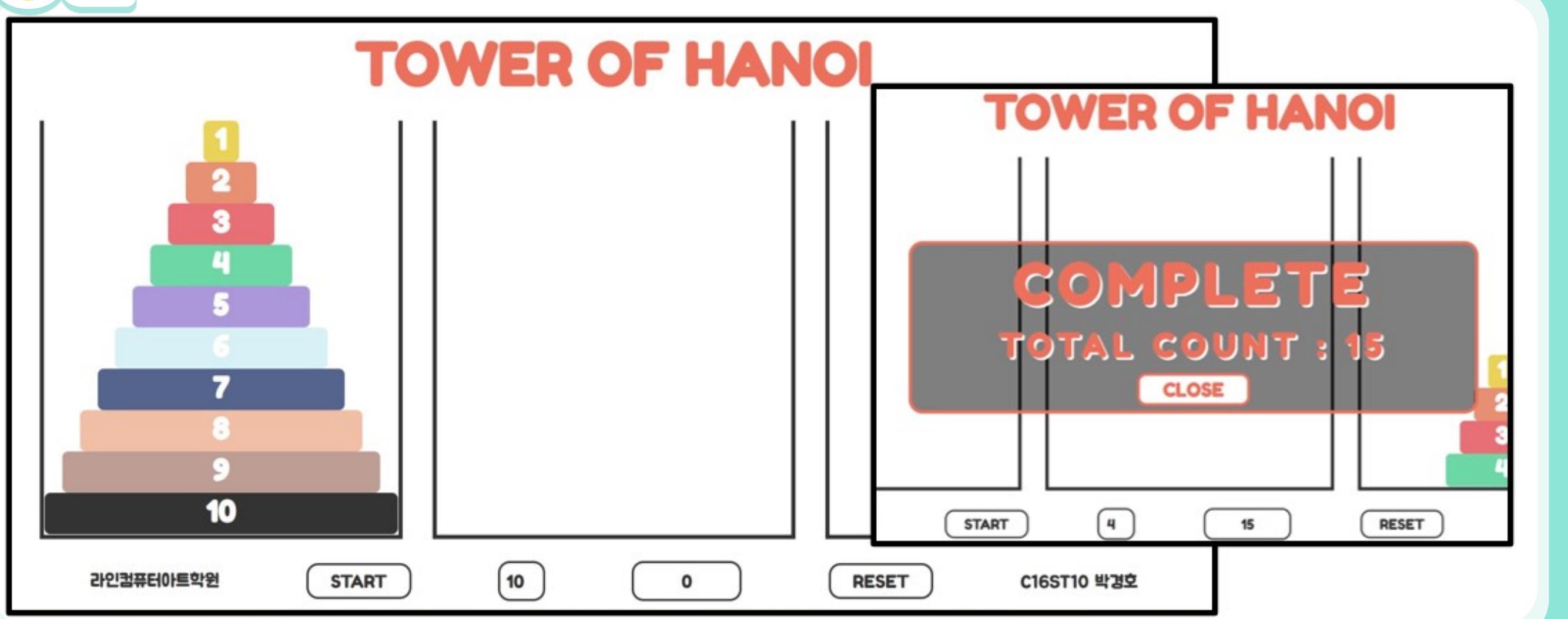
SKETCH

RING 수량 설정과 현재 이동 횟수 모니터링 부분 필요

START, RESET 버튼으로 조작의 편의성 제공

Animation 종료 시 완료 팝업창 구현 필요

02



LAYOUT

RING 순서별로 색상 차이를 주어 시각성 향상

Animation 종료 시 완료 팝업에 총 진행 횟수 표기

FOLDER STRUCTURE

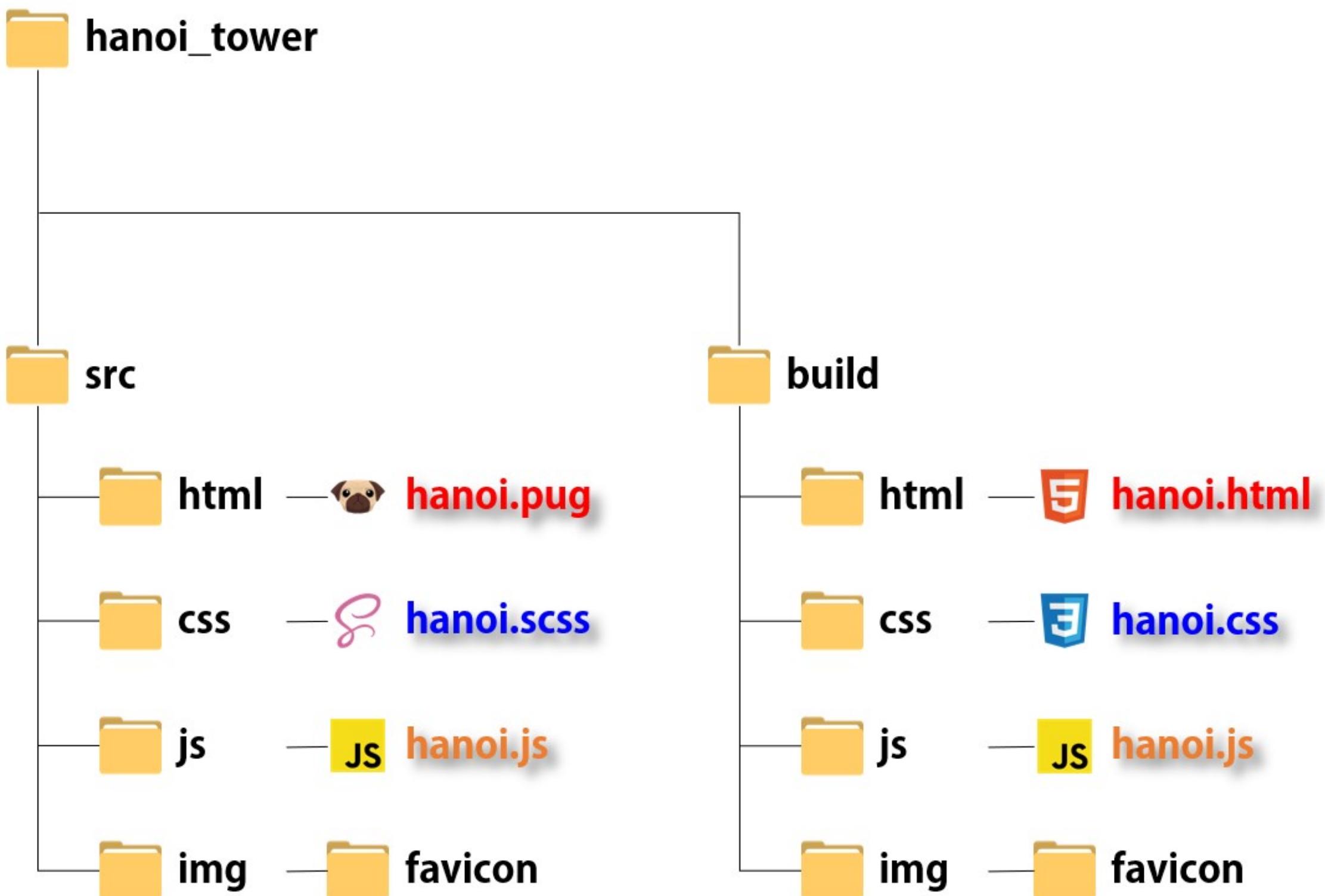
FOLDER STRUCTURE

폴더 구조

작업 영역 src 폴더와 배포 영역 build 폴더를 구분

PUG 작업 후 HTML으로 Compile 진행

SASS 작업 후 CSS으로 Compile 진행





CODE

STACK & QUEUE

JavaScript에서 Stack, Queue Class 생성

```
6 //--- STACK & QUEUE ---//  
7 class Stack {  
8     constructor(_num, set_stack=[]) {  
9         this._arr = set_stack;  
10        this.num = _num;  
11    }  
12    push(item) {  
13        this._arr.push(item);  
14    }  
15    pop() {  
16        return this._arr.pop();  
17    }  
18    peek() {  
19        return this._arr[this._arr.length - 1];  
20    }  
21    size() {  
22        return this._arr.length;  
23    }  
24    all_arr(){  
25        return this._arr;  
26    }  
27 }  
28  
29 class Queue {  
30     constructor(set_list=[]) {  
31         this._arr = set_list;  
32     }  
33     offer(item) {  
34         this._arr.push(item);  
35     }  
36     poll() {  
37         return this._arr.shift();  
38     }  
39     peek(){  
40         return this._arr[0];  
41     }  
42     get(index){  
43         return this._arr[index];  
44     }  
45     size() {  
46         return this._arr.length;  
47     }  
48 }
```

```
50 //--- HANOI_TOWER CLASS ---//  
51 class Hanoi_Tower {  
52 >   constructor(_ring){...  
78  
79     get getter_tower(){  
80       return this.tower;  
81     }  
82     get getter_name(){  
83       return this.name.peek();  
84     }  
85 >   set setter_ring(set_ring){...  
93  
94 >   create_pattern(){ //--- create parttern list ---//...  
99  
100 >  create_ring(){ //--- create ring list ---//...  
109  
110 >  move_ring(){ //--- move ring 1 turn ---//...  
166  
167 >  async show_hanoi(){ //--- loop move_ring & end catch ---//...  
230  
231 >  click_event(){ //--- button & input event ---//...  
264  
265 }  
266 hanoi_game = new Hanoi_Tower(1);  
267 hanoi_game.click_event();  
268 hanoi_game.create_ring();
```

Class 구현

위에서 생성한 Stack, Queue Class 활용

Ring 및 Pattern 생성 Method 구현

한 턴 동작 Method 구현

End 조건 전까지 반복하는 Method 구현

Button, Input 관련 Event Method 구현

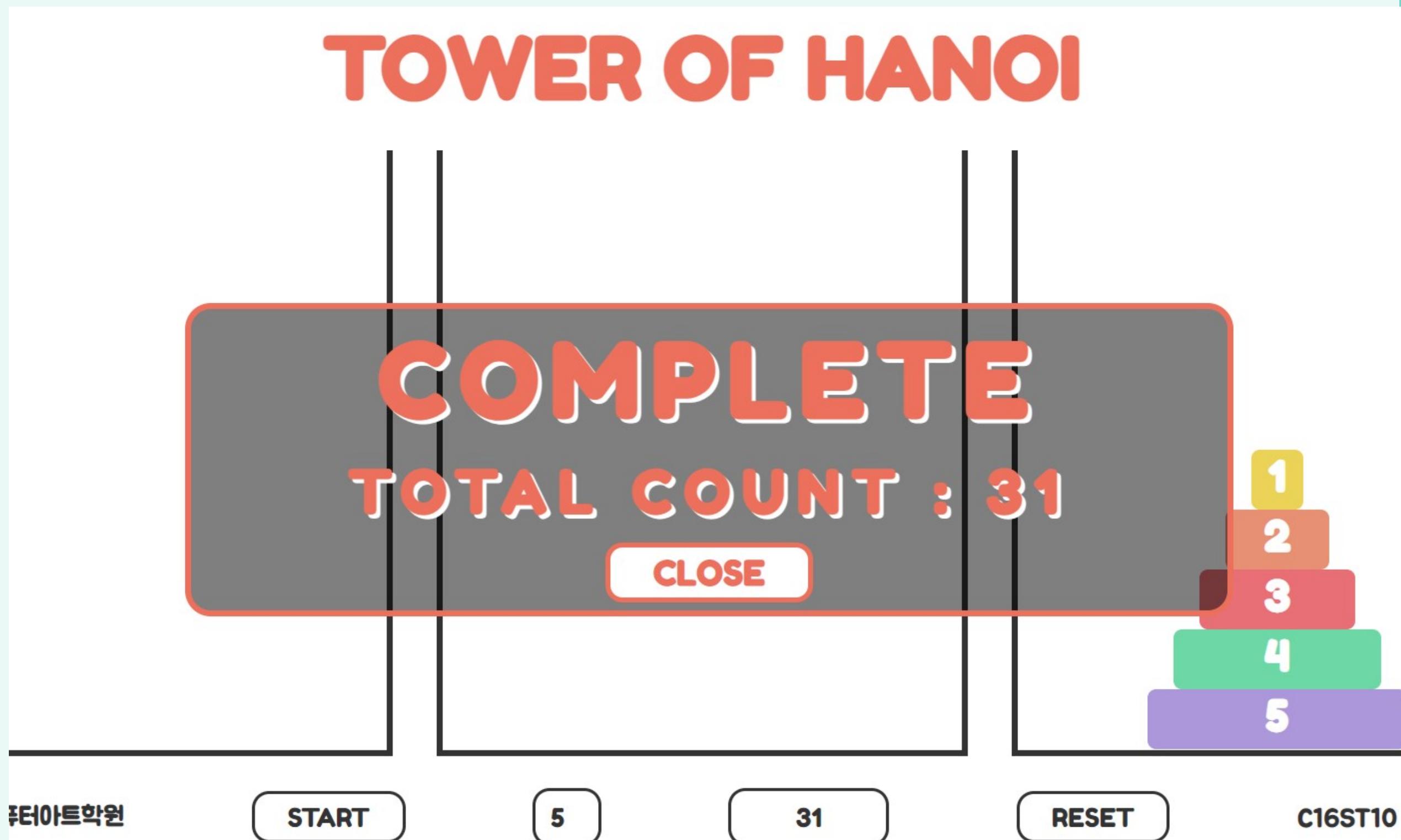
JavaScript 동작

RING 수량 설정 후 START 클릭 시 진행

RING이 해당 위치로 이동하는 모션 추가

RUN 중에는 RESET 버튼 외 조작 불가

완성 시 팝업창에 총 횟수 출력



```
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    while(true) {
        Stack<Integer> bar_1 = new Stack<>();
        Stack<Integer> bar_2 = new Stack<>();
        Stack<Integer> bar_3 = new Stack<>();
        int ring = scan.nextInt();
        for(int i=ring; i>0; i--) {
            bar_1.push(i);
        }
        if(ring == 0) {
            System.out.println("1 이상의 정수를 입력 해주시기 바랍니다.");
            System.out.println("--종료--");
            break;
        }
        move_ring(bar_1, bar_2, bar_3, ring);
    }
}
```

JAVA 구현

Scanner로 Console 창에 RING 수량 입력

Tower를 Stack으로 생성

A 탑에 RING 쌓기 진행

하노이탑 동작 Method 실행

```

public static void move_ring(Stack<Integer> bar1, Stack<Integer> bar2) {
    List<Stack<Integer>> bar_arr = new LinkedList<>(Arrays.asList(bar1, bar2));
    List<Queue<Integer>> odd = new LinkedList<>();
    List<Queue<Integer>> even = new LinkedList<>();
    List<String> tower = new LinkedList<>(Arrays.asList("A", "B", "C"));
    for(int i=1; i<=_ring; i++) {
        odd.add(new LinkedList<>(Arrays.asList(2,3,1)));
        even.add(new LinkedList<>(Arrays.asList(3,2,1)));
    }
    for(int i=1; ; i++) {
        System.out.println();
        System.out.println("===== " + i + "번째 =====");
        for(int j=_ring-1; j>=0; j--) {
            if(i%Math.pow(2,j) == 0) {
                if(_ring%2 == 1) {
                    if(j%2 == 1) {
                        int prev_pos = -1;
                        for(int k=0; k<3; k++) {
                            if(k == 2) prev_pos = odd.get(j).peek();
                            odd.get(j).offer(odd.get(j).poll());
                        }
                        bar_arr.get(odd.get(j).peek()-1).push(bar_arr.get(prev_pos-1).pop());
                        System.out.println("> " + (j+1) + "번 RING0| " + tower.get(prev_pos-1) + " -> " + tower.get(odd.get(j).size() - 1));
                        odd.get(j).offer(odd.get(j).poll());
                    } else {
                        ●
                        ●
                        ●
                    }
                }
            }
            break;
        }
        if(bar3.size() == _ring) {
            System.out.println();
            System.out.println("*****");
            System.out.println("*****하노이타워가 완성 되었습니다.*****");
            System.out.println("*****");
            break;
        }
    }
}

```

JAVA 구현

Stack과 Ring 수량이 매개변수인 Method 구현

내부에서 패턴 List 구현

패턴 중 홀수, 짝수에 따른 반복 조건 로직 구현

C 타워에 RING0I 완성되면 종료

JAVA 동작

Ring의 수량을 입력하면 이동 순서와 몇 번째 Ring의 이동경로가 나타난다.
C 타워가 순서대로 다 채워지면 완성 문구를 나타내고 종료된다.

3

===== 1번째 =====
» 1번 RING이 A → C 타워로 이동 합니다.

===== 2번째 =====
» 2번 RING이 A → B 타워로 이동 합니다.

===== 3번째 =====
» 1번 RING이 C → B 타워로 이동 합니다.

===== 4번째 =====
» 3번 RING이 A → C 타워로 이동 합니다.

===== 5번째 =====
» 1번 RING이 B → A 타워로 이동 합니다.

===== 6번째 =====
» 2번 RING이 B → C 타워로 이동 합니다.

===== 7번째 =====
» 1번 RING이 A → C 타워로 이동 합니다.

*****하노이타워가 완성 되었습니다.*****

PYTHON 구현

Python Class 구조 구현

내부 재귀 함수로 하노이탑 로직 구현

```
1  class Hanoi_Tower:
2      def __init__(self, _count):
3          self.__count = int(_count)
4          self.__from_tower = "A"
5          self.__save_tower = "B"
6          self.__to_tower = "C"
7
8
9      def move_ring(self, _count, _from, _to, _save):
10         if _count == 1:
11             print(f"{_count}번 링이 {_from} -> {_to}으로 이동합니다.")
12         if _count != 1:
13             self.move_ring(_count-1, _from, _save, _to)
14             print(f"{_count}번 링이 {_from} -> {_to}으로 이동합니다.")
15             self.move_ring(_count-1, _save, _to, _from)
16
17     def play_game(self):
18         self.move_ring(self.__count, self.__from_tower, self.__to_tower, self.__save_tower)
19
20     count = input("원판의 개수를 설정해주세요. >> ")
21     hanoi_game = Hanoi_Tower(count)
22     hanoi_game.play_game()
```

OPERATION-PYTHON

PYTHON 동작

터미널 창에서 RING 수량 입력

순서대로 해당 RING의 이동경로 출력

원판의 개수를 설정해주세요. >> 3
1번 링이 A → C으로 이동합니다.
2번 링이 A → B으로 이동합니다.
1번 링이 C → B으로 이동합니다.
3번 링이 A → C으로 이동합니다.
1번 링이 B → A으로 이동합니다.
2번 링이 B → C으로 이동합니다.
1번 링이 A → C으로 이동합니다.

REVIEW



코드를 작성하기에 앞서 하노이탑의 규칙을 먼저 이해하고
직접 하노이탑 게임을 해보며 패턴을 작성하였습니다.

먼저 규칙과 패턴에 대해 완벽히 이해한 뒤 코드 작업을 함으로써
로직을 전개하는데 더욱 수월하였습니다.

이번 하노이탑을 작업하면서 코드보다 먼저 작업해야 할 내용을
정리 및 숙지를 하는 게 더욱 중요하다는 것을 깨달았습니다.



THANK
YOU

박경호