Professional Coding Specialist

COS Pro JAVA 1 급

5 강-10 강. 모의고사 1 차

1. 모의고사 1 차(1-10 번)

과정 소개

COS Pro 1 급 JAVA 모의고사 1 차를 풀어보며 문제 유형을 익히고, JAVA 를 이용하여 알고리즘을 구현하기 위해 필요한 관련 지식을 익혀보도록 한다.

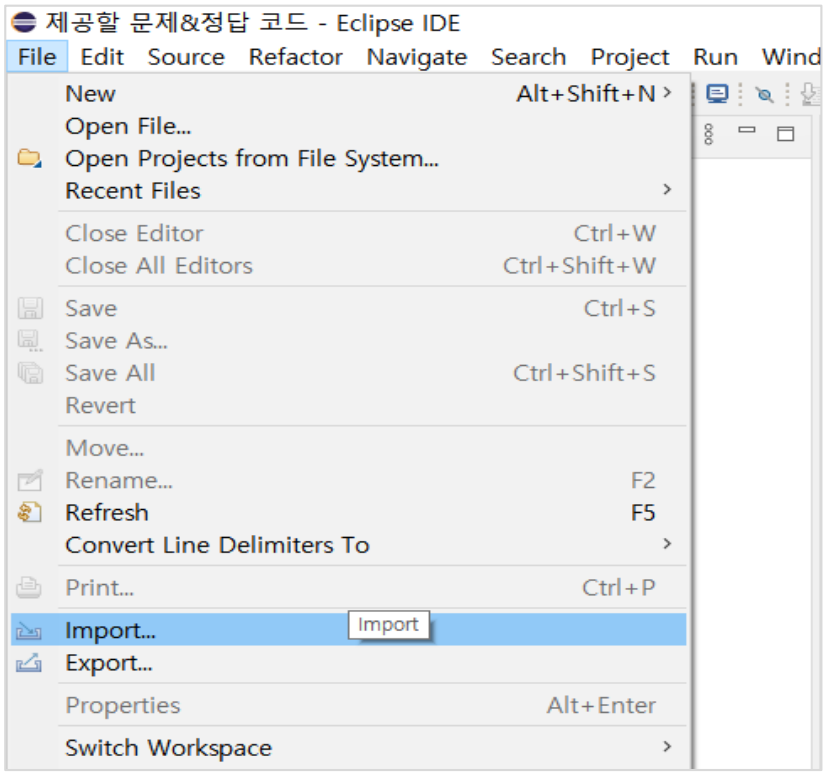
학습 목차

학습 목표

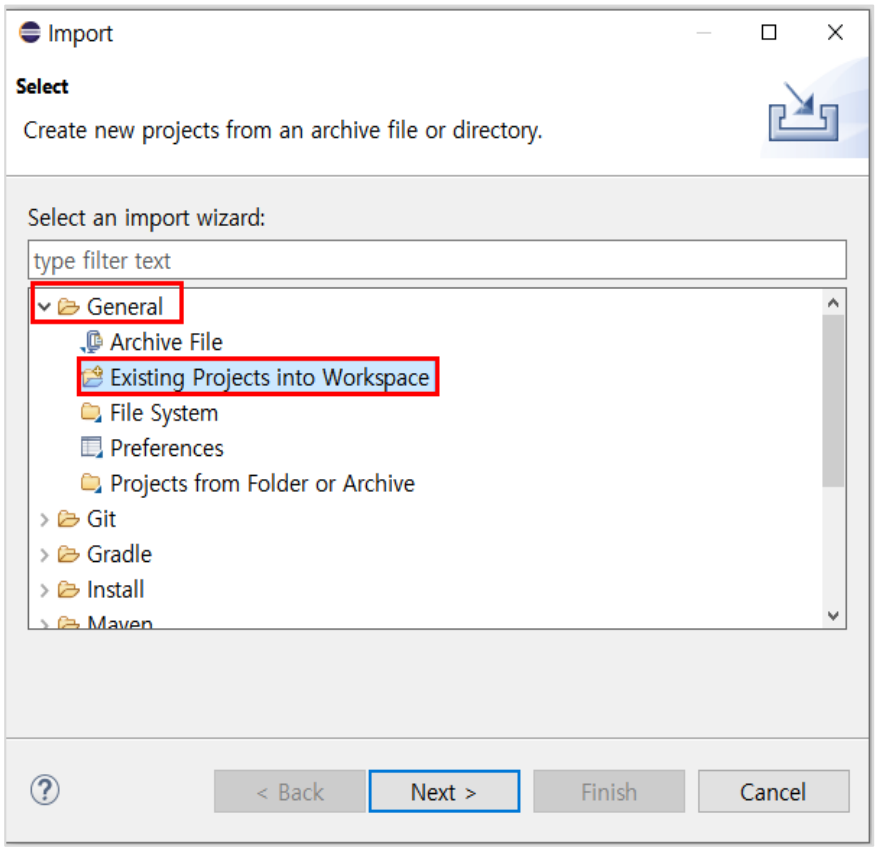
1. YBM IT(www.ybmit.com) 에서 제공하는 COS Pro 1 급 JAVA 모의고사(샘플 문제)를 풀어보며 JAVA 를 이용하여 주어진 문제를 해결하기 위한 알고리즘을 구성하는 능력을 배양한다.
2. 많이 등장하는 문제 유형을 익혀서 COS Pro 1 급 시험에 대비한다.

■ 문제 소스 파일 준비하기

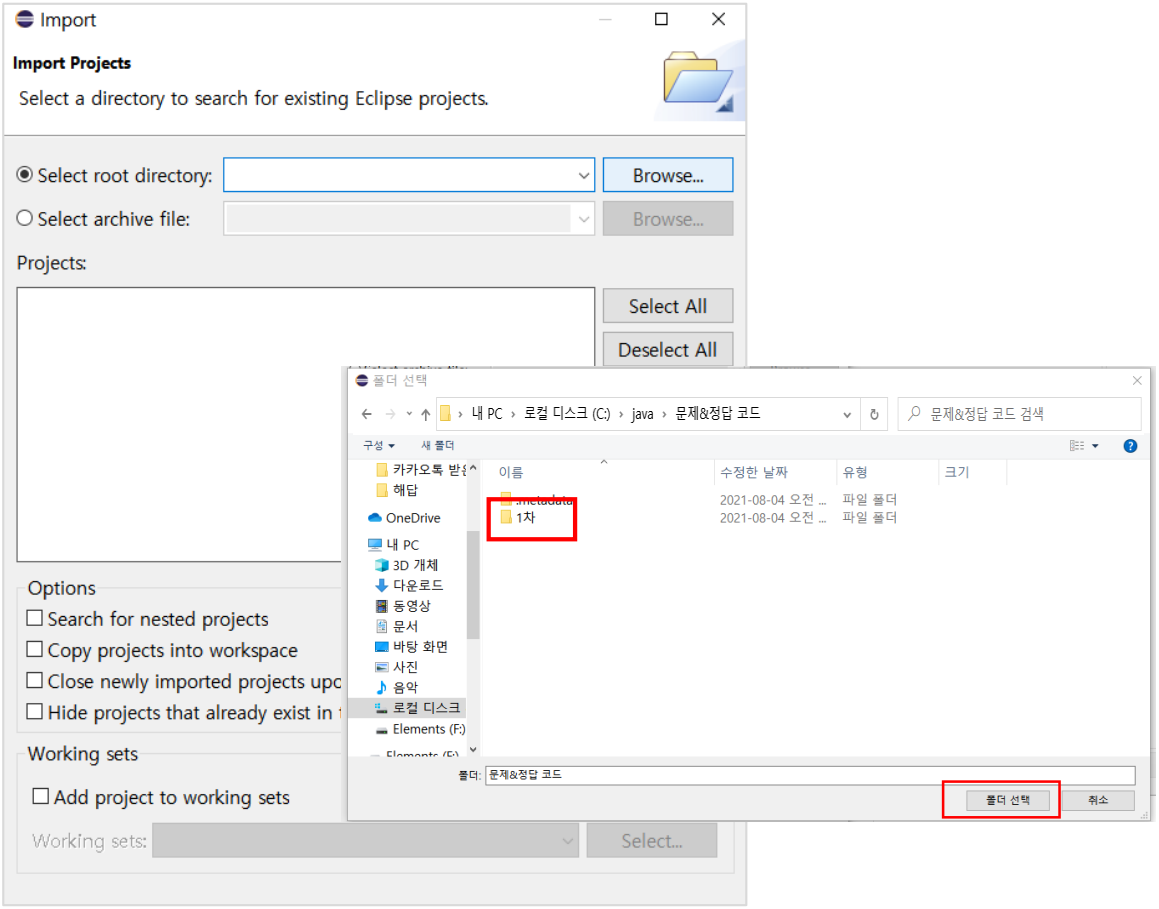
1. **[File] - [Import] 클릭**



2. **[General] - [Existing Projects into Workspace] 클릭**

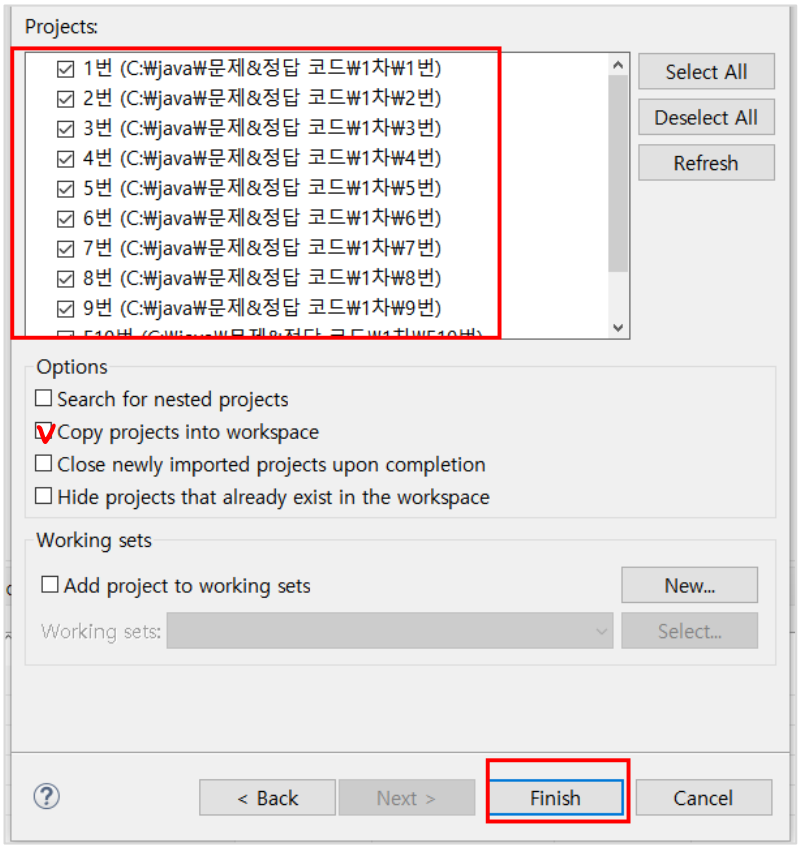


3. **[Browse] 버튼 클릭 - 각 차시별 폴더 선택**

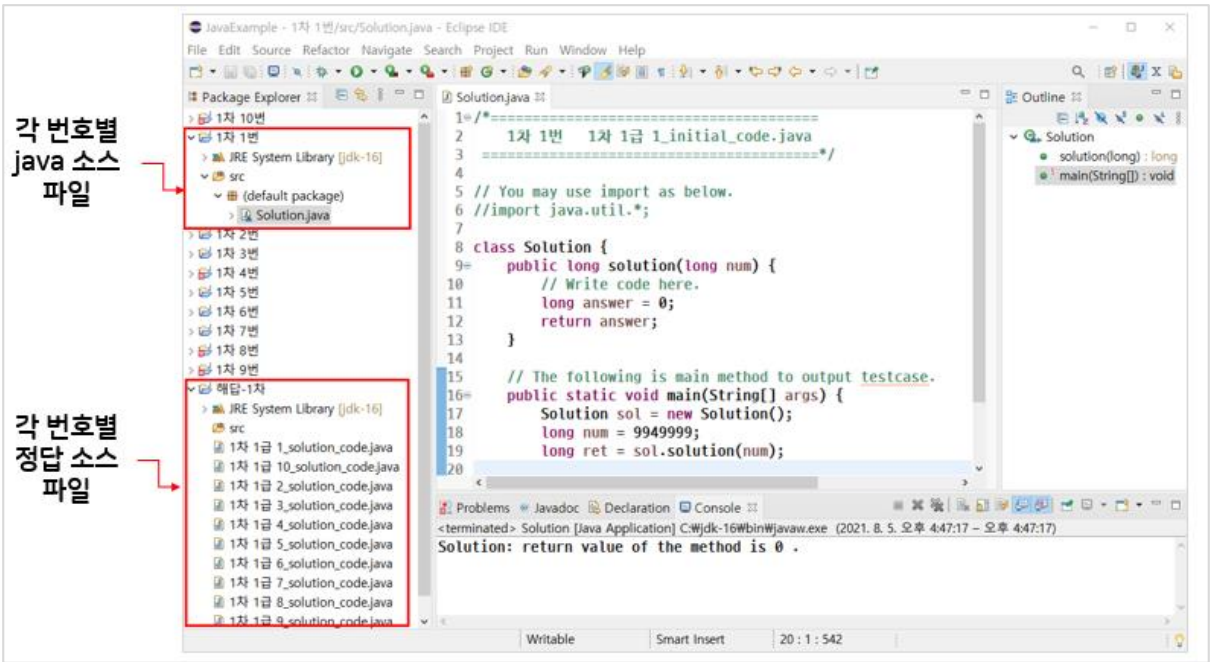


※ 다운 받은 위치에 따라 폴더 경로명은 다를 수 있음

4. **[Copy projects into workspace] 체크 후 [Finish] 버튼 클릭**



※ **각 차시별 소스 파일 구성**



1. 문제 1

1) 문제 코드

```
/*=======================================
    1차 1번   1차 1급 1_initial_code.java
 =======================================*/

// You may use import as below.
//import java.util.*;

class Solution {
    public long solution(long num) {
        // Write code here.
        long answer = 0;
        return answer;
    }

    // The following is main method to output testcase.
    public static void main(String[] args) {
        Solution sol = new Solution();
        long num = 9949999;
        long ret = sol.solution(num);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret + " .");
    }
}
```

1) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제
- 매개변수 num 으로 전달된 값에 1 을 더해서 나온 결과값에서 0 을 1 로 바꿔야 함

2) 십진수 9950000 에서 각 자리에 사용된 숫자 추출하는 방법

$9950000/10^0$ 을 10 으로 나눈 나머지	0	1 의 자리수
$9950000/10^1$ 을 10 으로 나눈 나머지	0	10 의 자리수
$9950000/10^2$ 을 10 으로 나눈 나머지	0	100 의 자리수
$9950000/10^3$ 을 10 으로 나눈 나머지	0	1000 의 자리수
$9950000/10^4$ 을 10 으로 나눈 나머지	5	10000 의 자리수
$9950000/10^5$ 을 10 으로 나눈 나머지	9	100000 의 자리수
$9950000/10^6$ 을 10 으로 나눈 나머지	9	1000000 의 자리수

3) 십진수 9950000 의 각 자리에 있는 모든 0 을 1 로 변경하는 절차

- 일의 자리에 있는 0 을 1 로 변경 → 10 의 0 제곱을 더함 : $9950000 + 10^0 = 9950001$
- 십의 자리에 있는 0 을 1 로 변경 → 10 의 1 제곱을 더함 : $9950001 + 10^1 = 9950011$
- 백의 자리에 있는 0 을 1 로 변경 → : $9950011 + 10^2 = 9950111$

10 의 2 제곱을 더함

- 천의 자리에 있는 0 을 1 로 변경 → 9950111 + 10^3 = 9951111
  10 의 3 제곱을 더함

※ 참고 자료 : 정수 타입 및 범위

자료형	크기	범위
byte	1byte	-128 ~ 127
short	2byte	-32,768 ~ 32,767
int	4byte	-2,147,483,648 ~ 2,147,483,647
long	8byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807

4) 정답

```
class Solution {
    public long solution(long num) {
        num++;
        long digit = 1;
        while (num / digit % 10 == 0) {
            num += digit;
            digit *= 10;
        }
        return num;
    }
}
```

①. 제시된 과제와 같이 인수로 전달받은 num 값을 1 만큼 증가
digit 는 1,10, 100 순으로 증가하며 num 의 각 자릿수별 숫자를 추출하기 위해 사용하는 변수

②. num 값의 1 의 자리 숫자, 10 의 자리 숫자, 100 의 자리 숫자가 0 인 동안 while 문을 실행하여 num 에 저장되어 있는 수에서 숫자가 0 인 것을 1 로 변경

- num 값의 1 의 자리수가 0 이면 1 을 더하고, num 값의 10 의 자릿수가 0 이면 10 을 더하고, num 값의 100 자릿수가 0 이면 100 을 더하여 각 자리에 있는 0 을 1 로 교체

5) 다른 코드 제안

```
public long solution(long num) {
        long answer = 0;
        String str=null;

        num++;
        str=num+"";
        str=str.replace("0", "1");
        answer= Long.parseLong(str);
        return answer;
    }
```

- 제시된 과제대로 인수로 전달받은 num 값을 1 만큼 증가
- 증가시킨 num 값을 문자열로 변환한 뒤 replace( ) 메소드를 이용하여 '0' 을 '1' 로 바꾸고 다시 Long 형으로 변환

6) 시간복잡도 알아보기

```
import java.time.Duration;
import java.time.LocalTime;

LocalTime startTime = LocalTime.now();  시작 시간

for (long i=0; i<=100000000L; i++) {
    int num=9950000;
    long digit = 1;
    while (num / digit % 10 == 0) {
        num += digit;
        digit *= 10;
    }
}

LocalTime endTime = LocalTime.now();  종료 시간

Duration d = Duration.between(startTime, endTime);
System.out.println(d.getSeconds()+"."+ d.getNano());
```

```
2.105091600
```

- Java.time 패키지의 now( )를 사용하여 프로그램 실행 시간을 측정할 수 있음.
- 반복문을 사용하여 시간이 오래 걸릴 것 같지만 생각보다 많은 데이터를 빠르게 계산

```
import java.time.Duration;
import java.time.LocalTime;

LocalTime startTime = LocalTime.now();  시작 시간
for (long i=0; i<=100000000L; i++) {
    long answer = 0;
    String str=null;

    int num=9950000;
    str=num+"";
    str=str.replace("0", "1");
    answer= Long.parseLong(str);
}
LocalTime endTime = LocalTime.now();  종료 시간

Duration d = Duration.between(startTime, endTime);
System.out.println(d.getSeconds()+"."+ d.getNano());
```

```
4.520126300
```

- Replace() 메소드를 사용하면 코드는 간편하지만, 많은 데이터를 변환하는 경우에는 숫자 계산 방식으로 작성한 정답으로 제시된 코드가 실행 속도가 빠른 것을 확인할 수 있음

2. 문제 2

1) 문제 코드

```
/*=======================================
    1차 2번   1차 1급 2_initial_code.java
 =======================================*/

// You may use import as below.
//import java.util.*;

class Solution {
    public int solution(int n) {
        // Write code here.
        int answer = 0;
        return answer;
    }

    // The following is main method to output testcase.
    public static void main(String[] args) {
        Solution sol = new Solution();
        int n1 = 3;
        int ret1 = sol.solution(n1);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret1 + " .");

        int n2 = 2;
        int ret2 = sol.solution(n2);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret2 + " .");
    }
}
```

※ 문제 출처 : https://www.ybmit.com/cos_pro/cos_pro_r_test.jsp

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제
- N x N 형태의 2 차원 배열에 소용돌이 형태로 수를 할당한 후 (1,1) 부터 (N, N) 까지 대각선 위치에 있는 수의 합을 구해야 함

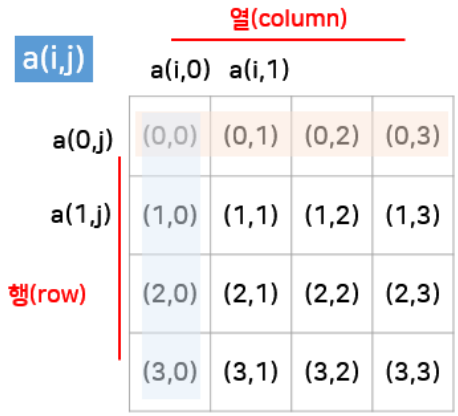
3) 보충 학습

① 그리디(Greedy) 알고리즘

- 각 단계에서 최적의 값을 찾는 알고리즘
- 대부분 뛰어난 결과를 도출하지 못하지만 때때로 최적 해를 제시하는 경우도 있음
- 대표적 사례 : 거스름돈 문제, 배낭에 짐 넣기 문제

② 행렬(Matrix)

❖ 가로 줄은 행, 세로 줄은 열로 구성한 데이터 구조로 2 차원 배열로 구현함



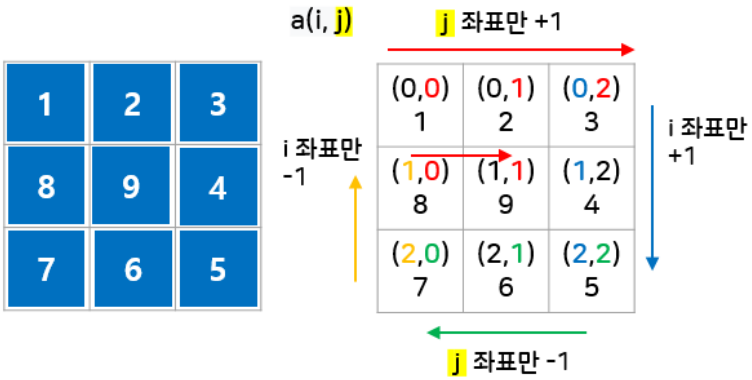
❖ JAVA 로 구현한 2 차원 배열 행렬 예

```
int[][] a= {
        {1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}
};

for(int i=0; i<a.length; i++) {
    for(int j=0; j<a[i].length; j++) {
        System.out.print(a[i][j]+"\t");
    }
    System.out.println();
}
```

```
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  16
```

4) 풀이

① N*N 행렬에 1 부터 N*N 까지의 수를 채우는 절차 이해



❖ 소용돌이 좌표 변화의 규칙: di[]={0,1,0,-1} #i 좌표가 변하는 규칙
dj[]={1,0,-1,0} #j 좌표가 변하는 규칙

- 수를 저장할 2 차원 배열을 0 으로 초기화
- 소용돌이 방향으로 좌표를 움직이기 위한 좌표 변화 규칙을 설정
- 시작 위치 좌표를 (0,0) 으로 설정
- 배열에 처음 저장하는 숫자를 1 로 지정하여 입력
- 규칙 di[0], dj[0]에 맞게 위치를 바꾸면서 숫자(+1) 입력
- 다음에 입력할 j 의 위치가 없는 경우(j >= N), 이동 방향을 바꿔야 함
- 규칙 di[1], dj[1] 에 맞게 위치를 바꾸면서 숫자(+1) 입력
- 다음에 입력할 i 의 위치가 없는 경우(i >=N), 이동 방향을 바꿔야 함
- 규칙 di[2], dj[2] 에 맞게 위치를 바꾸면서 숫자(+1) 입력
- 다음에 입력할 j 의 위치가 없는 경우(j < 0), 방향을 바꿔야 함
- 규칙 di[3], dj[2] 에 맞게 위치를 바꾸면서 숫자(+1) 입력
- 다음에 입력할 j 위치에 0 이 아닌 값이 입력되어 있으면, 방향을 바꿔야 함
- 규칙 di[0], dj[0]에 맞게 위치를 바꾸면서 숫자(+1) 입력

5) 강의에서 풀이한 코드

- 2 차원 배열을 초기화
- '소용돌이 방향으로 움직이기 위한 좌표 변화 규칙'을 배열로 설정
- 현재 위치를 (0, 0)으로 설정
- 숫자를 1 부터 입력
- k = 0 으로 초기화(소용돌이 좌표 인덱스)
- 아래를 반복(좌표가 범위 안에 있고 현재 위치의 값이 0 인 동안)
  - 소용돌이 변화 규칙[k]에 맞게 위치를 바꾸면서 숫자 값을 1 씩 증가하며 입력
  - (다음 입력 좌표가 범위를 벗어났거나) or (다음 위치의 값이 0 이 아닌 경우) : 방향을 바꾼다. → 규칙[(k+1)%4]

```
class T_Solution2{

int[][] pane;

int di[] = {0, 1, 0, -1};
int dj[] = {1, 0, -1, 0};

boolean inRange(int i, int j, int n){
    return 0 <= i && i < n && 0 <= j && j < n;
}

public int solution(int n){
    pane = new int[n][n];
    int ci = 0;
    int cj = 0;
    int num = 1;
    int k=0;

    while(inRange(ci, cj, n) && pane[ci][cj] == 0){
                pane[ci][cj] = num++;
                int ni = ci + di[k];
                int nj = cj + dj[k];
                if(!inRange(ni, nj, n) || pane[ni][nj] != 0) {
                    k=(k + 1) % 4;
                }
                ci += di[k];
                cj += dj[k];
    }

    for(int i = 0; i < n; i++) {
        for(int j=0; j<n; j++) {
            System.out.print(pane[i][j]+"\t");
         }
        System.out.println();
    }

    int ans = 0;
    for(int i = 0; i < n; i++) ans += pane[i][i];
    return ans;
}
```

- **전역 변수와 함수**
  - pane 을 n x n 형태의 2 차원 배열로 생성. 전역 변수로 선언
  - inRange() 함수 선언. 설정된 위치가 범위 안에 있다면 true 를 return
  - 가로 방향으로 진행할 때 변화하는 값은 순서대로 {1, 0, -1, 0}, 세로 방향으로 진행할 때 변화하는 값은 순서대로 {0, 1, 0, -1}. → 이것을 dj[] = {1, 0, -1, 0}, di[] = {0, 1, 0, -1} 로 구현

- **Solution 메소드**
  - ci, cj 는 배열의 현재 위치를 가리키는 인덱스를 나타내고, num 은 항목에 할당하는 값을 나타냄 → ci = 0, cj = 0, num = 1 로 초기화
  - ci, cj 가 배열 범위 내에 들어가고, ci, cj 가 가리키는 배열의 항목 값이 0 인 동안 while 문을 이용하여 배열의 항목 값을 할당하는 작업을 반복 실행

    배열에 소용돌이 순서대로 값을 할당하기 위해서

    ① 좌표 변화 규칙 di, dj 를 순차적으로 적용. 범위를 벗어났거나, ci, cj 가 가리키는 배열의 항목 값이 0 이 아니라면 규칙 적용을 중지

    ② ci, cj 가 가리키는 배열 항목에 num 값을 할당

    ③ 다음 번 값을 할당할 위치인 ni, nj 에 di, dj 의 규칙을 적용

    ④ ni, nj 가 범위를 벗어났거나 ni, nj 가 가리키는 배열의 항목 값이 0 이 아니라면, di, dj 에서 다음 인덱스의 규칙([(k+1)%4])을 ci, cj 에 새로 적용한 후 현재 while()을 빠져나감. 그렇지 않으면 ci, cj 에 ni, nj 를 그대로 할당하여 현재 규칙을 적용
  - 2 차원 배열에 모든 값을 할당한 후 for 문을 이용하여 대각선 위치에 있는 항목의 합을 계산

6) 정답지 코드

- 좌표 변화 규칙을 나타내는 변수 이름이 di, dj 대신 dx, dy 이고, if 문에 break 사용

```
int[][] pane;
int dx[] = {0, 1, 0, -1};
int dy[] = {1, 0, -1, 0};
boolean inRange(int i, int j, int n){
    return 0 <= i && i < n && 0 <= j && j < n;
}
public int solution(int n){
    pane = new int[n][n];
    int ci = 0;
    int cj = 0;
    int num = 1;
    while(inRange(ci, cj, n) && pane[ci][cj] == 0){
        for(int k = 0; k < 4; k++){
            if(!inRange(ci, cj, n) || pane[ci][cj] != 0) break;
            while(true){
                pane[ci][cj] = num++;
                int ni = ci + dy[k];
                int nj = cj + dx[k];
                if(!inRange(ni, nj, n) || pane[ni][nj] != 0){
                    ci += dy[(k + 1) % 4];
                    cj += dx[(k + 1) % 4];
                    break;
                }
                ci = ni;
                cj = nj;
            }
        }
    }
    int ans = 0;
    for(int i = 0; i < n; i++) ans += pane[i][i];
    return ans;
}
```

3. 문제 3

1) 문제 코드

```
/*=======================================
    1차 3번   1차 1급 3_initial_code.java
 =======================================*/

// You may use import as below.
//import java.util.*;

class Solution {
    public int solution(String pos) {
        // Write code here.
        int answer = 0;
        return answer;
    }

    // The following is main method to output testcase.
    public static void main(String[] args) {
        Solution sol = new Solution();
        String pos = "A7";
        int ret = sol.solution(pos);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret + " .");
    }
}
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제
- 수평 1 칸 + 수직 2 칸 혹은 수평 2 칸 + 수직 1 칸으로 이동 가능한 기물인 나이트(knight)가 8 x 8 체스판에서 범위를 벗어나지 않고 이동할 수 있는 위치의 개수를 집계

ex) 나이트의 현재 위치가 D4 인 경우 나이트가 이동할 수 있는 위치
: 현재 위치 D4 를 (x, y)좌표로 나타내면 (3, 3)

현재 위치에서 오른쪽 1 칸+위로 2 칸 이동	(x+1, y+2) → (4, 5)
현재 위치에서 오른쪽 1 칸+아래로 2 칸 이동	(x+1, y-2) → (4, 1)
현재 위치에서 왼쪽 1 칸+위로 2 칸 이동	(x-1, y+2) → (2, 5)
현재 위치에서 왼쪽 1 칸+아래로 2 칸 이동	(x-1, y-2) → (2, 1)
현재 위치에서 오른쪽 2 칸+위로 1 칸 이동	(x+2, y+1) → (5, 4)
현재 위치에서 오른쪽 2 칸+아래로 1 칸 이동	(x+2, y-1) → (5, 2)
현재 위치에서 왼쪽 2 칸+위로 1 칸 이동	(x-2, y+1) → (1, 4)
현재 위치에서 왼쪽 2 칸+아래로 1 칸 이동	(x-2, y-1) → (1, 2)

➔ 체스판을 벗어나지 않으면서 이동할 수 있는 위치 개수 = 8 개

3) 정답

```
public int solution(String pos) {
    int dx[]= {1,2,2,1,-1,-2,-2,-1};
    int dy[]= {2,1,-1,-2,-2,-1,1,2};

    int cx = pos.charAt(0)-'A';
    int cy = pos.charAt(1)-'0'-1;

    int answer = 0;

    for(int i=0;i<8;i++) {
        int nx = cx + dx[i];
        int ny = cy + dy[i];
        if (nx>=0 && nx<8 && ny>=0 && ny<8) {
            answer++;
        }
    }
    return answer;
```

① dx 에는 가로방향으로 바꿀 수 있는 변화값을 순서대로 나열해서 배열로 생성
dy 에는 세로방향으로 바꿀 수 있는 변화값을 순서대로 나열해서 배열로 생성

② cx = (pos 변수로 전달된 값의 첫 번째 문자를 숫자로 변환한 값 - 'A' 의 숫자 값) 을 계산하여 현재 위치에 대한 x 좌표값을 구함
cy = (pos 변수로 전달된 값의 두 번째 문자를 숫자로 변환한 값 - '0' 의 숫자 값) - 1 혹은 cy = int(pos 변수로 전달된 값의 두 번째 문자) - 1 을 계산하여 현재 위치에 대한 y 좌표 값을 구함

③ for 문을 이용하여 현재 좌표인 (cx, cy) 에 dx 와 dy 를 이용하여 8 가지 변화 값을 적용한 좌표 (nx, ny)가 체스판의 범위인 0 이상 8 미만에 해당하는지 확인하며 answer 에 집계

4) 문법 정리

- 문자는 ASCII 코드 값을 가지고 있으므로 int 로 형변환하면 ASCII 코드 값을 확인

```
System.out.println((int)('A'));
System.out.println((int)('0'));
```

```
65
48
```

- index 번호에 해당하는 문자 하나 가져오기 : charAt(int index)

```
String s="ABC";

System.out.println(s.charAt(0));
System.out.println(s.charAt(1));

System.out.println(s.charAt(0)-'A');
System.out.println(s.charAt(1)-'A');
```

```
A
B

0
1
```

- 문자와 문자 사이의 차이를 구하면 ASCII 코드 값의 차이 값이 출력

```
char c='A';
System.out.println(c-'A');
```
0

```
char i='3';
System.out.println(i-'1');
```
2

4. 문제 4

1) 문제 코드

```
/*=======================================
    1차 4번   1차 1급 4_initial_code.java
 =======================================*/

import java.util.*;

class Solution {
    public int[] solution(int[] arrA, int[] arrB) {
        int arrA_idx = 0, arrB_idx = 0;
        int arrA_len = arrA.length;
        int arrB_len = arrB.length;
        int answer[] = new int[arrA_len + arrB_len];
        int answer_idx = 0;
        while(@@@){
            if(arrA[arrA_idx] < arrB[arrB_idx])
                answer[answer_idx++] = arrA[arrA_idx++];
            else
                answer[answer_idx++] = arrB[arrB_idx++];
        }
        while(@@@)
            answer[answer_idx++] = arrA[arrA_idx++];
        while(@@@)
            answer[answer_idx++] = arrB[arrB_idx++];
        return answer;
    }

    // The following is main method to output testcase.
    public static void main(String[] args) {
        Solution sol = new Solution();
        int[] arrA = {-2, 3, 5, 9};
        int[] arrB = {0, 1, 5};
        int[] ret = sol.solution(arrA, arrB);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + Arrays.toString(ret) + " .");
    }
}
```

2) 문제 개요

- 두 개의 정렬된 배열을 오름차순으로 정렬된 상태로 합치도록 프로그램 빈 칸의 구문을 완성하는 문제(병합 정렬)

3) 정답

```
public int[] solution(int[] arrA, int[] arrB) {

    int arrA_idx = 0, arrB_idx = 0;
    int arrA_len = arrA.length;
    int arrB_len = arrB.length;
    int answer[] = new int[arrA_len + arrB_len];
    int answer_idx = 0;

    while( arrA_idx < arrA_len && arrB_idx < arrB_len){
        if(arrA[arrA_idx] < arrB[arrB_idx])
            answer[answer_idx++] = arrA[arrA_idx++];
        else
            answer[answer_idx++] = arrB[arrB_idx++];
    }

    while( arrA_idx < arrA_len )        A가 남은 경우
        answer[answer_idx++] = arrA[arrA_idx++];

    while( arrB_idx < arrB_len )        B가 남은 경우
        answer[answer_idx++] = arrB[arrB_idx++];

    return answer;
}
```

① arrA, arrB 배열의 항목을 가리키는 인덱스 변수 arrA_idx, arrB_idx 를 각각 0 으로 초기화
arrA, arrB 배열의 길이를 구하여 arrA_len, arrB_len 에 저장

② 두 개의 배열 항목을 가리키는 인덱스 변수 값이 모두 배열의 길이보다 작은 동안 반복 실행
➔ arrA 배열의 항목과 arrB 배열의 항목의 크기를 비교해서 작은 값을 answer 에 추가하고 해당 배열 항목을 가리키는 인덱스 값을 1 만큼 증가시킴

③ arrA 를 가리키는 인덱스 arrA_idx 가 arrA 의 길이보다 작은 동안 남아 있는 나머지 항목을 answer 에 추가
arrB 를 가리키는 인덱스 arrB_idx 가 arrB 의 길이보다 작은 동안 남아 있는 나머지 항목을 answer 에 추가

4) 다른 코드 제안

- System.arraycopy()를 이용해 배열을 병합한 후 Arrays.sort()를 활용하여 배열을 정렬하는 방법을 사용

```java
import java.util.Arrays;
import java.util.Collections;

public class T_Solution4 {
    public static void main(String[] args) {


        int[] arrA = {-2, 3, 5, 9};
        int[] arrB = {0, 1, 5};
        int[] arrS=new int[arrA.length+arrB.length];

        //배열 병합
        System.arraycopy(arrA, 0, arrS, 0, arrA.length);
        System.arraycopy(arrB, 0, arrS, arrA.length, arrB.length);


        for(int i =0; i<7; i++) System.out.print(arrS[i]+"\t");
        System.out.println();

        //정렬
        Arrays.sort(arrS);

        for(int i =0; i<7; i++) System.out.print(arrS[i]+"\t");

        //역순 정렬
        Integer[] arrS2= Arrays.stream(arrS).boxed().toArray(Integer[]::new);

        Arrays.sort(arrS2,Collections.reverseOrder());

        for(int k:arrS2) System.out.print(k+"\t");
        }
}
```

- System.arraycopy() : 배열 복사를 이용해 배열 병합

```
System.arraycopy(src, srcPos, dest, destPos, length)
```

원본, 원본 시작 위치, 복사본, 복사본 시작 위치, 총 길이

- Arrays.sort() : 배열 정렬 수행하면 두 배열이 병합됨

```
Arrays.sort(배열)
```

- 참고 : 역순 정렬하기

```java
Integer[] arrS2= Arrays.stream(arrS).boxed().toArray(Integer[]::new);

Arrays.sort(arrS2,Collections.reverseOrder());
```

5. 문제 5

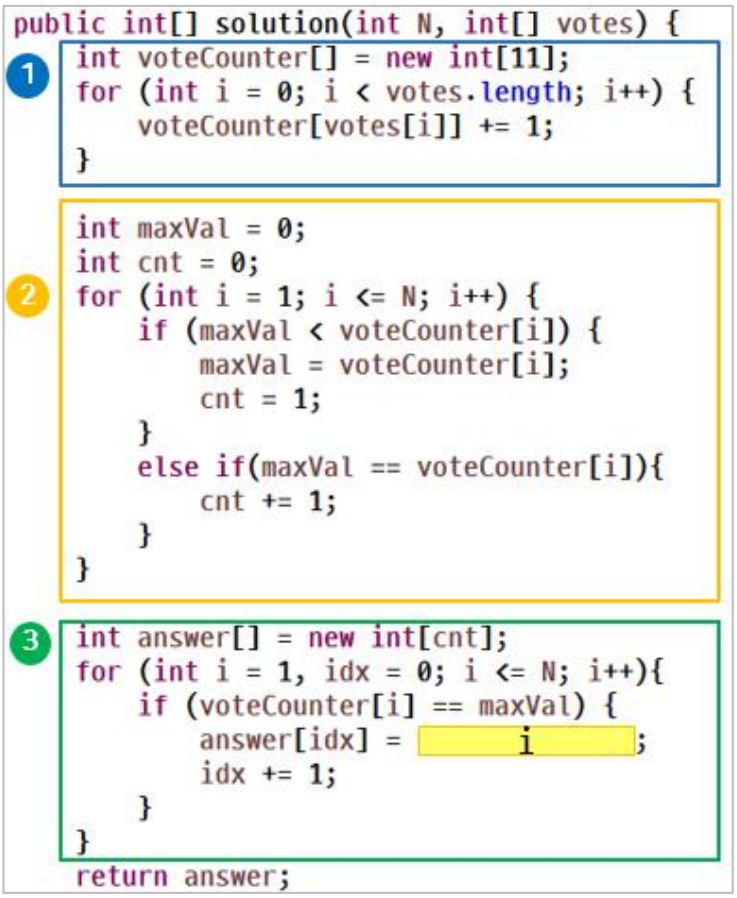
1) 문제 코드

```java
public int[] solution(int N, int[] votes) {
    int voteCounter[] = new int[11];
    for (int i = 0; i < votes.length; i++) {
        voteCounter[votes[i]] += 1;
    }
    int maxVal = 0;
    int cnt = 0;
    for (int i = 1; i <= N; i++) {
        if (maxVal < voteCounter[i]) {
            maxVal = voteCounter[i];
            cnt = 1;
        }
        else if(maxVal == voteCounter[i]){
            cnt += 1;
        }
    }
    int answer[] = new int[cnt];
    for (int i = 1, idx = 0; i <= N; i++){
        if (voteCounter[i] == maxVal) {
            answer[idx] = voteCounter[i];
            idx += 1;
        }
    }
    return answer;
}
```

2) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제
- 투표 결과를 저장하고 있는 votes 배열의 항목 값을 voteCounter 배열의 인덱스로 활용하여 득표수를 집계

3) 정답



```
public int[] solution(int N, int[] votes) {
    int voteCounter[] = new int[11];
    for (int i = 0; i < votes.length; i++) {
        voteCounter[votes[i]] += 1;
    }

    int maxVal = 0;
    int cnt = 0;
    for (int i = 1; i <= N; i++) {
        if (maxVal < voteCounter[i]) {
            maxVal = voteCounter[i];
            cnt = 1;
        }
        else if(maxVal == voteCounter[i]){
            cnt += 1;
        }
    }

    int answer[] = new int[cnt];
    for (int i = 1, idx = 0; i <= N; i++){
        if (voteCounter[i] == maxVal) {
            answer[idx] =        i       ;
            idx += 1;
        }
    }
    return answer;
```

① N+1 개의 항목을 갖는 voteCounter 를 생성하고 모든 항목값을 0 으로 초기화 (문제에서는 11 개를 고정하여 초기화 하였음)
for 문을 이용하여 매개변수 votes 에 있는 투표값을 하나씩 가져와 votes 의 항목값과 동일한 인덱스로 갖는 voteCounter 의 항목을 1 만큼 증가 → 투표 번호별 득표 수를 voteCounter 배열의 항목값으로 집계하게 됨

② 최대 득표수를 찾기 위하여 voteCounter 에 저장되어 있는 항목값 중 최댓값을 찾아서 maxVal 에 저장

③ for 문을 사용하여 voteCounter 의 항목들을 하나씩 가져와 최대 득표수와 동일한 항목 값을 찾아서 최대 득표수와 동일한 항목 값을 갖는 인덱스를 answer 배열에 추가하는 것으로 수정

6. 문제 6

1) 문제 코드

```java
/*=======================================
    1차 6번   1차 1급 6_initial_code.java
 =======================================*/

class Solution{
    public int func(int record){
        if(record == 0) return 1;
        else if(record == 1) return 2;
        return 0;
    }

    public int solution(int[] recordA, int[] recordB){
        int cnt = 0;
        for(int i = 0; i < recordA.length; i++){
            if(recordA[i] == recordB[i])
                continue;
            else if(recordA[i] == func(recordB[i]))
                cnt = cnt + 3;
            else
                cnt = cnt - 1;
        }
        return cnt;
    }

    // The following is main method to output testcase. The main method is correct a
    public static void main(String[] args) {
        Solution sol = new Solution();
        int[] recordA = {2,0,0,0,0,0,1,1,0,0};
        int[] recordB = {0,0,0,0,2,2,0,2,2,2};
        int ret = sol.solution(recordA, recordB);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret + " .");
    }
}
```

2) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제
- 가위, 바위, 보를 했을 때 이긴 경우는 계단 위치를 나타내는 cnt 변수를 3 만큼 증가시키고, 졌을 경우는 cnt 변수 값을 1 만큼 감소. 단, 현재 위치가 계단의 제일 아래인 경우에는 졌을 때도 cnt 변수값을 감소시키지 않는다는 것을 구현해야 함

3) 정답

```
① public int func(int record){

      if(record == 0) return 1;
      else if(record == 1) return 2;
      return 0;
  }                          "B가 낸 것에 대해 승리하기 위해 A가 내야 하는 것"

public int solution(int[] recordA, int[] recordB){
    int cnt = 0;

② for(int i = 0; i < recordA.length; i++){

      if(recordA[i] == recordB[i])
          continue;                          무승부
      else if(recordA[i] == func(recordB[i]))
          cnt = cnt + 3;                     A 승리
      else
          cnt = Math.max(0,cnt-1)            A 패배
  }
    return cnt;
}
```

① func( ) 함수는 전달받은 가위/바위/보 에 대해서 승리하는 가위/바위/보를 리턴

- 매개변수 record 의 값이 0(가위)인 경우 0(가위)을 이기는 값인 1(바위)을 리턴
- 매개변수 record 의 값이 1(바위)인 경우 1(바위)을 이기는 값인 2(보)를 리턴
- 매개변수 record 의 값이 0(가위), 1(바위) 이 아닌 2(보)인 경우 2(보)를 이기는 값인 0(가위)을 리턴

② for 문을 이용하여 recordA 와 recordB 에 있는 가위/바위/보 항목들을 순서대로 비교

- recordA 의 항목과 recordB 항목 값이 같으면 비김 : 다음 번 반복 수행
- recordA 의 항목 값이 recordB 의 항목을 이기는 값과 같은 경우 cnt 에 3 을 더함
- recordA 의 항목이 recordB 의 항목에 지는 값인 경우 cnt 를 1 만큼 감소. 단, 계단이 제일 아래인 경우에는 0 으로 유지해야 하므로 Math.max( ) 를 이용하여 0 과 (cnt-1) 중 큰 값을 cnt 에 할당하도록 코드를 수정

4) 다른 코드 제안

① A 가 승리하기 위한 패턴 규칙

A	B	
가위 0	보 2	(2+1)%3 = 0
바위 1	가위 0	(0+1)%3 = 1
보 2	바위 1	(1+1)%3 = 2

- B 가 낸 값에 1 을 더한 것을 3 으로 나눈 나머지가 A 와 같다면 B 가 패배(A 가 승리)

② 작성한 코드

```java
  int[] recordA = {2,0,0,0,0,0,1,1,0,0};
  int[] recordB = {0,0,0,0,2,2,0,2,2,2};

  int cnt=0;

  for(int i=0; i<recordA.length ; i++) {

      if(recordA[i]==((recordB[i]+1)%3))
          cnt+=3;
      else if ((recordA[i]!=recordB[i])&& (cnt>0))
          cnt-=1;
  }

 System.out.println(cnt);
```

7. 문제 7

1) 문제 코드

```java
/*=======================================
    1차 7번   1차 1급 7_initial_code.java
 =======================================*/

class Solution{
    int solution(int[] prices){
        int INF = 1000000001;
        int tmp = INF;
        int answer = -INF;
        for(int price : prices){
            if(tmp != INF)
                answer = Math.max(answer, tmp - price);
            tmp = Math.min(tmp, price);
        }
        return answer;
    }

    // The following is main method to output testcase. The main method is correct and
    public static void main(String[] args) {
        Solution sol = new Solution();
        int[] prices1 = {1, 2, 3};
        int ret1 = sol.solution(prices1);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret1 + " .");

        int[] prices2 = {3, 1};
        int ret2 = sol.solution(prices2);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret2 + " .");
    }
}
```

2) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제
- 주식 가격 배열에서 매도 값(뒤에 있는 항목) - 매수 값(앞에 있는 항목)을 계산한 것 중 가장 큰 값 찾기
- 주식을 팔 때는 반드시 먼저 매수한 후에 매도해야 하므로 주식 가격 배열의 최댓값 - 최솟값을 하게 되면 원하는 값을 구할 수 없음

3) 정답

```
int solution(int[] prices){
    int INF = 1000000001;
    int tmp = INF;
    int answer = -INF;
    for(int price : prices){
        if(tmp != INF)
            answer = Math.max(answer, price-tmp );
        tmp = Math.min(tmp, price);
    }
    return answer;
```

① answer 변수에는 임의의 최솟값을, tmp 변수(이전에 가져온 주식 가격 중 최솟값을 저장하기 위해 사용)에는 임의의 최댓값을 할당

② for 문을 이용하여 prices 배열에 있는 주식 가격을 차례로 가져와서 tmp 변수가 초기 값이 아닌 경우 (현재 가져온 주식 가격 - 이전에 가져온 주식 가격 중 최솟값)을 계산한 것과 answer 에 저장된 값 중 큰 값을 Math.max( ) 를 이용하여 answer 변수에 할당

③ for 문 마지막에는 tmp 값과 현재 가져온 주식 가격 중 작은 값을 Math.min( )을 이용하여 tmp 변수에 저장

4) 다른 코드 제안

```
int solution(int[] prices){
    int answer = -1000000001;
    int tmpProfit;

    for(int i=0; i< prices.length ; i++){
        for(int j=i+1; j<prices.length; j++) {
            tmpProfit=prices[j]-prices[i];
            answer=Math.max(answer,tmpProfit);
        }
    }
    return answer;
}
```

- 선택 정렬 방법처럼 첫 번째 for 문에 의해서 매수 금액을 선택하면 두 번째 for 문을 이용하여 매수 금액 뒤에 등장하는 매도 금액을 모두 가져와 (매도 금액 - 매수 금액) 을 계산한 것 중 큰 값을 answer 에 저장
- 시간 복잡도 = $O(n^2)$ 이므로 정답으로 제시된 코드보다 시간이 더 오래 걸림

8. 문제 8

1) 문제 코드

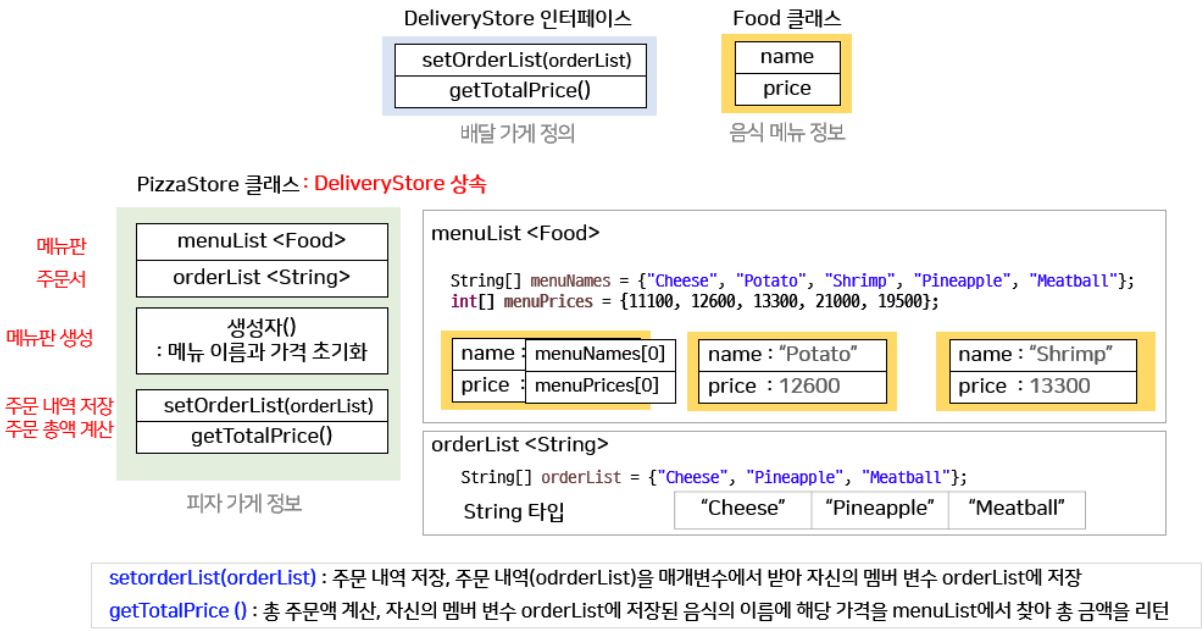
```
/*
    1차 8번   1차 1급 8_initial_code.java
 ========================================*/

import java.util.ArrayList;
import java.util.Iterator;

//DeliveryStore interface, Food and PizzaStore class are written as Inner Class. Read the
class Solution2 {
    interface DeliveryStore{
        public void setOrderList(String[] orderList);
        public int getTotalPrice();
    }

    class Food{
        public String name;
        public int price;
        public Food(String name, int price){
            this.name = name;
            this.price = price;
        }
    }

    class PizzaStore @@@ {
        private ArrayList<Food> menuList;
        private ArrayList<String> orderList;

        public PizzaStore(){
            //init menuList
            menuList = new ArrayList<Food>();
            String[] menuNames = {"Cheese", "Potato", "Shrimp", "Pineapple", "Meatball"};
            int[] menuPrices = {11100, 12600, 13300, 21000, 19500};
            for(int i = 0; i < 5; i++)
                menuList.add(new Food(menuNames[i], menuPrices[i]));

            //init orderList
            orderList = new ArrayList<String>();
        }

        public @@@{
            for(int i = 0; i < orderList.length; i++)
                this.orderList.add(orderList[i]);
        }
```

```
        public @@@{
            int totalPrice = 0;
            Iterator<String> iter = orderList.iterator();
            while (iter.hasNext()) {
                String order = iter.next();
                for(int i = 0; i < menuList.size(); i++)
                    if(order.equals(menuList.get(i).name))
                        totalPrice += menuList.get(i).price;

            }
            return totalPrice;
        }
    }

    public int solution(String[] orderList) {

        DeliveryStore deliveryStore = new PizzaStore();

        deliveryStore.setOrderList(orderList);
        int totalPrice = deliveryStore.getTotalPrice();

        return totalPrice;
    }

    // The following is main method to output testcase.
    public static void main(String[] args) {
        Solution2 sol = new Solution2();
        String[] orderList = {"Cheese", "Pineapple", "Meatball"};
        int ret = sol.solution(orderList);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret + " .");
    }
}
```

2) 문제 개요

- 인터페이스 DeliveryStore 를 상속 받는 PizzaStore 클래스를 정의하는 문제
- 인터페이스 DeliveryStore 에 있는 추상 메소드인 setOrderList( ), getTotalPrice( )를 자식 클래스인 PizzaStore 클래스에서 정의해야 PizzaStore 클래스에 대한 객체를 생성해서 사용할 수 있음

3) 전체 구조도



setorderList(orderList) : 주문 내역 저장, 주문 내역(odrderList)을 매개변수에서 받아 자신의 멤버 변수 orderList에 저장
getTotalPrice () : 총 주문액 계산, 자신의 멤버 변수 orderList에 저장된 음식의 이름에 해당 가격을 menuList에서 찾아 총 금액을 리턴

4) 정답

1
```
interface DeliveryStore{
    public void setOrderList(String[] orderList);
    public int getTotalPrice();
}
```

2
```
class Food{
    public String name;
    public int price;
    public Food(String name, int price){
        this.name = name;
        this.price = price;
    }
}
```
생성자
멤버 변수
매개 변수

```
class PizzaStore @@@ {    implements DeliveryStore
3 private ArrayList<Food> menuList;
  private ArrayList<String> orderList;

  public PizzaStore(){
      //init menuList
      menuList = new ArrayList<Food>();
      String[] menuNames = {"Cheese", "Potato", "Shrimp", "Pineapple", "Meatball"};
      int[] menuPrices = {11100, 12600, 13300, 21000, 19500};
      for(int i = 0; i < 5; i++)
          menuList.add(new Food(menuNames[i], menuPrices[i]));

      //init orderList
      orderList = new ArrayList<String>();
  }

  public @@@{  void setOrderList(String[] orderList)
      for(int i = 0; i < orderList.length; i++)
          this.orderList.add(orderList[i]);
  }

  public @@@{  int getTotalPrice()
      int totalPrice = 0;
      Iterator<String> iter = orderList.iterator();
      while (iter.hasNext()) {
          String order = iter.next();
          for(int i = 0; i < menuList.size(); i++)
              if(order.equals(menuList.get(i).name))
                  totalPrice += menuList.get(i).price;
      }
      return totalPrice;
  }
```

① DeliveryStore 클래스를 인터페이스 형태로 정의
- 추상 메소드 setOrderList( ) 와 getTotalPrice( )를 생성
- 두 개의 추상 메소드는 DeliveryStore 를 상속받는 자식 클래스에서 재정의되야 함

② Food 클래스를 정의
- 생성자를 이용하여 멤버변수 name, price 를 초기화

③ 인터페이스 DeliveryStore 를 상속하는 PizzaStore 클래스 정의 : 빈 칸(@@@)에 DeliveryStore 인터페이스를 implements
- menuList 를 멤버 변수로 생성하고, Food 클래스를 기초로 객체를 생성하여 menuList 의 항목으로 추가
- orderList 를 멤버 변수로 생성하여 빈 배열을 할당
- 부모 인터페이스 DeliveryStore 안에 있는 추상 메소드인 setOrderList( )를 재정의
  : void setOrderList(String[] orderList) 주문 메뉴를 받아 orderList 에 저장
- 부모 인터페이스 DeliveryStore 안에 있는 추상 메소드인 getTotalPrice( )를 재정의
  : int getTotalPrice( ) orderList 에 있는 음식 가격의 총합을 return

9. 문제 9

1) 문제 코드

```
/*=====================================
    1차 9번   1차 1급 9_initial_code.java
 =====================================*/

class Solution {
    public String func_a(String str, int len){
        String padZero = "";
        int padSize = @@@;
        for(int i = 0; i < padSize; i++)
            padZero += "0";
        return padZero + str;
    }

    public int solution(String binaryA, String binaryB) {
        int maxLength = Math.max(binaryA.length(), binaryB.length());
        binaryA = func_a(binaryA, maxLength);
        binaryB = func_a(binaryB, maxLength);

        int hammingDistance = 0;
        for(int i = 0; i < maxLength; i++)
            if(@@@)
                hammingDistance += 1;
        return hammingDistance;
    }

    // The following is main method to output testcase.
    public static void main(String[] args) {
        Solution sol = new Solution();
        String binaryA = "10010";
        String binaryB = "110";
        int ret = sol.solution(binaryA, binaryB);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret + " .");
    }
}
```

2) 문제 개요

- 프로그램의 알고리즘에 맞는 구문을 완성하는 문제
- 같은 길이의 문자열에서 같은 위치에 있지만, 서로 다른 문자의 개수를 구하는 프로그램에서 빈 칸을 채우는 문제

3) 정답

```
② public String func_a(String str, int len){
    String padZero = "";
    int padSize = len - str.length()
    for(int i = 0; i < padSize; i++)
        padZero += "0";
    return padZero + str;
}
```

```
public int solution(String binaryA, String binaryB) {
① int maxLength = Math.max(binaryA.length(), binaryB.length());
② binaryA = func_a(binaryA, maxLength);
② binaryB = func_a(binaryB, maxLength);

③ int hammingDistance = 0;
  for(int i = 0; i < maxLength; i++)
      if(@@@)        binaryA.charAt(i)!=binaryB.charAt(i)
          hammingDistance += 1;
  return hammingDistance;
}
```

① solution( ) 메소드에서 두 문자열 간의 차이를 구하여 return

- 두 개의 문자열을 저장한 binaryA 와 binaryB 중 길이가 더 긴 문자열 길이를 maxLength 에 저장
- func_a( ) 함수를 이용하여 두 문자열의 차이만큼 '0' 을 붙여서 binaryA 와 binaryB 에 저장되어 있는 문자열의 길이를 동일하게 맞춤

② func_a( ) 함수에서는 두 문자열의 길이 차이만큼 문자열 앞에 '0' 을 붙여 길이를 맞춘 문자열을 return

- func_a( ) 함수의 인수로 받은 len - str.length() 값 만큼 '0' 을 문자열에 붙여 넣도록 padSize 의 비어 있는 수식을 완성

③ solution( ) 메소드에서 두 문자열 간의 차이를 구하여 return

- solution() 메소드의 if 문에서 문자열의 각 문자를 순서대로 하나씩 가져와 그 문자들이 서로 같지 않으면 hammingDistance 변수 값을 1 만큼 증가시키도록 if 문의 조건식을 완성

4) 다른 코드 제안

문자열로 들어온 값을 정수로 바꾼 후 xor(^) 비트 연산

```
String binaryA = "10010";
String binaryB = "110";

int A = Integer.parseInt(binaryA);
int B = Integer.parseInt(binaryB);

String D= Integer.toString(A^B);
int hammingDistance=0;
System.out.println(D);

for(int i=0; i< D.length();i++) {
    System.out.println(D.charAt(i));
    if (D.charAt(i)=='1')  hammingDistance+=1;
}

System.out.println(hammingDistance);
```

※ 문제 출처 : https://www.ybmit.com/cos_pro/cos_pro_r_test.jsp

10. 문제 10

1) 문제 코드

```java
/*=======================================
  1차 10번  1차 1급 10_initial_code.java
 =======================================*/
class Solution {
    class Pair{
        public int firstNum;
        public int secondNum;
    }

    public int func_a(int numA, int numB, char exp){
        if (exp == '+')
            return numA + numB;
        else if (exp == '-')
            return numA - numB;
        else
            return numA * numB;
    }

    public int func_b(String exp){
        for(int i = 0; i < exp.length(); i++){
            char e = exp.charAt(i);
            if(e == '+' || e == '-' || e == '*')
                return i;
        }
        return -1;
    }

    public Pair func_c(String exp, int idx){
        Pair ret = new Pair();
        ret.firstNum = Integer.parseInt(exp.substring(0, idx));
        ret.secondNum = Integer.parseInt(exp.substring(idx + 1));
        return ret;
    }

    public int solution(String expression) {
        int expIndex = func_@@@(@@@);
        Pair numbers = func_@@@(@@@);
        int result = func_@@@(@@@)
        return result;
    }
```

```
    // The following is main method to output testcase.
    public static void main(String[] args) {
        Solution sol = new Solution();
        String expression = "123+12";
        int ret = sol.solution(expression);

        // Press Run button to receive output.
        System.out.println("Solution: return value of the method is " + ret + " .");
    }
}
```

2) 문제 개요

- 문제 코드 안에 작성된 함수의 알고리즘을 파악하여 알맞은 함수를 호출하도록 코드를 완성하는 문제
- 문자열로 전달한 수식을 계산하여 결과를 받아 오는 프로그램으로, 프로그램 안에서 정의된 func_a( ), func_b( ), func_c( ) 함수들의 역할을 파악한 후 적절한 인수를 사용하여 필요한 함수를 호출하도록 빈 칸에 적어야 함

3) 문법 확인

① charAt() : 문자열에서 원하는 위치의 문자 1 개를 리턴

```
String str="abcd";
char c = str.charAt(2);
```
c

② indexOf() : 문자열에서 특정 문자열의 시작 위치 리턴

```
String str2="abcd";
int i = str2.indexOf("cd");
i = str2.indexOf("c");
```
2

③ substring() : 부분 문자열 추출

- substring(시작 인덱스) : 시작 인덱스부터 끝까지 문자열 반환
- substring(시작 인덱스, 종료 인덱스) : 시작 인덱스부터 종료 인덱스 -1 까지 잘라서 반환

```
String exp="123+12";

System.out.println(exp.substring(2));
System.out.println(exp.substring(1,4));
```
3+12
23+

4) 정답

```
2
class Pair{
    public int firstNum;
    public int secondNum;
}

3
public int func_a(int numA, int numB, char exp){
    if (exp == '+')
        return numA + numB;
    else if (exp == '-')
        return numA - numB;
    else
        return numA * numB;
}

1
public int func_b(String exp){
    for(int i = 0; i < exp.length(); i++){
        char e = exp.charAt(i);
        if(e == '+' || e == '-' || e == '*')
            return i;
    }
    return -1;
}

2
public Pair func_c(String exp, int idx){
    Pair ret = new Pair();
    ret.firstNum = Integer.parseInt(exp.substring(0, idx));
    ret.secondNum = Integer.parseInt(exp.substring(idx + 1));
    return ret;
}
```

```
public int solution(String expression) {
    1 int expIndex = func_ b(expression)
    2 Pair numbers = func_ c(expression, expIndex)
    3 int result = func_ a(numbers.firstNum,
                           numbers.secondNum,
                           expression.charAt(expIndex))
      return result;
}
```

① func_b( ) 함수는 문자열에서 연산자가 위치한 인덱스를 찾아 return
- for 문을 이용하여 exp 에 저장되어 있는 문자열에서 문자를 e 에 받아 옴
- e 에 받아 온 문자가 '+', '-', '*' 에 해당되면 인덱스를 return

② func_c( ) 함수는 문자열에서 연산자의 인덱스를 기준으로 앞과 뒤의 문자열을 추출하고, 추출한 수를 정수로 변환하여 return
- 문자열이 저장된 exp 에서 첫 번째 문자부터 idx 앞에 있는 문자까지 substring( )을 사용하여 firstNum 에 저장
- 문자열이 저장된 exp 에서 idx 가 가리키는 문자의 다음 문자부터 마지막 문자까지 substring( ) 을 사용하여 secondNum 에 저장
- firstNum 과 secondNum 을 Integer.parseInt()를 사용하여 Integer 로 변환하여 클래스 Pair 의 변수 ret 에 저장
- 인스턴스 변수 ret 를 return

③ func_a( ) 함수는 전달받은 연산자 문자에 해당하는 계산을 실행하여 return

- 세 번째 매개변수 exp 의 값이 '+' 이면 numA + numB 를 실행
- 세 번째 매개변수 exp 의 값이 '-' 이면 numA - numB 를 실행
- 세 번째 매개변수 exp 의 값이 '*' 이면 numA * numB 를 실행

④. solution( ) 메소드에서 func_b( )를 이용하여 연산자의 인덱스를 찾은 후 func_c( )를 이용하여 연산자를 기준으로 두 수를 찾고, func_a( )를 이용하여 계산을 실행하도록 순서대로 알맞은 인수를 전달하며 함수를 호출하는 구문을 완성