Professional Coding Specialist

COS Pro JAVA 1 급

15 강-18 강. 모의고사 3 차

1. 모의고사 3 차(1-10 번)

과정 소개

COS Pro JAVA 1 급 모의고사 3 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 JAVA 1 급 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.

1. 문제 1

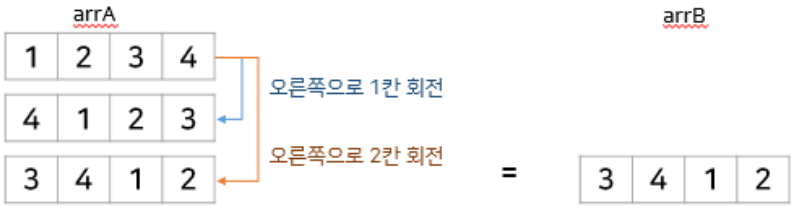
```
int[] func_a(int[] arr) {
    int length = arr.length;
    int[] ret = new int[length*2];
    for(int i = 0; i < length; i++)
        ret[i + length] = ret[i] = arr[i];
    return ret;
}

boolean func_b(int[] first, int[] second){
    int[] counter = new int[1001];
    for(int i = 0; i < first.length; i++){
        counter[first[i]]++;
        counter[second[i]]--;
    }
    for(int i = 0; i < 1001; i++)
        if(counter[i] != 0)
            return false;
    return true;
}

boolean func_c(int[] first, int[] second){
int length = second.length;
for(int i = 0; i < length; i++){
    int j;
    for(j = 0; j < length; j++)
        if(first[i + j] != second[j])
            break;
    if(j == length)
        return true;
}
return false;
}
```

```
public boolean solution(int[] arrA, int[] arrB) {
    if(arrA.length != arrB.length)
        return false;
    if(func_@@@(@@@)) {
        int[] arrAtemp = func_@@@(@@@);
        if(func_@@@(@@@))
            return true;
    }
    return false;
}

// 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다.
public static void main(String[] args) {
    Solution sol = new Solution();
    int[] arrA1 = {1, 2, 3, 4};
    int[] arrB1 = {3, 4, 1, 2};
    boolean ret1 = sol.solution(arrA1, arrB1);

    // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
    System.out.println("solution 메소드의 반환 값은 " + ret1 + " 입니다.");

    int[] arrA2 = {1, 2, 3, 4};
    int[] arrB2 = {1, 4, 2, 3};
    boolean ret2 = sol.solution(arrA2, arrB2);

    // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
    System.out.println("solution 메소드의 반환 값은 " + ret2 + " 입니다.");
}
```

1) **문제 코드**

2) 문제 개요

- 문제 코드 안에 작성된 함수를 파악한 후 제시된 과제를 해결하기 위한 알고리즘대로 알맞은 함수를 호출하도록 코드를 완성하는 문제
- 아래 그림과 같이 arrA 배열을 회전하여 arrB 배열을 만들 수 있는지 확인하는 프로그램



3) 정답

```
int[] func_a(int[] arr) {
    int length = arr.length;
    int[] ret = new int[length*2];
    for(int i = 0; i < length; i++)
        ret[i + length] = ret[i] = arr[i];
    return ret;
}

boolean func_b(int[] first, int[] second){
    int[] counter = new int[1001];
    for(int i = 0; i < first.length; i++){
        counter[first[i]]++;
        counter[second[i]]--;
    }
    for(int i = 0; i < 1001; i++)
        if(counter[i] != 0)
            return false;
    return true;
}

boolean func_c(int[] first, int[] second){
int length = second.length;
for(int i = 0; i < length; i++){
    int j;
    for(j = 0; j < length; j++)
        if(first[i + j] != second[j])
            break;
    if(j == length)
        return true;
}
return false;
}
```

```
public boolean solution(int[] arrA, int[] arrB) {
1   if(arrA.length != arrB.length)
        return false;
2   if(func_@@@(@@@)) {        b(arrA, arrB)
3       int[] arrAtemp = func_  a(arrA)
4       if(func_@@@(@@@))      c(arrAtemp, arrB)
            return true;
    }
    return false;
}
```

- func_a( ) 함수는 매개변수 arr 로 받은 배열을 두 번 이어 붙여서 길이가 2 배인 배열로 생성하여 return
- func_b( ) 함수 안에서 1000 개의 0 을 항목으로 갖는 배열 counter 를 생성
  - first 에서 가져온 항목과 동일한 인덱스를 갖는 counter 의 항목값은 1 만큼 증가
  - second 에서 가져온 항목과 동일한 인덱스를 갖는 counter 의 항목값은 1 만큼 감소
    : first 의 항목과 second 의 항목이 동일하면 항목과 같은 값의 인덱스를 갖는 counter 의 항목은 값을 0 으로 유지하게 됨
  - counter 배열에 있는 항목값이 0 이 아닌 값이 존재하면 false 를 return
- func_c( ) 함수는 두 개의 매개변수 first, second 를 받아서 second 가 first 의 일부분인지 확인하여 그 결과를 return
  - for 문을 이용하여 first 의 0 번 항목부터 second 길이만큼 자른 것이 second 와 같은 지를 확인하여 같으면 true 를 return
- solution( ) 메소드
  ①. 두 배열의 길이가 다르면 false 를 리턴
  ②. func_b( ) 를 이용하여 arrA 와 arrB 의 항목이 동일한 지 확인
  ③. func_a( ) 를 이용하여 arrA 를 2 번 이어 붙인 arrAtemp 를 생성
  ④. func_c( ) 를 이용하여 arrB 가 arrAtemp 의 일부분인지 확인하도록 함수를 호출

4) 다른 코딩 제안

- **같은 배열 두 번 복사**

```
int[] func_a(int[] arr) {

    int length = arr.length;
    int[] ret = new int[length*2];
    System.arraycopy(arr, 0, ret, 0, length);
    System.arraycopy(arr, 0, ret, length, length);
    return ret;
}
```

- **두 배열의 구성 성분 확인하기**

```
boolean func_b(int[] first, int[] second){
    int[] tempfirst, tempsecond;
    tempfirst =Arrays.copyOf(first,first.length);
    tempsecond =Arrays.copyOf(second,second.length);
    Arrays.sort(tempfirst);
    Arrays.sort(tempsecond);

    if (Arrays.equals(tempfirst,tempsecond))
        return true;
    return false;
}
```

- **두 배로 된 배열(A)과 배열(B)가 같은 부분이 있는지 판단**

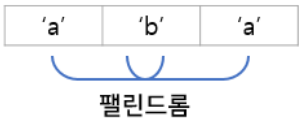
```
boolean func_c(int[] first, int[] second){
    int length = second.length;
    int[] temp;

    for(int i = 0; i < length; i++){
        temp =Arrays.copyOfRange(first, i,length+i);
        if (Arrays.equals(temp,second))
            return true;
    }
    return false;
}
```

2. 문제 2

1) 문제 코드

※ 팰린드롬(palindrome) : 문자열에 있는 앞뒤 문자의 위치를 바꾸어도 똑같은 문자열

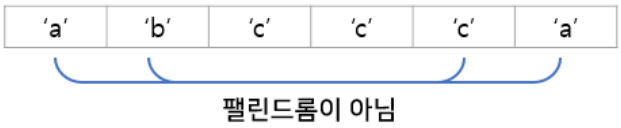
예시 1)



- 원본 문자열 : 'aba'
- 앞뒤 바꾼 문자열 : 'aba'

→ 원본과 앞뒤를 바꾼 문자열이 동일하기 때문에 팰린드롬

예시 2)



- 원본 문자열 : 'abccca'
- 앞뒤 바꾼 문자열 : 'acccba'

→ 원본과 앞뒤를 바꾼 문자열이 다르기 때문에 팰린드롬이 아님

```java
import java.util.*;

public class Solution {
    public boolean func_a(ArrayList<String> list, String s) {
        for (int i = 0; i < list.size(); i++)
            if (s.equals(list.get(i)))
                return true;
        return false;
    }

    public boolean func_b(String s) {
        int length = s.length();
        for (int i = 0; i < length / 2; i++)
            if (s.charAt(i) != s.charAt(length - i - 1))
                return false;
        return true;
    }

    public String func_c(ArrayList<String> palindromes, int k) {
        Collections.sort(palindromes);
        if (palindromes.size() < k)
            return "NULL";
        else
            return palindromes.get(k-1);
    }
```

```java
  public String solution(String s, int k) {
      ArrayList<String> palindromes = new ArrayList<String>();
      int length = s.length();
      for (int startIdx = 0; startIdx < length; startIdx++) {
          for (int cnt = 1; cnt < length - startIdx + 1; cnt++) {
              String subStr = s.substring(startIdx, startIdx + cnt);
              if (func_@@@(@@@)) {
                  if (func_@@@(@@@) == false)
                      palindromes.add(subStr);
              }
          }
      }
      String answer = func_@@@(@@@);
      return answer;
  }

  // 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다.
  public static void main(String[] args) {
      Solution sol = new Solution();
      String s1 = new String("abcba");
      int k1 = 4;
      String ret1 = sol.solution(s1, k1);

      // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
      System.out.println("solution 메소드의 반환 값은 " + ret1 + " 입니다.");

      String s2 = new String("ccddcc");
      int k2 = 7;
      String ret2 = sol.solution(s2, k2);

      // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
      System.out.println("solution 메소드의 반환 값은 " + ret2 + " 입니다.");
  }
```

2) 문제 개요

- 문제 코드 안에 작성된 함수를 파악한 후 제시된 과제를 해결하기 위한 알고리즘대로 알맞은 함수를 호출하도록 코드를 완성하는 문제
- 인수로 받은 문자열의 부분 문자열에서 '팰린드롬' 문자열들을 찾아서 배열에 저장하고 k 번째로 큰 '팰린드롬' 문자열을 리턴하는 프로그램

```
import java.util.*;

public class Solution {
    public boolean func_a(ArrayList<String> list, String s) {
        for (int i = 0; i < list.size(); i++)
            if (s.equals(list.get(i)))
                return true;
        return false;
    }

    public boolean func_b(String s) {
        int length = s.length();
        for (int i = 0; i < length / 2; i++)
            if (s.charAt(i) != s.charAt(length - i - 1))
                return false;
        return true;
    }

    public String func_c(ArrayList<String> palindromes, int k) {
        Collections.sort(palindromes);
        if (palindromes.size() < k)
            return "NULL";
        else
            return palindromes.get(k-1);
    }
```

```
public String solution(String s, int k) {
1   ArrayList<String> palindromes = new ArrayList<String>();
    int length = s.length();
2   for (int startIdx = 0; startIdx < length; startIdx++) {
        for (int cnt = 1; cnt < length - startIdx + 1; cnt++) {
            String subStr = s.substring(startIdx, startIdx + cnt);
3           if (func_@@@(@@@)) {          b(subStr)
                if (func_@@@(@@@) == false)   a(palindromes, subStr)
                    palindromes.add(subStr);
            }
        }
    }
4   String answer = func_@@@(@@@);   c(palindromes, k)
    return answer;
}
```

3) 정답

- func_a( ) 함수는 s 가 list 안에 존재하는지 여부를 return
- func_b( ) 는 s 가 팰린드롬 문자열인지 확인하여 return

- for 문을 이용하여 문자열 s 의 절반 길이만큼 반복하며 문자를 비교
- s 의 첫 번째 문자와 s 의 마지막 문자를 비교하여 만일 다르면 false 를 return 하고, 그렇지 않으면 그 다음 반복 구문으로 넘어가서 s 의 두 번째 문자와 s 의 마지막에서 두 번째 문자를 비교하는 작업을 s 의 절반 항목까지 반복

◆ func_c( ) 함수의 매개변수 palindromes 에 저장된 배열을 sorted( ) 함수를 이용하여 정렬

- func_c( ) 함수의 매개변수 k 보다 palindromes 에 저장된 배열 항목 개수가 적으면 "NULL" 을 return 하고, 그렇지 않으면 k 번째 항목을 return

① solution( ) 메소드에서 팰린드롬 문자열을 저장할 palindromes 를 선언

② 중첩 for 문을 이용하여

- 인수로 받은 s 의 인덱스 0 번째 문자부터 시작하여 문자열 길이가 1,2,3,..인 부분 문자열을 생성하여 sub_s 에 저장
- func_b( ) 를 이용하여 subStr 가 '팰린드롬' 인지 확인
- func_a( ) 를 이용하여 palindromes 배열에 subStr 가 없으면 palindromes 의 항목으로 추가

③ func_c( ) 를 이용하여 palindromes 배열에서 k 번째 문자열을 가져와 answer 에 저장

3. 문제 3

1) 문제 코드

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

// 다음과 같이 import를 사용할 수 있습니다.
import java.util.*;

class Solution {
    public int solution(String[] bishops) {
        // 여기에 코드를 작성해주세요.
        int answer = 0;
        return answer;
    }

    // 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다.
    public static void main(String[] args) {
        Solution sol = new Solution();
        String[] bishops1 = {new String("D5")};
        int ret1 = sol.solution(bishops1);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret1 + " 입니다.");

        String[] bishops2 = {new String("D5"), new String("E8"), new String("G2")};
        int ret2 = sol.solution(bishops2);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret2 + " 입니다.");
    }
}
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제
- 배열로 전달된 비숍에게 잡히지 않고 기물을 놓을 수 있는 체스판의 위치의 수를 집계
- 비숍은 자신의 현재 위치에서 대각선 방향에 있는 모든 기물들을 잡을 수 있음

3) 정답-제안 1

```
/*=========================================
   3차 1급 3_solution_code.java
   ybmit.com 사이트의 샘플 파일에는 정답이 제시되어 있지 않습니다.
   아래와 같은 정답을 제안해 드리며, 더 좋은 알고리즘으로 구현해 보시기 바랍니다.
 =========================================*/
```

```
public int solution(String[] bishops) {
    int answer = 0;
    int x, y, dx, y_up,y_dn;
    int[][] chess_map = new int[8][8];
①   for (int i =0; i<chess_map.length;i++)          // 모두 1로 초기화
        Arrays.fill(chess_map[i], 1);

    for (int i =0; i<bishops.length;i++) {
        String temp_x= bishops[i].substring(0,1);  //D5에서 D 문자열 추출
        x=(int)(temp_x.charAt(0))-(int)('A') ;     //D 문자열을 char로 변환 후
                                                   // ascii 코드값 추출하여 A와의 차이 구하기
        String temp_y= bishops[i].substring(1);    //D5에서 5 문자열 추출
        y=Integer.parseInt(temp_y)-1 ;             //5 문자열을 숫자로 변환
②
        chess_map[y][x]=0;
        dx=x-1;
        y_up=y;
        y_dn=y;
③       while (dx>=0) {
            y_up+=1;
            y_dn-=1;
            if (y_up <8)
                chess_map[y_up][dx]=0;
            if (y_dn>=0)
④               chess_map[y_dn][dx]=0;
            dx-=1;
        }
        dx=x+1;
        y_up=y;
        y_dn=y;
⑤       while(dx<8) {
            y_up+=1;
            y_dn-=1;
            if (y_up <8)
                chess_map[y_up][dx]=0;
            if (y_dn>=0)
⑥               chess_map[y_dn][dx]=0;
            dx+=1;
        }
        answer=0;
        for(int k = 0; k < chess_map.length;k++) {
            for(int h = 0;h < chess_map[k].length;h++) {
                if (chess_map[k][h]==1)
⑦                   answer += 1;
            }
        }
    }
    return answer;
```

※ 문

① 체스판 크기에 맞춰 각 항목값이 1 인 8 X 8 크기의 2 차원 배열 chess_map 을 생성.

- 항목값 = 1 : 비숍에 잡히지 않고 기물을 배치할 수 있는 위치를 의미.

chess_map = [[1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1]]

② for 문을 이용하여 bishops 배열에 있는 비숍의 위치값을 가져와 가로 위치에 해당하는 인덱스와 세로 위치에 해당하는 인덱스 x, y 값을 구하고, chess_map 에서 현 비숍의 위치에 해당하는 인덱스의 항목값을 0 으로 변경(항목값 = 0 은 비숍에 잡히는 위치를 의미)

❖ for 문 종료 후 chess_map =

```
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1]
```

③ 현재 비숍의 왼쪽 방향으로 비숍에 의해 잡히는 위치를 찾기 위해 dx 변수에 x-1 을 할당하고, y_up 변수와 y_dn 변수에는 현 비숍의 y 값을 할당

④ dx 값이 0 이상인 동안(체스판 왼쪽 끝을 벗어나지 않는 동안)

- y_up 은 1 만큼 증가시키고, y_dn 은 1 만큼 감소시킨 후 두 변수의 값이 체스판의 영역을 벗어나지 않으면 chess_map[y_up][dx] 항목과 chess_map[y_dn][dx] 항목값을 0 으로 변경
- dx 의 값을 1 만큼 감소시킴

❖ while 문 종료 후 chess_map =

```
[1, 1, 1, 1, 1, 1, 1, 1]
[0, 1, 1, 1, 1, 1, 1, 1]
[1, 0, 1, 1, 1, 1, 1, 1]
[1, 1, 0, 1, 1, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1, 1]
[1, 1, 0, 1, 1, 1, 1, 1]
[1, 0, 1, 1, 1, 1, 1, 1]
[0, 1, 1, 1, 1, 1, 1, 1]
```

⑤ 현재 비숍의 오른쪽 방향으로 비숍에 의해 잡히는 위치를 찾기 위해 dx 변수에 x+1 을 할당하고, y_up 변수와 y_dn 변수에는 현 비숍의 y 값을 할당

⑥ dx 값이 8 미만인 동안(체스판 오른쪽 끝을 벗어나지 않는 동안)

- y_up 은 1 만큼 증가시키고, y_dn 은 1 만큼 감소시킨 후 두 변수의 값이 체스판의 영역을 벗어나지 않으면 chess_map[y_up][dx] 항목과 chess_map[y_dn][dx] 항목값을 0 으로 변경
- dx 의 값을 1 만큼 증가시킴

❖ while 문 종료 후 chess_map =

```
[1, 1, 1, 1, 1, 1, 1, 0]
[0, 1, 1, 1, 1, 1, 0, 1]
[1, 0, 1, 1, 1, 0, 1, 1]
[1, 1, 0, 1, 0, 1, 1, 1]
[1, 1, 1, 0, 1, 1, 1, 1]
[1, 1, 0, 1, 0, 1, 1, 1]
[1, 0, 1, 1, 1, 0, 1, 1]
[0, 1, 1, 1, 1, 1, 0, 1]
```

⑦. for 문을 이용하여 2 차원 배열인 chess_map 배열에서 항목값이 1 인 항목의 개수를 answer 변수에 누적 집계

4) 정답 –제안 2

```
/*==========================================
  3차 1급 T_Solution3.java
 ybmit.com 사이트의 샘플 파일에는 정답이 제시되어 있지 않습니다.
 아래와 같은 정답을 제안해 드리며, 더 좋은 알고리즘으로 구현해 보시기 바랍니다.
```

```
==========================================*/
public int solution(String[] bishops) {
    int answer = 0;
    int[] dx= {1,1,-1,-1};
    int[] dy= {1,-1,-1,1};

    int[][] chess_map = new int[8][8];

    for (int i =0; i<bishops.length;i++) {
        for (int j=0 ; j <4; j++) {
            String temp_x= bishops[i].substring(0,1);  //D5에서 D 문자열 추출
            int x=(int)(temp_x.charAt(0))-(int)('A') ; //D 문자열을 char로 변환 후
                                                       // ascii 코드값 추출하여 A와의 차이 구하기
            String temp_y= bishops[i].substring(1);    //D5에서 5 문자열 추출
            int y=Integer.parseInt(temp_y)-1 ;         //5 문자열을 숫자로 변환

            while(0<=x && x<8 && 0<=y && y<8) {
                chess_map[y][x] = 1;
                x = x + dx[j];
                y = y + dy[j];
            }
        }
    }

    for(int k = 0; k < chess_map.length;k++) {
        System.out.println(Arrays.toString(chess_map[k]));
        for(int h = 0;h < chess_map[k].length;h++) {
            if (chess_map[k][h]==0)
                answer += 1;
        }
    }
    return answer;
}
```

- 체스판 크기(0 부터 7)에 맞춰 8 X 8 크기의 2 차원 배열 chess_map 의 모든 값을 0 으로 초기화하여 생성
- for 문을 이용하여 bishops 배열에 있는 비숍에 모든 변화 규칙을 적용
  - 비숍의 위치를 가져와 아스키 정수 연산을 이용하여 정수로 변환하여 x, y 에 저장
  - 비숍의 위치가 체스판의 영역을 벗어나지 않는 동안 chess_map 의 해당 위치를 1 로 표시하고, x, y 에 변환 규칙을 적용하여 비숍의 위치를 변경
- for 문을 이용하여 2 차원 배열 chess_map 에서 항목값이 0 인 항목의 개수를 answer 변수에 누적 집계

4. 문제 4

1) 문제 코드

```
/*=======================================
    3차 4번   3차 1급 4_initial_code.java
 =======================================*/

// 다음과 같이 import를 사용할 수 있습니다.
import java.util.*;

class Solution {
    public int solution(String s1, String s2) {
        // 여기에 코드를 작성해주세요.
        int answer = 0;
        return answer;
    }

    // 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다.
    public static void main(String[] args) {
        Solution sol = new Solution();
        String s1 = new String("ababc");
        String s2 = new String("abcdab");
        int ret = sol.solution(s1, s2);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret + " 입니다.");
    }
}
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제
- 두 개의 문자열의 앞뒤 부분에서 공통된 문자 패턴을 찾고, 공통된 부분은 아래 그림과 같이 한 번만 보이도록 두 문자열을 붙여서 그 문자열의 길이를 구하는 문제

3) 정답-제안

```
/*=========================================
    3차 1급 4_solution_code.java
    ybmit.com 사이트의 샘플 파일에는 정답이 제시되어 있지 않습니다.
    아래와 같은 정답을 제안해 드리며, 더 좋은 알고리즘으로 구현해 보시기 바랍니다.
```

```
① int overlap(String s1, String s2) {
    int len = Math.min(s1.length(), s2.length());
    int over_len = 0;

  ② for ( int i =1 ; i <= len ; i++) {
        String f = s1.substring(s1.length() - i);
        String s = s2.substring(0,i);

        if (f.equals(s))
            over_len = i;
    }
  ③ return s1.length() + s2.length() - over_len;
}

public int solution(String s1, String s2) {
    // 여기에 코드를 작성해주세요.
    int answer = 0;
  ④ answer = Math.min(overlap(s1,s2), overlap(s2,s1));
    return answer;
}
==========================================*/
```

① overlap( ) 함수를 작성. 두 문자열의 앞뒤 부분에서 공통 패턴이 있는지 확인하고, 공통 패턴이 존재하면 두 문자열 길이의 합에서 공통 패턴이 있는 문자열의 길이를 뺀 길이를 리턴 하는 함수

② for 문을 이용하여 비교 길이를 늘려가면서 문자열 s2 의 앞 부분과 문자열 s1 의 뒷부분을 substring 을 활용하여 구하고, 구한 부분 문자열이 일치할 경우 비교 길이를 over_len 에 저장

③ 두 문자열의 길이 합에서 공통된 패턴이 있는 문자열의 길이를 뺀 값을 리턴

④ solution( ) 메소드에서 overlap( )함수를 두 번 호출. 문자열 s1 의 뒷부분과 문자열 s2 앞 부분의 공통 패턴의 길이를 뺀 두 문자열 길이와 문자열 s2 의 뒷부분과 문자열 s1 앞 부분의 공통 패턴의 길이를 뺀 두 문자열의 길이를 구함. 구한 두 길이 중 더 작은 값을 리턴.

※ **현재는 전체 코딩하는 문제의 경우 코드를 원하는 곳에 추가 가능하므로 overlap() 이라는 함수를 정의할 수 있습니다. 그런데, 테스트 환경이 바뀌어 함수 추가가 어려울 경우 overlap() 함수를 카피하여 2 번 적으면서 s1 과 s2 순서를 바꾸면 됩니다.**

5. 문제 5

1) 문제 코드

```
/*=======================================
    3차 5번   3차 1급 5_initial_code.java
 =======================================*/

// 다음과 같이 import를 사용할 수 있습니다.

import java.util.*;

class Solution {
    public String solution(String phrases, int second) {
        // 여기에 코드를 작성해주세요.
        String answer = "";
        return answer;
    }

    // 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다.
    public static void main(String[] args) {
        Solution sol = new Solution();
        String phrases = new String("happy-birthday");
        int second = 3;
        String ret = sol.solution(phrases, second);
        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret + " 입니다.");
    }
}
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 solution( )에 프로그램 코드를 작성하는 문제
- 14 자까지 표시하는 화면에 '_____' 으로 시작하여 'happy-birthday'의 문자를 1 글자씩 왼쪽으로 이동하며 표시하고, 모든 문자를 표시한 뒤에는 다시 '_____' 부터 시작하여 한 글자씩 나타내도록 프로그램을 구성해야 함

3) 정답

```
public String solution(String phrases, int second) {
    String answer = "";
1   String display = "";
    display = "______________" + phrases;
2   for(int i = 0; i < second; ++i) {
        display = display + Character.toString(display.charAt(0));
        display = display.substring(1);
    }
3   answer = display.substring(0,14);
    return answer;
}
```

①. 시작할 때 보여줘야 하는 문자열인 '_' 문자 14 개와 시간이 경과하면서 한 개씩 보여줘야 하는 문자열을 담은 phrases 를 붙인 것을 display 변수의 값으로 초기화

②. 매개변수 second 의 값만큼 반복하며 현재 display 변수가 갖는 두 번째 문자부터 마지막 문자까지 현재 display 변수가 갖는 첫 번째 문자를 붙인 것을 다시 display 에 할당

→display 변수가 갖는 문자열의 가장 왼쪽 문자는 오른쪽 끝으로 이동하고 나머지 문자들은 왼쪽으로 한 자리씩 이동

③. 마지막 결과로 산출된 display 문자열에서 14 개의 문자만 잘라서 return

※ 28 개의 문자가 회전하며 나타나기 때문에 for 문의 반복횟수를 second → secode%28 로 바꿔 코드 가능

```
String display = "______________" +phrases+"______________";

answer= display.substring(second%28,14+second%28);
```

6. 문제 6

1) 문제 코드

```java
/*=======================================
    3차 6번   3차 1급 6_initial_code.java
 =======================================*/
import java.util.ArrayList;

class Solution {
    public int solution(int n) {
        int answer = 0;
        ArrayList<Integer> primes = new ArrayList<Integer>();
        primes.add(2);
        for (int i = 3; i <= n; i += 2) {
            boolean isPrime = true;
            for (int j = 2; j < i; j++)
                if (i % j == 0){
                    isPrime = false;
                    break;
                }
            if (@@@)
                primes.add(i);
        }

        int primeLen = primes.size();
        for (int i = 0; i < primeLen - 2; i++)
            for (int j = i + 1; j < primeLen - 1; j++)
                for (int k = j + 1; k < primeLen; k++)
                    if (@@@)
                        answer++;
        return answer;
    }

    // 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다.
    public static void main(String[] args) {
        Solution sol = new Solution();
        int n1 = 33;
        int ret1 = sol.solution(n1);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret1 + " 입니다.");

        int n2 = 9;
        int ret2 = sol.solution(n2);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret2 + " 입니다.");
    }
}
```

2) 문제 개요

- 제시된 과제를 해결하기 위해 작성한 solution( ) 메소드에서 비어 있는 구문을 채워 완성하는 문제
- n 보다 작은 소수 3 개를 더하여 인수로 받은 n 을 만들 수 있는 개수를 찾는 프로그램에서 비어 있는 부분을 채워야 함

3) 정답

```java
import java.util.Scanner;

class Solution {
    public int solution(int n) {
        int answer = 0;
        int i, j, k;
        ArrayList<Integer> prime = new ArrayList<Integer>();
        prime.add(2);
        for (i = 3; i <= n; i += 2) {
            for (j = 2; j < i; j++) {
                if (i % j == 0) {
                    break;
                }
            }
            if (j == i) {
                prime.add(i);
            }
        }
        int prime_n = prime.size();
        for (i = 0; i < prime_n - 2; i++) {
            for (j = i + 1; j < prime_n - 1; j++) {
                for (k = j + 1; k < prime_n; k++) {
                    if (prime.get(i) + prime.get(j) + prime.get(k) == n) {
                        answer += 1;
                    }
                }
            }
        }
        return answer;
    }
}
```

① 1

② 2

③ 3

① ArrayList 로 prime 변수를 <Integer> 타입으로 선언
prime 변수에 소수인 2 를 항목으로 추가

② for 문을 이용하여 3 이상 n 이하의 홀수를 i 로 가져옴

- 중첩 for 문을 이용하여 2 부터 i 보다 작은 정수를 j 로 가져옴
  - i 를 j 로 나눈 나머지가 0 이면 j 는 i 의 약수이므로 i 는 소수가 아니기 때문에 break 명령을 이용하여 중첩 for 문 종료
- i 와 j 가 값이 같으면 위의 break 문을 만나지 않았기 때문에 소수이므로 prime 변수에 1 값을 추가

③ 중첩 for 문을 이용하여 prime 배열에 있는 소수 3 개를 중복되지 않게 가져옴

- 첫 번째 for 문으로 prime 의 항목을 0 번째부터 가져와 i 에 저장
- 두 번째 for 문으로 prime 의 항목을 i 다음부터 가져와 j 에 저장
- 세 번째 for 문으로 prime 의 항목을 j 다음부터 가져와 k 에 저장
- i + j + k 의 결과가 n 과 같으면 answer 값을 1 만큼 증가하여 개수 계산

7. 문제 7

1) 문제 코드

```java
/*======================================
    3차 7번   3차 1급 7_initial_code.java
 ======================================*/

import java.util.*;

class Solution {
    public int[] solution(int k) {
        int[] kaprekarArr = new int[k];
        int count = 0;
        for (int i = 1; i <= k; i++) {
            long squareNum = i * i;
            long divisor = 1;
            while (squareNum % divisor != 0) {
                long front = squareNum / divisor;
                long back = squareNum % divisor;
                divisor *= 10;
                if (back != 0 && front != 0)
                    if (front + back == i) {
                        kaprekarArr[count] = i;
                        count++;
                    }
            }
        }
        int[] answer = new int[count];
        for (int i = 0; i < count; i++)
            answer[i] = kaprekarArr[i];
        return answer;
    }

    // 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다. main 메소드는 잘못된 부분이 없으니, s
    public static void main(String[] args) {
        Solution sol = new Solution();
        int k = 500;
        int[] ret = sol.solution(k);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + Arrays.toString(ret) + " 입니다.");
    }
}
```

2) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제
- 어떤 수를 제곱한 결과를 두 부분으로 나눈 후 나눠진 두 수를 더한 것이 원래의 수와 같은 것을 '카프리카 수'라고 하는데, 인수로 받은 k 보다 작거나 같은 '카프리카 수'를 찾아오는 프로그램에서 잘못된 부분을 수정해야 함
- 제곱한 수를 두 부분으로 나누는 방법에 대한 이해가 필요함

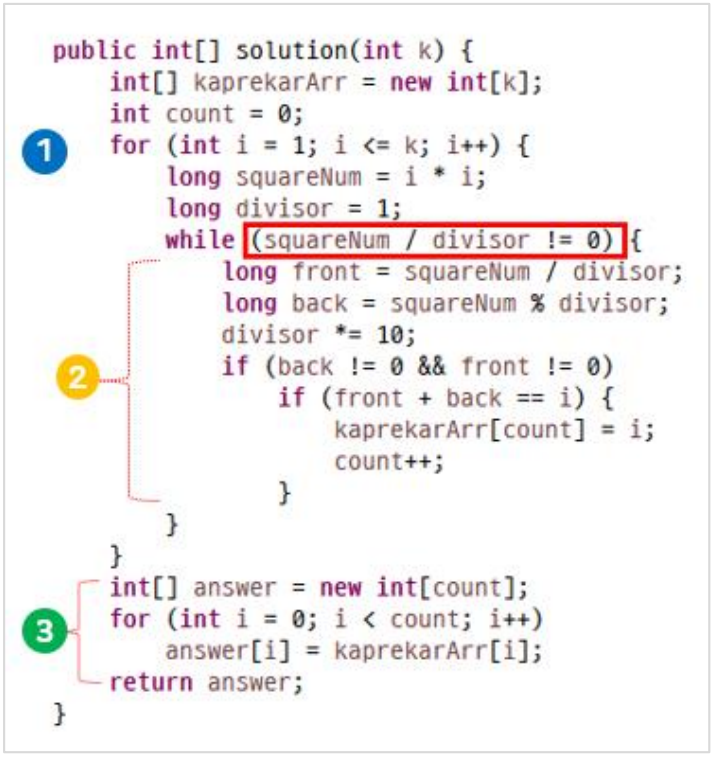
  예) 3125 를 두 부분으로 나누는 방법

  ① 312 와 5 로 나누기 : 3125//10 = 312 ; 3125%10 = 5

  ② 31 와 25 로 나누기 : 3125//100 = 31 ; 3125%100 = 25

  ③ 3 와 125 로 나누기 : 3125//1000 = 3 ; 3125%1000 = 125

3) 정답



```
public int[] solution(int k) {
    int[] kaprekarArr = new int[k];
    int count = 0;
1   for (int i = 1; i <= k; i++) {
        long squareNum = i * i;
        long divisor = 1;
        while (squareNum / divisor != 0) {
            long front = squareNum / divisor;
            long back = squareNum % divisor;
            divisor *= 10;
2           if (back != 0 && front != 0)
                if (front + back == i) {
                    kaprekarArr[count] = i;
                    count++;
                }
        }
    }
    int[] answer = new int[count];
3   for (int i = 0; i < count; i++)
        answer[i] = kaprekarArr[i];
    return answer;
}
```

①. for 문을 이용하여 1 부터 k 이하의 정수를 가져옴

- 제곱수를 두 부분으로 나누기 위한 기준 수를 갖는 divisor 를 10^0= 1 로 초기화

②. while 문 : 제곱수를 divisor(기준 수)로 나눈 몫이 0 이 아닌 동안 반복하도록 조건식을 수정

- front = 제곱수를 divisor(기준 수)로 나눈 몫
- back = 제곱수를 divisor(기준 수)로 나눈 나머지

  → 제곱수를 두 부분으로 나누어 front 와 back 에 저장
- divisor 에 10 을 곱하여 다음 반복에 사용할 기준 수를 계산
- front 와 back 의 값이 모두 0 이 아니면서 두 변수를 더한 값이 본래의 수와 같으면 카프리카 수 배열 kaprekaArr 에 넣고, count 에 추가

③. answer 변수에 카프리카 수 배열 kaprekaArr 을 복사

8. 문제 8

1) 문제 코드

```java
/*=======================================
    3차 8번   3차 1급 8_initial_code.java
 =======================================*/

class Solution {
    public int solution(int k, int[] student) {
        int answer = 0;
        for(int i = 0; i < student.length; i++){
            student[i] -= 4*k;
            if(student[i] <= 0)
                break;
            answer += (student[i] + k - 1) / k;
        }
        return answer;
    }

    // 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다. main 메소드는 잘못된
    public static void main(String[] args) {
        Solution sol = new Solution();
        int k1 = 1;
        int[] student1 = {4, 4, 4, 4};
        int ret1 = sol.solution(k1, student1);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret1 + " 입니다.");

        int k2 = 3;
        int[] student2 = {15, 17, 19, 10, 23};
        int ret2 = sol.solution(k2, student2);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret2 + " 입니다.");
    }
}
```

2) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제
- 교실에 선풍기 4 대가 있다는 조건에서 한 대의 선풍기가 송풍할 수 있는 학생 수 k 와 각 반의 학생 수 배열 student 를 매개변수로 받아서 바람을 받지 못하는 학생 수를 구한 다음 더 필요한 선풍기 수를 구해야 함

3) 정답

```java
public int solution(int k, int[] student) {
    int answer = 0;
1   for(int i = 0; i < student.length; i++){
        student[i] -= 4*k;
2       if(student[i] <= 0)
            continue;
3       answer += (student[i] + k - 1) / k;
    }
    return answer;
}
```

① for 문을 이용해 학급 당 학생 수만큼 돌면서, 교실에 있는 4 대의 선풍기의 바람을 받지 못하는 학생 수를 구함

: 학생 수 – (선풍기 4 대 * 한 대의 선풍기가 송풍하는 학생 수) 를 계산하면 바람을 받지 못하는 학생 수가 계산됨

② 바람을 받지 못하는 학생 수가 0 이하이면 현재 학급에서는 더 필요한 선풍기 수를 구할 필요가 없으므로 다음 학급의 학생 수를 가져와서 작업을 계속하기 위한 명령인 continue 를 사용

← 문제 코드에 있던 break 를 사용하면, 바람을 받지 못한 학생 수가 0 이하인 경우 반복문은 더 이상 실행되지 않고 강제 종료되어 다음 학급에서 필요한 선풍기 대수를 구할 수 없음

③ 바람을 받지 못하는 학생들을 위해 추가로 구입해야 하는 선풍기 대수를 계산

- **바람을 받지 못한 학생 수를 k(선풍기 한 대가 송풍하는 학생 수)로 나눈 몫 + k 로 나눈 나머지가 1 이상 k-1 이하인 경우에도 제공해야 하는 선풍기 1 대** 의 계산이 이루어져야 함
- **바람을 받지 못하는 학생수 + (k 로 나누었을 때 나타날 수 있는 최대 나머지 = k-1)** 값을 k 로 나눈 몫을 구함

9. 문제 9

1) 문제 코드

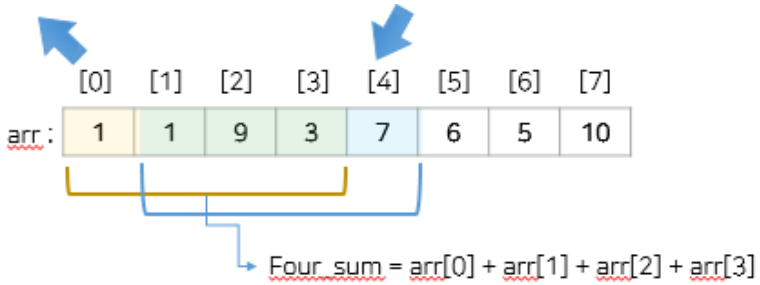
```
/*=======================================
    3차 9번   3차 1급 9_initial_code.java
 =======================================*/

class Solution {
    public int solution(int[] revenue, int k) {
        int answer = 0;
        int n = revenue.length;
        int sum = 0;
        for (int i = 0; i < k; i++) {
            sum += revenue[i];
        }
        answer = sum;
        for (int i = 0; i < n; i++) {
            sum = sum - revenue[i - k] + revenue[i];
            if (answer < sum)
                answer = sum;
        }
        return answer;
    }

    // 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다. main 메소드는 잘못된
    public static void main(String[] args) {
        Solution sol = new Solution();
        int[] revenue1 = {1, 1, 9, 3, 7, 6, 5, 10};
        int k1 = 4;
        int ret1 = sol.solution(revenue1, k1);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret1 + " 입니다.");

        int[] revenue2 = {1, 1, 5, 1, 1};
        int k2 = 1;
        int ret2 = sol.solution(revenue2, k2);

        // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
        System.out.println("solution 메소드의 반환 값은 " + ret2 + " 입니다.");
    }
}
```

2) 문제 개요

- 제시된 과제가 바르게 수행되도록 문제 코드를 수정하는 문제
- 슬라이딩윈도우 방식을 사용하여 매출액 배열 revenue 에서 연속된 k 개의 매출액 합계가 최대인 값을 찾는 코드에서 잘못된 곳을 찾아 수정해야 함
- 슬라이딩 윈도우 방식

  : 일정한 너비의 창문을 옆으로 미는 것처럼 연속된 일정 개수의 항목을 이용하여 문제를 풀 때 사용하는 방법

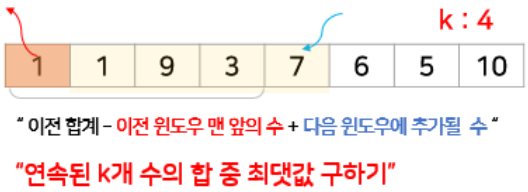
  예) 연속된 4 개 항목의 합을 구하는 경우



- 배열 arr 에서 연속된 4 개의 항목 합계를 구하여 Four_sum 에 저장한 후, 다음 4 개의 항목 합을 구하기 위해서 Four_sum 에서 arr[0]을 빼고, arr[4] 를 더하면 원하는 값을 구할 수 있음.

3) 정답

```java
public int solution(int[] revenue, int k) {
    int answer = 0;
    int n = revenue.length;
    int sum = 0;
    for (int i = 0; i < k; i++) {
        sum += revenue[i];
    }
    answer = sum;
    for (int i = k; i < n; i++) {
        sum = sum - revenue[i - k] + revenue[i];
        if (answer < sum)
            answer = sum;
    }
    return answer;
}
```



"연속된 k개 수의 합 중 최댓값 구하기"

①. sum 변수에 첫 번째 매출액부터 k 번째 매출액까지의 합을 구하기

②. for 문을 이용하여 k 부터 (매출액 항목 수 – 1) 까지의 값을 i 로 받아 옴

- 새로운 k 개의 매출액 합계를 계산 : 현재 구한 매출액 합계에서 가장 앞에 위치한 항목값을 빼고 현 매출액 합계에서 마지막 항목의 다음 값을 더해서 새로운 k 개의 매출액 합계를 계산
- 합계 중 가장 큰 값을 answer 에 넣어 리턴

4) 보충 학습 : 연속된 k 개의 합 중 최댓값 구하기 - 브루트 포스 방식 vs 슬라이딩 윈도우 방식

① 브루트 포스 방식

기준 항목을 선택한 후 중첩 for 문을 이용하여 기준 항목부터 k 개의 항목을 합산하여 연속된 k 개의 합 중 최댓값을 찾음

```
public int solution(int[] revenue, int k) {
    int answer = 0;
    int n = revenue.length;
    int sum = 0;
    for (int i = 0; i <= n-k; i++) {
        sum=0;

        for (int j = i; j < i+k ;j++) {
            sum = sum+revenue[j];
            if (answer < sum)
                answer = sum;
        }
    }
    return answer;
```

➔ 시간 복잡도 = O(k*n)

② 슬라이딩 윈도우 방식

처음 k 개의 합계를 구한 뒤 for 문을 이용해 반복하면서 이전에 구했던 합계 구간에서 이전 구간의 첫 번째 항목값은 빼고 이전 구간의 바로 다음에 있는 항목값을 더하면서 연속된 k 개의 합계 중 최댓값을 찾음

➔ 이전에 구한 합계 값 일부를 사용함으로써 반복되는 계산 횟수를 줄일 수 있음

```
public int solution(int[] revenue, int k) {
     int answer = 0;
     int n = revenue.length;
     int sum = 0;

     for (int i = 0; i < k ; i++) {
        sum += revenue[i];
     }

     answer = sum;

     for (int i = 1; i<=n-k; i++  ) {
        sum = sum - revenue[i-1] + revenue[i+k-1];
        if ( answer < sum)
            answer = sum;
     }
     return answer;
}
```

➔ 시간 복잡도 = O(n)

5) 이론 정리 : 브루트 포스, 슬라이딩 윈도우, 투 포인터

① 특징

브루트 포스 **(Brute Force)**	**슬라이딩 윈도우** **(Sliding Window)**	**투 포인터** **(Two Pointer)**
모든 경우를 탐색해야 할 때	연속된 구간의 데이터에 대한 문제일 때	정렬되어 있고, 중복된 값이 없을 때
중첩 반복문을 이용해 순회	일정 구간을 정해서 이동시킴	포인터 두 개를 활용

② 시간 복잡도

구분	**브루트 포스**	**슬라이딩 윈도우**	**투 포인터**
세 수의 합이 K 의 배수가 되는 경우 구하기	$O(n^3)$		
연속된 K 개 수의 합의 최댓값 구하기	O(k*n)	O(n)	
세 수의 합이 K 가 되는 수 구하기	$O(n^3)$		$O(n^2)$

10. 문제 10

```
/*=======================================
    3차 10번   3차 1급 10_initial_code.java
 =======================================*/
import java.util.ArrayList;
import java.util.Iterator;

//Shop 인터페이스와 HairShop, Restaurant 클래스는 Inner Class로 작
class Solution {
    class Shop{
        protected ArrayList<Customer> reserveList;
        public Shop() {
            this.reserveList = new ArrayList<Customer>();
        }
        public boolean reserve(Customer customer){
            reserveList.add(customer);
            return true;
        }
    }
    class Customer{
        public int id;
        public int time;
        public int numOfPeople;
        public Customer(int id, int time, int numOfPeople){
            this.id = id;
            this.time = time;
            this.numOfPeople = numOfPeople;
        }
    }
    class HairShop @@@ {
        public HairShop(){
            super();
        }

        @@@{
            if(@@@ != 1)
                return false;

            Iterator<Customer> iter = reserveList.iterator();
            while (iter.hasNext()) {
                Customer r = iter.next();
                if(@@@)
                    return false;
            }
            reserveList.add(customer);
            return true;
        }
    }
```

1) 문제 코드

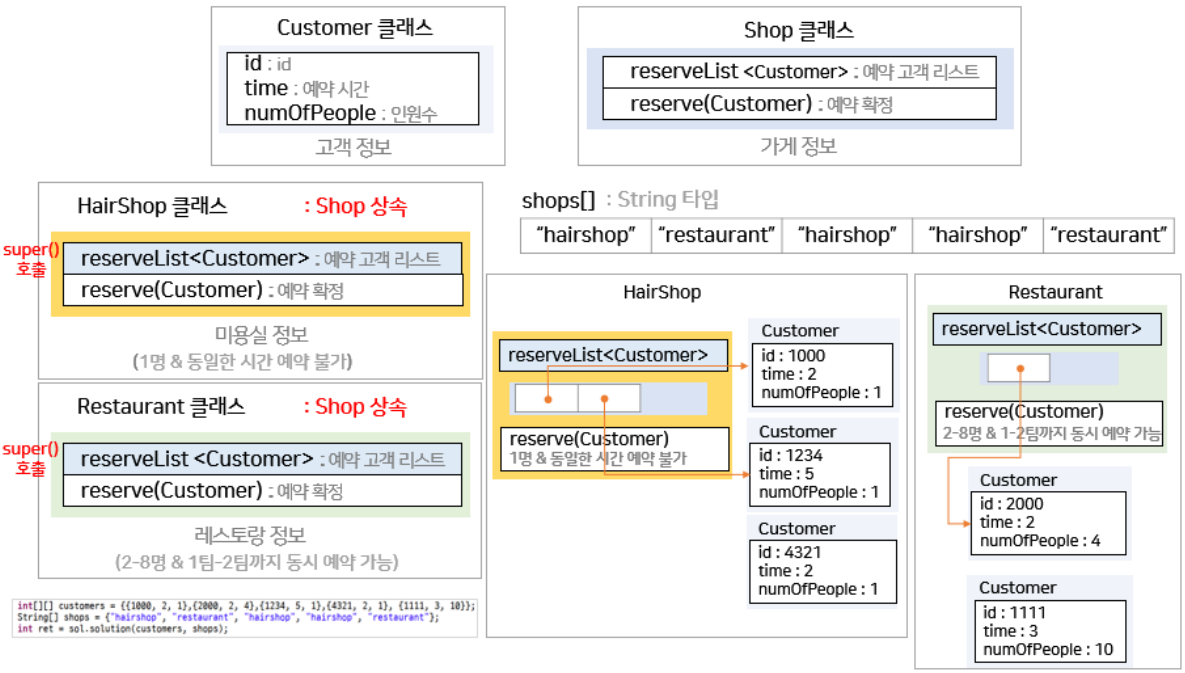
```
class Restaurant @@@ {
    public Restaurant(){
        super();
    }
    @@@{
        if(@@@)
            return false;
        int count = 0;

        Iterator<Customer> iter = reserveList.iterator();
        while (iter.hasNext()) {
            Customer r = iter.next();
            if(@@@)
                count += 1;
        }
        if(count >= 2)
            return false;
        reserveList.add(customer);
        return true;
    }
}
```

```
public int solution(int[][] customers, String[] shops) {
    Shop hairshop = new HairShop();
    Shop restaurant = new Restaurant();
    int count = 0;
    for(int i = 0; i < shops.length; i++){
        if(shops[i].equals("hairshop")){
            if(hairshop.reserve(new Customer(customers[i][0], customers[i][1], customers[i][2])))
                count += 1;
        }
        else if(shops[i].equals("restaurant")){
            if(restaurant.reserve(new Customer(customers[i][0], customers[i][1], customers[i][2])))
                count += 1;
        }
    }
    return count;
}

// 아래는 테스트케이스 출력을 해보기 위한 main 메소드입니다.
public static void main(String[] args) {
    Solution sol = new Solution();
    int[][] customers = {{1000, 2, 1},{2000, 2, 4},{1234, 5, 1},{4321, 2, 1}, {1111, 3, 10}};
    String[] shops = {"hairshop", "restaurant", "hairshop", "hairshop", "restaurant"};
    int ret = sol.solution(customers, shops);

    // [실행] 버튼을 누르면 출력 값을 볼 수 있습니다.
    System.out.println("solution 메소드의 반환 값은 " + ret + " 입니다.");
}
}
```

2) 문제 개요

- 일반 클래스인 Shop 클래스를 상속하는 HairShop 클래스와 Restaurant 클래스를 정의하는 코드를 완성하는 문제
- HairShop 클래스의 reserve( ) 메소드에서는 신규 예약 고객 수가 1 명인 경우에 예약 가능하지만, 신규 예약 고객의 예약 시간이 기존 예약 손님들의 예약 시간과 중복되면 예약할 수 없도록 코딩해야 함
- Restaurant 클래스의 reserve( ) 메소드에서는 신규 예약 고객 수가 2 명 이상 8 명 이하인 경우에 예약 가능하지만, 신규 예약 고객의 예약 시간에 기존 예약 건수가 2 건이 존재하면 예약할 수 없도록 코딩해야 함

- solution( ) 메소드는 [고객 식별번호, 고객이 신청한 예약 시간, 예약 인원 수]를 항목으로 갖는 배열 customers 와 'hairshop' 혹은 'restaurant' 로 구성된 배열 shops 를 인수로 받고, HairShop 클래스의 객체와 Restaurant 클래스의 객체를 생성하고 reserve( ) 메소드를 실행해서 고객 예약 횟수를 집계



3) 정답

- Customer 클래스의 생성자 메소드에 의해서 세 개의 멤버 변수를 매개변수로 전달된 값으로 초기화
  - id : 고객 id (1 이상 10,000 이하의 자연수)를 매개변수 id 의 값으로 초기화
  - time : 예약 시간 (0 이상 23 이하의 자연수)를 매개변수 time 의 값으로 초기화
  - numOfPeople : 예약 인원 수 (1 이상 10 이하의 자연수)를 매개변수 numOfPeople 로 초기화

- Shop 클래스 정의
  - 생성자 메소드에서 멤버 변수 reserveList 를 생성하여 빈 배열을 할당
  - reserve( ) 메소드에서 매개변수로 받은 customer 를 reserveList 에 추가하고 true 를 return

- Shop 클래스를 상속하도록 HairShop 클래스 정의

```
class HairShop @@@ {        extends Shop
    public HairShop(){
        super();
    }

    public boolean reserve(Customer customer) {
        if(@@@ != 1)          customer.numOfPeople
            return false;

        Iterator<Customer> iter = reserveList.iterator();
        while (iter.hasNext()) {
            Customer r = iter.next();
            if(@@@)    r.time == customer.time
                return false;
        }
        reserveList.add(customer);
        return true;
    }
}
```

- 생성자 메소드는 부모 클래스의 생성자 메소드를 그대로 실행하므로 HairShop 클래스의 멤버 변수로 reseveList 가 빈 ArrayList 로 생성됨
- reserve( ) 메소드를 오버라이드
  - Customer 클래스 타입의 매개변수 customer 의 멤버 변수 numOfPeople 값이 1 이 아니면 false 를 리턴
  - reserveList 에 있는 예약 고객 데이터에서 customer 의 멤버 변수 time 과 중복된 예약 시간이 존재하면 false 를 return
  - 위의 두 가지 조건에 해당하지 않으면 reserveList 에 매개변수 customer 를 추가하고 true 를 return

- Shop 클래스를 상속하도록 Restaurant 클래스 정의

```
class Restaurant @@@ {          extends Shop
    public Restaurant(){
        super();
    }

    public boolean reserve(Customer customer) {
        if(@@@)                          customer.numOfPeople < 2 ||
            return false;                customer.numOfPeople > 8
        int count = 0;

        Iterator<Customer> iter = reserveList.iterator();
        while (iter.hasNext()) {
            Customer r = iter.next();
            if(@@@)                      r.time == customer.time
                count += 1;
        }
        if(count >= 2)
            return false;
        reserveList.add(customer);
        return true;
    }
}
```

  - 생성자 메소드는 부모 클래스의 생성자 메소드를 그대로 실행하므로 Restaurant 클래스의 멤버 변수로 reseveList 가 빈 ArrayList 로 생성됨
  - reserve( ) 메소드를 오버라이드
    - ◆ Customer 클래스 타입의 매개변수 customer 의 멤버 변수 numOfPeople 값이 2 보다 작거나 8 보다 크면 false 를 리턴
    - ◆ reserveList 에 iterator( )를 이용하여 예약 고객 데이터에서 매개변수 customer 의 멤버 변수 time 값과 같은 예약 고객 수를 count 변수에 집계한 후, count 변수의 값이 2 이상이면 false 를 return
    - ◆ 아니면 예약 리스트 reserveList 에 고객을 추가

- solution( ) 메소드

```
public int solution(int[][] customers, String[] shops) {
    Shop hairshop = new HairShop();
    Shop restaurant = new Restaurant();
    int count = 0;
    for(int i = 0; i < shops.length; i++){
        if(shops[i].equals("hairshop")){
            if(hairshop.reserve(new Customer(customers[i][0], customers[i][1], customers[i][2])))
                count += 1;
        }
        else if(shops[i].equals("restaurant")){
            if(restaurant.reserve(new Customer(customers[i][0], customers[i][1], customers[i][2])))
                count += 1;
        }
    }
    return count;
}
```

- shops 의 값이 'hairshop' 이면 hairshop 객체의 reserve( ) 메소드를 실행하고 그 결과가 true 이면 예약 건수를 집계하는 count 값을 1 만큼 증가
: hairshop 의 reserve( ) 메소드의 매개변수에 customer 의 항목 3 개(id,time,numOfPeople)를 이용하여 생성한 Customer 클래스의 객체를 전달
- shop 의 값이 'restaurant' 이면 restaurant 객체의 reserve( ) 메소드를 실행하고 그 결과가 true 이면 예약 건수를 집계하는 count 값을 1 만큼 증가
:restaurant 의 reserve( ) 메소드의 매개변수에 customer 의 항목 3 개 (id,time,numOfPeople)를 이용하여 생성한 Customer 클래스의 객체를 전달