> 07. 이중 for문 사용하기

반복문안에 반복문을 넣을 수 있다. for문 안에 for문을 중복해서 사용 할 수있다.

0123456789012345678901234567890123456789012345678901234567890123456789…

0123456789를 출력하는 이작업을 10번 반복하여 출력하고 싶으면 다음과 같이 기술 하면 될 것이다. for문 을 이용해서 0123456789를 10번 출력하는 코드로 다음과 같이 기술하면 될 것이다.

```
for(int i=0;i<10;i++) {
    System.out.print("0123456789");
}
```

잘 생각해 보면 for문 사이에 있는 프린트 문을 for문을 이용해서 0부터 9까지 출력해서 화면에 0123456789가 출력 되도록 할 수 있다. print문으로 0123456789를 출력하는 대신에 다음과 같은 for문을 이용하여 같은 결과를 만들 수 있을 것이다.

```
for(int j=0;j<10;j++) {
    System.out.print(j);  // 출력결과 0123456789
}
```

이 2가지 결과를 합친 결과는 다음과 같다.

```
for(int i=0;i<10;i++) {
    for(int j=0;j<10;j++){
        System.out.print(j);
    }
}
```

이중for 문을 사용할 때 조심해야 할 점은 초기식으로 선언되는 i,j 같은 변수들의 이름이 겹치거나 잘못 사용하면 문제가 발생 한다는 점이다.

바깥쪽 for문의 반복 코드 부분이 새로 시작 하게 되면 내부에 선언한 변수와 for문의 초기문도 새로 실행되어 변수의 값들이 초기화 되어 이전에 가지고 있던 값이 사라진다. 코드에서 j 값은 i가 0부터 변환식에 도착해서 하나씩 증가 할때 마다 0으로 초기화 되어 다시 하나씩 증가한다.

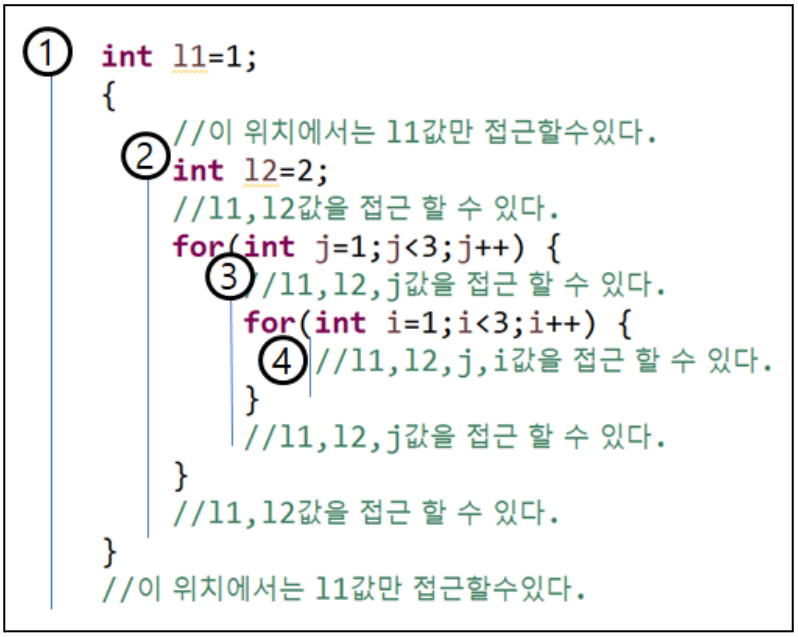
변수는 지역 변수와 전역변수로 구성되어 있는데 지역변수는 특정 지역에서만 사용할 수 있는 변수이고 전역변수는 모든 지역에서 사용할 수 있는 변수이다. 지금 까지 사용한 변수는 모두 지역 변수이고 전역 변수 앞에는 static이 붙는다. 나중에 좀더 자세히 살펴볼 예정이고 일단 지역변수의 특성만 좀 기억해 보자.

지역변수는 기본적으로 중괄호 블럭안에 선언한 변수로 선언되면 해당 중괄호 블럭이나 해당 중괄호 블럭 안쪽의 중괄호 블럭들에서만 사용할 수 있다.

지역변수가 선언된 범위에서 다시 같은 이름의 지역 변수를 선언하면 동일한 지역 변수를 식별 할 수 없어서 문제가 발생 하니 주의하자.

메소드의 매개변수는 해당 메소드 중괄호 블록 안에서 지역변수로 사용된다.

for문의 초기화 식은 해당 for문 중괄호 블록 안에서 지역변수로 사용된다.



왼쪽 이미지의 1번 부분은 메인 메소드 선언부에 중괄호안 이여서 l1은 main 메소드에 선언된 지역변수 이다.

이 중괄호 부분 에서 선언된 l1는 중괄호 내부인 왼쪽 이미지에서 표시된 1, 2, 3, 4 중괄호 안쪽 부분에서 사용할 수 있다.

2번 부분에 선언된 l2의 경우 1번 부분은 2번 중괄호 부분의 바깥쪽 중괄호 이므로 1번을 제외한 2,3,4 중괄호 부분에서 사용할 수 있다.

3번에 선언된 변수 j같은 경우 3번 중괄호 블록에서 사용하려고 선언한 변수이므로 3,4에서만 사용할 수 있다.

4번 블록에서 선언된 i값은 4번 블록에서만 사용할 수있다.

상위 모든 주석 부분에 l1,l2,i,j 변수을 찍을수 있는지 코드를 통해서 직접 확인해보자.

다음과 같이 아래로 출력하는 방법에 대해서 생각해 보자.

```
0123456789      //012345678901234567890123456… 옆으로 출력되는 것과
0123456789      //아래로 출력되는 것의 차이점을 생각해보자.
0123456789      //옆으로 출력되는 것은 엔터가 없는것이고
0123456789      //아래로 출력되는 것은 엔터가 있는것이여서
0123456789      // '0123456789'가 반복되는 것이아니라.
0123456789      // '0123456789엔터'가 반복되는 것이다.
```

```
for(int i=0;i<10;i++) {
    //System.out.print(i+"\n");
    System.out.print(i);

}
System.out.print("\n");
```

하나의 for문으로 "0123456789엔터"를 출력하는 프로그램을 구현하려면 왼쪽과 같이 하면 된다. 간혹 주석 부분처럼 잘못 구현하는 경우가 있는데 잘 생각해 보면 주석 한 부분은 다음과 같이 잘못 출력 되는 걸 알 수 있다.

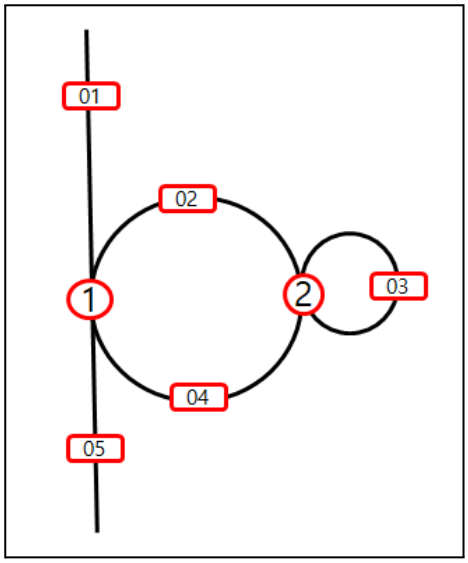
“0엔터1엔터2엔터3엔터4엔터5엔터6엔터7엔터8엔터9엔터엔터”

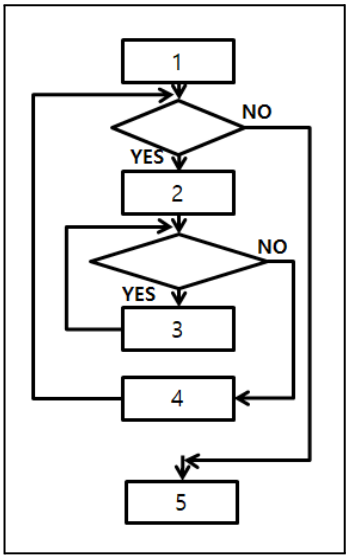
올바르게 구현하려면 상위 코드처럼를 이용해서 “0123456789엔터”가 여섯번 반복되도록 코드를 구현하면 우리가 원하는 출력 결과를 얻을 수 있다.

```
for(int j=0;j<10;j++) {
    for(int i=0;i<10;i++) {
        //System.out.print(i+"\n");
        System.out.print(i);

    }
    System.out.print("\n");
}
```

“0123456789엔터”이 아래로 열개 출력되도록 구현해 보자. 다음과 같이 구현 가능할 것이다.






```
1

while( ){
        2
        while( ){
                3
        }
        4
}

5
```

상위 맨왼쪽 이미지를 보고 순서도와 의사 코드를 작성 하시오.

상위 순서도에서 출력 되는 모든 경우를 기술해 보자. 반복되는 부분을 소괄호로 묶었다.

1번 부부의 반복문이 거짓인 경우 출력 결과는 01, 05가 된다.

1번 부분이 반복문이 참이고 2 번 부분이 반복무늬 거짓이면 출력 결과는 01, 04, 02, 05가된다.

1번 부분이 반복문이 참이고 2 번 부분이 반복무늬 거짓인 상태에서 1번 부분이 여러번 반복되면 출력 결과는 01, (04, 02), 04, 02, 04, 02,..., 05가 된다.

1번 부분이 반복문이 참이고 2 번 부분이 참 일때 출력 결과는 01, 04, 03, 02, 05가 된다.

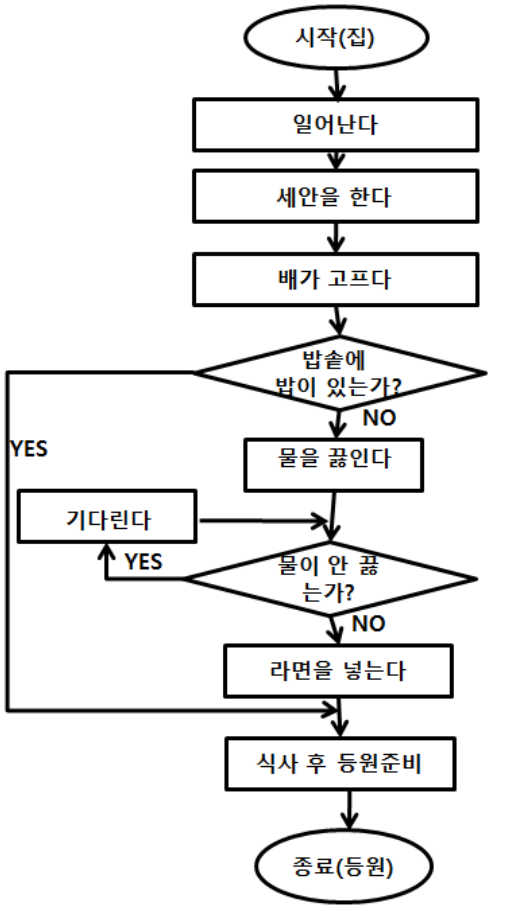
1번 부분이 반복문이 참이고 2 번 부분이 참 일때 반복문 1번은 1번 반복문 2번은 여러번 반복된다면 출력 결과는 01, 04, (03), 03, 03, ..., 02, 05가 된다.

1번 부분이 반복문이 참이고 2 번 부분이 참인 일때 반복문 1번은 여러번 반복문 2번은 한번 반복된다면 출력 결과는 01, (04, 03, 02),04, 03, 02,04, 03, 02, ..., 05,가

된다.

1번 부분이 반복문이 참이고 2 번 부분이 참인 일때 반복문 1번은 여러번 반복문 2번도 여러번 반복된다면 출력 결과는 01, (04, (03), 03, 03,... 02),04, 03, 02,04, 03, 02, ..., 05가 된다.

문제1) 아래의 순서도 의사코드에서 뭐가 잘못되었는지 확인해서 올바르게 고쳐 보자.



의사코드

시작(집)

일어난다

세안을 한다

배가 고프다

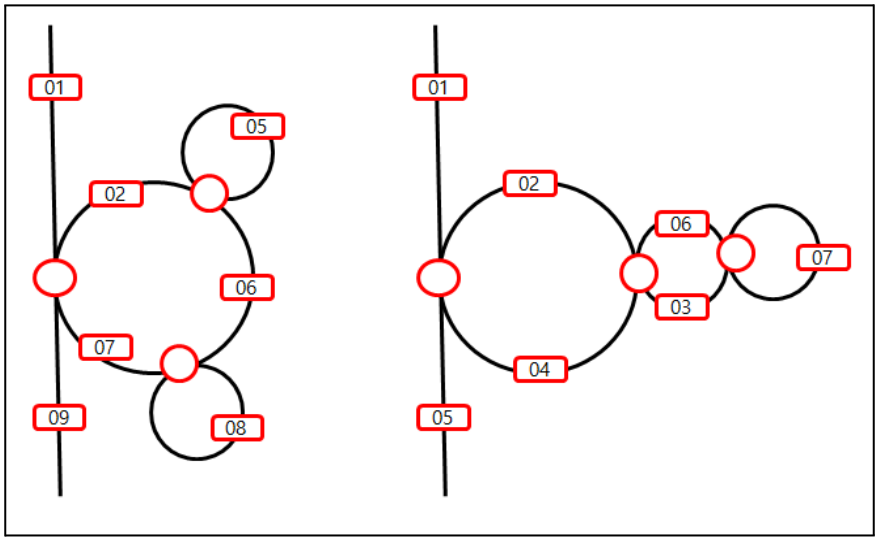
if(밥솥에 밥이 있는가?) {
식사 후 등원준비
}else{
물을 끓인다
}

While(물이 안 끓는가?){
기다린다
}

라면을 넣는다

식사 후 등원준비

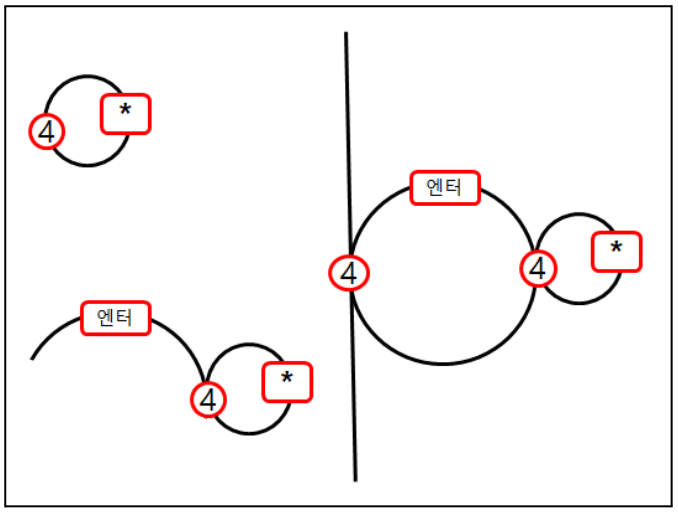
종료(등원)



문제 2)왼쪽 기찻길 이미지 두개를 순서도와 의사 코드로 구현하여 보자.

```
****
****
****
****
```

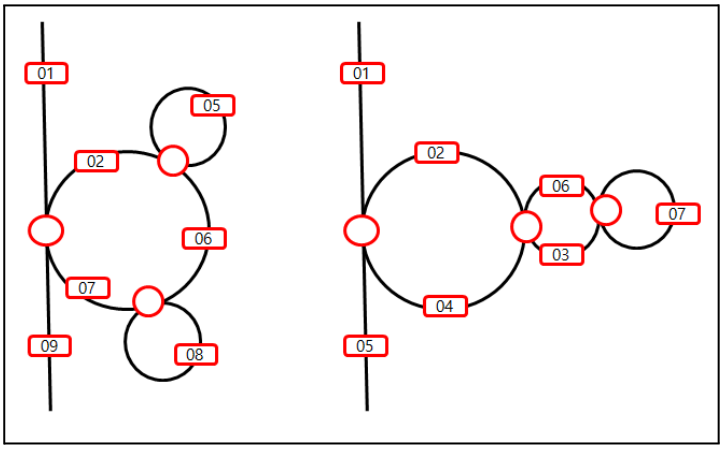
다음을 생각해 보자. 별을 4번찍다가 엔터를 한번치고 별을 4번찍다가 엔터를 한번치고 이 작업을 4번 반복하면 ' ****엔터 ****엔터 ****엔터 ****엔터 '와 같이 출력이 될 것이고 실제로 엔터가 줄을 바꾼다면 왼쪽 처럼 직사각형 모양이 될 것이다. 이것을 기찻길로 만든다고 생각해보자. ****은 별한개 출력하는 작업을 4번 반복하는것과 같다. 이는 아래 왼쪽 상단의 기찻길 모양과 같이 될것이다. 그 다음에 엔터를 쳐야 할 것이다.



이것은 왼쪽 아래 이미지처럼 별 4 번 찍은 것에 엔터 한번 친 것이 될 것이다. 별 4번 찍고 엔터 1 번 치는 것을 총네 번 반복해야 된다. 빨간색 동그라미 안에 숫자가 반복 횟수 라면 최종 결과가 왼쪽 이미지 처럼 될 것이고 이것은 사각형 모양을 화면에 출력하는 기찻길이 될 것이다. 왼쪽 이미지 기찻길을 순서도와 의사 코드로 만들어 보자.

문제 1) 아래처럼 출력될 수 있도록 기찻길을 만들고 순서도를 만든 다음 의사 코드를 작성해보자.

1. 1**********1
2. *****1*****1*****1*****1
3. 2*****2*****2*****2*****
4. 21**********1
5. 1****21****21****21****2
6. 1111****21111****21111****21111****2
7. 1111****22221111****22221111****2222
8. 11112222333344441111222233334444111122223333444

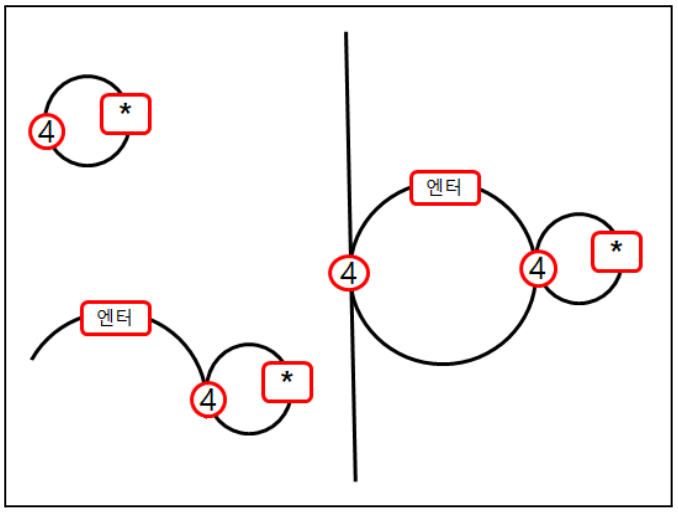


문제 1)왼쪽 기찻길 이미지 두개를 순서도와 의사 코드로 구현하여 보자.

이중 반복문을 이용해서 아래 이미지 처럼 정사각형 별을 찍는 프로그램을 작성하는 방법을 생각해보자.

```
****
****
****
****
```

다음을 생각해보자. 별을 4번찍다가 엔터를 한번치고 별을 4번찍다가 엔터를 한번치고 이 작업을 4번 반복하면 ' ****엔터 ****엔터 ****엔터 ****엔터 '와 같이 출력이 될 것이고 실제로 엔터가 줄을 바꾼다면 왼쪽 처럼 직사각형 모양이 될 것이다. 이것을 기찻길로 만든다고 생각해보자. ****은 별한개 출력하는 작업을 4번 반복하는것과 같다. 이는 아래 왼쪽 상단의 기찻길 모양과 같이 될것이다. 그 다음에 엔터를 쳐야 할 것이다.



이것은 왼쪽 아래 이미지처럼 별 4 번 찍은 것에 엔터 한번 친 것이 될 것이다. 별 4번 찍고 엔터 1 번 치는 것을 총네 번 반복해야 된다. 빨간색 동그라미 안에 숫자가 반복 횟수 라면 최종 결과가 왼쪽 이미지 처럼 될 것이고 이것은 사각형 모양을 화면에 출력하는 기찻길이 될 것이다. 코드로 구현해 보자.

별을 한개 찍는다. `System.out.print("*");`

상위 왼쪽상단 이미지 처럼 별을 4번 찍는다.

```
for(int i=0;i<4;i++) {
    System.out.print("*");
}
```

상위 왼쪽하단 이미지 처럼 별을 4번 찍고 엔터를 한번 친다.

```
for(int i=0;i<4;i++) {
    System.out.print("*");
}
System.out.println();
```

상위 오른쪽 이미지 처럼 별을 4번 찍고 엔터를 한번 치는 작업을 4번 반복한다.

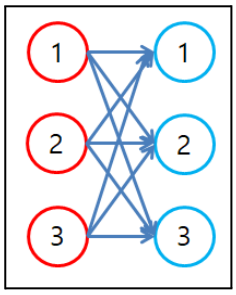
```
for(int j=0;j<4;j++) {
    for(int i=0;i<4;i++) {
        System.out.print("*");
    }
    System.out.println();
}
```

문제 1) 아래처럼 출력될 수 있도록 기찻길을 만들고 순서도를 만든 다음 코드를 작성해보자. 구현이 어려우면 기찻길을 그려보고 하자. 그래도 어려우면 순서도를 그려보고

하자.

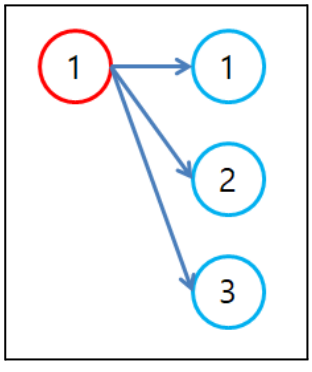
1. 1**********1
2. *****1*****1*****1*****1
3. 2*****2*****2*****2*****
4. 21**********21
5. 1****21****21****21****2
6. 1111****21111****21111****21111****2
7. 1111****22221111****22221111****2222
8. 11112222333344441111222233334444111122223333444

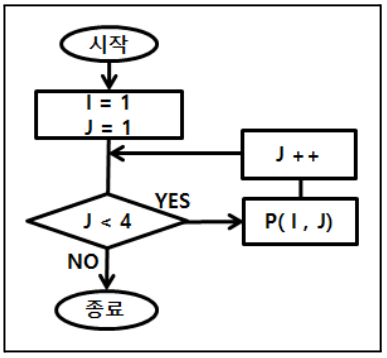
빨간공과 파란공을 모두 짝지어 찍는 프로그램을 구현해 보자.



빨간공 1,2,3 3개와 파란공 1,2,3 3개를 모두 짝지어 보자. 모든 결과를 출력해 보면 다음과 같은 결과를 가진다. (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)

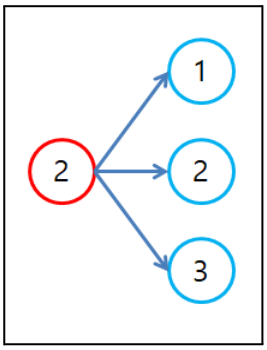
다음 3가지를 확인해 보자. 빨간색 1번 공을 가지고 파란공들과 짝을 지어 보면 (1,1), (1,2), (1,3) 과 같다. 빨간색 2번 공을 가지고 파란공들과 짝을 지어 보면 (2,1), (2,2), (2,3) 과 같다. 빨간색 3번 공을 가지고 파란공들과 짝을 지어 보면 (3,1), (3,2), (3,3)과 같다. 이 3가지를 하나씩 나눠서 순서도로 만들어 보면 다음과 같고 결국 3가지 결과를 합치면 빨간색 공과 파란색 공의 모든 짝이 된다.

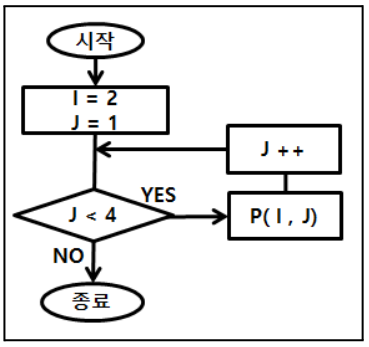




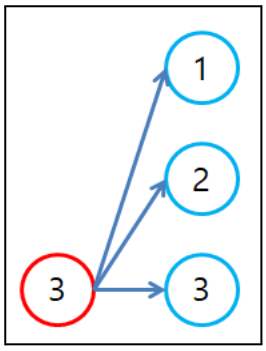
맨 왼쪽 이미지 처럼 빨간색 공 1개는 파란색공 3개와 짝을 지어보면 다음과 같이 될 것이다. 앞이 빨간공 뒤가 파란공이라면 (1,1), (1,2), (1,3) 와 같은 짝이 만들어 질 것이다. 자세히 보면 빨간 공은 1인 상태로 고정하고 파란 공을 1부터 3까지 하나씩 증가 하면서 짝을 지었다. 빨간색 공을 i라 하고 파란색 공을 j라 한다면 반복문을 이용해서 j값을 하나씩 증가시 키는 방법은 이미 공부 하여서 충분히 생각해 볼 수 있을 것이다. 고정된 값 i는 반복문 밖에 선언하고 반복문 안에서는 j 값을 1부터 하나씩 증가 시키며 i와 j 값을 출력하면 원하는 결과를 얻을 수 있을 것이다. 상위 순서도를 확인하고 기찻길과 의사코드를 만들어 보자. 순서도를 잘 따라 가며

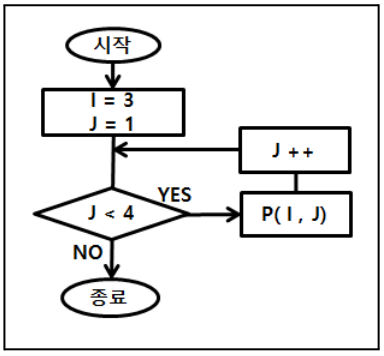
출력값을 확인해 보자.



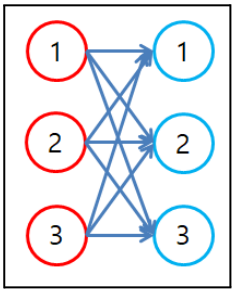


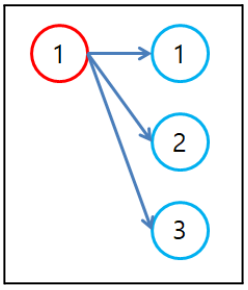
왼쪽 첫 번째 이미지 처럼 빨간 공이 2라면 (2, 1),(2, 2),(2, 3)와 같은 짝이 만들어 질 것이다. 이전 상황과 잘 비교해 보면 i 값이 2로 바뀐 것 말고는 변한게 없다는 것을 알 수 있다.

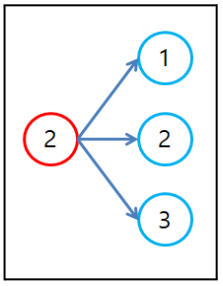


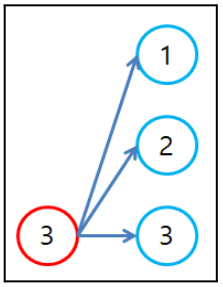


왼쪽 첫 번째 이미지 처럼 빨간 공이 3이라면 (3, 1),(3, 2),(3, 3)와 같은 짝이 만들어 질 것이다. 이전 상황과 잘 비교해 보면 i 값이 3로 바뀐 것 말고는 변한게 없다는 것을 알 수 있다.



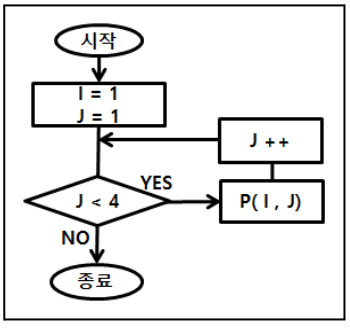


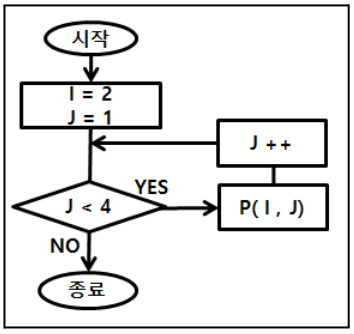


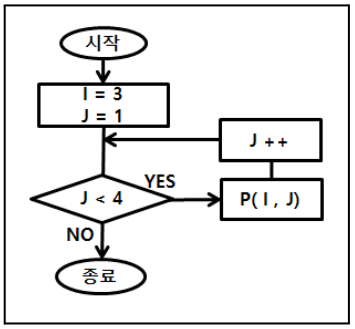


왼쪽 3개의 이미지를 합치면 맨 왼쪽 이미지 처럼 된다. 짝을 짓는다면 다음과 같은 결과가 나올 것이다. (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3), 상위와 같이 출력 되려면 어떻게 순서도와 코드를 구현 할 것인지 생각해 보자.

잘 살펴보면 맨 왼쪽 이미지는 나머지 세 개 이미지를 합친 것과 같다. 따라서 오른쪽 세 개 이미지를 순서도로 그리고 순서대로 실행 한다면 같은 결과가 나올 것이다.







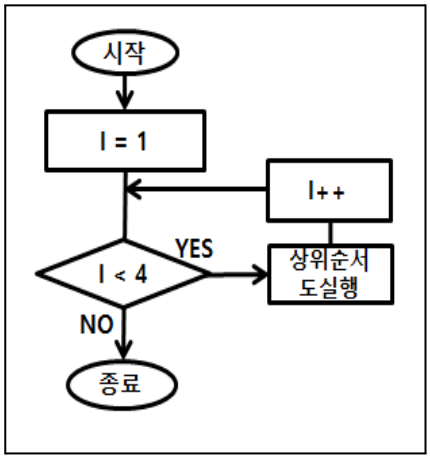
출력 결과를 잘 살펴보면 상위 첫 번째 순서도 출력결과는 (1, 1),(1, 2),(1, 3) 이다. 두 번째 순서도 출력결과는 (2, 1),(2, 2),(2, 3)이다. 세 번째 순서도 출력결과는 (3, 1),(3, 2),(3, 3) 이다. 세 가지 순서도를 왼쪽부터 순서대로 실행한다면 우리가 원하는 결과를 얻을 수 있고, 결국 (i, 1),(i, 2),(i, 3) 과 같음을 알 수 있다.

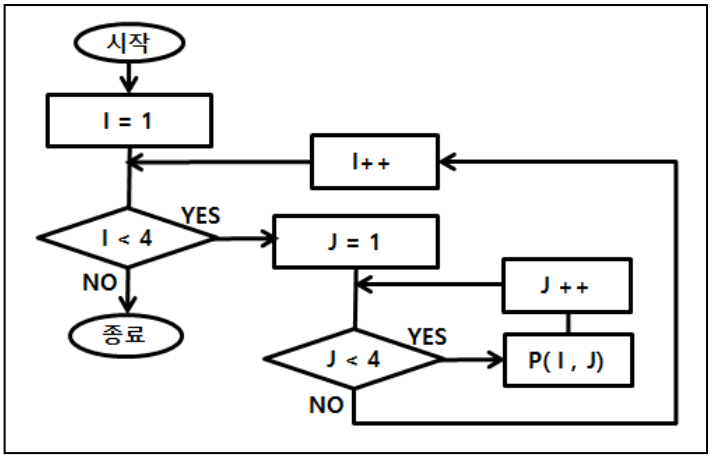
```
int i=1;
for(int j=1;j<4;j++) {
    System.out.println(i+":"+j);
}
```

다음을 자바코드로 변경하면 다음과 같다. i가 무엇이냐에 따라서 빨간색 공의 번호가 결정되는 것과 같다.

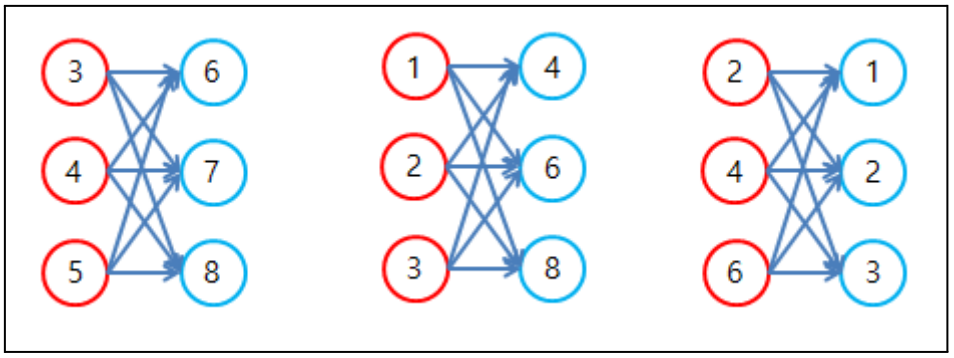
잘 생각해 보면 세 개의 순서도가 상당히 유사하다는 것을 알 수 있고 유일한 차이점을 찾는다면 첫 번째 순서도는 i는 1이고 두 번째 순서도는 i가 2이고 세 번째 순서도는 i가 3이라는 것이다. i 값이 하나씩 1, 2, 3 증가 하고 있다는 것만 빼고는 달라진 것이 하나도 없다. 결국에 i 값이 하나씩 증가 하면서 반복 되고 있다는 것이다. 숫자 하나씩 증가되면서 반복되는 것은 반복문으로 바꿀수 있다는 사실은 무수히 경험해 왔을 것이다. 반복문으로 만들어 보면 i값이 증가하는 형태의 반복문안에 상위 순서도 1개가 들어가 있으면 될 것이다.

상위 왼쪽 순서도를 확인해 보면 i가 1부터 3까지 3번 반복하면서 상위 순서도가 3번 실행된다. i가 1일때 i가 1인 상태에서 상위순서도가 실행이 되고 i가 2일때 i가 2인 상태에서 상위순서도가 실행이 되고 i가 3일때 i가 3인 상태에서 상위순서도가 실행이 되면 우리가 원하던 (1, 1),(1, 2),(1, 3) (2, 1),(2, 2),(2, 3) (3, 1),(3, 2),(3, 3) 결과를 얻을 수 있다. 다음 이미지는 상위 3개의 이미지를 합쳐서 하나의 순서도로 만들었다. i가 1씩 증가 하면서 상위 순서도를 하나씩 반복되는 형태이다. 천천히 확인해 보자.





오른쪽 순서도는 왼쪽 순서도에서 상위 순서도 실행 부분을 실제로 붙여서 만든 순서도로 하나하나 따라 가며 출력값을 확인해 보면 원하는 결과가 출력 된다는 것을 알 수 있다. 다음에 코드가 있는데 보지말고 for문과 while문으로 만들어 보자. 반복문으로 바꾸어 보자.



문제 1) 상위 3개의 이미지를 보고 어떤 결과가 출력되는지 기술 해 보고, 기찻길을 그려보고,순서도를 그려보고, 의사 코드를 만들어 보자.

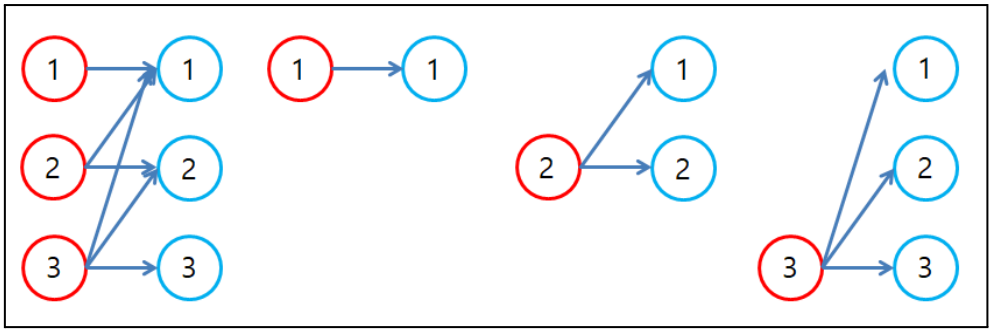
1*1=1	1*2=2	1*3=3	1*4=4	1*5=5	1*6=6	1*7=7	1*8=8	1*9=9
2*1=2	2*2=4	2*3=6	2*4=8	2*5=10	2*6=12	2*7=14	2*8=16	2*9=18
3*1=3	3*2=6	3*3=9	3*4=12	3*5=15	3*6=18	3*7=21	3*8=24	3*9=27
4*1=4	4*2=8	4*3=12	4*4=16	4*5=20	4*6=24	4*7=28	4*8=32	4*9=36
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30	5*7=35	5*8=40	5*9=45
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36	6*7=42	6*8=48	6*9=54
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49	7*8=56	7*9=63
8*1=8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48	8*7=56	8*8=64	8*9=72
9*1=9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54	9*7=63	9*8=72	9*9=81

1단 1*1=1 1*2=2 1*3=3 1*4=4 1*5=5 1*6=6 1*7=7 1*8=8 1*9=9 2단 2*1=2 2*2=4 2*3=6 2*4=8 2*5=10 2*6=12 2*7=14 2*8=16 2*9=18 3단 3*1=3 3*2=6 3*3=9 3*4=12 3*5=15 3*6=18 3*7=21 3*8=24

1*1=1	2*1=2	3*1=3
1*2=2	2*2=4	3*2=6
1*3=3	2*3=6	3*3=9
1*4=4	2*4=8	3*4=12
1*5=5	2*5=10	3*5=15
1*6=6	2*6=12	3*6=18
1*7=7	2*7=14	3*7=21
1*8=8	2*8=16	3*8=24
1*9=9	2*9=18	3*9=27
4*1=4	5*1=5	6*1=6
4*2=8	5*2=10	6*2=12
4*3=12	5*3=15	6*3=18
4*4=16	5*4=20	6*4=24
4*5=20	5*5=25	6*5=30
4*6=24	5*6=30	6*6=36
4*7=28	5*7=35	6*7=42
4*8=32	5*8=40	6*8=48
4*9=36	5*9=45	6*9=54
7*1=7	8*1=8	9*1=9
7*2=14	8*2=16	9*2=18
7*3=21	8*3=24	9*3=27
7*4=28	8*4=32	9*4=36
7*5=35	8*5=40	9*5=45
7*6=42	8*6=48	9*6=54
7*7=49	8*7=56	9*7=63
7*8=56	8*8=64	9*8=72
7*9=63	8*9=72	9*9=81

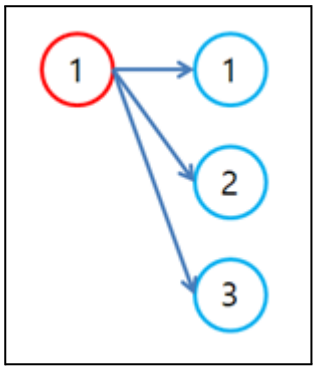
문제 2) 이미지와 같은 형태의 구구단을 출력할 수 있는 코드를 만들어 보자.

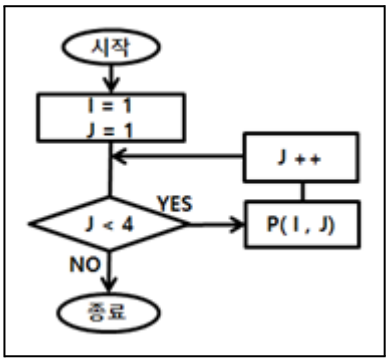
다음은 빨간색 i값이 파란색 j값에 영향을 주는 경우이다.

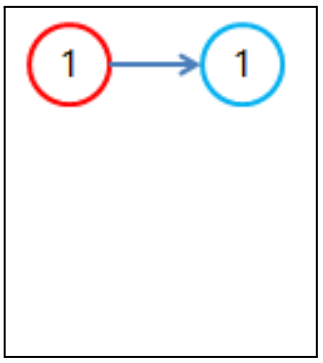


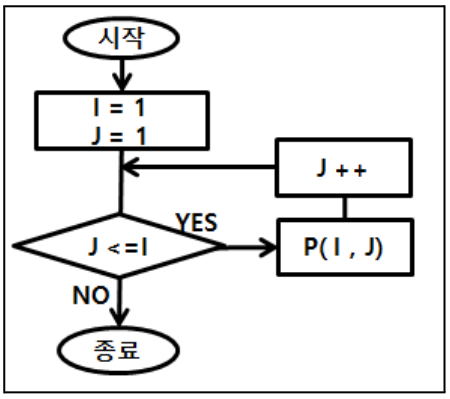
상위를 실행시키면 다음과 같은 결과가 나올 것이다. (1, 1) (2, 1),(2, 2) (3, 1), (3, 2),(3, 3) 빨간색 공을 i라 하고 파란색 공을 j라 한다면 i값은 j값을 찍는데 영향을 준다. i값이 1일때 j값은 1 한개 i값이 2일때는 j값 1,2 두 개 i값이 3일 때는 j값은 1, 2, 3 세개가 출력 된다. 잘 생각해 보면 파란공 j가 빨간공 i보다 같거나 작을때만 실행이

되는 것을 확인할 수 있다. i값에 따라 출력되는 결과가 규칙성 있게 달라지고 있고, 이렇게 i값이 j값 반복에 영향을 주려면 어떻게 해야 하는지 생각해 보자. i값이 j값 반복에 영향을 주려면 j값 반복문 안의 조건식으로 i값을 사용해야 한다.

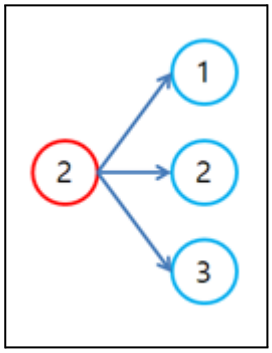


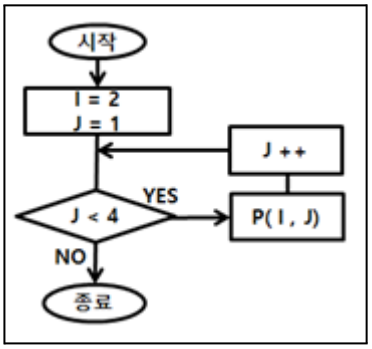


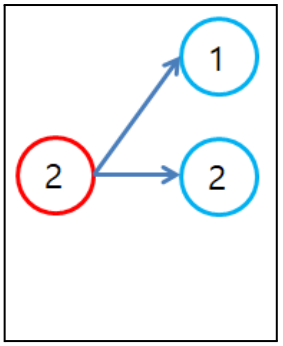


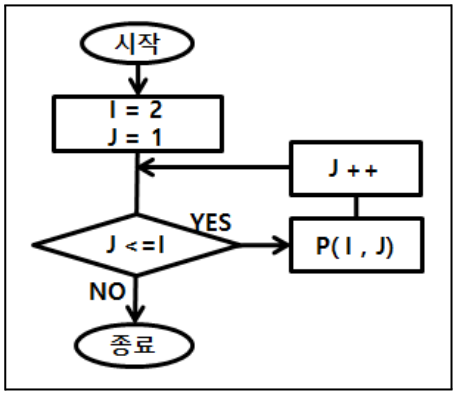


상위 첫번째 이미지와 두번째 이미지에서 반복문 밖의 i값이 반복문의 안쪽에서 출력 할때 영향을 주었다. 두번째 순서도 구조를 유지하면서 세번째 이미지 처럼 (1,1) 만 출력되도록 하려면 i값을 가지고 반복문의 조건식에 영향을 주어야 한다. 무조건 3번 반복되는 상태에서 i값이 j값보다 같거나 작을때만 반복 되도록 해야 한다. 그래서 조건식을 왼쪽 처럼 J<=i 과 같이 바꾸거나 J<i+1로 바꾸어 주면 된다. 이렇게 하면 조건식을 첫번째 돌때 j값이 1이고 i값은 1이므로 1<=1 비교가 true여서 반복 부분를 반복하고 두번째 돌때 j값이 2이고 i값은 여전이 1이여서 2<=1 비교가 false여서 반복문을 빠져나와 반복문이 종료 된다. 출력 결과는 (1, 1) 이다.



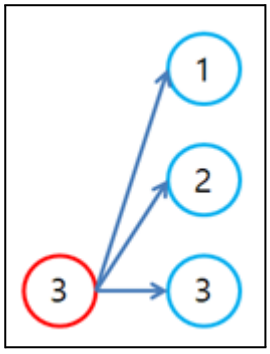


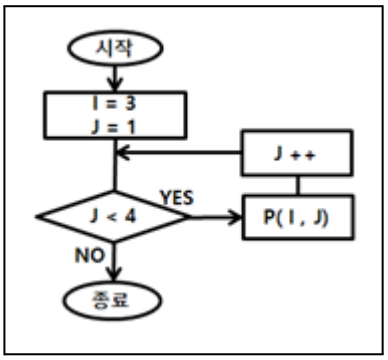


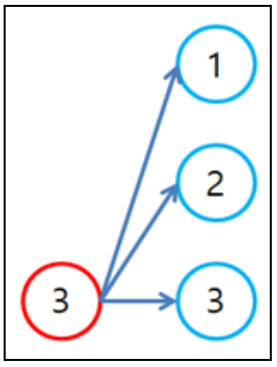


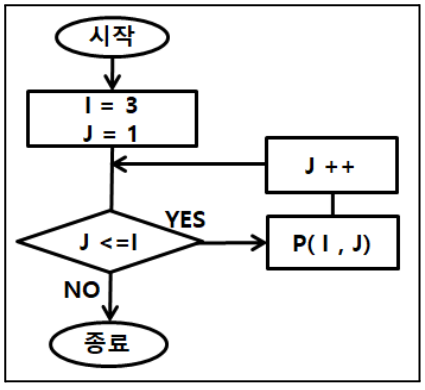
상위 첫번째 이미지와 두번째 이미지에서 반복문 밖의 i값이 반복문의 안쪽에서 출력 할때 영향을 주었다. 두번째 이미지 순서도 구조를 유지하면서 세번째 형태로 만들려면 i값을 가지고 반복문의 조건식에 영향을 주어야 한다. 무조건 3번 반복한 것을 i값에 따라서 반복 횟수를 줄여야 한다. 그래서 조건식을 왼쪽 처럼 J<i+1 나 J<=i로 바꾸어 주면 이전 빨간공이 1일때와 동일한 구조를 가지며 원하는 결과를 출력 할 수 있게 된다. 이렇게 하면 조건식을 첫번째 돌때 j값이 1이고 i값은 2이므로 1<=2 비교가 true여서 반복 부분을 반복하고 두번째 돌때 j값이 2이고 i값은 여전히 2이여서

2<=2 비교가 true여서 반복 부분을 반복하고 세번째 돌대 j값이 3이고 i값은 여전히 2여서 3<=2 비교가 false이므로 반복문을 빠져나와 반복문이 종료 된다.



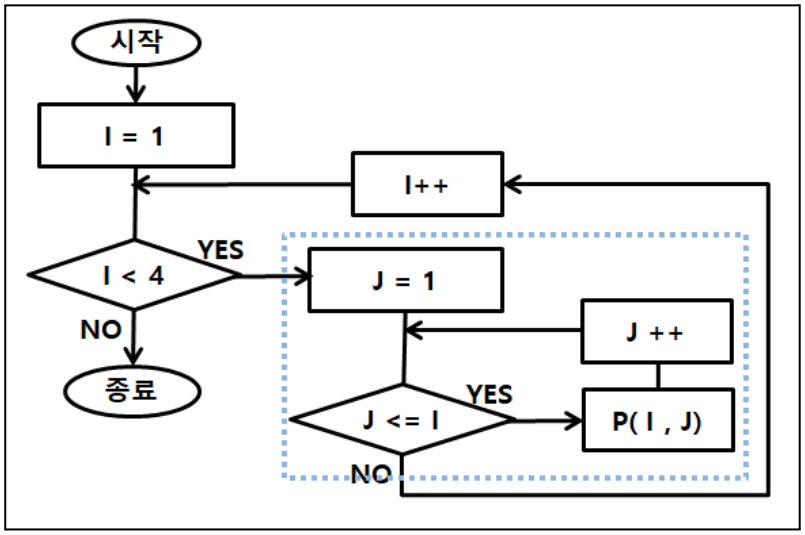






상위 첫번째 이미지와 세번째 이미지를 확인해 보면 동일하지만 네번째 이미지 순서도를 이용 해서 같은 결과를 얻을 수 있다.. 이렇게 하면 조건식을 첫번째 돌때 j값이 1이고 i값은 3이므로 1<=3 비교가 true여서 반복 부분을 반복하고 두번째 돌때 j값이 2이고 i값은 여전히 3이여서 2<=3 비교가 true여서 반복 부분을 반복하고 세번째 돌때 j값이 3이고 i값은 여전히 3이여서 비교식 3<=3의 결과가 true여서 반복 부분을 반복하고 네번째 돌대 j값이 4이고 i값은 여전히 3이여서 4<=3 비교 결과가 false이므로 반복문을 빠져나와 반복문이 종료 된다.

잘 생각해보면 상위 3개의 순서도는 i값이 1에서 하나씩 증가하여 3이 될 때까지 반복하는 것과 동일하여 다음과 같이 순서도를 변경할 수 있다.



왼쪽은 i값을 1, 2, 3씩 변경하면서 점선 부분을 반복하는 순서도이다.

1) 다음 문자열을 출력 할 수 있는 기찻길 순서도 의사 코드를 만들어 보자.

1. *엔터**엔터***엔터****엔터*****엔터******엔터*******엔터

2. *******엔터******엔터*****엔터****엔터***엔터**엔터*엔터

3. 공간공간공간공간공간공간*엔터공간공간공간공간공간**엔터공간공간공간공간**
*엔터공간공간공간****엔터공간공간*****엔터공간******엔터*******엔터

공간을 진짜 공간 키보드의 스페이스 ‘ ‘로 찍고 *은 *로 찍고 엔터는 실제 엔터

줄바꿈으로 표시하면 아래와 같은 삼각형 모양이 나올 것이다.

```
*          *******          *
**          ******          **
***          *****          ***
****          ****          ****
*****          ***          *****
******          **          ******
*******          *          *******
```

2)웹에서 자바 별찍기로 검색을 해서 다양함 별을 찍어 보자.

```
11111 12345 ******  *   *****  *****    *
22222 23456         **  ****   ****    ***
33333 34567 *       *** ***    ***    *****
44444 45678 *       **** **    **    *******
55555 56789 *       *****  *   *    *********
            *                  **    *******
12345 56789         *****  *   ***    *****
12345 45678 *****   ****   **  ****    ***
12345 34567 *****   ***    *** *****    *
12345 23456 *****   **     ****
12345 12345 *****   *      *****
            *****
```

다음 6,5,1,8,7,4,2,3을 값으로 가지는 배열이 아래와 같이 있을 때 배열의 내용을 1,2,3,4,5,6,7,8순으로 오름차 순으로 배열안의 내용을 바꾸어 정렬해 보자.

다음 버블정렬을 이해해 보자.

1. 배열의 인덱스 0과 1에 들어 있는 수를 비교하여 큰수를 배열의 인덱스 1쪽으로 교환하여 이동 시킨다.
2. 0이 작으면 교환되지 않는다.

```
int arr[]={6,5,1,8,7,4,2,3};
if(arr[0]>arr[1]){
      int temp;
      temp=arr[0];
      arr[0]=arr[1];
      arr[1]=temp;
}
```

3. 배열의 인덱스 1과 2에 들어 있는 수를 비교하여 큰수를 배열의 인덱스 2쪽으로 교환하여 이동 시킨다.

```
//{5,6,1,8,7,4,2,3};
if(arr[1]>arr[2]){
      int temp;
      temp=arr[1];
      arr[1]=arr[2];
      arr[1]=temp;
}
//{5,1,6,8,7,4,2,3};
```

4. 이작업을 배열이 끝날때 까지 반복한다.
5. 이 작업을 배열이 끝날때 까지 반복하면 배열안에 가장 큰 수가 맨 마지막에 정렬된다.

```
for(int i=0;i<arr.lenth-1;i++){
      if(arr[i]>arr[i+1]){
          int temp;
          temp=arr[i];
          arr[i]=arr[i+1];
          arr[i]=temp;
      }
}
```

6. 이 작업을 2번 하면 1번째에 가장큰 수가 배열의 맨 마지막으로 이동하고 2번째에 그다음 큰수가 배열의 뒤에서 2번째 위치로 이동 한다.

```
for(int i=0;i<arr.lenth-1;i++){
      if(arr[i]>arr[i+1]){
            int temp;
            temp=arr[i];
            arr[i]=arr[i+1];
            arr[i]=temp;
      }
```

```
}
for(int i=0;i<arr.lenth-1;i++){
	if(arr[i]>arr[i+1]){
		int temp;
		temp=arr[i];
		arr[i]=arr[i+1];
		arr[i]=temp;
	}
}
```

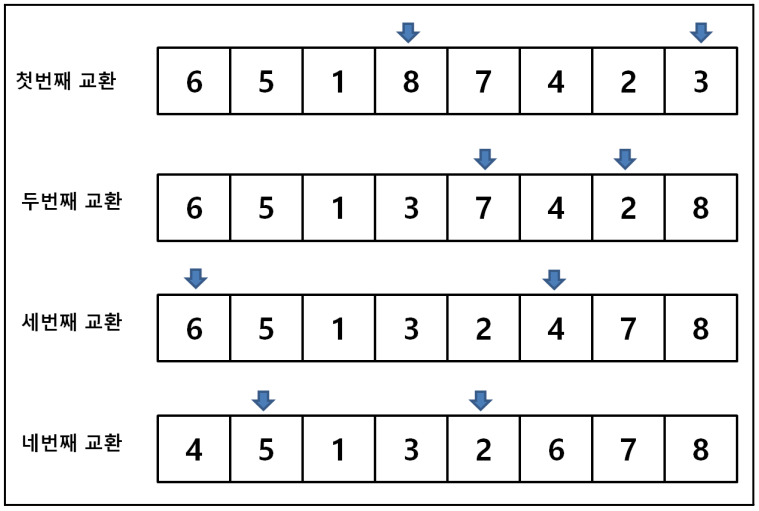
7. 반복문을 1번 돌때마다 1개의 데이터가 배열의 뒤쪽부터 정렬되므로 배열의 개수만큼 반복하면 배열의 모든 데이터가 정렬 된다.

```
for(int j=0;j<arr.lenth-1;j++){
	for(int i=0;i<arr.lenth-1;i++){
		if(arr[i]>arr[i+1]){
			int temp;
			temp=arr[i];
			arr[i]=arr[i+1];
			arr[i]=temp;
		}
	}
}
```

8. 내림차수로 정렬하고 싶다면 작은수를 뒷쪽으로 보내면 된다.

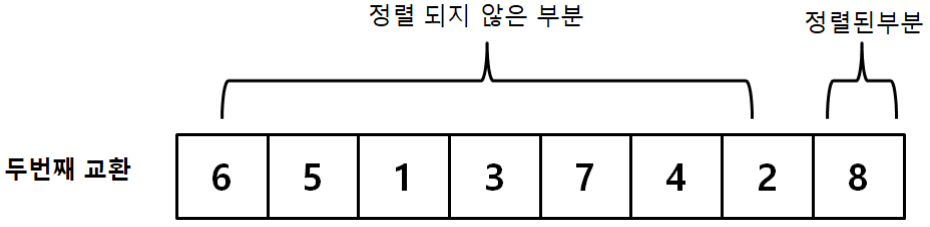
```
int [] arr= {7,5,9,0,3,1,6,2,4,8};
for(int i=0;i<arr.length;i++) {
	System.out.print(arr[i]+" ");
}
System.out.println();
for(int j=0;j<arr.length-1;j++) {
	for(int i=0;i<arr.length-2;i++) {
		if(arr[i] <arr[i+1]) {
			int temp=arr[i];
			arr[i]=arr[i+1];
			arr[i+1]=temp;
		}
	}
}
```

다음 선택정렬을 이해해 보자.



1. 배열안에서 가장 큰 수(첫번째 교환 부분의 첫 번째 화살표)를 찾고 배열의 마지막 위치(첫번째 교환 부분의 두 번째 화살표)와 두 값을 교환한다. 다음을 진행하고 나면 배열 안에 어떤 수가 들어 있더라 하더라도 가장 큰 수가 배열의 맨 마지막에 들어 간다.

2. 첫번째 교환이 이루어 지고 난 다음에는 배열의 맨마지막에 가장 큰 수가 들어가고 배열 안의 내용은 두번째 교환 부분과 같은 모양이 된다. 여기서 오름차순으로 배열 내용을 만든다고 생각해 보자. 맨 마지막 부분은 이미 정렬이 끝난 상태이므로더 이상 손을 댈 필요가 없다. 다음에 어떤 작업을 해할지 고민해보자.



3. 두번째 교환 부분에서 정렬 된 부분과 정렬되지 않은 부분을 구분한 다음 정렬된 부분은 더 이상 손대지 않으면 되고, 정렬되지 않은 부분중에서 가장 큰수 (두 번째 교환 부분의 첫 번째 화살표)를 찾아 정렬되지 않은 부분의 마지막 배열 부분(두 번째 교환 부분의 두 번째 화살표)의 값과 교환하면 세 번째 교환 부분 배열처럼 정렬이 된다.

4. 세번째 교환 부분에서 정렬 된 부분과 정렬되지 않은 부분을 구분한 다음 정렬된 부분은 더 이상 손대지 않으면 되고, 정렬되지 않은 부분중에서 가장 큰수 (세 번째 교환 부분의 첫 번째 화살표)를 찾아 정렬되지 않은 부분의 마지막 배열 부분(세 번째 교환 부분의 두 번째

화살표)의 값과 교환하면 네 번째 교환 부분 배열처럼 정렬이 된다. 이런 방법으로 계속 반복하다 보면 배열안의 모든 내용이 정렬될 것이다.

총 몇번 반복해야 정렬이 되는가?

3개라면 2개만 정렬되면 나머지 하나는 저절로 정렬이 된다. 따라서, 배열크기-1 한 만큼 반복하면 배열은 정렬이 된다. 이후 부터 배열의 크기를 배열이름.length로 표기 한다.

상위 내용을 정리해 보면 정렬하기 위해서 다음과 같은 과정을 거치면 된다.

0~7 까지 8개 중에 가장 큰수를 찾아 배열 인덱스 7에들어 있는 수와 교환해야 한다.

0~6 까지 7개 중에 가장 큰수를 찾아 배열 인덱스 6에들어 있는 수와 교환해야 한다.

0~5 까지 6개 중에 가장 큰수를 찾아 배열 인덱스 5에들어 있는 수와 교환해야 한다.

0~4 까지 5개 중에 가장 큰수를 찾아 배열 인덱스 4에들어 있는 수와 교환해야 한다.

0~3 까지 4개 중에 가장 큰수를 찾아 배열 인덱스 3에들어 있는 수와 교환해야 한다.

0~2 까지 3개 중에 가장 큰수를 찾아 배열 인덱스 2에들어 있는 수와 교환해야 한다.

0~1 까지 2개 중에 가장 큰수를 찾아 배열 인덱스 1에들어 있는 수와 교환해야 한다.

반복되는 숫자를 확인해 보면 반복문으로 바꿀수 있을 것이다.

> 08. 다양한 예제

문제 1) 1~50까지의 짝수를 출력하는 코드를 만들어 보자.

문제 2) 1~100사이의 10의 배수을 출력하는 코드를 만들어 보자. 어떤 숫자가 배수 인지 아닌지 알고 싶으면 7로 나눈 나머지가 0이면 7의 배수이고 아니면 7의 배수가 아니다.

문제 3) 30~300까지의 6의 배수의 합을 출력하는 코드를 만들어 보자.

문제 4) 숫자를 하나 입력 받아 1부터 입력한 수까지 순서대로 화면에 출력 되도록 코드를 만들어 보자.

문제 5) 사용자에게 두 수를 입력받아 두 수의 사이에 있는 모든 수를 오름차 순으로 출력하는 순서도와 프로그램을 만들어 보자. 예)5 9 입력시 6 7 8 더한 결과를 얻음

문제 6) 두수를 입력 받아 사이에 있는 짝수를 화면에 오름차 순으로 출력 되도록 순서도와 프로그램을 만들어 보자.

문제 7) 1-2+3-4+5-6……+99-100의 결과를 구하는 프로그램을 작성해 보자.

문제 8) 1/2+2/3+3/4+4/5+ 5/6+6/7+....+99/100의 결과를 구하는 프로그램을 작성해 보자.

문제 9) 피보나치 수열을 10개를 순서대로 출력하는 프로그램을 작성해 보자. 피보나치 수열이 무엇인지는 웹사이트를 검색해서 스스로 알아보자.

문제 10) 원하는 색의 전구와 밝기를 입력 받아 처리하는 프로그램을 만들어 보자. 사용자가 원하는 밝기와 색상을 입력받아 원하는 결과를 출력 받자. 아래 설명한 내용대로 운영 되도록 기술 하자.

초기 변수값 : color=”빨강” brightness =50

값의 범위 : color=빨강,노랑,파랑 brightness는 0~100

제안사항 : brightness의 숫자 변경은 1씩 가능하다. brightness값이 10이라면 다음 brightness값은 11 이나 9 만 가능하다. 10을 50으로 변경하려면 반복문을 사용해야 한다.

사용자입력변수 : colorInput, brightnessInput

최종결과값출력 : p(“현재 색상은”+color+”밝기는”+brightness+”이다”);

문제 11) 1-2+3-4+5-6……+99-100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

문제 12) 1/2+2/3+3/4+4/5+ 5/6+6/7+....+99/100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

문제 13) 다음과 같이 출력 되도록 프로그램을 완성해 보자.

1 2 3 4 5

10 9 8 7 6

11 12 13 14 15

20 19 18 17 16

21 22 23 24 25

문제 14) 해당 달의 시작 요일과 일수를 입력 받아 달력을 출력해 보자.\t 탭을 이용해서 만들어 보자.

문제 15) 배열 1,2,3,4,5,6,7,8,9 에서 이동방향, 이동칸수, 채울수자를 입력 받아 배열의 내용을 변경후 출력해보자.

ex)입력 왼쪽 3 2 결과 4,5,6,7,8,9,2,2,2

ex)입력 오른쪽 3 4 결과 4,4,4,1,2,3,4,5,6

문제 16)배열 1,2,3,4,5,6,7,8,9 에서 회전방향과 회수를 입력받아 배열 내용을 회전시키고 출력해보자.

ex)입력 왼쪽 2 결과 3,4,5,6,7,8,9,1,2

ex)입력 오른쪽 3 결과 7,8,9,1,2,3,4,5,6

문제 17)배열을 100개 선언하여 0~99까지 넣은 다음 i=2 부터 50까지 i를 제외한 i의 배수와 같은 인덱스에 0를 넣은 다음 배열에 0이 아닌 수를 출력해 보자. 출력 결과가 모두 소수인데 이유를 생각해 보자.

ex)i가 2이면 2를 제외한 2의 배수는 4,6,8,10,12,14,16… 등이 있고 해당 인덱스에 0를 넣으면된다.

ex)i가 3이면 3를 제외한 3의 배수는 6,9,12,15,18… 등이 있고 해당 인덱스에 0를 넣으면된다.

문제 18) 17번에서 배열의 값이 0이 아닌 수를 출력해보면 나오는 결과물이 소수이다. 소수란 1과 자기 자신만으로 나눠지는 수를 의미한다.

```java
double sum=0;
for(int i=1;i<100;i++) {
	sum+=(double)i/(i+1);
}
System.out.println(sum);

String color="빨강";
int brightness=50;

String colorInput="노랑";
int brightnessInput=70;

color=colorInput;
while(brightness!=brightnessInput) {
	if(brightness>brightnessInput) {
		brightness--;
	}else {
		brightness++;
	}
}
System.out.println("현재 색상은"+color+"밝기는"+brightness+"이다");


//1   1   2   3   5   8
int preValue=1;
int curValue=1;
System.out.println(preValue);
for(int i=0;i<9;i++) {
	System.out.println(curValue);
	int temp=curValue;
	curValue=preValue+curValue;
	preValue=temp;
}
```

> 09. 2중 메뉴

메뉴 아래 하위메뉴가 존재할때 처리방법

프로그램 설명:

1. **메인 메뉴**:
    - 사용자는 커피 메뉴(1) 또는 디저트 메뉴(2)를 선택할 수 있습니다.
    - `0`을 입력하면 프로그램이 종료됩니다.
2. **커피 메뉴**:
    - 아메리카노, 카페라떼, 카푸치노, 에스프레소 중 선택할 수 있습니다.
    - `0`을 입력하면 메인 메뉴로 돌아갑니다.
3. **디저트 메뉴**:
    - 케이크, 마카롱, 쿠키 중 선택할 수 있습니다.
    - 각 디저트는 하위 메뉴를 가지고 있습니다.
    - `0`을 입력하면 메인 메뉴로 돌아갑니다.
4. **하위 메뉴**:
    - 케이크, 마카롱, 쿠키의 하위 메뉴에서 추가 옵션을 선택할 수 있습니다.
    - `0`을 입력하면 디저트 메뉴로 돌아갑니다.

실행화면

```
=== 메인 메뉴 ===

1. 커피 메뉴

2. 디저트 메뉴

0. 종료

메뉴를 선택하세요: 2


=== 디저트 메뉴 ===

1. 케이크
```

2. 마카롱

3. 쿠키

0. 메인 메뉴로 돌아가기

디저트 메뉴를 선택하세요: 1

케이크를 선택하셨습니다. 가격은 5000원입니다.

=== 케이크 하위 메뉴 ===

1. 초코케이크

2. 치즈케이크

3. 당근케이크

0. 디저트 메뉴로 돌아가기

케이크 종류를 선택하세요: 1

초코케이크를 선택하셨습니다. 추가 가격은 1000원입니다.

```
package com.the.ex;

import java.util.Scanner;

public class CoffeeAndDessertMenu {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String inputString = "";

        while (!inputString.equals("0")) {
            System.out.println("=== 메인 메뉴 ===");
            System.out.println("1. 커피 메뉴");
            System.out.println("2. 디저트 메뉴");
            System.out.println("0. 종료");
            System.out.print("메뉴를 선택하세요: ");

            inputString = scanner.nextLine();

            switch (inputString) {
                case "1":
                    coffeeMenu(scanner);
                    break;
                case "2":
                    dessertMenu(scanner);
                    break;
                case "0":
```

```
                System.out.println("프로그램을 종료합니다.");
                break;
            default:
                System.out.println("잘못된 입력입니다. 다시 선택해주세요.");
        }
        System.out.println(); // 줄바꿈
    }

    scanner.close();
    System.out.println("프로그램이 종료되었습니다.");
}

// 커피 메뉴 처리
private static void coffeeMenu(Scanner scanner) {
    String inputString = "";

    while (!inputString.equals("0")) {
        System.out.println("=== 커피 메뉴 ===");
        System.out.println("1. 아메리카노");
        System.out.println("2. 카페라떼");
        System.out.println("3. 카푸치노");
        System.out.println("4. 에스프레소");
        System.out.println("0. 메인 메뉴로 돌아가기");
        System.out.print("커피 메뉴를 선택하세요: ");

        inputString = scanner.nextLine();
```

```
            switch (inputString) {
                case "1":
                    System.out.println("아메리카노를 선택하셨습니다. 가격은
3000원입니다.");
                    break;
                case "2":
                    System.out.println("카페라떼를 선택하셨습니다. 가격은
3500원입니다.");
                    break;
                case "3":
                    System.out.println("카푸치노를 선택하셨습니다. 가격은
4000원입니다.");
                    break;
                case "4":
                    System.out.println("에스프레소를 선택하셨습니다. 가격은
2500원입니다.");
                    break;
                case "0":
                    System.out.println("메인 메뉴로 돌아갑니다.");
                    break;
                default:
                    System.out.println("잘못된 입력입니다. 다시 선택해주세요.");
            }
            System.out.println(); // 줄바꿈
        }
    }
```

```
    // 디저트 메뉴 처리
    private static void dessertMenu(Scanner scanner) {
        String inputString = "";

        while (!inputString.equals("0")) {
            System.out.println("=== 디저트 메뉴 ===");
            System.out.println("1. 케이크");
            System.out.println("2. 마카롱");
            System.out.println("3. 쿠키");
            System.out.println("0. 메인 메뉴로 돌아가기");
            System.out.print("디저트 메뉴를 선택하세요: ");

            inputString = scanner.nextLine();

            switch (inputString) {
                case "1":
                    System.out.println("케이크를 선택하셨습니다. 가격은
5000원입니다.");
                    cakeSubMenu(scanner);
                    break;
                case "2":
                    System.out.println("마카롱을 선택하셨습니다. 가격은
3000원입니다.");
                    macaronSubMenu(scanner);
                    break;
                case "3":
                    System.out.println("쿠키를 선택하셨습니다. 가격은
2000원입니다.");
```

```
                cookieSubMenu(scanner);
                break;
            case "0":
                System.out.println("메인 메뉴로 돌아갑니다.");
                break;
            default:
                System.out.println("잘못된 입력입니다. 다시 선택해주세요.");
        }
        System.out.println(); // 줄바꿈
    }
}

// 케이크 하위 메뉴
private static void cakeSubMenu(Scanner scanner) {
    String inputString = "";

    while (!inputString.equals("0")) {
        System.out.println("=== 케이크 하위 메뉴 ===");
        System.out.println("1. 초코케이크");
        System.out.println("2. 치즈케이크");
        System.out.println("3. 당근케이크");
        System.out.println("0. 디저트 메뉴로 돌아가기");
        System.out.print("케이크 종류를 선택하세요: ");

        inputString = scanner.nextLine();
```

```
            switch (inputString) {
                case "1":
                    System.out.println("초코케이크를 선택하셨습니다. 추가 가격은
1000원입니다.");
                    break;
                case "2":
                    System.out.println("치즈케이크를 선택하셨습니다. 추가 가격은
1500원입니다.");
                    break;
                case "3":
                    System.out.println("당근케이크를 선택하셨습니다. 추가 가격은
1200원입니다.");
                    break;
                case "0":
                    System.out.println("디저트 메뉴로 돌아갑니다.");
                    break;
                default:
                    System.out.println("잘못된 입력입니다. 다시 선택해주세요.");
            }
            System.out.println(); // 줄바꿈
        }
    }

    // 마카롱 하위 메뉴
    private static void macaronSubMenu(Scanner scanner) {
        String inputString = "";

        while (!inputString.equals("0")) {
```

```
            System.out.println("=== 마카롱 하위 메뉴 ===");
            System.out.println("1. 딸기 마카롱");
            System.out.println("2. 초코 마카롱");
            System.out.println("3. 바닐라 마카롱");
            System.out.println("0. 디저트 메뉴로 돌아가기");
            System.out.print("마카롱 종류를 선택하세요: ");

            inputString = scanner.nextLine();

            switch (inputString) {
                case "1":
                    System.out.println("딸기 마카롱을 선택하셨습니다. 추가 가격은
500원입니다.");
                    break;
                case "2":
                    System.out.println("초코 마카롱을 선택하셨습니다. 추가 가격은
500원입니다.");
                    break;
                case "3":
                    System.out.println("바닐라 마카롱을 선택하셨습니다. 추가 가격은
500원입니다.");
                    break;
                case "0":
                    System.out.println("디저트 메뉴로 돌아갑니다.");
                    break;
                default:
                    System.out.println("잘못된 입력입니다. 다시 선택해주세요.");
            }
```

```
            System.out.println(); // 줄바꿈
        }
    }

    // 쿠키 하위 메뉴
    private static void cookieSubMenu(Scanner scanner) {
        String inputString = "";

        while (!inputString.equals("0")) {
            System.out.println("=== 쿠키 하위 메뉴 ===");
            System.out.println("1. 초코칩 쿠키");
            System.out.println("2. 오트밀 쿠키");
            System.out.println("3. 버터 쿠키");
            System.out.println("0. 디저트 메뉴로 돌아가기");
            System.out.print("쿠키 종류를 선택하세요: ");

            inputString = scanner.nextLine();

            switch (inputString) {
                case "1":
                    System.out.println("초코칩 쿠키를 선택하셨습니다. 추가 가격은
300원입니다.");
                    break;
                case "2":
                    System.out.println("오트밀 쿠키를 선택하셨습니다. 추가 가격은
400원입니다.");
                    break;
```

```
                case "3":
                    System.out.println("버터 쿠키를 선택하셨습니다. 추가 가격은
350원입니다.");
                    break;
                case "0":
                    System.out.println("디저트 메뉴로 돌아갑니다.");
                    break;
                default:
                    System.out.println("잘못된 입력입니다. 다시 선택해주세요.");
            }
            System.out.println(); // 줄바꿈
        }
    }
}
```