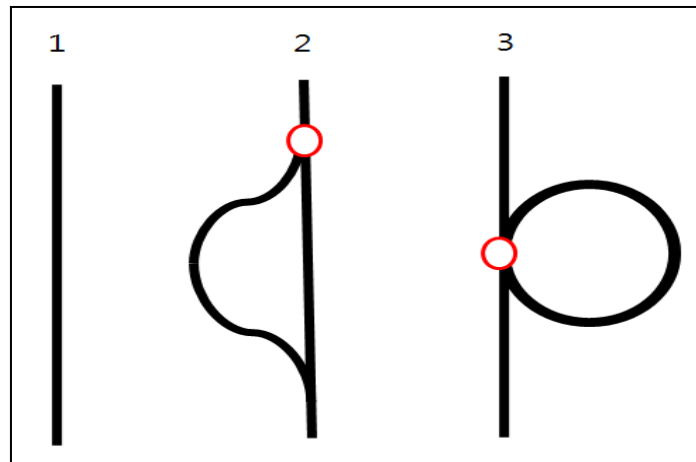


## 03. 제어문

### > 04. 프로그램의 3요소

컴퓨터 프로그램은 크게 3가지로 구성되어 있는데 ‘절차문 반복문 제어문’ 이다. 이 3요소는 프로그램 실행 순서와 관계가 있다. 실행 순서를 이야기 할 때 철도를 이야기 하는 경우가 많은데 기차가 철도를 따라 순서대로 진행되기 때문이다. 프로그램도 위에서 아래로 순서대로 진행된다. 프로그램 3요소를 철도 그림으로 그리면 아래와 같이 되는데 1번이 절차문 2번이 조건문 3번이 반복문에 해당 한다. 다음 3개의 기차길의 차이를 설명해 보자.



#### 1번 절차문

기차가 라인을 따라서 순서대로 움직이듯 실행 코드도 사람이 책을 읽을 때 처럼 위에서 아래로 순서대로 실행 된다.

냉장고에 코끼리 넣는 방법을 3단계로 생각해 보자. 1단계 냉장고 문을 연다. 2단계 코끼리 넣는다 3단계 냉장고 문 닫는다. 순서에 따라 순서대로 진행하면 냉장고에 코끼리를 넣을 수 있다. 간단해 보이지만 초임들은 순서없이 해석 하려 할 수 있다. 집중해서 한줄 한줄 순서대로 코드를 확인하는 습관을 가지자.

냉장고 안에 기린 넣는 방법을 4단계로 생각해 보자. 1단계 냉장고 문을 연다 2단계 코끼리를 뺀다. 3단계 기린을 넣는다 4단계 냉장고 문을 닫는다. 이전 이야기가 종료된 것이 아니어서 이전 처럼 냉장고 문 열고 기린 넣고 냉장고 문 닫는 방법으로는 해결이 안 된다.

프로그램도 종료되기 전에 실행한 결과가 이후에 계속 영향을 준다. 당연한 이야기 처럼 들리 겠지만 초보자들은 익숙하지 않아서 쉽게 잊어 버리고 코끼리를 꺼내지 않고 기린을

집어 넣으려고 하는 경우가 자주 발생 한다. 냉장고 안에 들어 있는 데이터가 중요도가 있다면 냉장고 안에 무엇이 들어 있는지 항상 파악하고 있어야 한다. 초보자들은 순서대로 프로그램을 진행하면서 냉장고에 뭐가 들어 있는지 인지 하고 있어야 한다.

이야기를 이어서 문제를 하나 내 보겠다. 사자가 생일 파티를 열어서 모든 동물한테 반드시 참여 하라고 초대장을 보냈는데 한 동물이 참여하지 않았다. 참여 하지 않은 동물은 어떤 동물일까? 정답은 기린이다. 이유는 냉장고에 갇혀 있어서다. 프로그램이 종료되지 않는 한 지금 해 준 이야기 처럼 계속 연결 된다. 프로그램 실행중 가지고 있던 데이터들은 지속적으로 유지, 보관, 변경 된다. 초보자들은 프로그램이 순서대로 실행 된다는 것을 알면서도 중간 중간 건너 뛰어서 잘못된 코드 분석을 하는 경우가 많다. 프로그램은 위에서 아래서 순서대로 진행된다는 사실을 잊지 말자.

다시 한번 이야기하지만 프로그램은 순서대로 실행이 되고 데이터의 변화도 순서대로 변경되어 프로그램이 종료되기 전까지 계속된다. 아무리 긴 코드를 만나도 당황하지 말고 책을 읽듯이 순서대로 한 줄 한 줄씩 코드를 보면서 데이터가 어떻게 변화 되는지 확인 하자.

프로그램 언어도 결국에는 언어 여서 한 줄 한 줄에 의미가 있다. 따라서, 코드 한 줄 한 줄의 의미를 반드시 이해하고 넘어가야 한다. 한 줄 한 줄 천천히 책을 읽듯이 읽어 나가면 언젠가는 코드를 이해 할 수 있을 것이다.

복잡한 상황이어도 냉장고의 상태를 항상 파악하고 있어야 하듯 프로그램 코드도 실행 순서에 따라 한줄 한줄 해석해 나가면 아무리 복잡한 프로그램이라도 해석 할 수 있을 것이다.

## 2번 조건문

기차길에서 둘 중에 하나의 길을 선택해야 할 경우 조건문에 해당한다. 한쪽 길을 선택해서 해당 길로 이동하면 다른쪽 길은 갈수 없는 길이 된다.

상위 이미지에서 중간에 있는 빨간색 원 모양이 노선 변경 스위치이다. 노선 변경 스위치는 스위치가 왼쪽을 가리키고 있다면 기차가 왼쪽으로 이동 하고 스위치가 오른쪽을 가리키고 있다면 기차는 오른쪽으로 이동한다. 스위치 선택을 통해서 특정 노선으로 이동하게 하거나 이동하지 못하게 할 수 있다. 스위치는 기차가 오른 쪽으로 갈 것인지 왼쪽으로 갈 것인지 결정할 수있다. 기차가 한쪽으로 이동하게 되면 반대쪽 기차길은 지나갈 수 없는 기차길이 된다. 둘중 하나의 기차길만 이동한다.

프로그램에서 조건식은 true나 false를 생성하고 true이면 왼쪽 false이면 오른쪽 이런 식으로 특정 코드 블록을 실행시키는 기차길에서 스위치 역할을 한다.

프로그램에서 조건문은 if문 switch 문으로 사용하여 표현한다. if 나 switch문을 사용하여 위에서 아래로만 실행되는 코드 진행을 특정 코드 부분만 실행 하거나 실행 하지 않도록 할 수 있다. if,switch 라는 용어를 기억해 두자.

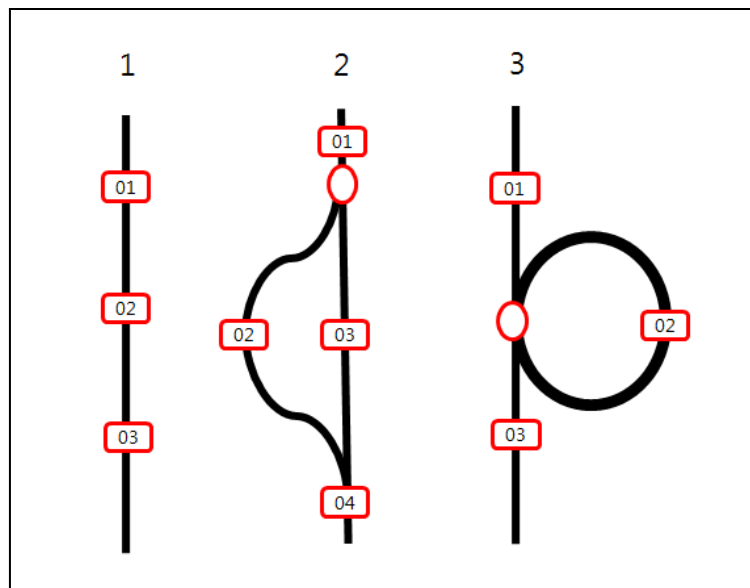
### 3번 반복문

반복문은 특정 기차길을 반복적으로 이동 하고자 할 때 사용 한다. 반복문에서 중간의 빨간 원은 노선 변경 스위치이고 2번 조건문과의 차이는 조건문은 왼쪽으로 갈 것인지 오른쪽으로 갈 것인지 결정 할때 스위치를 사용 하지만 3번 반복문은 스위치를 기준으로 오른쪽 원 부분을 반복할 것인지 말 것 인지를 결정한다. 결정적으로 반복문은 스위치로 돌아오고 조건문은 그렇지 않다. 원형태의 기차길은 상황에 따라서 기차가 여러번 돌 수도 있지만 한번 또는 돌지 않을 수도 있다. 반복문은 조건문과 마찬가지로 스위치의 온(true), 오프(false) 처럼 반복 여부를 true false값으로 결정 할 수 있고, true false 값을 가지는 값을 boolean값 이라고 한다. 결과적으로 결정된 값이 true면 특정 부분을 반복하고 false 이면 반복문을 빠져 나가게 된다. 프로그램에서 반복문은 for 또는 while 문, do~while 문을 사용하여 표현 한다.

그림을 보고 검정색 라인이 철도 길이라고 생각 하고 기차가 각각의 철도 길에서 위에서 아래로 이동 했을 경우 어떤 숫자를 지나왔는지 기술해 보자.

프로그래머들은 결과를 화면에 보여줄때 출력 한다고 이야기 한다.

기차가 지나온 순서대로 화면에 기차길의 숫자를 출력해 보자. 빨간원 부분 스위치는 상황에 따라 왼쪽이나 오른쪽 둘중에 하나를 선택해서 이동한다. 모든 경우를 고려해 기차길에서 발생 할 수 있는 모든 숫자를 출력 경우를 생각해 보자.



그림을 보고 만약 사각형 안의 숫자가 화면에 출력 된다면 시작 부분은 위 부분 01이고 아래 부분 끝까지 이동하면 종료라 생각 할 때 각각 화면에 출력되는 숫자를 기술 해보자.

1번의 경우 01,02,03이 출력 되고 다르게 출력이 되는 경우는 없을 것이다. 그래서, 항상 순서대로 실행된다 하여 절차문이라 한다.

2번의 경우 빨간점의 스위치에 따라서 01,02,04 혹은 01,03,04 가 출력 될 것이다. 상황에 따라 2가지 경우중 하나가 선택이 되어 조건문이라고 한다. 프로그램에서 조건문을

사용할때 if문이나 switch문으로 사용된다.

3번의 경우 빨간점의 스위치에 따라 01,03 혹은 01,02,03 혹은 01,02,02,03 혹은 01,02,02, .. 02, 03과 같이 02 부분이 여러번 반복될수 있는 결과를 얻게 될것이다. 02 부분이 반복되고 있어서 이부분을 반복문이라고 한다. 반복문의 경우 반복 하고 있는 도중 스위치를 변경하지 않으면 02부분을 무한히 반복하게 되는데 이럴 경우를 무한루프 라고 한다. 무한루프가 발생하지 않도록 02 반복문에서 스위치를 변경하는 프로그램을 구현 하여야 한다. 반복문을 프로그램에서 사용 할때 for문이나 while문, do while문으로 사용 한다.

다음 문제를 풀어 보자. 02 부분이 안녕을 출력하는 부분 이라면 기차길 1,2,3번에서 몇 번씩 출력되는지 기술해 보자.

02부분이 화면에 안녕이 출력되는 부분이라고 생각해 보면 1번의 경우에 반드시 한 번만 출력이 되고 2번의 경우 상황에 따라서 한번도 출력 되지 않을 수 있고 한 번만 출력 될 수도 있다. 3번의 경우 상황에 따라서 한번도 출력되지 않을 수 있고, 한번 출력 될 수도 있고, 2번 출력 될 수도 있고, 3번 출력 될 수도 있고 여러번 출력 될 수도 있다.

2번 조건문이나 3번 반복문에서 동그라미 원모양 스위치는 프로그램에서  $4>3$ ,  $4==3$ ,  $4<3$ 과 같은 비교 연산자가 사용되고 이들을 조건식 이라고 이야기한다. 해당 조건식 실행 결과에 따라 true인지 false 인지가 선택되고 이에 따라 진행 방향이 결정된다.

프로그램언어에서 반복문으로 사용되는 While문, for문, do~while문과 조건문으로 사용되는 if에 true, false를 생성하는 조건식을 사용한다.

지금 까지 설명한 3가지 문법을 조합하여 컴퓨터에서 돌아가는 모든 프로그램을 만들 수 있다. 프로그램을 만들다가 문제가 발생하면 알고리즘을 사용하여 문제를 해결할 수 있다. 알고리즘에 대해서 알아보자.

알고리즘이란? 어떤 문제를 해결하는 방법을 알고리즘 이라고 한다. 예) 라면을 먹어야 한다면, 슈퍼에가서 라면을 사서 집 부엌에 물을 올리고 라면을 넣어 끓인후 다익으면 먹는다. 여기서 ‘라면을 먹어야 한다면’이 해결 해야 할 문제라면 라면을 먹기 까지의 과정들이 알고리즘에 해당 한다.

프로그램 알고리즘이란? 프로그램에서 주어진 문제를 프로그램을 구현하여 해결하는 것을 의미한다.

컴퓨터 프로그램이란? 현실 세계의 데이터 정보를 컴퓨터에 넣고 가공 처리 하여 사용자가 원하는 결과로 출력하는 것을 의미 한다. 본인이 물건을 구매하는 웹사이트를 생각해 보자. 사용자로 부터 물건 정보를 입력 받아 저장시켜 놓은 다음에 다른 사용자가 원하는 형태로 저장되어 있는 상품을 보여준다. 대부분의 프로그램의 목적이 이와 같은 정보 처리이다. 여기서 중요한 것이 정보와 처리 부분이다.  $3 + 5$  라는 계산에서 3, + , 5 각각은 정보에 해당하고 계산한 결과 8은 처리된 결과에 해당 한다.

현실세계의 데이터를 컴퓨터에서 다룰때 컴퓨터 자료형 에 넣어 사용하고 데이터 양이

많아지면 데이터 베이스로 관리 한다. 나중에 데이터베이스도 공부해 보자.

처리는 프로그램 언어로 사용자가 원하는 데로 결과를 보여주는 작업을 해준다. 이때 원하는 결과 대로 보여 주기 위해서는 다양한 방법이 존재 하는데 이때, 어떤 방법을 사용하여 처리하는 것이 좋을지 고민하게 될 것이고 이 문제를 해결하는 과정을 알고리즘이라고 한다. 알고리즘이란? 문제를 해결하는 방법이다.

컴퓨터 프로그램은 크게 3가지로 구성되어 있다. 절차문, 조건문, 반복문 절차문은 코드가 기술된 순서대로 진행 한다, 조건문은 둘중에 하나을 선택해서 하나만 실행 한다, 반복문은 같은 작업을 반복한다. 놀랍게도 상위 3가지 방법을 사용하여 컴퓨터에서 발생하는 모든 문제를 해결 할 수 있다. 결론은 컴퓨터 알고리즘은 절차문, 조건문, 반복문 3가지로 컴퓨터에서 발생한 모든 문제를 해결 할 수 있다.

#### 연습문제

1. 이전 3개의 기차길의 차이점을 설명하시오.
2. 프로그램 3요소는 무엇이고 프로그램에 어떻게 사용되는지, 프로그램에서 키워드는 뭔지 설명하시오.
3. 불리언 자료형이란?
4. 조건식의 실행결과는 어떻게 되는가?

## > 01. 알고리즘과 순서도

컴퓨터에서 알고리즘을 직접 코드로 작성해서 구현 할 수 있지만 복잡한 알고리즘을 구현할 때 단순히 코드로 작성하기 보다는 그림 형태로 표현하는 순서도(flow chart)로 만드는 것이 중요 하다.

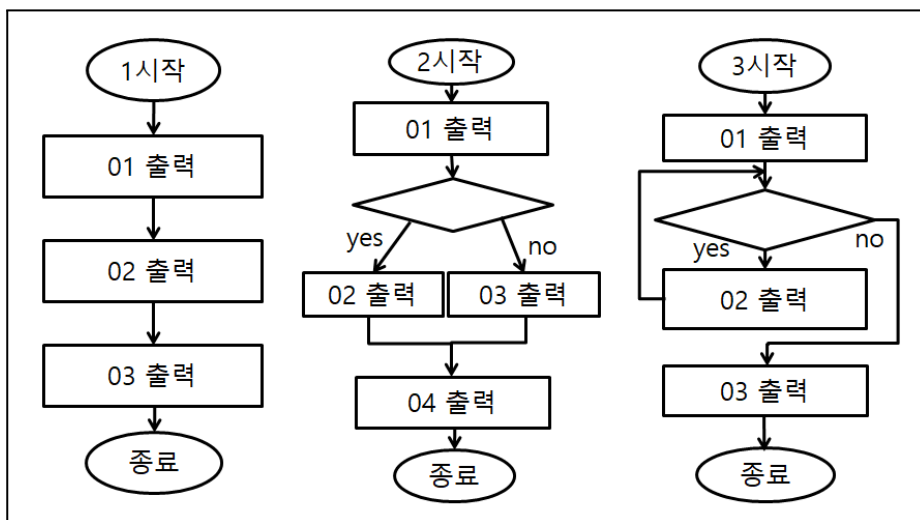
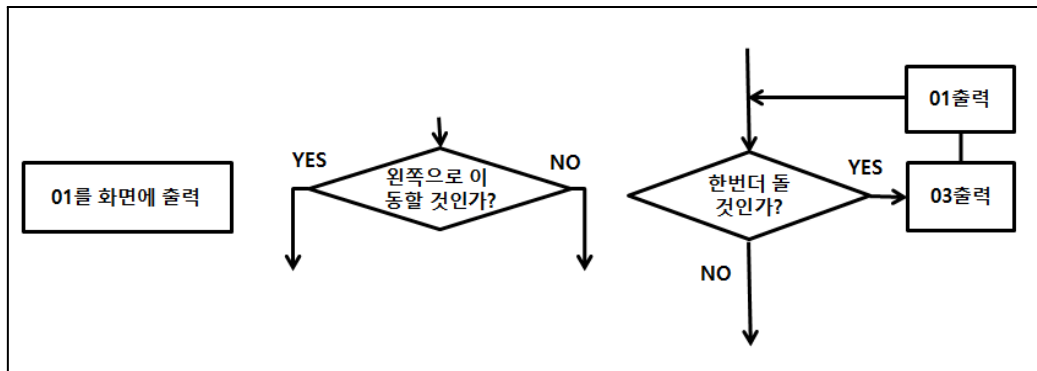
순서도란? 어떤 결과를 얻기 위한 과정(알고리즘)을 그림으로 표현한 것이다. 여기에서는 프로그램 알고리즘을 그림으로 표현 하는데 사용 된다. 영어로 flow chart라고 한다.

웹에 순서도와 flowchart를 검색해서 확인해 보자. 웹에 있는 이미지 몇개만 보면 어느 정도 사용 방법이 이해가 될 것이다.

순서도는 기본적으로 위에서 아래로 순서대로 진행된다. 절차적이다. 질문에 이르면 선택에 의해서 두 방향중 하나의 방향으로 진행된다. 조건에 해당한다. 순서 도를 따라가다 보면 반복적으로 특정 내용이 실행 된다. 반복문이다. 순서도는 컴퓨터 프로그램에서 사용되는 절차문, 조건문, 반복문 3가지를 그림으로 표현하기 좋다.

상위 3개의 기차길을 순서도로 그리면 다음과 같은 모양을 가진다.



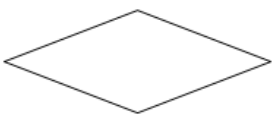
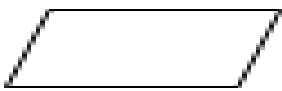
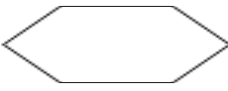


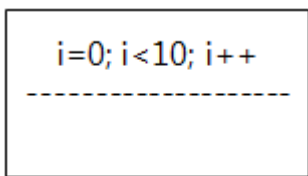
아래 맨 왼쪽 이미지 사각형은 사각형안의 내용이 실행됨을 의미하고 다음 이미지는 조건문으로 마름모 조건 결과가 true인지 false인지 에 따라 실행되는 방향이 달라지고, 마지막 이미지는 조건에 따라 특정 부분이 반복된다.



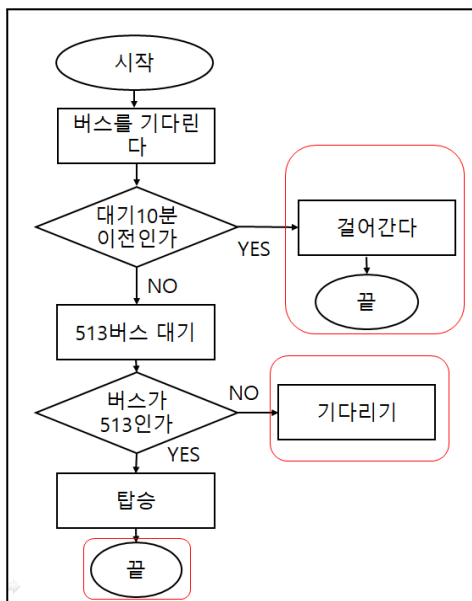
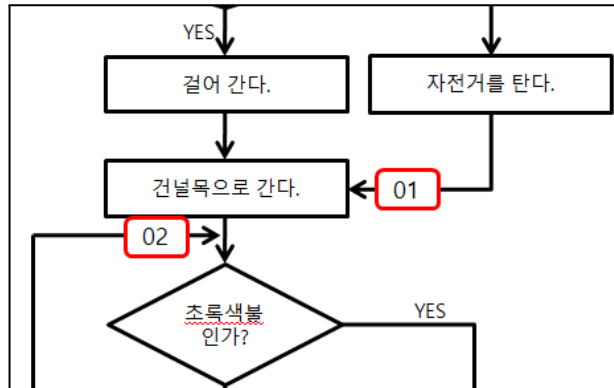
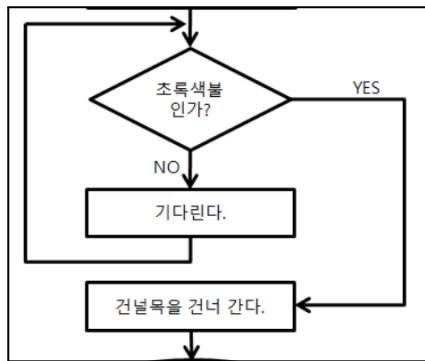
이전 그림 기차길 3개에 기차가 지나갈 때 기차길에 써있는 숫자가 출력된다고 생각했을 때 동일한 형태로 숫자가 출력되는 순서도를 작성해보면 왼쪽 이미지와 같다.

조건문과 반복문은 비슷한 부분이 많다 좀 더 자세히 구분해 보면 “조건문”은 true, false로 갈라진 두 길이 언젠가 만나서 종료 쪽으로 향하는 구조이고, “반복문”은 한부분은 종료 다른 한 부분은 다시 조건문으로 돌아오는 구조로 되어 있어 특정 부분이 반복된다. 일반적으로 반복문은 true일 때 반복하게 만들어야 프로그램으로 구현하기 쉽다.

복잡한 알고리즘을 순서도를 이용하여 그림으로 표현한다고 이야기하였다. 다음은 순서도에서 어떤 기호를 사용하여 그리는지 다음 표를 확인해 보자.

기 호	이름	의 미
	시작, 끝	순서도의 시작과 끝을 나타낸다. 보통 원안에 시작이나 끝이라고 기술해서 만든다.
	처리	각종 연산, 데이터 처리, 변수의 값변경 등의 작업을 하는데 사용한다. 박스안에 처리 내용을 기술한다. 기차가 01를 지나간다. 기차가 02를 지나간다.
	판단	조건 비교 판단하여 상황에 따라 로직의 흐름을 변경함 보통 true 나 false가 리턴될 수 있는 비교 연산자가 기술됨
	입출력	사용자의 데이터의 입력과 출력이 필요할때 사용한다. 보통 처리에 상황을 기술하여 기술한 내용으로 대처 해서 사용한다.
	준비	데이터 저장 공간을 확보하기 위해 준비 할때 사용한다. 보통 처리로 대처 해서 사용한다.
	서류	종이 같은 실제 출력 최종결과물을 표현한다. 보통 처리로 대처 해서 사용한다.
	흐름선	순서도의 이동 방향을 의미한다.
	반복	사각형 안에 위부분은 몇번 반복 할 수 있는지 조건을 넣고 아래부분에는 반복할 순서도를 넣는다. 왼쪽은 상자안의 순서도를 i=0부터 10보다 작은동안 10번 반복하라는 이야기이다.

동그라미 모양은 프로그램 시작과 끝에 사용되어 프로그램의 시작과 종료를 의미 한다. 하나의 프로그램에는 하나의 시작과 끝이 있도록 구현하여야 한다. 다이아몬드 모양인 판단은 상황에 따라 실행 방향이 달라야 할때 사용한다. 직사각형 모양인 처리는 발생하는 작업을 기술할 때 사용한다. 판단, 입출력, 준비, 서류 같은 다양한 경우가 있는데 보통 편의를 위해서 직사각형 모양인 처리로 대체해서 사용 하는 경우가 대부분 이다. 이 책에서도 입출력, 준비, 서류 같은 경우 정해진 여러가지 도형으로 하지 않고 직사각형 모양에 처리 내용을 기술하여 표현고 있다.

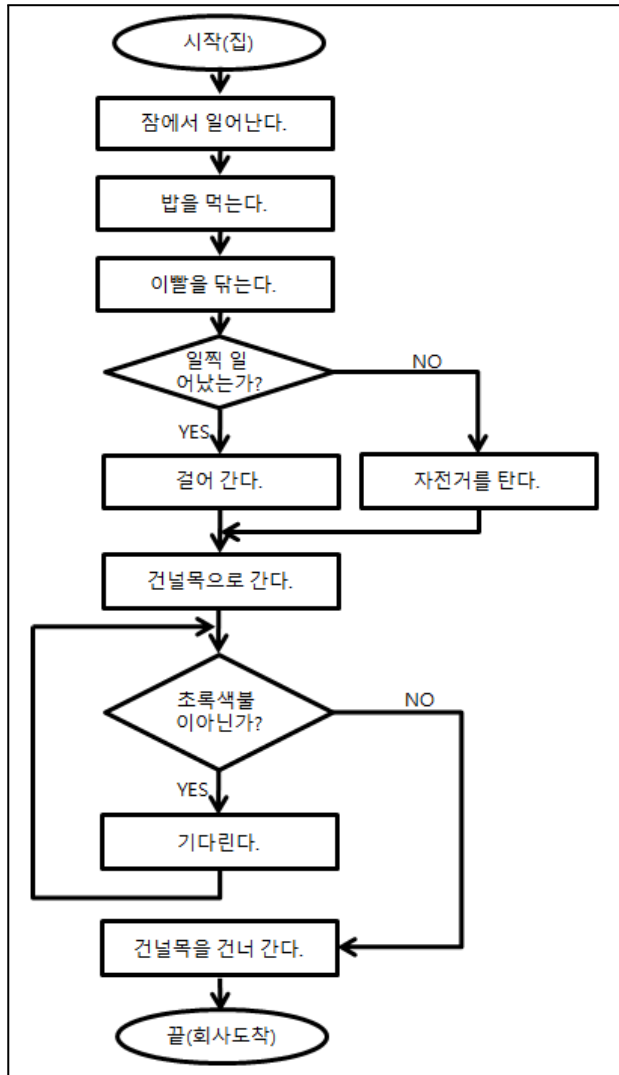


순서도 그릴때 조심해야 할 점은 다음과 같다.

1. 반복문은 true부분이 반복 되도록 기술한다.
2. 처리선은 되도록 라인 쪽으로 그리자. 상위 오른쪽 이미지에서 01 보다는 02로 표현하자.
3. 종료 지점은 하나만 존재해야 한다.

다음 연습문제를 풀어 보자.

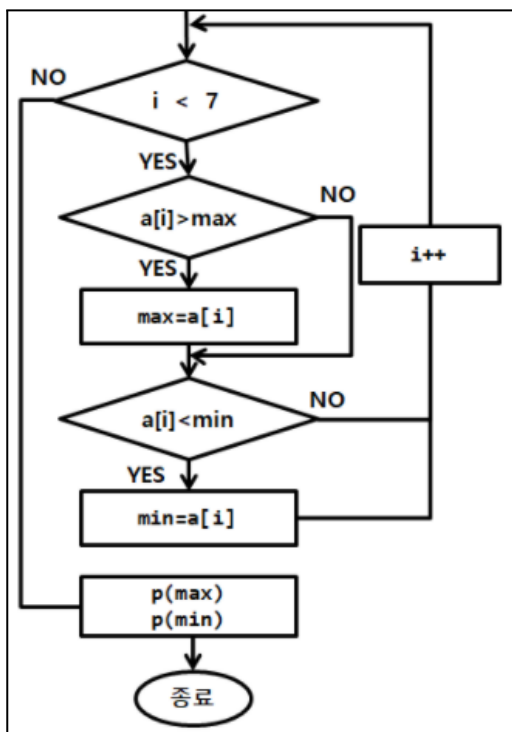




문제1) 다음 페이지의 순서도는 집에서 회사까지 가는 순서도에서 1,2,3번 기차길처럼 동작하는 부분을 각각 찾아 보자.

문제2) 직사각형 부분이 화면에 출력된다고 가정 할때 만약, 일찍 일어나고 초록색불을 처음 확인하였을때는 빨간불이고 2번째 확인하였을때는 초록불이 이었다면 화면에 무엇이 출력되는 결과를 기술하시오.

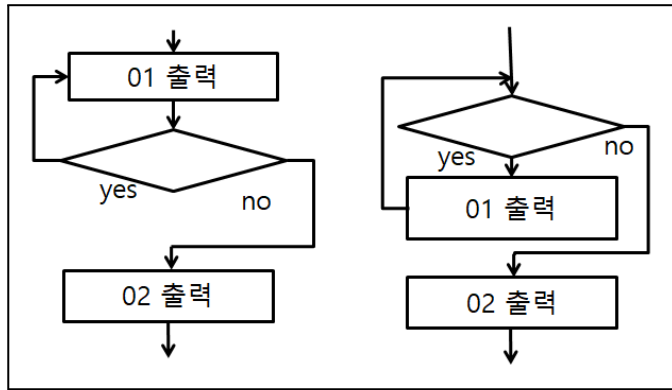
문제3)왼쪽 순서도에서 조건문과 반복문을 찾아보자.



문제4) 왼쪽 이미지에서 각각의 다이아몬드가 조건인지 반복인지 정의해 보자.

첫 번째 다이아몬드는 Yes 쪽을 따라가다 보면 결국에는 조건식 다이아몬드로 다시 돌아오는 걸 확인할 수 있다. No 쪽으로 따라가보면 결국 프로그램이 종료되는 것을 확인할 수 있다. 따라서 첫 번째 다이아몬드는 반복문이다.

두 번째 다이아몬드 와 세 번째 다이아몬드는 어느 쪽으로 이동 하든 결국에 한 곳에서 만난다.



순서도 그릴때 주의해야 하는 부분을 확인해 보자. 반복문을 순서도로 그릴 때 왼쪽 이미지의 두 순서도 처럼 그리는 일이 있는데 초보자들은 두 순서도를 같은 결과가 나올 것으로 생각하는 경우가 있는데 실질적으로는 다르다. 기차길 따라가듯 잘 따라가 보면 왼쪽의 경우 01이 반드시 1번 이상

출력되지만 오른쪽 순서도는 한번도 출력되지 않는 결과를 얻을 수 있다. 따라서 전혀 다른 순서도이고 프로그램으로 구현 할 때 반복문은 보통 while이나 for문을 사용 하는데 왼쪽의 경우 do~while문을 사용 해야 하는 경우이다. 반드시 1번 꼭 출력되어야 할 경우는 왼쪽 순서도를 사용 해야 하지만, 그렇지 않을 경우에는 오른쪽 순서도를 사용하여야 한다. 실질적으로 do while문을 프로그램에서 구현할 경우는 거의 없다. 아무 생각 없이 잘못 그린 경우가 대부분이니 1번 보다는 2번 형태의 반복문으로 그리자.

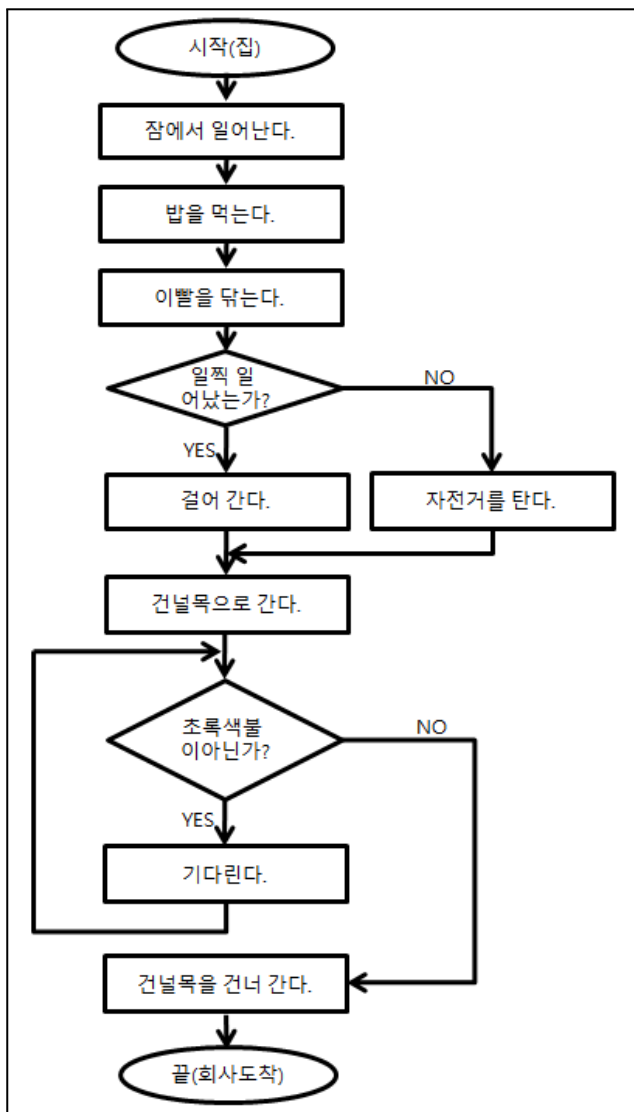
do~while은 1~N번까지 반복할때 while은 0~N번까지 반복할 때 사용한다.

## > 02. 의사 코드

의사 코드란? 프로그램은 아니지만 프로그래밍 언어 처럼 알아보기 쉽게 기술한 코드를 의미한다.

의사코드에서 절차문은 그냥 순서대로 글로 기술하면 되고, 조건문은 if, 반복문은 while 키워드로 만들어 보았다. 의사 코드는 순서도를 글로 옮겨 놓은 것이라 생각해도 된다. 특정한 문법은 없고 의사 코드라는 의미는 사람이 알아보기 쉽게 기술 하였다고 생각하면 된다.

다음 이미지 순서도는 집에서 회사까지 가는 과정을 순서도로 만든 것 이고 이미지 오른쪽에 있는 코드는 의사 코드이다. 다음을 확인해 보자. 중요한점은 탭을 이용해서 줄을 맞춰서 기술해야 한다. 중괄호{ 안에 기술하는 내용은 탭만큼 띄워서 기술해야 한다. 키보드 맨 왼쪽 부분에 tab이라고 쓰여져 있는 키를 누르면 탭 만큼 띄워 쓸수 있다.



의사코드

p(시작(집))

p(잠에서 일어난다.)

p(밥을 먹는다.)

p(이빨을 닦는다.)

if(일찍 일어났는가?){

    p(걸어간다.)

}else{

    p(자전거를 탄다.)

}

p(건널목으로 간다.)

while(초록색불이 아닌가?){

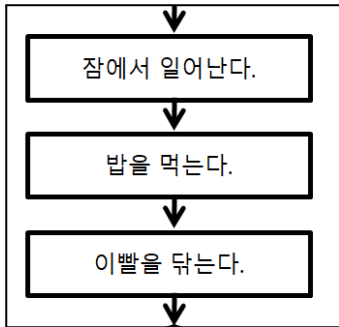
    p(기다린다.)

}

p(건널목을 건너 간다.)

p(회사도착)

상위 순서도에서 절차문, 조건문, 반복문을 찾아보자.



왼쪽은 절차문에 해당한다. 다음은 의사코드 이다.

p(잠에서 일어난다)

p(밥을 먹는다)

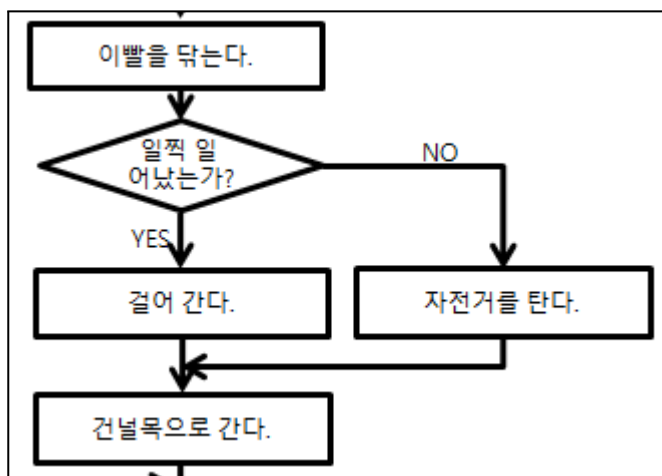
p(이빨을 닦는다)

좀더 실제 프로그램 코드와 유사한 형태로 의사코드 작성을 해 보겠다.

p()의 의미는 소괄호 안의 내용을 화면에 출력 하라는 의미로 생각 하면 된다.

프로그램 언어에서는 조건문을 if문으로 사용 한다. if문 사용 방법은 다음과 같다.

```
if(조건식){
    //참 일때 실행되는 코드 블록
}else{
    //거짓일때 실행되는 코드 블록
}
```



다음 부분은 조건문에 해당하는 부분이다.

p(이빨을 닦는다.)

if(일찍 일어났는가?){

p(걸어간다.)

}else{

p(자전거를 탄다.)

}

p(건널 먹으러 간다.)

조건문은 일찍 일어났는가? 와 같은 조건식이 참 인지 거짓 인지에 따라 실행 부분이 결정된다.

if문의 조건식에는 true,false의 결과를 가지는 수식만 올 수 있다

```

p(이빨을 닦는다);

if(일찍 일어났는가){
    p(걸어간다);
}
else{
    p(자전거를 탄다);
}

p(건널목으로 간다);

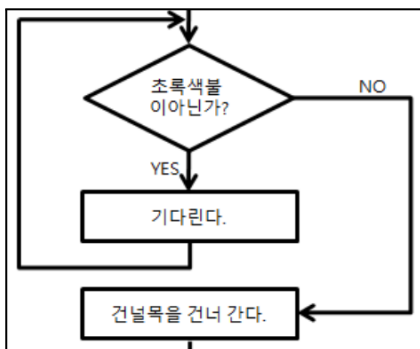
```

if문 다음에는 조건식을 포함한 소괄호가 오고 소괄호 안에 조건식이 참인지 거짓 인지에 따라 실행 되는 부분이 달라진다. 참일 경우에 두 개의 중괄호중에 else 키워드 이전 부분인 첫번째 중괄호 부분이 실행되고 거짓일 때는 else 다음에 오는 두번째 중괄호 블록 부분이 실행된다. 실행 할 거짓 부분이 없다면 else{} 부분을 생략할 수 있다. 왼쪽 이미지는 프로그램 언어로 기술한 결과물이다.

왼쪽 의사코드를 보고 순서도를 그려보자.

아래 이미지 부분은 반복문에 해당한다.

관련 의사코드와 순서도를 그려서 제출해 보자.



의사코드

```

while(초록색 불이 아냐?){
    p(기다린다.)
}

p(건널목을 건너 간다.)

```

프로그램 언어에서 반복문은 while과 for 문을 사용 하는데 while문의 축약된 형태가 for문 이여서 순서도를 그대로 옮기기에는 while문이 편하다. 사용 방법은 다음과 같다.

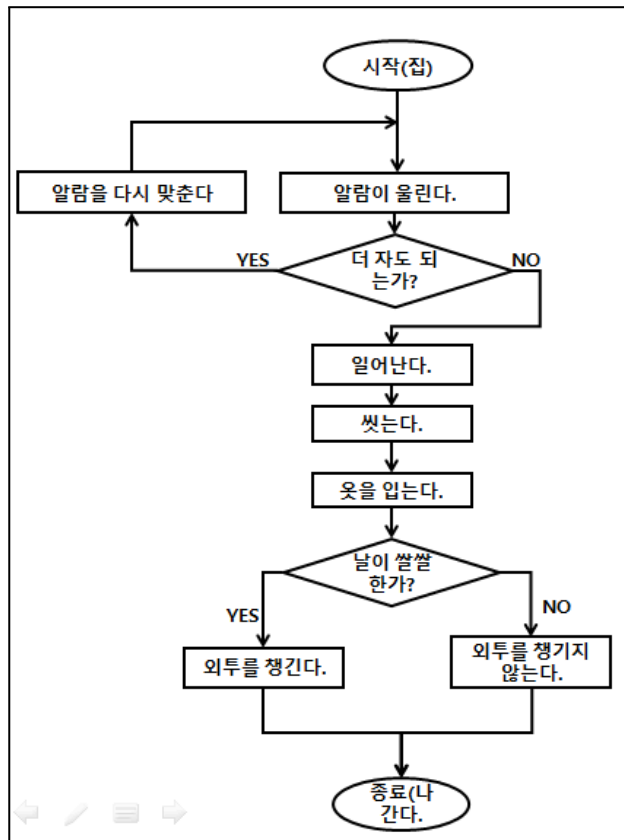
```

while(조건식){
    //반복할 코드블록
}
// while 문을 빠져 나오면 이어서 실행될 부분 이다.

```

while문 사용 방법은 while 다음의 소괄호에 조건식을 쓰고 중괄호에 반복할 코드를 기술한다. 조건식이 true면 반복할 코드 블록을 반복하고 false이면 while문을 빠져 나간다.

while문의 조건식 부분이 코드블록을 반복 여부를 결정한다. true이면 반복 할 코드 블록이 실행되고 false이면 while문 중괄호 부분을 빠져나간다.



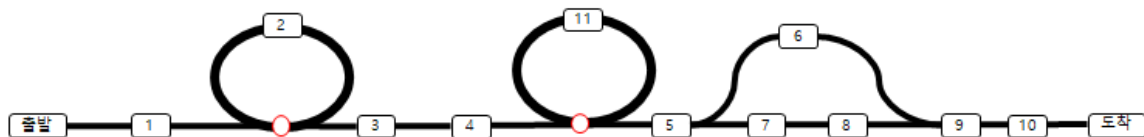
문제1. 왼쪽 순서도 의사코드를 작성하시오.

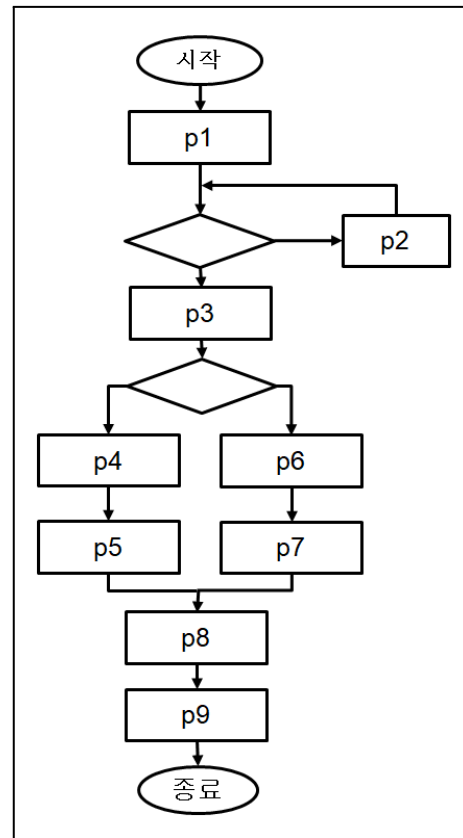
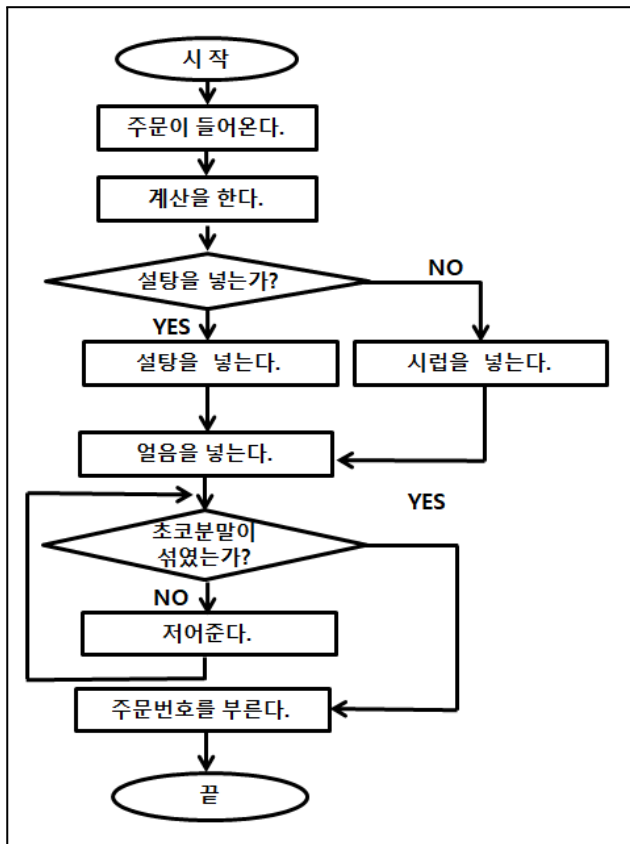
문제2. 다음 의사코드를 순서도로 만드시오.

```

p(PPT제출 전)
p(컴퓨터를 켜다)
If(PPT가 깔려 있는가?){
    p(PPT 슬라이드를 만든다.)
}else{
    p(PPT를 다운로드 받는다.)
}
p(PPT를 완성한다.)
While(PPT 제출일이 아닌가?){
    p(기다린다.)
}
p(PPT를 제출한다)
p(PPT 제출 완료)
  
```

문제3. 다음 기차길의 순서도와 의사코드를 작성해 보자. 조건안의 질의로 스위치가 오른쪽인가?, 반복할 것인가?를 사용해 보자.





문제5. 왼쪽 순서도에서 잘못된 부분은 반복문이 false일때 반복된다는 것이다. 찾아서 true가 반복되게 고친 다음 의사코드를 만들어 보자.

문제6. 오른쪽 순서도를 보고 기차길과 의사 코드를 만들어 보자 .

문제7. 라면끓이는 방법을 순서도와 의사코드로 만들어 보자.

---

## > 03. 의사 코드를 자바코드로 구현하기

---

p(잠에서 일어난다) : 자바 프로그램에서는 `System.out.println("잠에서 일어난다");`라고 기술하면 화면에 “잠에서 일어난다” 문자열이 출력 되는데 프로그램 언어마다 출력 방법이 조금씩 다르다. `System.out.println("잠에서 일어난다");`로 기술 하면 화면에 소괄호 안의 내용이 화면에 출력된다는 의미이다.

다음 조건을 자바로 변경하면 변경하려면 `if(조건)`에서 조건이 `true`이면 걸어간다 `false`이면 자전거를 탄다. 다음 예제들을 확인해보자.

```
if(일찍 일어났는가){  
    p(걸어간다.)  
}  
else{  
    p(자전거를 탄다.)  
}
```

걸어간다는 출력하고 싶다면 다음을 기술 한다.

```
if(true) {  
    System.out.println("걸어간다.");  
} else {  
    System.out.println("자전거를 탄다.");  
}
```

자전거를 탄다는 출력하고 싶다면 다음을 기술 한다.

```
if(false) {  
    System.out.println("걸어간다.");  
} else {  
    System.out.println("자전거를 탄다.");  
}
```

다음은 반복문으로 변경하려면 자바에서는 다음과 같이 하면 된다.

```
while(초록색 불이 아닌가){  
    p(기다린다.)
```



```
}
```

일단 다음과 같이 기술하면 3번 기다린다가 출력된다.

```
for (int i = 0; i < 3; i++) {  
    System.out.println("기다린다.");  
}
```

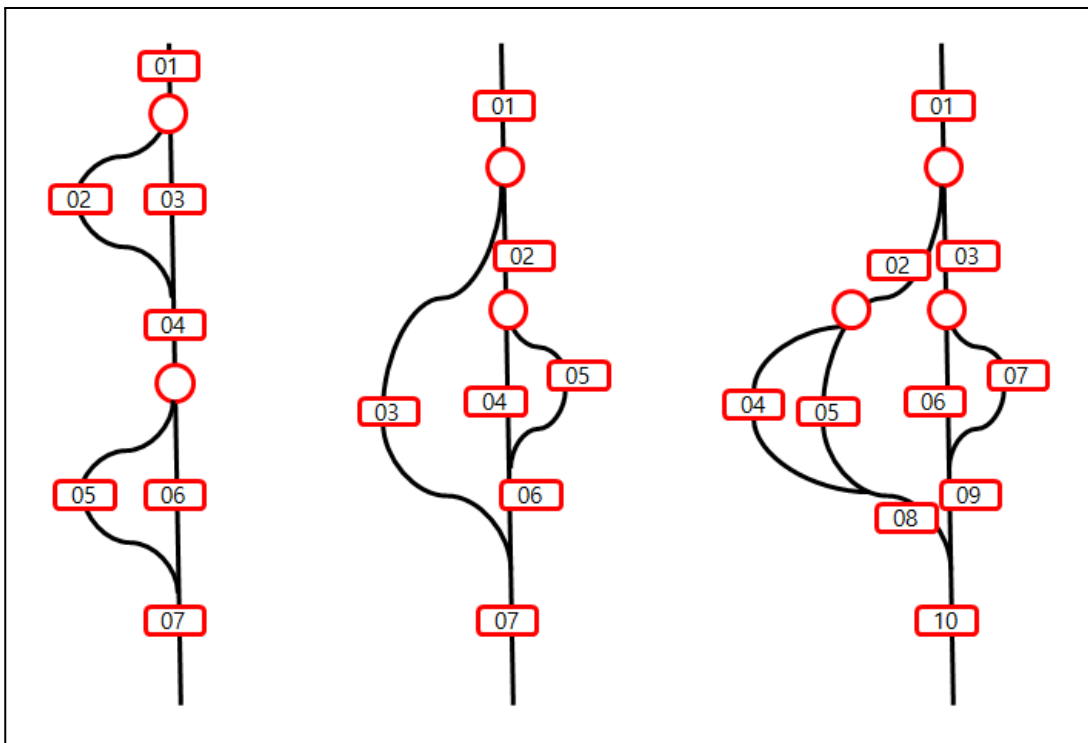
이후 제공되는 의사 코드를 자바 코드로 변경해서 결과가 나오도록 만들어 보자.

```
if(true), if(false), for(int i=0;i<3;i++)
```

이전 챕터의 연습문제 답안 의사 코드를 자바 코드로 구현해서 실행 시켜 보자.

절차식, 조건식, 반복식은 서로 내부에 넣어 반복해서 사용할 수 있다.

조건문 안에 조건문을 넣을 수 있다. 각 기차길에서 if문이 몇개 사용되는지 확인해 보자.



각 빨간색 동그라미 부분이 if문의 시작 부분이므로 동그라미 개수 만큼 if문이 존재 한다. 각 if문의 시작과 끝 부분을 생각해 보자.

각 if문의 시작 부분은 빨간색 동그라미가 있는 부분이고 끝나는 부분은 빨간 동그라미에서 갈라진 선이 최초 만나는 지점이다.

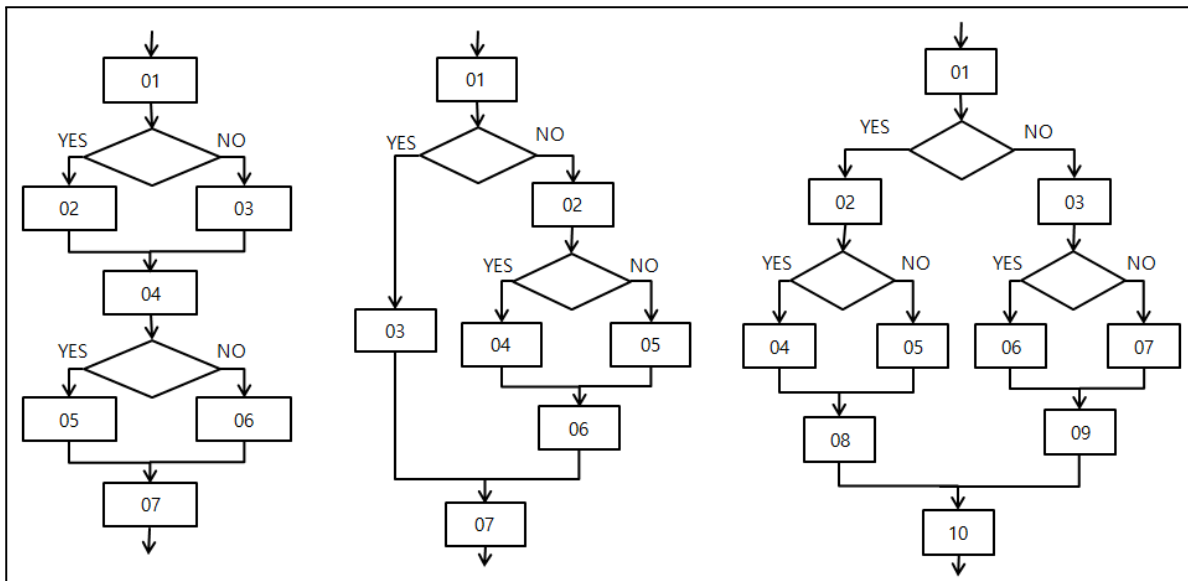
첫번째 기차길은 01 다음에 시작하여 갈라진 기차길이 04에서 만나고 있고, 04 다음에서 갈라진 기차길은 07에서 합쳐지고 있다. if문 2개가 중복되지 않고 순차적으로 기술된다.

두번째 기찻길은 01 다음에 시작하여 07 이전에 합쳐지고, 02다음에 갈라진 기찻길은 06 이전에 합쳐진다. 01다음에서 if문이 끝나는 부분은 07이고, 02 다음에 if문이 끝나는 부분은 06이전이다. 2개의 if문이 중복되어 있다.

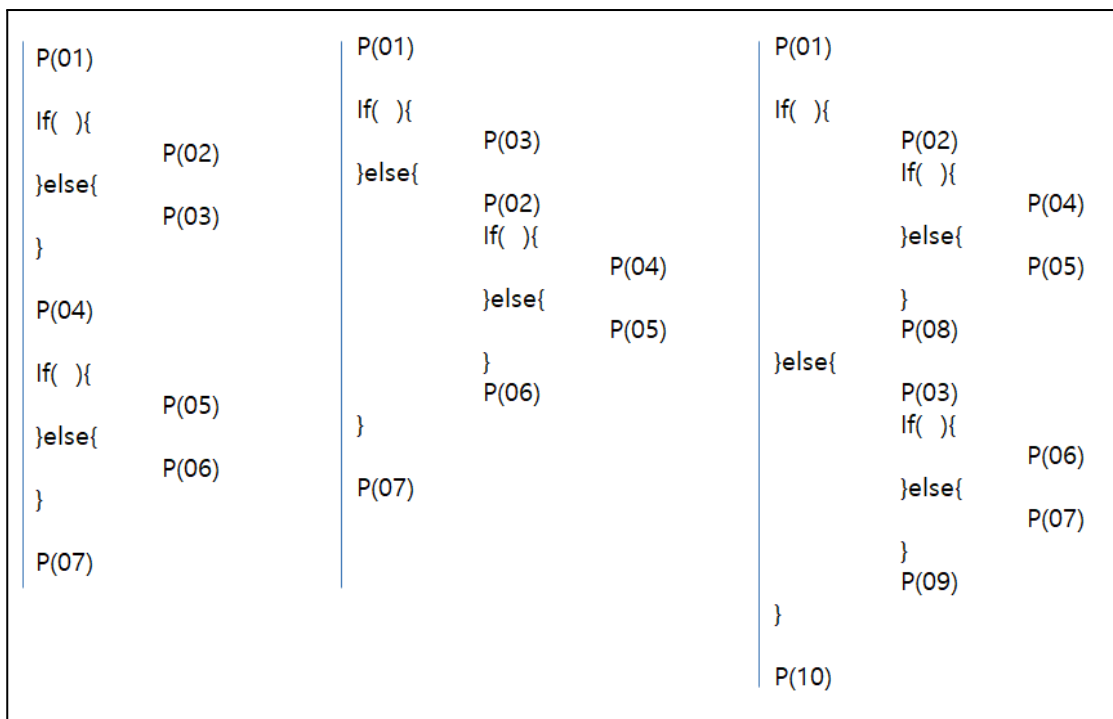
세번째 기찻길은 01다음에 시작한 if문이 10에서 끝나면 왼쪽길 안에 if문 하나 오른쪽길 안에 if문 하나가 있는 상태이다. 3개의 if문이 중복되어 있다.

상위 이미지를 보고 의사 코드와 순서도를 만들어 보자. 다음 페이지에 정답이 있는데 보지 말고 의사코드와 순서도를 만들고 자바 코드로 구현해서 출력 결과를 예측해 보자.

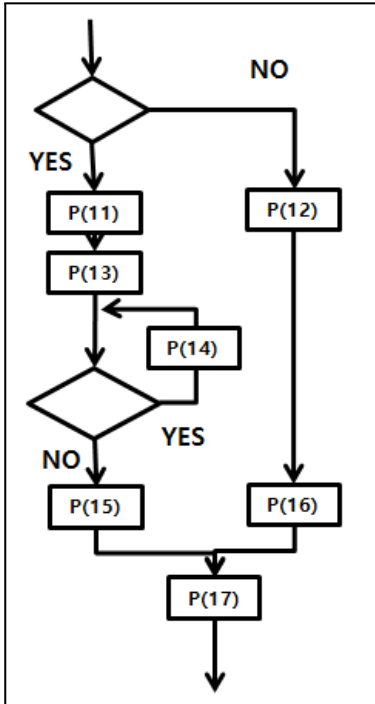
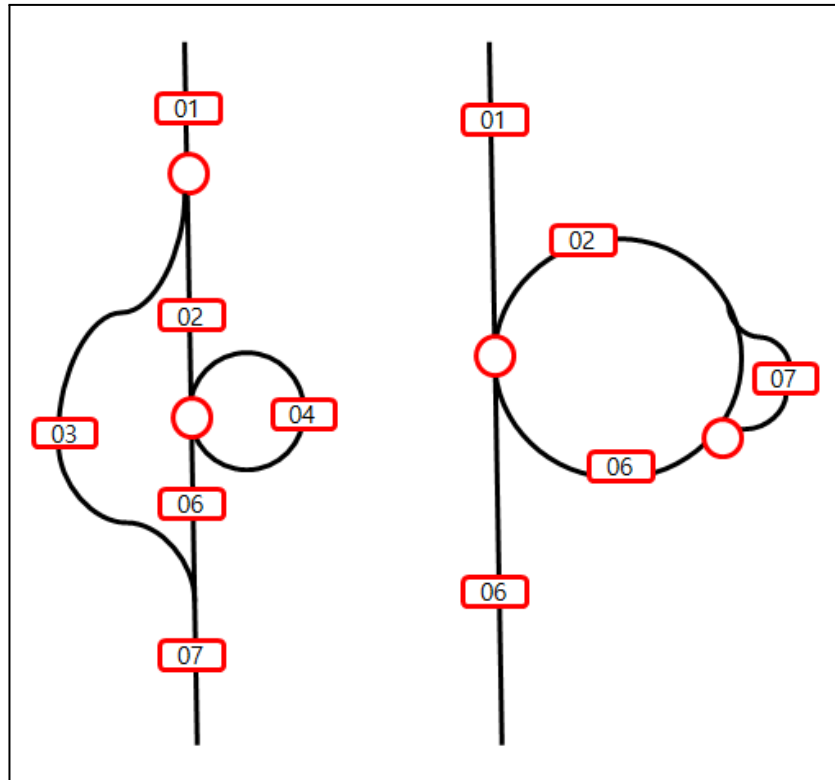
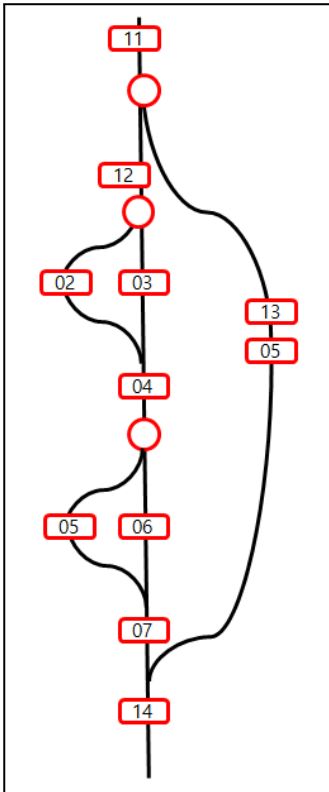
다음 페이지에 정답이 있으니 다 만든 다음에 확인해 보자.



상위 기찻길을 의사코드로 만들어 보자. 중괄호 부분 위치를 확인해서 기술해 보자.



if문과 반복문을 섞어서 중복되게 사용할 수 있다. 다음 페이지의 문제들을 풀어보고 자바코드를 만들어 보자.



1. 상위 왼쪽의 이미지를 확인하여 의사 코드와 순서도를 만들어 보자.

2. 조건문안에 반복문을 사용 할 수 있다. 왼쪽 순서도를 보고 기차길과 의사 코드를 구현해 보자.

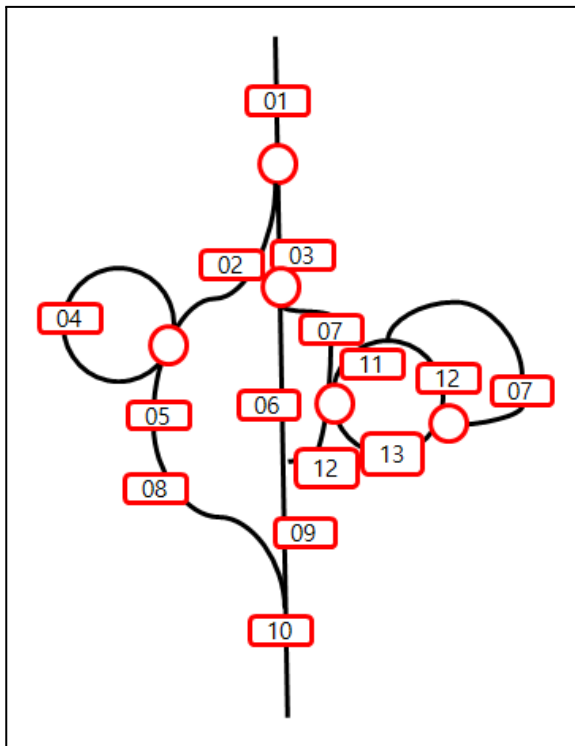
3. 조건문안에 반복문을 사용할 수있고 반복문안에 조건문을 사용할 수 있다. 상위 가운데 이미지를 보면 조건문안에 반복문이 있는 것을 확인할 수 있다. 상위 맨오른쪽을 보면 반복문 안에 조건문이 있는것을 확인할 수 있다. 상위 왼쪽 2개의 이미지를 순서도와 의사 코드로 만들어 보자.

4. 다음을 순서도로 만들어 보자. 513번 버스 대기 시간이 10분 이상 넘으면 걸어서 목적지에 가고 10분 이전이면 도착한 버스가 513번 버스인지 확인하여 513번 버스를 타고 목적지에 가는 순서도를 만들어 보자.

5. 다음 페이지의 왼쪽 이미지를 보고 순서도와 의사코드를

작성하시오.

6. 다음 페이지의 오른쪽 이미지를 보고 기차길과 순서도를 작성하시오.



```

1  if( ){
3
4
} else{
3  while( ){
6
8
5
7
} else{
11
while{
12
13
16
}
}
13 while( ){
17
if( ){
22
} else{
21
18
19
}
}

```

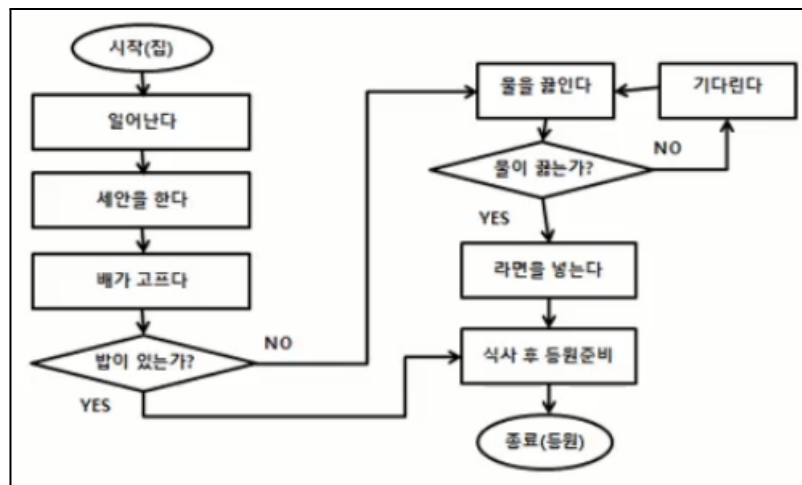
```

P(1)
If( ){
P(2)
if( ){
P(4)
} else{
P(5)
}
P(6)
} else{
P(3)
if( ){
P(7)
} else{
P(8)
}
P(9)
if( ){
P(13)
} else{
P(10)
while( ){
P(11)
P(12)
}
P(15)
}
P(16)

```

7. 왼쪽 의사코드를 보고 기차길과 순서도를 만들 어보자.

8. 다음 기차길과 의사 코드를 작성해 보자.



---

## > 03. 조건문 if 사전문제 -04

---

다음 입력 부분의 코드를 메인에 넣고 실행 시켰을때 출력 부분이 출력되는지 확인해 보자.

1.입력

```
if(5>3){  
    System.out.println(true);  
}  
출력 : true
```

2.입력

```
if(5<3){  
    System.out.println(true);  
}  
출력 : 출력 없음
```

3.입력

```
System.out.println(1);  
if(5<3){  
    System.out.println(true);  
}  
System.out.println(2);  
출력  
1  
2
```

4.입력

```
if(5>3){  
    System.out.println(true);  
}else{  
    System.out.println(false);  
}  
출력 : true
```

5.입력

```
if(5==3){  
    System.out.println(true);  
}else{  
    System.out.println(false);  
}  
출력 : false
```

6.입력

```
System.out.println(1);  
if(5>3){  
    System.out.println(true);  
}else{
```

```
        System.out.println(false);
    }
    System.out.println(2);
출력 : true
```

#### 7. 입력

```
int a=10;
int b=5;
if(a<b){
    System.out.println(true);
}
출력
?
```

#### 8. 입력

```
int a=10;
int b=5;
if(a>b){
    System.out.println(true);
}else{
    System.out.println(false);
}
출력
?
```

#### 9. 입력

```
Scanner scanner = new Scanner(System.in);

System.out.print("문자열을 입력하세요: ");

String string = scanner.nextLine();

if (string.contains("java")) { //java 문자열이 들어 있으면 true

    System.out.println("java가 포함되어 있습니다.");

} else {

    System.out.println("java가 포함되어 있지 않습니다.");

}

출력

?
```

---

## > 04. 조건문 if

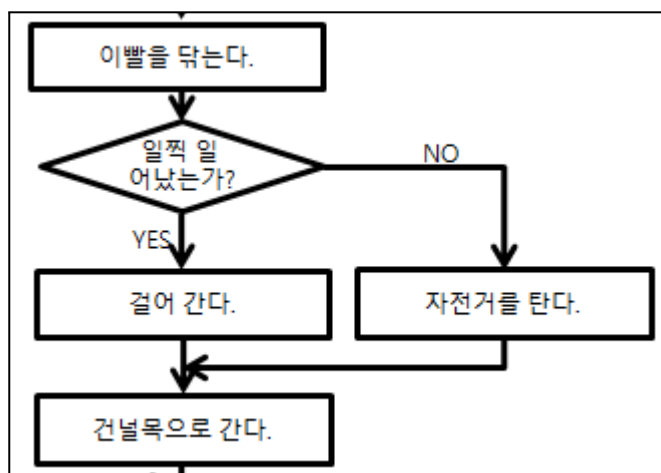
---

컴퓨터 프로그램은 크게 절차문, 조건문, 반복문 3가지로 구성되어 있다고 하였는데  
절차문은 순서대로 위에서 아래로 프로그램이 진행 된다는 이야기이다. 조건문은 둘중에  
하나를 선택해서 실행해야 할 경우에 사용한다. 반복문은 특정 코드를 반복할때 사용한다.

둘 중 하나의 기차길을 선택해서 원하는 목적지에 가는 경우가 있다. 이럴 때 프로그램  
언어에서는 if 문을 사용한다.

프로그램 언어는 기본적으로 위에서 아래로 순서대로 모든 명령어(코드)가 한 줄 한 줄  
실행된다. 하지만, if문을 만나면 상황에 따라 특정 코드가 실행 될 수도, 실행 되지 않을  
수도 있다.

다음 부분은 조건문에 해당하는 부분의 순서도와 의사코드 이다.



이빨을 닦는다.  
`if(일찍 일어났는가?){`  
    걸어간다.  
`}else{`  
    자전거를 탄다.  
`}`  
건널목으로 간다.

상위 순서도를 확인해 보면 일찍 일어났는가? 라는 질문에 yes 인지 no인지에 따라 걸어  
갈 것 인지 자전거를 탈 것인지 진행 방향이 달라지는데 이런 상태를 조건문이라 한다.

프로그램에서 yes는 true, no는 false로 boolean 자료형을 의미한다. 자바에서 조건문을  
표현 할 때 if라는 키워드를 사용한다.

조건식의 조건문은 if(‘일찍 일어났는가?’), if(true), if(a>1) 이와 같이실행 결과  
true,false 값으로 가지는 구문만 올수 있다. (10>3) (10!=2) (4<6)와 같이 조건식은  
보통 true, false를 만들어 내는 비교연산자 (==, <, > 등등)를 사용하는 식이 온다.

다음을 확인해 보면 조건식이 참이면 참일 때 실행되는 코드 블록이 실행되고 거짓 이면  
거짓일 때 실행되는 코드 블록이 실행 된다.

```
if(조건식){//조건식은 실행결과 boolean값을 가진 식만 올수 있다.  
    //참 일때 실행되는 코드 블록  
}else{  
    //거짓일때 실행되는 코드 블록  
}
```

if문 다음에는 조건식을 포함한 소괄호가 오고 소괄호 안에 조건식이 참인지 거짓 인지에 따라 실행 되는 부분이 달라 진다. 참일 경우에 첫번째 중괄호 부분이 실행되고 거짓일 때는 else 다음에 오는 두번째 중괄호 블록 부분이 실행된다. If문은 상황에 따라 다른 결과를 얻어 내고자 할 때 사용한다. 두 코드 블록중 하나의 코드 블록이 실행 된다.

else 다음에 오는 중괄호에 코드가 없을 경우에 아래 처럼 else 부분을 생략할 수 있다.

```
if(조건식){  
    //참 일때 실행되는 코드 블록  
}
```

코드 블록에 코드가 한줄이면 {}호를 생략 할 수 있다.

```
if(true)  
    System.out.println("true");  
else  
    System.out.println("false");
```

{ } 중괄호를 생략해서 쓰다가 보면 나중에 여러줄 쓸때도 중괄호를 생략하게 되어서 아래 처럼 잘못된 형태로 코드를 작성하게 된다. 문법적으로는 문제가 없지만 원하는 결과와 다른 결과가 출력되는 프로그램이 될 수 있다. 다음 코드를 작성하여 생각한거와 동일하게 실행 되지는지 확인해 보자.

```
boolean isFlag=true;//boolean isFlag=false; false로 변경한 경우도 실행해 보자.  
if(isFlag){  
    System.out.println("출력값은");  
    System.out.println("true");  
}  
else  
    System.out.println("출력값은");  
    System.out.println("false");//isFlag값에 관계없이 무조건 출력
```



```

boolean isFlag=false;
if(isFlag){//중괄호를 하였지만 ; 때문에 원하는 결과가 나오지 않음
    System.out.println("출력값은");
    System.out.println("true");
}
if(isFlas); //;이 없어야 원하는결과가 나온다.
    System.out.println("출력값은");//isFlag값에 관계없이 무조건 출력

```

상위 코드는 원하는 결과는 나오지 않으나 문법적으로는 문제가 없기 때문에 잘못된 부분을 찾아내기 어렵다. 어떤 문제가 발생한 건지는 본인이 코드를 작성해서 결과를 확인한 다음 찾아 보자. 이런 문제를 사전에 방지하기 위해서 코드가 한 줄이라 할지라도 중괄호를 사용하는 습관을 가지자. 세미콜론 때문에 if문이 종료 되고 이후 코드가 if문과 상관없이 그냥 실행 되고 있다.

크다,작다,같다 등을 나타내는 >,==,< 등을 비교 연산자라고 한다. 비교 연산자의 실행 결과는 true, false 값을 갖는 불리언 (boolean), bool 자료형을 생성하여 조건문에서 조건식으로 사용할 수 있다. 다음 표를 보고 다음 문제를 풀어 보자.  $5 < 23$ ,  $9 \neq 23$ ,  $34 >= 34$ ,  $53 <= 23$ ,  $9 == 5$  정답이 필요 하다면 다시 한번 표를 확인해 보고 그래도 이해가 가지 않는다면 웹을 검색해서 공부해 보자.

기 호	설명	예제
A > B	A가 B보다 크면 true	3>4:false 3>3:false 4>3:true
A >= B	A가 B보다 같거나 크면 true	3>=4:false 3>=3:true 4>=3:true
A < B	A가 B보다 작으면 true	3<4:true 3<3:false 4<3:false
A <= B	A가 B보다 작거나 같으면 true	3<=4:true 3<=3:true 4<=3:false
A==B	A와 B가 같으면 true	3==4:false 3==3:true
A!=B	A와 B가 다르면 true	3!=4:true 3!=3:false

다음 코드 처럼 표에서 확인한 것들이 맞게 출력되는지 다양한 조건식을 이용해 if문을 완성해 보자. a값을 10 대신에 1를 넣으면 어떻게 실행되는 지도 확인해 보자.

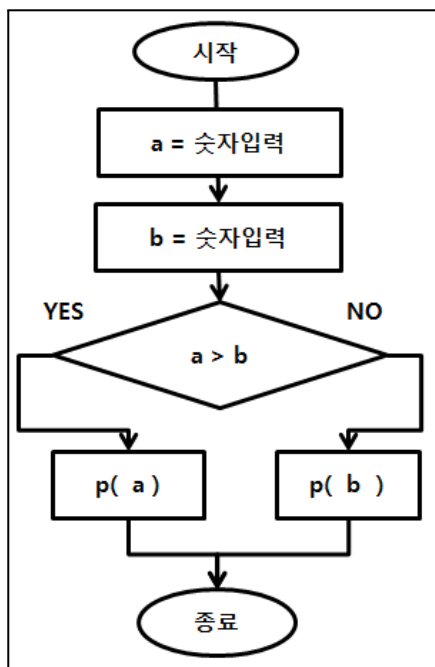
```

int a=10;
if(a>5) {
    System.out.println("수식결과 true여서 a가 5보다크다.");
}else {
    System.out.println("수식결과 false여서 a가 5보다 크지 않다.");
}

if(1>5) {
    System.out.println("수식결과 true이다");
}else {
    System.out.println("수식결과 false이다.");
}

```

다음 순서도와 코드를 확인해 보자.



사용자에게 두 수 a, b를 입력 하여 둘 중 큰수를 화면에 출력하는 순서도와 코드이다. a가 크면 a가 출력 되고 b가 크면 b가 출력 된다. 코드를 입력하고 실행해서 생각한 대로 동작 되는지 확인해 보자.

```

java.util.Scanner scanner =
    new java.util.Scanner(System.in);
int a=Integer.parseInt(scanner.nextLine());
int b=Integer.parseInt(scanner.nextLine());
if(a>b){
    System.out.println(a);
}else{
    System.out.println(b);
}

```

if문 조건식의 실행 결과에 따라 출력되는 결과가 a가 될수도 있고 b가 될수 도 있다.

```

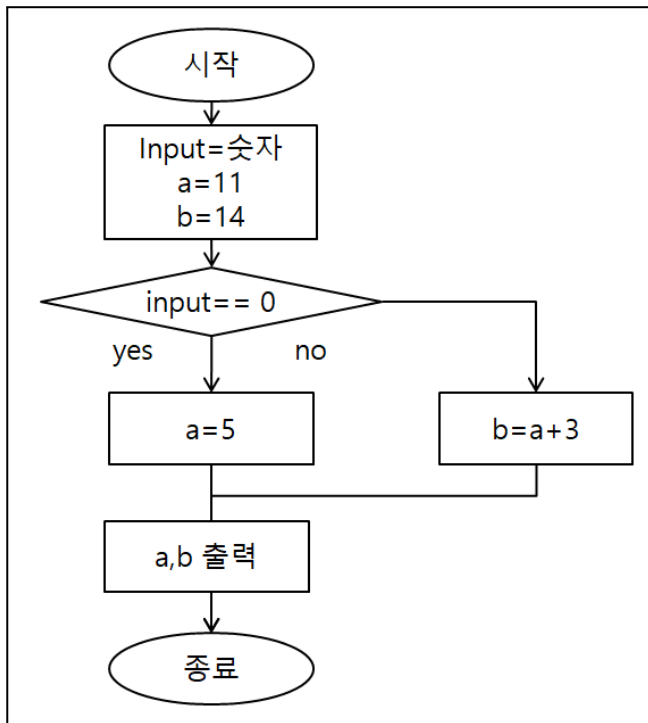
//true, false 1,0 일반적인 언어에서 0이 아니면 true취급
//if(1){} 조건식이 실행
//if(0){} 조건식이 실행 안됨
//if(23){} 조건식이 실행됨
if(a=10) {
    System.out.println("true");
}

```

왼쪽 이미지는 조건식에 true, false을 가지는 식이 오지 않아서 컴파일 에러가 났다.

a=1에서 =은 할당 연산자여서 a에 1를 넣으라는 이야기이다.

a==1은 비교연산자로 a가 1이랑 같은지 물어보는 것이고 a가 1이면 실행 결과가 true가 된다. 자바의 경우 조건식은 반드시 ==를 사용해야 한다.



if문 문제풀이가 어렵다면 책이나 웹에 있는 if문 예제를 여러개 실행해보고 문제를 천천히 풀어 보고 해결이 안되면 답안을 보고 풀어 보거나 gtp를 이용해 보자.

1. 왼쪽 순서도를 프로그램으로 구현해서 출력 결과를 확인해보자.

2. 입력 받은 숫자의 절대값을 출력하는 프로그램을 만들어 보자. (힌트: 0보다 작으면 -1를 곱하면 양수가 된다.)

3. 어떤 수를 나누어 떨어지게 하는 수를 약수라고 한다. 100을 2로 나누면 0이된다. 따라서, 2는 100의 약수이다. 사용자에게 숫자를 하나 입력받아 126의 약수인지 아닌지 출력해 보자. (힌트: 사용자가

입력한 수를 126로 나눈 나머지가 0이면 약수이다.  $126 \% x == 0$  이 true이면 x는 126의 약수이다.)

4. 두수를 입력받아 첫번째수가 두번째수의 약수인지 아닌지 확인하는 프로그램을 구현해 보자.

5. 국영수과목의 점수를 입력받아 평균이 80이상이면 합격 이하면 불합격을 출력해 보자. 4. (힌트: 과목수만큼 과목점수를사용자 입력을 받아 평균을 구한후 80보다 작으면 불합격 크면 합격)

6. 사용자에게 입력받은 수 x가 5이하이면 x+15의 계산결과가, 5초과 이면 x+5의 계산 결과가 출력되는 프로그램을 만들어 보자.

(힌트: 상황에 따라  $x=x+5$ 나  $x=x+15$ 를 사용하면 된다. 중괄호 안에 선언한 변수는 해당 중괄호 안에서만 사용 할 수 있다. 되도록 메인 메소드 시작 부분에 변수를 선언해서 사용하자. if문 안의 블록에 x를 선언하면 if문 밖에서 접근할 수 없다.)

```

boolean b1=5!=3;
boolean b2=5==3;
if(b1)
    if(b2)
        System.out.println("1번");
    else
        System.out.println("2번");
else
    System.out.println("3번");
System.out.println("4번");
  
```

7. 왼쪽의 if문이 어떻게 출력되는지 생각해보고 출력 결과를 확인해 보자.

생각과 다른 예매한 결과가 나올 수 있으니 될수 있으니 if문의 중괄호를 생략하지 말자.

8. 변수 a에 20, b에 0을 넣은 다음 만약에 a가 10보다 크면 a에 b를 넣고 아니면 b에 a를 넣어서 a,b값을 출력하는 프로그램을 만들어 보자.
9. 입력한 숫자가 10보다 큰 수인지 아닌지 출력하는 코드를 만들어 보자.
10. result라는 변수를 선언하고 사용자에게 수를 입력 받아 0이면 0을 0이 아니면 1를 넣어서 출력하는 프로그램을 구현해 보자.
11. 두수를 입력해서 큰수에서 작은수를 뺀 차이를 출력하는 프로그램을 구현해 보자.
12. 입력받은 3개의 숫자 중 가장 큰 수를 출력하는 코드를 구현하시오. 3개중에서 2개를 선택해서 큰수를 구한후 나머지 하나를 비교해 보면된다.

답안 3번

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("약수를 판별할 수를 입력하세요: ");
        String input = scanner.nextLine();
        int num = Integer.parseInt(input);
        if (126 % num == 0) {
            System.out.println(num + "은/는 126의 약수입니다.");
        } else {
            System.out.println(num + "은/는 126의 약수가 아닙니다.");
        }
        scanner.close();
    }
}
```

3.

```
Scanner scanner = new Scanner(System.in);
System.out.print("숫자를 입력하세요: ");
String input = scanner.nextLine();
int num = Integer.parseInt(input);
int result;
if (num == 0) {
    result = 0;
} else {
    result = 1;
}
System.out.println(result);
scanner.close();
```

산술 연산자는 수학적 연산을 하는 +, -, \* ,/ ,% 등이 있다. 산술 연산자중 %연산자는 두수 a,b를 a%b연산 하였을때 결과값은 a를 b로 나눈 나머지가 된다. 7%5==2, 10%5==0, 2%7==2, %연산자의 사용 용도를 확인해 보자.

1. 나머지 연산자(%)를 사용하여 어떤 숫자가 특정 수의 배수인지 확인할 수 있습니다.

100이라는 숫자가 5의 배수라면 100을 5로 나눈 나머지가 0이 된다. 5의 배수란 5을 곱해서 나온 수 이기 때문에 어떤수 a가 5의 배수라면 a를 5로 나눈 나머지가 0이 된다. a%5==0 이 true 라면 a는 5의 배수이다.

100이 5의 배수인지 알아보는 코드

```
int number = 100;
// 5의 배수인지 확인
if (number % 5 == 0) {
    System.out.println(number + "는 5의 배수입니다.");
} else {
    System.out.println(number + "는 5의 배수가 아닙니다.");
}
```

2. 나머지 연산자(%)를 사용하여 어떤 숫자가 홀수인지 짝수인지 확인할 수 있습니다.

어떤 숫자가 2로 나누어지면 짝수이다. 2,4,6,8... 모두 2로 나뉜다. a%2==0이 true라면 a는 짝수 이다.

```
int number = 7;
// 홀수인지 짝수인지 확인
if (number % 2 == 0) {
    System.out.println(number + "는 짝수입니다.");
} else {
    System.out.println(number + "는 홀수입니다.");
}
```

3. 약수를 구할 때는 해당 수를 1부터 자기 자신까지의 숫자로 나누어 보고, 나머지가 0인 경우가 약수입니다.

어떤 수가 약수인지 아닌지 확인하는 방법은 1~어떤수 사이의 숫자중 어떤수를 나누어 떨어지는 수인지 확인하면 된다. 8의 약수는 1,2,4,8 이다. 8의 약수 a는 8이하의 수중 8%a==0이 true인 a를 찾으면 된다.

4의 약수를 구하는 코드

```
int number = 4;
```

```

System.out.println(number + "의 약수: ");
int i = 1;
if (number % i == 0) {
    System.out.print(i + " ");
}
i = 2;
if (number % i == 0) {
    System.out.print(i + " ");
}
i = 3;
if (number % i == 0) {
    System.out.print(i + " ");
}
i = 4;
if (number % i == 0) {
    System.out.print(i + " ");
}
System.out.print(i + "입니다. ");

```

#### 4. 잔돈을 계산할 때 주로 나눗셈과 나머지 연산자를 활용합니다. |

5700원을 500원짜리로 잔돈을 거슬러 주고 싶다면 5700원을 500으로 나눈 몫이 거슬러줄 500원짜리 개수가 된다.  $5700/500==11$ 이므로 5700은 500원 11개를 거슬러주고 200원이 남는다. 이때 200원은 %연산자로 구할 수 있다. 5700원을 500원으로 나눈 나머지이므로  $5700\%500==200$ 과 같다.

```

int totalAmount = 5700;
int coin500Count = totalAmount / 500; // 500원짜리 동전의 개수 계산
int remainingAmount = totalAmount % 500; // 나머지를 계산하여 남은 잔액 구하기
System.out.println("500원짜리 개수: " + coin500Count);
System.out.println("남은 잔액: " + remainingAmount + "원");

```

#### 5. 초를 시간, 분, 초로 변환할 때 나눗셈과 나머지 연산자를 활용할 수 있습니다.

10000초가 몇 시간 인지 알고 싶으면 1시간은 3600초 이므로  $10000/3600==2$  에서 2시간 하고 2800초가 남는다. 여기서 2800초를 구하고 싶다면  $10000\%3600==2800$  하면된다.

```

int totalSeconds = 10000;
// 시간 계산
int hours = totalSeconds / 3600;
// 남은 초 계산
int remainingSeconds = totalSeconds % 3600;
// 분 계산
int minutes = remainingSeconds / 60;
// 남은 초 계산

```

```
int seconds = remainingSeconds % 60;
System.out.println("시간: " + hours + "시간");
System.out.println("분: " + minutes + "분");
System.out.println("초: " + seconds + "초");
```

totalSeconds / 3600은 시간을 나타내며, totalSeconds % 3600은 나머지 초를 나타냅니다. 그 후, 나머지 초를 이용하여 분과 초를 계산합니다. 이를 출력하면 10000초가 2시간 46분 40초로 변환되었음을 확인할 수 있습니다.

다음 문제를 확인해 보자.

1. 숫자를 하나 입력 받아 홀수 인지 짝수 인지 구하는 프로그램을 구하여라.
2. 숫자 2개를 입력 받아 첫번째 숫자가 두번째 숫자의 배수인지 아닌지 출력하는 프로그램을 구하여라.
3. a 원을 동전으로 바꾸면 500원짜리 몇 개 100원짜리 몇 개가 되는가?
4. 500원짜리 n개 100원짜리 m개 총 얼마인가?
5. n초를 입력 받아 시분초로 바꿔보자.
6. n시간 m분 1초는 총 몇초인가?

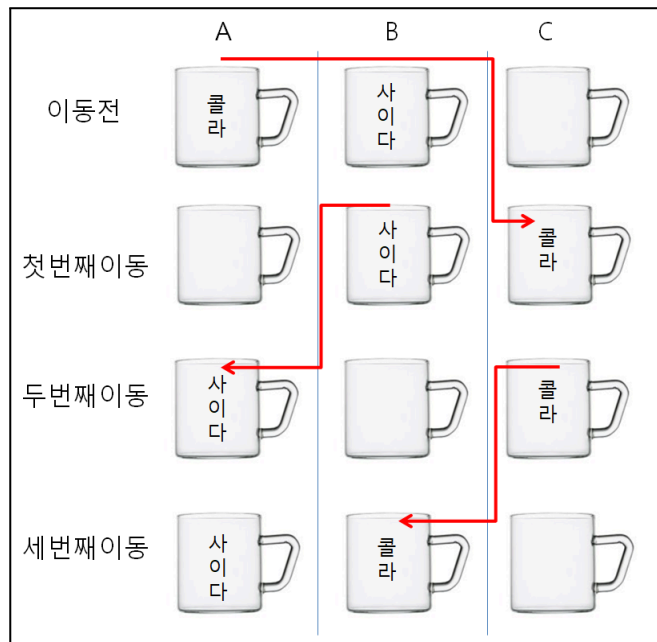
- 변수 a,b에 들어있는 값을 서로 교환하는 방법에 대해서 생각해보자.

이 문제의 해답은 변수 2개를 가지고 처리할 수 없다. 임시 저장 할 변수 c를 하나더 사용하여야 교환이 가능하다.

두 수 a=5, b=4가 있고 두수를 교환하여 a=4, b=5가 되게 만들려면 어떻게 해야 할 것인지를 고민해 보자. a=b, b=a 라고 하면 될거 같지만 a=5, b=4, a=b 에서 a,b값은 이미 둘다 4가 되어 이후 b=a 작업의 값은 여전히 둘다 4가 된다. 작업 이후 우리가 원하는 값 교환이 이루어지지 않는다.

```
int a = 5;          int b = 4;
System.out.println("교환 전: a=" + a + ", b=" + b);

// 잘못된 교환
a = b;              b = a;
System.out.println("잘못된 교환후: a=" + a + ", b=" + b); // a=4 b=4
```



우리가 원하는 형태의 교환을 위해서는 변수  $c$ 를 하나 더 선언하여  $c$ 를 임시 저장 공간으로 사용하면  $c=a, a=b, b=c$  와 같은 방법으로  $a, b$  값을 교환 할 수 있다.

왼쪽 이미지에는 A, B, C 세개의 컵이 있다. 처음에는 A 컵에 콜라가 담겨져 있고 B 컵에 사이다가 담겨져 있는데 두 컵이 내용물을 교환해서 A 컵에는 사이다, B 컵에는 콜라가 담기게 하려고 한다. A 컵 B 컵에 내용물이 담겨져 있는 상태여서 내용물을 서로 교환 할 수 없기 때문에 새로운 C 컵이 필요하다.

첫 번째 이동에서 A 컵에 들어있는 콜라를 다른 C 컵에 옮겨 두어야 B 컵에

있는 사이다를 A 컵에 담을 수 있기 때문에 A 컵에 있는 콜라를 C 컵으로 옮겼다.  $C=A$ ;

두 번째 이동에서 B 컵에 있는 사이다를 A 컵에 옮겼다.  $A=B$ ;

세 번째 이동에서 C 컵에 있는 콜라를 B 컵에 옮겼다.  $B=C$ ;

결과 적으로 우리가 원하던 교환을 통해 a 컵에 사이다 B 컵에 콜라가 담겨 있도록 만들었다.

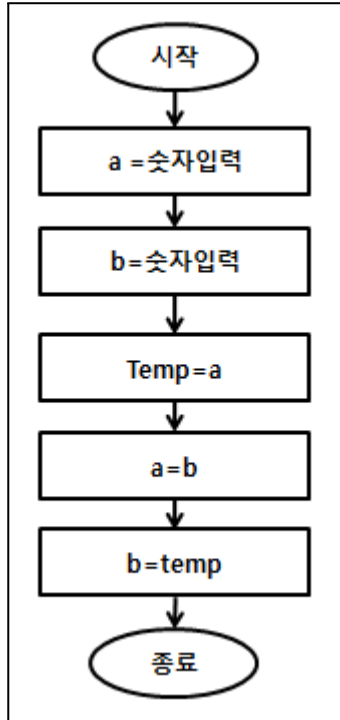
$a=5, b=6$  일때 임시 저장 공간 변수  $c$ 를 하나 선언해서  $c=a, a=b, b=c$  와 같은 방법을 통해 두수를 교환하게 만들어  $b=6, a=5$  상태로 만들 수 있다.

```
public class Main {
    public static void main(String[] args) {
        int a = 5; // A 컵의 콜라
        int b = 6; // B 컵의 사이다

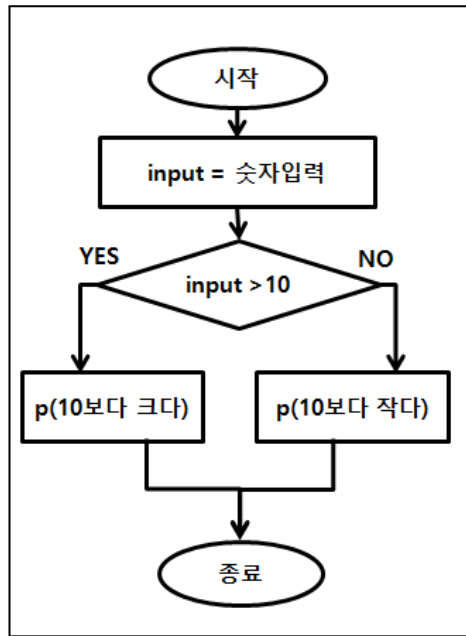
        int c;
        c = a; // 첫 번째 이동: A 컵의 콜라를 C 컵으로 옮김
        a = b; // 두 번째 이동: B 컵의 사이다를 A 컵으로 옮김
        b = c; // 세 번째 이동: C 컵의 사이다를 B 컵으로 옮김
        // 결과 출력
        System.out.println("A 컵의 내용물: " + a);
        System.out.println("B 컵의 내용물: " + b);
    }
}
```



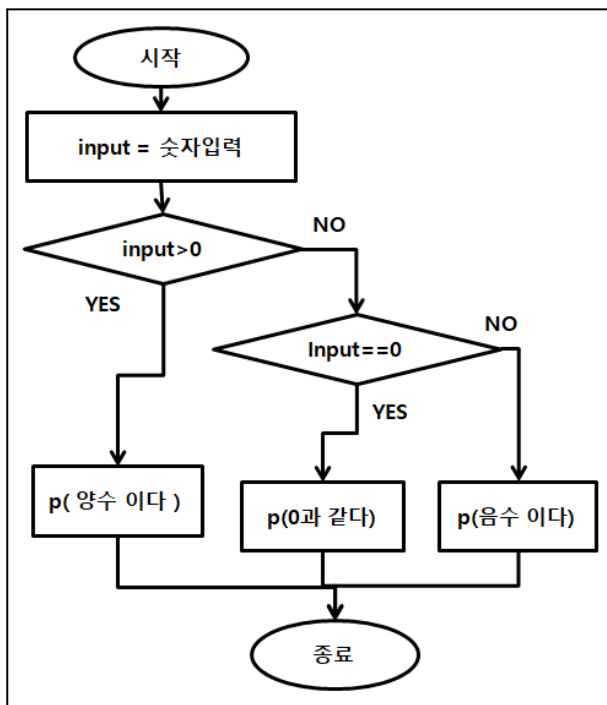
다음 문제를 풀어보자.



1. 왼쪽 순서도는 a, b의 수를 입력 받아 두 수를 교환 후 두 수를 출력하는 순서도이다. 참고 해서 코드를 작성해 보자.



2. 입력한 숫자가 10보다 큰 수인지 아닌지 출력하는 순서도를 보고 프로그램으로 만들어 보자.



3. 입력 받은 숫자가 양수인지 0인지 음수인지 판단하는 순서도와 프로그램을 만들어 보자.

4. 3개의 수를 입력 받아 가장 작은 수를 a, 다음 작은 수를 b, 나머지 수를 c에 넣어 작은 수 부터 출력해 보자.

변수 a, b, c에 무작위로 각각 숫자를 입력받는다.

가장 작은 변수를 찾아 a값과 교환한다.

남은 변수를 비교해서 c가 더작으면 b와 교환 그렇지 않으면 교환하지 않는다.

a, b, c를 순서대로 출력한다.

다음 예제처럼 동작하는지 확인해 보자.

입력값 : a, b, c 에 임의수 를 입력받는다.

출력값: a, b, c 값을 출력하면 정렬된 값이 출력된다.

만약에 입력값이 5,1,2이면 출력값은 1,2,5가 된다.

만약에 입력값이 3,7,4이면 출력값은 3,4,7가 된다.

만약에 입력값이 3,5,6이면 출력값은 3,5,6가 된다.

---

## > 05. else if문과 switch문

---

if else문은 둘중에 하나를 실행해야 할때 사용하고 else if문은 여러개의 조건 중 하나만 실행해야 할 경우 사용한다.

else if문은 여러 개의 조건중 하나의 코드 블록만 선택해서 실행 하고 싶을때 사용한다. 조건에 맞는 부분이 없으면 else문 코드 블록이 실행된다. if문으로 구현 가능 하지만 여러개 조건 중 반드시 하나만 실행 된다면 if문 보다는 else if문을 사용하는 것이 좋다.

다음 코드는 여러개의 코드중 반드시 하나의 코드 블록이 실행되도록 if else문으로 구현한 코드이다.

```
int i=1;
if(i>0){
    System.out.println("i은 양수");
}else {
    if(i<0){
        System.out.println("i은 음수");
    }else {
        if(i==0) {
            System.out.println("i은 0");
        }else {
            System.out.println("i은 모두 아님");
        }
    }
}
```

i가 양수 인지 음수인지 0인지 판별하는 프로그램이다.

왼쪽 코드 이미지의 라인 처럼 if문 열고 닫는 중괄호를 잘 맞추자. 중괄호를 연부분의 if문 시작 부분과 같은 부분에서 괄호를 닫아야 한다. 초임자는 마음이 급해서 괄호를 잘 맞추지 않는데 매우 중요하므로 마음의 여유를 가지고 천천히 맞춰서 기술하자.

코드를 자동 정렬하는 방법은 컨트롤+A 를 통해서 모든 코드를

선택한다음에 컨트롤+시프트+F를 누르면 코드가 정렬 된다.

코드를 잘 확인해 보면 println 으로 출력된 양수, 음수, 0, “모두 아님” 중 반드시 하나만 출력 된다. 상위 처럼 if문이 중복되어 여러 경우중 반드시 하나만 출력 되는 경우 else if문으로 변경 가능하다.

```
int i=1;
if(i>0) {
    System.out.println("i는 양수");
}else if(i<0) {
    System.out.println("i는 음수");
}else if(i==0) {
    System.out.println("i는 0");
}else {
    System.out.println("i는 양수,음수,0아님");
}
```

상위 if문을 else if문으로 변경하면 다음과 같다.

else if 문의 코드 블록들이 두 개이상 출력되는 경우는 발생하지 않는다. 여러개 중 반드시 하나만 출력되는 프로그램을 구현 할 때 여러개의

if문을 사용하는 것보다 else if문을 사용하는 것이 편리하다.

```
if(조건1){
    실행문1;
}else if(조건문2){
    실행문2;
}else{
    //원하는 조건이
    없을경우 실행되는
    부분;
}
```

else if문 사용 방법은 다음과 같다.

처음에 if(조건문)으로 조건을 설정하고 다음에 오는 {} 블록 안에 조건에 만족 할때 실행할 실행문을 기술한다.

새로운 조건 범위를 추가하고 싶다면

else if(조건문){실행;} 부분을 새로운 조건 범위 만큼 반복하여 기술한다.

만족하는 조건문이 없을 경우 실행할 코드 부분을 else 부분에 기술 한다. else문은 생략 가능하다.

중요한 점은 추가한 조건 중 맞는 조건이 모두 실행되는 것이 아니고 1개만 실행 된다. 조건1 에  $a > 0$  조건2에  $a > 1$ 라 기술 한다면 조건1과 조건2가 범위가 겹치면 조건 1이 실행되고 조건2는 절대로 실행하지 않는 코드가 된다.

다음 예제를 실행해 보자.

1.입력 else 생략가능

```
if(5<3){
    System.out.println(1);
}else if(6<3){
    System.out.println(2);
}
```

출력

출력없음

2.입력 else if 여러개중 1개 만 출력

```
if(5>3){
    System.out.println("조건에 맞음");
}else if(6>3){
    System.out.println("조건이 맞으나 이미 맞는 조건이있어서 출력되지 않음");
}else{
    System.out.println("모든 조건에 맞지 않으면 출력");
}
```

출력

조건에 맞음

### 3. 입력

```
int hi1=4;
if(hi1>3){
    System.out.println(1);
}else if(hi1>5){
    System.out.println(2);
}else{
    System.out.println(3);
}
```

출력

?

### 4. 입력

```
int j1=0;int j2=1;int j3=2;
if(j1>3){
    System.out.println(1);
}else if(j2>j3){
    System.out.println(2);
}else{
    System.out.println(3);
}
```

출력

?

### 5. 입력

```
Scanner scanner = new Scanner(System.in);
System.out.print("점수를 입력하세요: ");
String input = scanner.nextLine();
int score = Integer.parseInt(input);
if (score >= 90) {
    System.out.println("A학점입니다.");
} else if (score >= 80) {
    System.out.println("B학점입니다.");
} else if (score >= 70) {
    System.out.println("C학점입니다.");
} else if (score >= 60) {
    System.out.println("D학점입니다.");
} else {
    System.out.println("F학점입니다.");
}
```

출력

?

### 6. 입력

else if문은 여러개의 조건중 하나의 조건만 실행한다.

```
Scanner scanner = new Scanner(System.in);
```

```

System.out.print("Enter a number between 1 and 3: ");
String input = scanner.nextLine();
int num = Integer.parseInt(input);
if (num == 1) {
    System.out.println("You entered 1");
} else if (num == 2) {
    System.out.println("You entered 2");
} else if (num == 3) {
    System.out.println("You entered 3");
} else {
    System.out.println("Invalid input");
}

```

## 7. 입력

사용자 입력값에 따라 다른 동작을 수행하는 예제

```
Scanner scanner = new Scanner(System.in);
```

```

System.out.print("Enter a command (start, stop, pause): ");
String command = scanner.nextLine();

if (command.equals("start")) {
    System.out.println("Starting the program");
} else if (command.equals("stop")) {
    System.out.println("Stopping the program");
} else if (command.equals("pause")) {
    System.out.println("Pausing the program");
} else {
    System.out.println("Invalid command");
}

```

switch문에 대해서 생각해 보자.

switch문은 여러 조건중 하나의 블록만 실행되는 제어문이다. else if문 처럼 조건에 맞는 코드 블록중 만족하는 하나의 코드 블록만 실행된다 는 공통점이 있고 차이점은 else if문은 범위 조회가 가능하지만 switch문에서는 범위 조회는 허용하지 않고 1:1 매칭이 되는 경우만 허용한다. 복잡한 else if문 보다 switch문을 사용하는 것이 코드를 보기에 좀더 낫다.

```

int i=5;
if(i==5) {
    System.out.println("최우수");
} else if(i==4) {
    System.out.println("우수");
} else if(i==3) {
    System.out.println("보통");
} else {
    System.out.println("불합격");
}

```

다음을 확인해 보자.

5~0 사이의 숫자를 입력 받아 5이면 최우수, 4이면 우수, 3이면 보통, 2,1,0 은 불합격이 출력 되도록 프로그램을 구현해 보자.

하나의 조건을 가지고 둘 중 하나의 코드 블록을 선택 하여 실행 할 때는 ifelse문을

사용하고 여러 개의 조건중 조건에 맞는 하나의 코드 블록을 선택해서 실행할 때에는 `elseif`문을 사용한다.

`elseif`문 중 조건이 1:1로 매핑되면 `switch`문으로 변경해 사용할 수도 있다. `switch`문을 사용하는 방법은 다음과 같다. 확인해 보고 이해 하자. 다음은 상위 코드를 `switch`문으로 변경한 것이다.

```
int i=5;
switch(i){
    case 5:
        System.out.println("최우수");
        break;
    case 4:
        System.out.println("우수");
        break;
    case 3:
        System.out.println("보통");
        break;
    default:
        System.out.println("불합격");
}
```

```
int i=4;
switch(i) {
case 5:
    System.out.println("최우수");
    break;
case 4:
    System.out.println("우수");
    break;
case 3:
case 2:
case 1:
case 0:
    System.out.println("불합격");
    break;

default:
    System.out.println("미응시");

    break;
}
```

`switch` 문은 특정 변수를 여러 경우(`case`)와 비교하여, 일치하는 경우에 해당하는 블록의 코드를 실행하는 제어문입니다.

각 `case`는 특정 값에 매칭되며, `switch` 옆의 변수 값과 일치하는 `case`를 찾아 해당하는 블록의 코드를 실행하다.

`break`를 만나 해당 `switch`문을 빠져 나온다.

`break`문이 없으면 다음 `case`문에 기술된 코드들이 `break`를 만날때 까지 계속 실행된다.

만약 어떤 `case`와도 일치하지 않는 경우에는 `default` 블록의 코드가 실행된다.

`case` 문 다음에 `break;`가 없으면 다음 줄이 이어서 실행된다. 해당 코드를 보면 `case 3,2,1,0`이 모두 같은 결과가 출력 되도록 만들어야 해서 `break;` 문을 사용하지 않고

다음 `case`문이 실행되게 만들어 `case 3,2,1,0`의 경우 모두 불합격이 출력되고 `switch`문을 빠져 나가게 만들었다. `default` 안에 있는 `break;`의 경우 기술하지 않아도 `switch`문의 맨 마지막 부분 이어서 스위치 문을 빠져 나가 생략이 가능하다.

```
switch(조건변수){//조건변수의 값에 일치하는 case 문으로 이동한다.
```

```
    case 변수값1:
        //조건변수가 변수값1일때 실행되는 코드
        break;           //break를 만나면 switch문을 빠져나감
```

```
    case 변수값2:
        //조건변수가 변수값2일때 실행되는 코드
        break;
```

```
    case 변수값3:
        //break가 없으면 빠져나가지 않고 아래로 흘러내린다.
```

```
    default:
        //어떤 조건도 맞지 않을 경우 실행되는 코드
```

```
}
```

switch문은 조건 변수로 double형과 범위 조회를 할 수 없다.

반드시 1:1 매핑되는 else if문만 switch문으로 변경할 수 있다.

문자열의 경우 1.8에서 지원 되지만 과거 버전은 문자열을 지원하지 않는다.

다음은 switch관련 예제를 확인해 보자.

#### 1.입력

int a=3;//a=1 a=6으로 변경해서 실행해 보자.

```
switch(a){
    case 1:
        System.out.println(1);
        break;
    case 2:
        System.out.println(2);
        break;
    case 3:
        System.out.println(3);
        break;
    case 4:
        System.out.println(4);
        break;
    default:
        System.out.println(5);
}
```

출력

3



## 2. 사용자 입력값에 따라 다른 메시지를 출력하는 예제

```
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a number between 1 and 3: ");
int num = Integer.parseInt(scanner.nextLine());
switch (num) {
    case 1:
        System.out.println("You entered 1");
        break;
    case 2:
        System.out.println("You entered 2");
        break;
    case 3:
        System.out.println("You entered 3");
        break;
    default:
        System.out.println("Invalid input");
        break;
}
```

## 3. 월을 입력받아 해당하는 계절을 출력하는 예제

```
Scanner scanner = new Scanner(System.in);
System.out.print("Enter a month (1-12): ");
int month = Integer.parseInt(scanner.nextLine());
switch (month) {
    case 3:
    case 4:
    case 5:
        System.out.println("It's spring");
        break;
    case 6:
    case 7:
    case 8:
        System.out.println("It's summer");
        break;
    case 9:
    case 10:
    case 11:
        System.out.println("It's fall");
        break;
    case 12:
    case 1:
    case 2:
        System.out.println("It's winter");
        break;
    default:
        System.out.println("Invalid input");
}
```

```

        break;
    }
4.
Scanner scanner = new Scanner(System.in);
System.out.print("Enter numbers: ");
double num1 = Double.parseDouble(scanner.nextLine());
System.out.print("Enter numbers: ");
double num2 = Double.parseDouble(scanner.nextLine());
System.out.print("Enter an operation (+, -, *, /): ");
String operator = scanner.nextLine();
switch (operator) {
    case "+":
        System.out.println("Result: " + (num1 + num2));
        break;
    case "-":
        System.out.println("Result: " + (num1 - num2));
        break;
    case "*":
        System.out.println("Result: " + (num1 * num2));
        break;
    case "/":
        if (num2 != 0) {
            System.out.println("Result: " + (num1 / num2));
        } else {
            System.out.println("Cannot divide by zero");
        }
        break;
    default:
        System.out.println("Invalid operator");
}

```

```

4.
Scanner scanner = new Scanner(System.in);
System.out.print("Enter an alphabet: ");
String alphabet = scanner.nextLine();
switch (alphabet) {
    case "a":
    case "A":
        System.out.println("You entered A");
        break;
    case "b":
    case "B":
        System.out.println("You entered B");
        break;
    // 알파벳 c ~ z까지 모두 동일한 방식으로 처리할 수 있음
    default:
        System.out.println("Invalid input");
        break;
}

```

```
}
```

## 연습문제

1. 다음 else if문은 switch문으로 switch문은 elseif문으로 변경해보자.

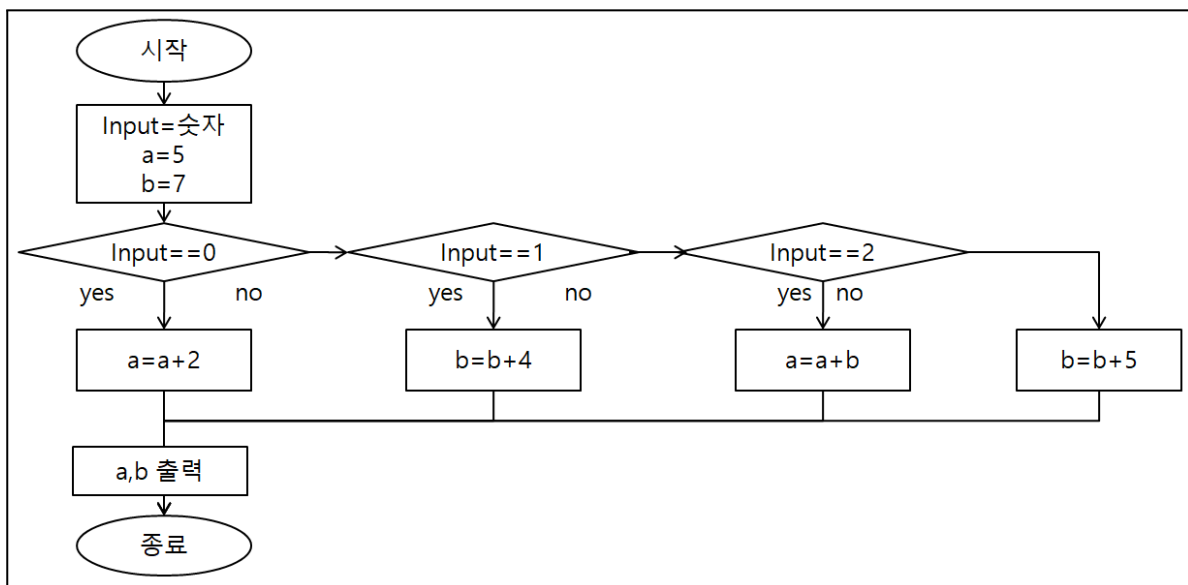
```
int i=0;
if(i==0) {
    i++;
}else if(i==2) {
    i=i-2;
}else if(i==3) {
    i=3;
}
System.out.println(i);
```

```
int i=0;
switch(i) {
    case 2: i++; break;
    case 3:
    case 4: i=i+2; break;
    case 6: i++; break;
}
System.out.println(i);
```

2. 1~5사이의 숫자를 변수 c에 저장하여 한글로 출력하는 switch문을 만들어 보자.

다음은 else if문과 switch문으로 풀어 보자.

3. 다음 순서도와 동일하게 실행되도록 else if문과 switch문으로 기술하시오.



4. 점수를 입력받아 90점이상은 수, 80점이상은 우, 70점 이상은 미 ..로 출력하는 프로그램을 else if문과 switch문으로 만들어 보자. 범위 조회여서 switch문으로 처리하기 어려울 것 처럼 보이지만 %연산자를 사용하면 switch문을 사용할 수 있다.  $100/10==10$ 이고  $99\sim90/10==9$ 이다. 정수 연산이기 때문에 소수점이 없다.  $89\sim80/10==8$ 이다.  $74/10==7$ 이 된다. 이를 이용하면 충분히 switch문으로 구현할 수 있다. 수우미양가를 출력하는 프로그램을 switch문으로 구현해 보자.

5. 1~5사이의 숫자를 입력받아 해당 숫자를 한글로 출력하는 프로그램을 작성해 보자.
6. 달을 입력받아 해당달이 28일인지 30일인지 31일인지 출력하는 프로그램을 구현해 보자.
7. 나이를 입력 받아 해당 나이의 때의 학력이 초,중,고 중 어디에 해당하는지 출력해보자.

---

## > 06. 조건문과 논리연산자 - 예제 연습

---

논리 연산자는 boolean값 true, false을 가지고 하는 연산이다.

논리연산자는 and,or,not이 있고 &&, ||, ! 으로 표기 한다.

&& and, || or 연산자는 두개의 boolean값을 가지고 하나의 boolean값을 만드는 일을 한다.

&& and연산자는 두 조건식이 모두 true일때 true를 생성한다.

|| or연산자는 두 조건식중 하나라도 true일때 true를 생성한다.

| 문자는 키보드의 엔터 위에 있다.

! not 조건식의 결과를 반전할때 사용한다. 조건식이 true이면 false, false이면 true가 된다.

남자이고 17세 이상일때 둘다 만족할때 if문을 실행하려면 && 으로 두 조건식을 연결하고  
남자이거나 17세 이상일때 둘중 하나만 만족할때 if문을 실행하려면 ||으로 두 조건식을  
연결하면 된다. 조건의 결과를 반전하여 if문을 실행하려면 !를 사용 할 수 있다.

1. AND 연산자(&&)를 이용한 예제:

```
int x = 4;
int y = 10;
if (x > 0 && y > 0) {
    System.out.println("x와 y는 모두 양수입니다.");
}
```

2. OR 연산자(||)를 이용한 예제:

```
int x = 5;
int y = -12;
if (x > 0 || y > 0) {
    System.out.println("x와 y 중에 적어도 하나는 양수입니다.");
}
```

3.NOT 연산자(!)를 이용한 예제:

```
boolean isTrue = true;
if (!isTrue) {
    System.out.println("isTrue는 false입니다.");
} else {
    System.out.println("isTrue는 true입니다.");
}
```

```
}
```

4. AND 연산자(&&)와 OR 연산자(||)를 함께 이용한 예제:

조건식이 3개 이상 되고 &&와 || 가 섞여 있으면 반드시 중괄호를 사용해서 기술 하는 습관을 가지자 두연산자는 우선순위가 달라 생각과 다른 결과를 얻을 수 있다. 섞어 사용하지 말자.

```
int x = 5;
int y = 10;
if ((x > 0 && y > 0) || (x < 0 && y < 0)) {
    System.out.println("x와 y는 둘 다 양수이거나 둘 다 음수입니다.");
} else {
    System.out.println("x와 y는 서로 다른 부호입니다.");
}
```

5. NOT 연산자(!)를 이용한 문자열 비교 예제:

```
boolean b=true;
if (!b) {
    System.out.println("true입니다.");
} else {
    System.out.println("false입니다.");
}
```

6.

```
int i=5;
if (!(i<3)) {
    System.out.println("true입니다.");
} else {
    System.out.println("false입니다.");
}
```

7. 삼항 연산자(?:)를 이용한 예제:

```
int x = 10;
int y = x > 5 ? 20 : 30;
System.out.println("y = " + y);
```

---

## > 07. 조건문과 논리연산자

---

논리 연산자는 boolean값 true, false을 가지고 하는 연산이다.

논리연산자는 and,or,not이 있고 &&, ||, ! 으로 표기 한다.

&& and, || or 연산자는 두개의 boolean값을 가지고 하나의 boolean값을 만드는 일을 한다.

&& and연산자는 두 조건식이 모두 true일때 true를 생성한다.

|| or연산자는 두 조건식중 하나라도 true일때 true를 생성한다.

| 문자는 키보드의 엔터 위에 있다.

! not 조건식의 결과를 반전할때 사용한다. 조건식이 true이면 false, false이면 true가 된다.

if문에 2개의 조건식을 합쳐서 하나로 표현할 때 사용한다.

논리 연산자를 이용해 모든 입력과 출력을 만들어 놓은 표를 진리표라 이야기하고 아래에 진리표가 있으니 확인해 보자.

!은 true일 때 false, false일 때 true를 만들어 내는 not이라는 단항 연산자이고 3개중에 우선순위가 가장 높다. not and or 순으로 우선순위를 가진다.

중복된 여러개의 if문을 하나의 if문으로 쓰고자 할때 논리연산자를 사용한다.

만약 a가 4보다 크고 b가 3보다 크면 합격 아니면 불합격을 출력하는 프로그램을 구현 하려면 다음과 같이 구현하여야 한다.

```
if (a>4) { //2
    if (b>3) {
        System.out.println("합격");
    }
}
```

```

        } else {
            System.out.println("불합격");
        }
    } else {
        if (b>3) {
            System.out.println("불합격");
        } else {
            System.out.println("불합격");
        }
    }
}

```

상위 코드에서 불합격을 찍는 부분의 코드가 중복해서 사용 되었다. 이런 형태의 코드를 개선해서 중복을 없애려면 논리 연산자를 사용하여  $a > 4$  와  $b > 3$ 이 모두 맞을때 출력 되도록 기술하려면  $a > 4 \&\& b > 3$ 을 하면 된다.

$\&\&$  연산자는 앞과 뒤에 각각 true 또는 false 값을 가지는 두 개의 조건식을 연결하여, 두 조건식이 모두 true일 경우에만 결과 값으로 true를 반환하는 논리 연산자다. 그 외의 경우에는 false를 반환한다. 즉,  $\&\&$  연산자는 논리적인 AND 연산을 수행하는 연산자이다.

이 연산자를 이용해서 상위 코드를 다음과 같이 변경할 수 있다.

```

if (a>4&&b>3) {//출력값이 상위 코드와 동일하게 동작
    System.out.println("합격");
} else {
    System.out.println("불합격");
}

```

a가 10이고 b가 20이라면  $a > 4 \&\& b > 3$  은 true  $\&\&$  true가 되어  $\&\&$ 앞뒤에 true가 오니 최종 실행 결과가 true가 된다.

a가 10이고 b가 1이라면  $a > 4 \&\& b > 3$  은 true  $\&\&$  false되어  $\&\&$ 앞뒤가 모두 true가 아니므로  $\&\&$  계산결과는 false가 된다.

$a > 4$ 이 true이고  $b > 3$  이 true이면  $a > 4 \&\& b > 3$ 이 결과가 true이다.  $a > 4$ 와  $a > 3$ 둘다 true일때 true이지 둘중 하나라도 true가 아니면 실행 결과는 false가 된다.

만약  $a > 4$  와  $b > 3$  중 하나만 만족하면 합격 아니면 불합격을 출력하는 프로그램을 구현하려면 다음과 같이 구현하여야 한다.

```

if (a>4) {//2
    if (b>3) {
        System.out.println("합격");
    } else {
        System.out.println("합격");
    }
} else {
    if (b>3) {

```



```

        System.out.println("합격");
    } else {
        System.out.println("불합격");
    }
}

```

상위 코드는 합격을 찍는 부분의 코드가 중복해서 사용 되었습니다. 이런 코드 중복을 제거하기 위해,  $a > 4$  또는  $b > 3$  중 하나라도 맞을 때 합격 메시지를 출력 하도록 코드를 수정하려면  $a > 4 \ || \ b > 3$ 을 사용하면 됩니다.  $\|$  연산자는 앞과 뒤에 각각 true 또는 false 값을 가지는 두 개의 조건식을 연결하여, 두 조건식 중 하나라도 true일 경우 결과값으로 true를 반환하는 논리 연산자입니다. 둘 다 false일 경우에는 false를 반환합니다. 즉,  $\|$  연산자는 논리적인 OR 연산을 수행하는 연산자입니다.

이 연산자를 이용해서 상위 코드를 다음과 같이 변경할 수 있다.

```

if (a>4 || b>3) {
    System.out.println("합격");
} else {
    System.out.println("불합격");
}

```

논리 연산자를 좀더 자세히 살펴보자.

논리 연산자는 불리언 (Boolean)데이터들을 가지고 새로운 불리언 결과를 만들어 내는 연산자이다. 조건문과 나중에 배울 반복문에서 조건식을 2개 이상 사용하여 하나의 결과를 얻고자 할 때 논리 연산자를 사용한다. 다음 같은 경우가 조건식이 2개 있는 경우이다.

집에들어 가는 방법이 비밀번호와 key 두 가지가 있다.

비밀번호가 111이고 key는 108호 키이다.

이때 집에 들어갈 수 있는 방법은 비밀번호와 key가 둘다 맞아야 가능한 경우가 있고, 둘중에 하나만 맞아도 들어갈 수 있는 방법이 있을 것이다. 결과가 true일때 집에 들어갈 수 있다고 본다면 다음과 같이 생각할 수 있다.

둘 다 맞아야 열어 줄 경우  $\&\&$  연산자, 둘 중에 하나만 맞아도 열어 줄 경우  $\|$  연산자를 사용한다. 전자의 경우 수식은 비밀번호==1111  $\&\&$  key==108 이고 후자의 경우 수식은 비밀번호==1111  $\|$  key==108 이다.  $\&\&$ ( 앤드) 연산자는 비교 대상이 둘 다 true 일 때만 결과가 true가 되고  $\|$ (오어) 연산자는 비교 대상 둘 중에 하나 라도 true가 존재하면 true를 결과로 남겨 준다.

둘다 맞아야 들어가는 경우 비밀번호==1111  $\&\&$  key==108 은 비밀번호는 1111이고 key는 108이어야 둘다 true가 되어 true  $\&\&$  true == true 다음과 같이 연산결과가 true가 된다. 비밀번호가 123 이면 key가 108이든 1111이든 관계없이 비밀번호, key 둘다 true가 아니므로 연산 결과는 false가 된다.

둘중에 하나만 맞아도 들어가는 경우 비밀번호==1111  $\|$  key==108 의 경우 비밀번호가 1111 이고 key가 100이면 true  $\|$  false 가 되어서 둘중에 하나라도 true가 있으면

true이므로 결과가 true가 되어 집에 들어갈 수 있다. 비밀번호가 112 이고 key가 123 이면 둘다 false여서 false || false == false 가되어 최종 결과가 false가 되어 집에 못들어 간다. 둘다 맞을 경우는 당연히 들어간다. 다음 코드를 확인해 보자.

```
int num=1111;
String key="108호";
//비밀번호가 111이고 key는 "108호"이다.
//1. 둘다 맞아야 들어갈 수 있음 &&
if(num==111&&key.equals("108호")) {//객체의 비교는 .equals를 사용한다.
    System.out.println("들어감");
}else {
    System.out.println("못들어감");
}
//2. 둘중에 하나만 맞으면 들어 갈 수 있음 ||
if(num==111||key.equals("108호")) {
    System.out.println("들어감");
}else {
    System.out.println("못들어감");
}
//3. 비밀번호만 맞으면 들어감
if(num==111) {
    System.out.println("들어감");
}else {
    System.out.println("못들어감");
}
//4. 키만 맞으면 들어감
if(key.equals("108호")) {
    System.out.println("들어감");
}else {
    System.out.println("못들어감");
}
```

다음은 진리표라고 하는 것이다. 불 연산은 연산 대상이 많지 않아 모든 연산 방법을 표로 만들어서 외우고 다니는 경우가 많다. 아래 표는 연산에 필요한 boolean자료형 a,b,c , 둘다 true일때만 true인 and연산자, 둘 중에 하나가 true면 true인 or연산자, true이면 false false이면 true 반전되는 not연산자를 가지고 연산한 결과표이다. 논리 연산자는 연산자 우선순위가 모두 다르다는 것을 조심하자.

!이 가장 높고 다음은 <,>,<=,>= 다음은 == 다음은 &&, 다음은 ||이다.

최상단 부분만 남겨놓고 나머지를 가린 다음 답을 맞출수 있을때 까지 공부해 보자.

a	b	a&&b	a  b	!a	c	a  b  c	a&&b&&c	a  b&&c
---	---	------	------	----	---	---------	---------	---------

true	true	true	true	false	f	true	false	true
true	false	false	true	false	f	true	false	true
false	true	false	true	true	f	true	false	false
false	false	false	false	true	f	false	false	false

논리 연산자를 사용하여 아래표의 왼쪽을 보고 오른쪽을 구현 할 수 있도록 확인해 보자.

조건식	판별식
6보다 큰수	a>6
두수 모두 100 이 넘는 경우	a>100 && b>100
세수중 적어도 하나가 100이 넘는 경우	a>100    b>100    c>100
부모 동의가 있거나 20살이 넘는 경우	isState==true    a>20
세수중 하나라도 음수가 있는 경우	a<0    b<0    c<0
30과 40 사이 숫자	a>=30 && a<=40 30<a<40 은 안됨
30과 40 사이를 제외한 모든 숫자	!(a>=30 && a<=40)
60보다 작거나 100보다 큰 숫자	a<60    a>100

다음 문제는 2개 씩 뽑아서 각의 프로그램의 ?표부분에 적절한 내용을 적어서 동일하게 동작 하도록 만들어 보자.

1. 같은 문자열이 2번 출력 되도록 적절한 조건식을 ?부분에 넣어보자.

```
// 입력값에 따라 동일한 결과가 나오도록 두 조건식을 완성 하시오.
boolean isFlag1 = true or false, isFlag2 = true or false;// 입력값
```

```

if (isFlag1) {
    if (isFlag2) {
        System.out.println("맞음");
    } else {
        System.out.println("틀림");
    }
} else {
    if (isFlag2) {
        System.out.println("틀림");
    } else {
        System.out.println("틀림");
    }
}
if (?) {//적절한 조건식을 ?부분에 넣어보자.
    System.out.println("맞음");//출력값이 상위 코드와 동일하게 동작하도록
} else {
    System.out.println("틀림");
}

```

2. 같은 문자열이 2번 출력 되도록 적절한 조건식을 ?부분에 넣어보자.

```

boolean isFlag1 = true or false, isFlag2 = true or false;// 입력값
if (isFlag1) {
    if (isFlag2) {
        System.out.println("맞음");
    } else {
        System.out.println("맞음");
    }
} else {
    if (isFlag2) {
        System.out.println("맞음");
    } else {
        System.out.println("틀림");
    }
}
if (?) {// 적절한 조건식을 입력하시오
    System.out.println("맞음");//출력값이 상위 코드와 동일하게 동작하도록
} else {
    System.out.println("틀림");
}

```

3. 같은 문자열이 2번 출력 되도록 적절한 조건식을 ?부분에 넣어보자. !연산자가 필요하다.

```

boolean isFlag1 = true or false, isFlag2 = true or false;// 입력값
if (isFlag1) {
    if (isFlag2) {
        System.out.println("틀림");
    } else {

```

```

        System.out.println("틀림");
    }
} else {
    if (isFlag2) {
        System.out.println("틀림 ");
    } else {
        System.out.println("맞음");
    }
}
}
if (?) { // 적절한 조건식을 입력하시오
    System.out.println("맞음"); // 출력값이 상위 코드와 동일하게 동작하도록
} else {
    System.out.println("틀림");
}

```

4. 같은 문자열이 2번 출력 되도록 적절한 조건식을 ?부분에 넣어보자.

```

boolean isFlag1 = true or false, isFlag2 = true or false; // 입력값
if (isFlag1) {
    if (isFlag2) {
        System.out.println("틀림");
    } else {
        System.out.println("맞음");
    }
} else {
    if (isFlag2) {
        System.out.println("맞음 ");
    } else {
        System.out.println("맞음");
    }
}
}
if (?) { // 적절한 조건식을 입력하시오.
    System.out.println("맞음"); // 출력값이 상위 코드와 동일하게 동작하도록
} else {
    System.out.println("틀림");
}
}

```

- |                        |                        |
|------------------------|------------------------|
| 1. isFlag1&&isFlag2    | 2. isFlag1  isFlag2    |
| 3. !(isFlag1  isFlag2) | 4. !(isFlag1&&isFlag2) |

다음 문제를 풀어보자. 문자열의 비교는 .equals를 사용해야 한다. 복잡한 내용이 있는데 일단은 참조 데이터는 .equals로 비교한다고 생각하면된다. scanner를 이용해서 사용자 입력으로 얻어온 데이터는 모두 문자열이므로 .equal 를 사용해야 한다.

```
String str1 = "Hello"; String str2 = "Hello";
```

```

if (str1.equals(str2) == true) {

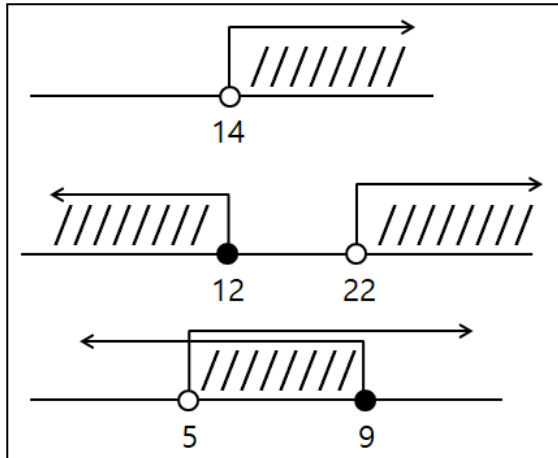
    System.out.println("str1과 str2는 같습니다.");

} else {

    System.out.println("str1과 str2는 다릅니다.");

}

```



1. 왼쪽 이미지에서 색칠한 부분의 값들이 들어 왔을때 true가출력 되는 조건식을 가진 if문을 각각 3개 만들어 색칠한 부분인지 아닌지 출력하는 프로그램 형태로 구현해 보자.

2. 왼쪽 이미지에서 색칠 안한 부분의 값들이 들어 왔을때 true가되는 조건식을 만들어 보자. not연산자는 사용방법과 not를 사용하지 않는 방법 각각 2개씩 총 6개 if문 수식에 넣어서 만들어 보자.

3. 입력받은 수가 3의 배수 이거나 7의 배수이면 ‘3또는 7의 배수’가 출력되고, 2의 배수도 되고 5의 배수도 되면 ‘2와 5의 배수’를 출력하고 두가지 경우 다 만족하면 2가지 다 출력하는 프로그램을 만들어 보자.

1번위치	2번위치
4번위치	3번위치

4. 왼쪽 이미지는 하나의 점이 x,y좌표를 가질때 위치할 수 있는 장소를 4개로 구분한 것이다. x,y가 모두 양수이면 2번 위치에 점이 존재하게 될것이고 모두 음수이면 4번위치에 점이 위치하게 될것이다. 해당 점의 좌표를 입력받아 해당 점이 어느 위치에 있는지 출력해 보자. x,y축 위에 있을 때는 x,y축

위에 있다고 출력하면 된다.

4. 버스요금계산하는 프로그램을 구현해 보자. 15세 이하는 1300원 초과는 1600원이고 버스카드로 결제하면 100원 할인해 준다.

문자열을 boolean형으로 변경하는 방법은 다음과 같다.

```
Boolean a= Boolean.parseBoolean("true");
```

입력: 16엔터 true엔터

출력: 버스요금은 1500원 입니다.

5. 다음을 구현하려면 어떤문을 사용해야 하는지 고민해 보고 사용자 입력을 받아 결과가 출력되도록 구현해 보자. (if elseif ifelse 논리연산자)

a. 밥을 먹었으면 ‘밥을 먹었음’ 이 출력 되고 밥을 먹지 않았으면 아무것도 출력되지 않는 형태의 프로그램구현.

b. 빵을 먹었으면 ‘빵을 먹었음’이 출력되고 빵을 먹지 않았으면 ‘밥을 먹었음’이 출력되는 형태의 프로그램 구현.

c. 밥을 먹었으면 ‘밥을 먹었음’, 빵을 먹었으면 ‘빵을 먹었음’ 아무것도 먹지 않았으면 ‘아무것도 먹지 않음’이 이 출력 되도록 프로그램 구현.

d. 식사로 ‘밥’과 ‘빵’이 있고 후식으로 국,우유, 아이스크림, 커피가 있는데 밥을 먹으면 국과 아이스크림중 하나를 빵을 먹으면 우유 커피 중 하나를 후식으로 먹을수 있도록 구현해 보자.

6. 다음과 같이 기술하면 오늘의 요일이 i에 숫자로 들어간다. i가 7일 경우 일요일이고 월화수목금토일은 1234567과 같다. i값을 가지고 오늘의 요일을 출력하는 프로그램을 만들어 보자.

//빨간줄에 마우스를 올리면 импорт 할 패키지가 뜬다.

//확인해서 해당 클래스를 클릭해서 자동으로 импорт하자.

```
LocalDateTime now = LocalDateTime.now();
```

```
System.out.println(now); //현재시간 출력
```

```
DayOfWeek dayOfWeek = now.getDayOfWeek();
```

//1 (월요일)부터 7 (일요일)까지의 값을 가집니다.

```
int i = dayOfWeek.getValue();
```

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

7. 한국사이즈를 입력받아 미국사이즈로 출력하시오.

미국 SIZE	S	M	L	XL	XXL
한국 SIZE	90~95	95~100	100~105	105~120	110이상

8. java,html,db과목의 점수를 입력받아. 평균이 60점 이상이면 합격, 평균이 60점 미만이면 불합격, 40점 이하인 과목이 하나라도 있으면 과락이 출력되도록 프로그램을 구현해 보자. 최종 결과 화면은 다음과 같다. java 35 과락, html 10과락,db 20과락 또는 평균 40로 60점미만 불합격 또는 과락 없이 평균 60이상 합격이 출력된다.

초·중·고교생			성인		
	분류	비고		분류	비고
95 미만	체중 미달	120 미만을 모두 정상 체중으로 나타내기도 함	100 미만	체중 미달	체중 미달을 95 미만으로 보기도 함
95 ~ 120	정상		100 ~ 110	정상	
120 ~ 130	경도 비만		110 ~ 120	과체중	
130 ~ 150	중도 비만		120 ~ 150	비만	
150 이상	고도 비만		150 이상	고도 비만	

9. 왼쪽은 비만도 산출식이다. 사용자 입력을 받아 결과가 아래 표와 같은 분류가 나오도록 프로그램을 구현하여라.

$$\frac{y}{(x-100) \times 0.9} \times 100$$

10. 다음 표를 확인하여 사용자가 알고 있는 신체사이즈중 하나를 입력받아 해당 미국사이즈와 한국사이즈를 출력하시오. 겹치는 부분은 본인이 원하는 쪽으로 해석 해서 출력하자. 명확하지 않은 경계는 본인이 적절히 결정하자.

미국 SIZE		S	M	L	XL	XXL
한국 SIZE		90~95	95~100	100~105	105~120	110이상
목둘레	(cm)	33~35	36~37	38~39	41~42	43~45
가슴둘레	(cm)	86~91	96~102	107~112	117~122	127~132
팔길이	(cm)	78~81	81~84	84~86	86~89	89~91
허리둘레	(inch)	28~29	30~31	32~34	34~37	38~40

11. 다음과 같이 실행되는 계산기 프로그램을 만들어보자.

첫번째수 입력>>5, 연산자선택 1. + 2. - 3. \* 4. / >>3 두번째수 입력>>12

5\*12=60 입니다.

12. 가위,바위,보 문자열중 하나를 랜덤하게 화면에 출력하는 프로그램을 구현해보자.하나를 내는 것 처럼 구현 하려면 0,1,2 3개의 수를 랜덤하게 리턴하게 만든 다음 가위,바위,보 3개에 1:1 매핑하면 된다.

13.가위 바위 보 게임을 만들어서 이겼는지 졌는지 출력하는 프로그램을 만들어 보자.

```
Random random = new Random();
```

```
int number = random.nextInt(10); // 0부터 9까지의 정수를 반환
```



- 매개변수를 통해 0 이상 10 미만의 정수를 반환할 수 있습니다.
- 예를 들어, `random.nextInt(10)`은 0부터 9까지의 랜덤한 정수를 반환합니다. (즉, 10상한값이 아니라 범위의 크기를 지정합니다.)
- 이 형태는 가장 많이 사용되며, 특정 범위의 양의 정수를 얻을 때 유용합니다.
- `int randomNum = random.nextInt(3) + 5; //5,6,7를 얻을 수 있다.`

14. 6~10사이의 랜덤한 수를 만들어 보자. 총 수에 범위는 6,7,8,9,10 5개 이므로 랜덤 함수에 5를 곱하고 시작을 6부터 하니 더하기 6를 하면 될 것이다. 제대로 동작하는지 확인해 보자.

15. 1~45까지의 수를 랜덤하게 만들어 3개의 수를 뽑은 다음 본인이 입력한 3개의 수와 몇개 같은지 확인하는 프로그램을 배열로 만들어 보자.

소득금액	세율	누진공제액
1,200만원 이하	6%	없음
1,200만원 ~ 4,600만원 이하	15%	1,080,000
4,600만원 ~ 8,800만원 이하	24%	5,220,000
8,800만원 ~ 1억 5천만원 이하	35%	14,900,000
1억 5천만원 ~ 3억원 이하	38%	19,400,000
3억원 ~ 5억원 이하	40%	25,400,000
5억원 초과	42%	35,400,000

16. 왼쪽표를 기본으로 본인이 낼 세금이 얼마인지 출력하는 프로그램을 구현해 보자. 만약에 본인이 번돈이 1600만원 이라면 1200만원은 세율이 6%이고 나머지 400만원은 세율이 15%이다.

문제 풀이

8.

```
package com.the.ex;

public class JavaAvg {
    public static final int cutLine=60;
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //java,html,db과목의 점수를 입력받아.
        //평균이 60점 이상이면 합격, 평균이 60점 미만이면 불합격, 40점 이하인
        과목이
        //하나라도 있으면 과락이 출력되도록 프로그램을 구현해 보자.
        //최종 결과 화면은 다음과 같다.
```

//java 35 과락, html 10과락,db 20과락 또는  
//평균 40로 60점미만 불합격 또는 과락 없이 평균 60이상 합격이 출력된다.

```
int java=95;
```

```
int html=80;
```

```
int db=61;
```

```
int avg=(java+html+db)/3;
```

```
if(JavaAvg.cutLine<=java&&JavaAvg.cutLine<=html
```

```
&&JavaAvg.cutLine<=db&& JavaAvg.cutLine<=avg) {//
```

```
System.out.println("과락 없이 평균 "+JavaAvg.cutLine+"이상 합격
```

```
:"+avg);
```

```
}else {
```

```
// if(java<JavaAvg.cutLine) {//java 35 과락, html 10과락,db  
20과락 또는
```

```
// System.out.println("java :"+java+" 과락");
```

```
// }else if(html <JavaAvg.cutLine) {
```

```
// System.out.println("html :"+html+" 과락");
```

```
// }else if(db <JavaAvg.cutLine) {
```

```
// System.out.println("db :"+db+" 과락");
```

```
// }else {
```

```
// System.out.println(avg+"로 "+JavaAvg.cutLine+"점미만");
```

```
// }
```

```
if(java<JavaAvg.cutLine) {//java 35 과락, html 10과락,db
```

```
20과락 또는
```

```
System.out.println("java :"+java+" 과락");
```

```
}
```

```
if(html <JavaAvg.cutLine) {
```

```
System.out.println("html :"+html+" 과락");
```

```
}
```

```
if(db <JavaAvg.cutLine) {
```

```
System.out.println("db :"+db+" 과락");
```

```
}
```

```
if(avg<JavaAvg.cutLine){
```

```
System.out.println(avg+"로 "+JavaAvg.cutLine+"점미만");
```

```
}
```

```

        System.out.println("불합격");
    }
}

```

15.

```

package com.the.ex;
import java.util.Arrays;
import java.util.Scanner;
public class Java01 {
    public static void main(String[] args) {
        //15. 1~45까지의 수를 랜덤하게 만들어 3개의 수를 뽑은 다음
        //본인이 입력한 3개의 수와 몇개 같은지 확인하는 프로그램을 배열로 만들어

```

보자.

```

        int num[] =new int[3];
        int count=0;
        int inputNum=0;
        num[0]=(int)(Math.random()*45+1);
        num[1]=(int)(Math.random()*45+1);
        num[2]=(int)(Math.random()*45+1);

        Scanner scanner =new Scanner(System.in);
        System.out.println("숫자 입력>>");
        inputNum= scanner.nextInt();
        if(num[0]==inputNum) {
            count++;
        }
        if(num[1]==inputNum) {
            count++;
        }
        if(num[2]==inputNum) {
            count++;
        }

        System.out.println("숫자 입력>>");
        inputNum= scanner.nextInt();
        if(num[0]==inputNum) {
            count++;

```

```

    }
    if(num[1]==inputNum) {
        count++;
    }
    if(num[2]==inputNum) {
        count++;
    }

    System.out.println("숫자 입력>>");
    inputNum= scanner.nextInt();
    if(num[0]==inputNum) {
        count++;
    }
    if(num[1]==inputNum) {
        count++;
    }
    if(num[2]==inputNum) {
        count++;
    }

    System.out.println("배열내용:" + Arrays.toString(num));
    System.out.println("총 "+count+"개수 만큼 같은 값을 가진다.");
}
}

```

---

## > 08. 반복문

---

반복문은 같은 작업을 여러번 반복 할때 사용한다. 화면에 자기 이름을 100번 찍는다고 생각해 보자. `System.out.println("홍길동");` 과 같은 코드를 100번 적어야 할 것이다. 프로그램에서는 반복되는 작업을 쉽게 처리할 수 있는 반복문을 제공한다.

대표적인 반복문 3가지는 `while`, `do~while`, `for`문 이다.

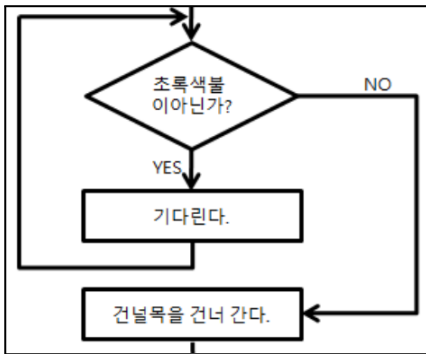
이 세 가지는 반복문은 사용 방법만 다를 뿐 결과는 동일 하다. 본인이 사용하기 편한 것 중 하나를 골라서 사용하면 된다.

// 다음 반복되는 코드를 `while`, `do~while`, `for` 반복문을 통해 간단히 기술 할 수 있다.

```
System.out.println("안녕하세요");
System.out.println("안녕하세요");
System.out.println("안녕하세요");
System.out.println("안녕하세요");
System.out.println("안녕하세요");
System.out.println("안녕하세요");
System.out.println("안녕하세요");
System.out.println("안녕하세요");
System.out.println("안녕하세요");
```

보통 `while`문은 반복 횟수가 명확하지 않을 때 사용하고, `for`문은 반복회수가 명확할때 사용하고, `do~while`은 반드시 1번 이상 실행해야 할 때 사용한다. 대부분 `for`문으로 대체 가능하여 `for`을 가장 많이 사용한다.

`while`문 사용 방법은 `while` 다음에 소괄호로 묶은 조건식을 쓰고 중괄호 안에 반복할 코드를 기술한다. 아래의 의사코드를 확인해 보자. 소괄호 안의 조건식이 `true`면 반복할 코드 블록을 반복하고 `false` 이면 `while`문 을 빠져 나간다. 상위에서 현상태가 빨간 불이면 ‘초록색 불이 아닌가?’라는 질문에 ‘예’라 답하면 기다린다 쪽으로 진행 방향이 이동 할 것이고 이 작업은 현 상태가 초록불이 될때 까지 반복하다가 초록 불이되면 반복문을 빠져 나와 ‘건널목을 건너 간다’ 쪽으로 진행 방향이 바뀔 것이다. 초록불인가? 대신에 초록불이 아닌가?로 기술한 이유는 `yes`일때 반복하게 구현해야 프로그램으로 변경하기 쉽다. 반복문은 조건이 `true`일때 반복한다.

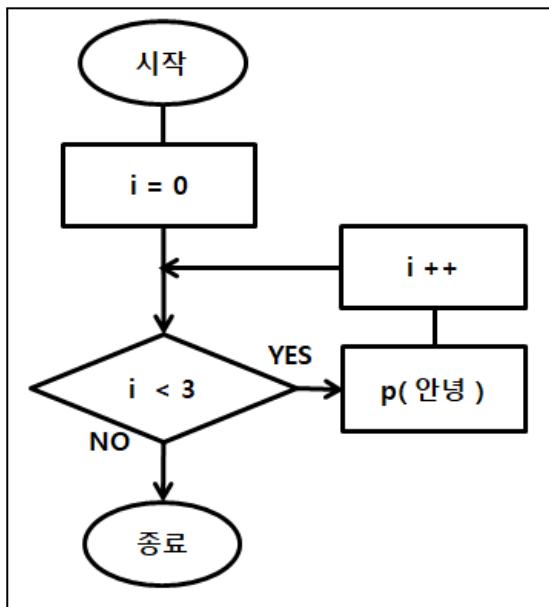


```

while(초록색 불이 아닌가?){
    기다린다.
}
건널목을 건너 간다.
  
```

while문의 조건식 부분이 코드블록을 반복할 것인지 말것 인지를 결정한다. true이면 반복 할 코드 블록이 실행되고 false이면 while문 종괄호 부분을 빠져나가 실행된다.

다음은 화면에 안녕을 3번 찍는 작업을 순서도와 while, for, do while문으로 자바 코드를 구현한 것이다. 순서도를 자세히 하나씩 따라가면서 화면에 안녕이 3번찍히는 것을 확인해 보자. i++은 i변수를 하나 증가 시킨다는 이야기 이다. i 가 3이라면 i++이후에 i값은 4가 된다.



맨 처음 i가 0으로 초기화 된 상태에서 반복문의 조건문에 도착하면 i 값은 0 이므로 조건식 i<3과 비교후 true이므로 Yes 부분으로 이동하여 p(안녕),i++ 가 순서대로 실행 되어 화면에 ‘안녕’이 출력되고 i 값은 1 증가하여 1이 된다. 그 다음 다시 반복문의 조건 i<3으로 이동하고 i값이 1이므로 여전히 참이다. 따라서, Yes 부분으로 이동 동하여 p(안녕),i++ 가 실행이 되어 화면에 ‘안녕’이 출력되고 i 값은 1 증가하여 2가 된다. 그 다음 다시 반복문의 조건 i<3으로 이동하고 i값이 2이므로 여전히 참이다. 따라서 Yes 부분으로 이동 동하여 p(안녕),i++ 가 실행이 되어 화면에 ‘안녕’이 출력되고 i 값은 1 증가하여 3이 된다. 그 다음 다시 반복문의 조건 i<3으로 이동하고 i값이 3이므로 거짓이

되고 반복 문을 빠져나와 결국 프로그램이 종료되고 이 프로그램은 실행중 화면에 ‘안녕’이 3번 출력 되게 된다.

자바코드를 while문으로 기술하면 다음과 같다.

```

int i=0; //초기식
while(i<3){//조건식
    System.out.println(i+"번째 안녕");
    i++;//변환식
}
  
```

자바코드를 for문으로 기술:

```
for(int i=0;i<3;i++){//for(초기식;조건식;변환식)
    System.out.println(i+"번째 안녕");
}
```

자바코드를 do~while문으로 기술:

```
int i=0;//선언식
do{
    System.out.println(i+"번째 안녕");
    i++;//변환식
}while(i<3)//조건식
```

상위 3개의 코드는 다음 설명을 읽어 보고 외워보자.

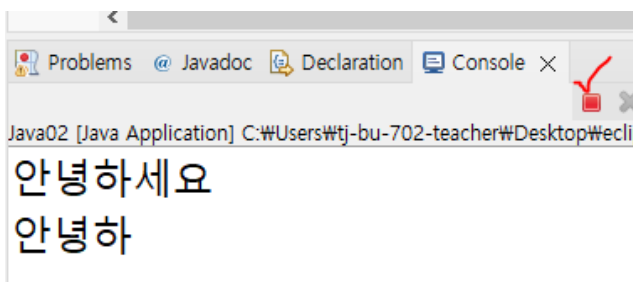
초기식은? 반복문의 조건식에 비교 값으로 사용되는 값을 초기화 하는 부분이다.

조건식은? 초기식을 가지고 반복 할 것인지를 결정하는 식이다.

변환식은? 초기식에 선언한 값을 변경하여 반복문을 빠져나가는 용도로 사용하는 식이다.

초기식이 변하지 않으면 반복문에서 조건식의 결과가 바뀌지 않아 무한 반복될 수 있다.  
이런 상황을 무한루프라한다. 일정 횟수 반복 하다가 while문을 빠져나갈 수 있도록 적절한 변환식을 만들어 주어야한다.

초기식, 변환식은 반복을 결정하는 조건식과 관련이 있어서 값을 잘못 결정하면 무한루프가 발생하거나 한번도 실행되지 않을 수 있어 잘 확인해서 사용해야 한다. 혹시 무한루프가 발생하면 콘솔창 상단에 빨간색 정사각형 아이콘을 클릭하면 프로그램 실행이 종료된다.



이런 문제가 발생하지 않게 초기식, 조건식, 변환식을 제대로 사용하고 있는지 확인하여야 한다.

```
초기식;
do{
    //반복할 코드블록
    변환식;
}while(조건식);
```

실행 순서를 확인해 보자. do while문의 경우 -초기식 -코드블록 -변환식 -조건식 순으로 진행 되다가 조건식이 참이면 (-코드블록 -변환식 -조건식) 부분을 반복 하다 조건식의 조건이 거짓이면 반복문을 빠져 나간다.

```
초기식;
while(조건식){
    //반복할 코드블록
    변환식;
}
```

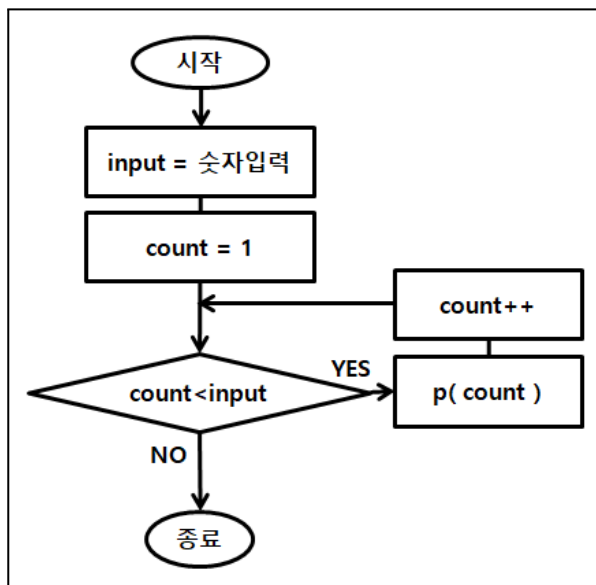
while문의 경우 -초기식 -조건식 -코드블록 -변환식 -조건식 순으로 진행 되다가 조건식이 참이면 (-조건식 -코드블록 -변환식) 순으로 괄호부분을 반복 하다가 조건이 거짓이면 반복문을 빠져 나간다.

```
for(초기식;조건식;변환식){
    //반복할 코드블록
}
```

for문의 경우 -초기식 -조건식 -코드블록 -변환식 -조건식 순으로 반복하다가 조건식이 참이면(-코드블록 -변환 -조건식) 순으로 괄호 부분을 반복 하다가 조건이 거짓이면 반복문을 빠져 나간다.

변환식은 보통 1개씩 증가해서  $i++$  같은 형태로 많이 사용 하지만 여러개 증가시키고 싶을때는  $i=i+2$ 나  $i=i+5$ 와 같은 변환식을 사용할 수도 있다.

상위 3개의 반복문에서 초기식 조건식 변환식이 어디에 있는지 명확히 확인해서 외워 쓸수 있도록 해보자. 순서도를 while문으로 어떻게 바뀌었는지 확인해 보고 나머지 for, do while은 그냥 외우자.



왼쪽 순서도를 확인해 보고 숫자를 하나 입력 받아 1부터 입력한 수까지 순서대로 화면에 출력 되도록 3가지 반복문으로 프로그램을 만들어 보자.



//반복문 없이 구현

//0부터 순서대로 3개의숫자를 출력해 보자.

```
int count=0;
```

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

```
count++;
```

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

```
count++;
```

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

```
count++;
```

//while문

```
int count=0;
```

```
while(count<3) {
```

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

```
    count++;
```

```
}
```

//for문

```
for(int count=0;count<3;count++) {
```

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

```
}
```

//do~while문

```
int count=0;
```

```
do {
```

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

```
    count++;
```

```
}while(count<3);
```

다음 추가적인 3가지 형태의 반복문을 확인해 보자.

// 1.초기식을 이용해서 원하는 숫자부터 실행 할 수 있다.

```
for(int i=5;i<10;i++) { //초기식 변수에 5를 넣으면 5부터 시작한다.
```

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

```
}
```

// 2.증감식을 이용해서 원하는 만큼 증가 시킬수 있다.

```
for(int i=0;i<10;i=i+2) { //초기식 변수를 2씩 증가 시킬수 있다.
```

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

```
}
```

// 3.반복문을 이용해서 합산 할 경우 합산할 변수는 반복문 밖에 선언해야 한다.

```
int sum1=0;
```

```
for(int i=1;i<11;i++) {
```

```
    sum1=sum1+i;
```

```
    System.out.println(sum1);
```

```

}
System.out.println("최종:"+sum1);

for(int i=1;i<11;i++) {
    int sum2=0;
    sum2=sum2+i;
    System.out.println(sum2);
}
//System.out.println(sum2); sum2는 지역변수라 접근할 수 없다.

```

다음 코드를 메인에 넣어 실행해 보자.

for문

1. 입력

```

for(int i=1;i<10;i++){
    System.out.print(i);
}

```

출력

123456789

2. 입력

```

for(int i=5;i<10;i++){
    System.out.print(i);
}

```

출력

56789

3. 입력

```

for(int i=5;i<10;i++){
    System.out.print(i);
}

```

출력

56789

4. 입력

```

for(int i=5;i<10;i=i+2){
    System.out.print(i);
}

```

```
}  
출력  
579
```

```
5. 입력  
for(int i=5;i<10;i=i+2){  
    System.out.print(i);  
}  
출력  
579
```

```
6. 입력  
for(int i=5;i<10;i=i+2){  
    System.out.print(i);  
}  
출력  
579
```

```
6. 입력  
for(int i=5;i<10;i=i+2){  
    System.out.print(i);  
}  
출력  
579
```

```
7. 입력  
int sum=0;  
for(int i=1;i<10;i++){  
    sum=sum+i;  
    System.out.print(sum);  
}  
출력  
?
```

```
8. 입력  
int sum=0;  
for(int i=5;i<10;i=i+2){  
    sum=sum+i;  
    System.out.print(sum);  
}  
출력  
?
```

do~while 문  
1.

```
int i=0;
int sum=0;
do{
    sum=sum+i;
}while(i<=10);
```

for문

1. 1부터 10까지의 합을 구하는 예제

```
int sum = 0;
for (int i = 1; i <= 10; i++) {
    sum += i;
}
System.out.println("1부터 10까지의 합: " + sum);
```

2. 구구단 2단을 출력하는 예제

```
for (int i = 1; i <= 9; i++) {
    System.out.println("2 x " + i + " = " + (2 * i));
}
```

3. 1부터 100까지의 홀수들의 합을 구하는 예제

```
int sum = 0;
for (int i = 1; i <= 100; i++) {
    if (i % 2 == 1) {
        sum += i;
    }
}
System.out.println(sum)
```

4. 1부터 100까지의 숫자 중 3의 배수만 출력하는 예제

```
for (int i = 1; i <= 100; i++) {
    if (i % 3 == 0) {
        System.out.println(i);
    }
}
```

5. 배열의 요소들 중에서 3의 배수만 출력하는 예제

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

6. 1부터 50까지의 숫자 중 5의 배수이면서 3의 배수인 숫자들의 합을 구하는 예제

```
int sum = 0;
for (int i = 1; i <= 50; i++) {
    if (i % 5 == 0 && i % 3 == 0) {
        sum += i;
    }
}
```

```

    }
}
System.out.println("1부터 50까지의 숫자 중 5의 배수이면서 3의 배수인 숫자들의 합: "
+ sum);

```

7. 1부터 100까지의 숫자 중 3과 5의 공배수인 숫자들의 합을 구하는 예제

```

int sum = 0;
for (int i = 1; i <= 100; i++) {
    if (i % 3 == 0 && i % 5 == 0) {
        sum += i;
    }
}
System.out.println("1부터 100까지의 숫자 중 3과 5의 공배수인 숫자들의 합: " + sum);

```

다음 문제를 3종류의 반복문으로 기술해 보자.

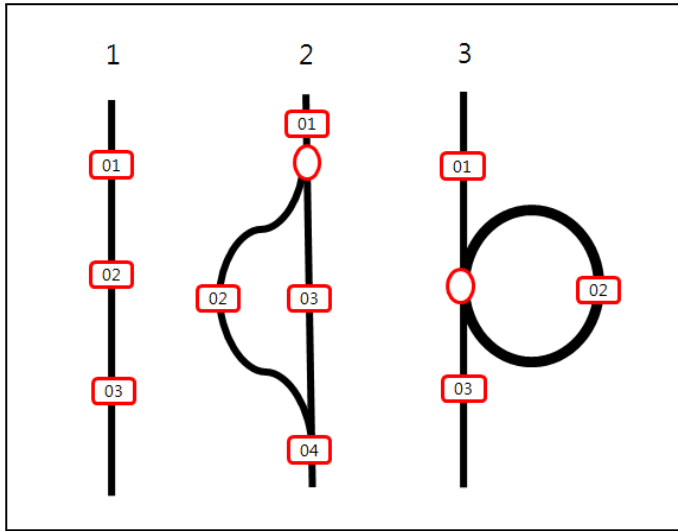
1. 1~10까지 출력 , 출력한 순의 총합 출력
2. 4~20까지 출력 , 출력한 순의 총합 출력
3. 5~29까지 출력 , 출력한 순의 총합 출력
4. 4~15까지 수중 4부터 3씩 증가한 수 출력
5. “안녕하세요”를 10번 찍어 보자.
6. 0~9까지 찍어보자.
7. 3~10까지찍어 보자.
8. 1~10의 합을 구해보자.
9.  $i=i+5$  변환식을 사용해서 10~100사이의  $i$ 값을 출력해 보자.

다음은 if문과 비슷한 3항 연산자이다. 많이 쓰이니 확인해 보자.

(조건식) ? (식1) : (식2); //조건식이 참이면 식1이 남고 조건식이 거짓이면 식2가 남는다. 삼항연산자 : ;은 if 조건문과 비슷하다.

$i=(a<20)?5+4:3+9$ ;이면  $a$ 가 10이면 조건식이 참이여서  $i$ 에  $5+4$ 가 들어가고  $a$ 가 30이면 거짓 이어서  $i$ 에  $3+9$ 가 들어간다.  $\text{int } b=(50>4)?1:3$ ;의 실행 결과는  $b=1$ 이다.

다음 문제들을 풀어보자.

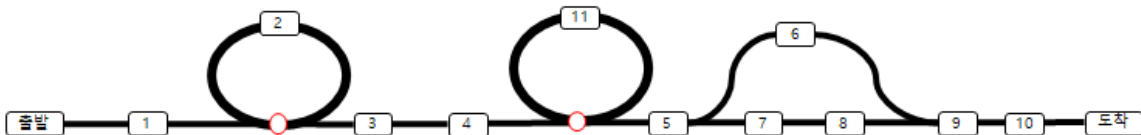


다음 이미지에서 조건 부분이  $a > 5$ 라고 할때 출력 결과가 다음과 같이 나오도록 a값 설정과 순서도와 코드를 만들어 보자.

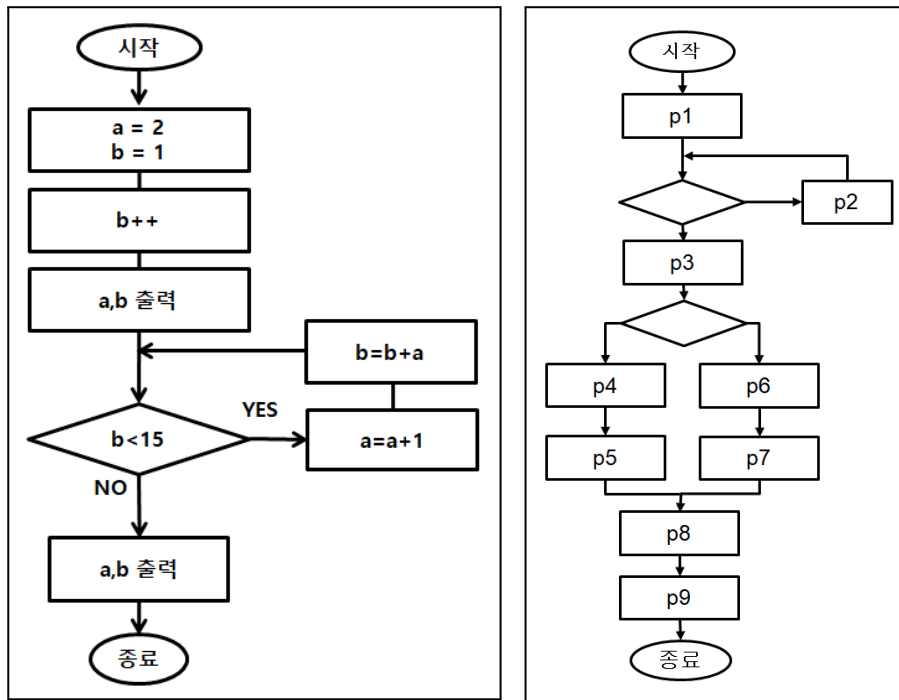
1. 왼쪽 이미지 에서 1번 절차문의 경우 01,02,03이 출력 되다.
2. 2번 조건문에서 01,02,04가 출력 되도록 a값을 설정하여 코드를 만들어 보자.
3. 2번 조건문에서 01,03,04가 출력 되도록 a값을 설정하여 코드를 만들어

보자.

4. 3번 반복문에서 01,03이 출력 되도록 a값을 설정하여 코드를 만들어 보자.
5. 3번 반복문에서 01,02,03이 출력 되도록 a값을 설정하여 코드를 만들어 보자.
6. 3번 반복문에서 01,02,02,02,03이 출력 되도록 a값을 설정하여 코드를 만들어 보자.



7. 상위와 같은 형태의 출력이 가능하도록 기차길을 만들어 출력 되도록 구현 해보자. 조건에 사용한 변수는 코드 맨 상위에 기술해서 출력 결과를 변경 할 수 있도록 해보자.
8. 아래 왼쪽 이미지 순서도 실행 결과가 어떻게 되는지 생각해 본 다음 코드로 작성해 실행 결과를 비교 보자.
9. 아래 오른쪽 이미지의 순서도에서 모든 조건식이  $a < 5$ 일대 a를 적절히 변경해서 다음과 같이 출력 되도록 프로그램을 구현해 보자.
  1. p1,p3,p4,p5,p8,p9
  2. p1,p2,p3,p6,p7,p8,p9
  3. p1,p2,p2,p3,p4,p5,p8,p9



10) 3을 10번 더하는 코드를 3가지로 만들어 보자.

11) 3부터 10을 출력하는 코드를 3가지로 만들어 보자.

12) 15~30까지의 사이 숫자를 더한 총합을 sum에 넣어 출력하는 코드를 3가지로 만들어 보자.

13) 사용자에게 1부터 9 사이의 숫자를 받아서 사용자가 입력한 숫자에 맞는 해당 구구단을 출력해 보자. 예를 들어 사용자가 4를 입력하면 4단을 출력하면 된다.

---

## > 09. break와 continue

---

break문은 반복문을 빠져 나갈때 사용된다.

continue문은 반복문에서 현재 진행 중인 반복 부분 진행을 중단하고 빠져나와 다음 반복 부분을 계속 진행 하는 용도로 사용된다.

다음 예제를 실행 시켜 보자.

break; 반복문을 빠져나옴

입력

```
int a=5;
for(int i=0;i<10;i++){
    if(i==a){
        break;
    }
    System.out.print(i);
}
```

출력 5에서 break를 만나 반복문을 빠져나와 더이상 출력되지 않음

01234

continue; 반복문 시작부분으로 이동

```
int a=5;
for(int i=0;i<10;i++){
    if(i==a){
        continue;
    }
    System.out.println(i);
}
```

출력 5에서 continue를 만나 5를 제외한 나머지 숫자만 출력됨

012346789

1. break를 사용 - while 문에서 5 이상의 값을 출력하고 반복문을 종료하는 예제

```
int i = 0;
while (true) {
    i++;
    if (i >= 5) {
        System.out.println(i);
        break;
    }
}
```

2. break를 사용한 예제 - for 문에서 4 이상의 값을 출력하고 반복문을 종료하는 예제

```
for (int i = 0; i < 10; i++) {
```



```

    if (i >= 4) {
        System.out.println(i);
        break;
    }
}

```

3. `continue`를 사용한 예제 - `while` 문에서 짝수를 출력하는 예제

```

int i = 0;
while (i < 10) {
    i++;
    if (i % 2 == 1) {
        continue;
    }
    System.out.println(i);
}

```

4. `continue`를 사용한 예제 - `for` 문에서 3의 배수를 출력하지 않는 예제

```

for (int i = 1; i <= 10; i++) {
    if (i % 3 == 0) {
        continue;
    }
    System.out.println(i);
}

```

5. `break`와 `continue`를 함께 사용한 예제 - `while` 문에서 3의 배수를 출력하지 않고, 5 이상의 값을 출력하고 반복문을 종료하는 예제

```

int i = 0;
while (true) {
    i++;
    if (i % 3 == 0) {
        continue;
    }
    if (i >= 5) {
        System.out.println(i);
        break;
    }
}

```

6. `break`와 `continue`를 함께 사용한 예제 - `for` 문에서 짝수를 출력하고, 6 이상의 값이 나오면 반복문을 종료하는 예제

```

for (int i = 0; i < 10; i++) {
    if (i % 2 == 1) {
        continue;
    }
    System.out.println(i);
    if (i >= 6) {
        break;
    }
}

```

```

}
}

```

7. **break**와 **continue**를 함께 사용한 예제 - **while** 문에서 2의 배수를 출력하지 않고, 8 이상의 값을 출력하고 반복문을 종료하는 예제

```

int i = 0;
while (true) {
    i++;
    if (i % 2 == 0) {
        continue;
    }
    if (i >= 8) {
        System.out.println(i);
        break;
    }
}

```

```

for(int i=0;i<10;i++) {
    if(i==5) {
        break;
    }
    System.out.println(i);
}

```

**break**문은 반복문 안에 기술해서 반복문을 빠져 나갈때 사용 한다. 왼쪽 0~9까지 출력 하는 **for**문에서 **i**값이 5가 되면 **break**를 만나 반복문을 빠져나와 5이후 숫자가 출력 되지 않는다.

```

27 for(;;) {
28
29     System.out.println(">>");
30     int a=Integer.parseInt(
31         (new java.util.Scanner(System.in))
32         .nextLine());
33     if(a==1004) {
34         break;
35     }
36 }

```

왼쪽 이미지는 **while**문에서 사용자 입력을 받아 1004를 입력할때 까지 반복하는 프로그램이다. 27라인은 무한루프를 **for**문으로 구현한 것이다. 초기식, 조건식, 변환식이 없어서 무한루프를 돌다가 **break**를 만나면 반복문을 빠져 나간다. **while**문에서 무한루프는 **while(true){}**로 기술한다.

```

while(true){

    System.out.println(">>");
    int a=Integer.parseInt(
        (new java.util.Scanner(System.in))
        .nextLine());
    if(a==1004) {
        break;
    }
}

```

30~32라인은 원래는 한줄로 기술하여야 하는데 내용이 많아서 여러 줄로 기술한 것이다. 여러줄 기술할 때는 괄호나 . 혹은 = 에서 줄을 바꾸어 기술하면 문제가 없다. 반복 회수가 명확하지 않을때 **while**문을 사용하고 명확할때 **for**문을 사용한다.

## continue문

```
for(int i=0;i<10;i++) {  
    if(i==5) {  
        continue;  
    }  
    System.out.println(i);  
}
```

반복문 안에서 현재 진행중인 반복 부분을 중단하고 다음 반복할 부분으로 이동 할 때 사용한다. 해당 반복 코드 블록에서 continue;를 만나면 진행하던 반복을 종료하고 변환문으로 이동하여 다음 반복 진행을 이어 나간다. 왼쪽은 0~9까지 출력 하는 for문에서 5를 제외한 다른 숫자들이 출력될 수 있도록 continue문을 만든 것이다. 출력 결과를 보고 한줄 한줄 따라가며

확인해 보자.

다음은 개선된 for문이다. 되도록 사용하지 말자.

1. for문안에 쉼표로 연결해서 여러 수식을 넣을 수 있다.

```
for(int count=1,c=3;count<10;count++) {
```

2. 조건식에 꼭 초기식이 오는 것은 아니다.

```
int count = 0;  
int i = 0;  
for ( i = 1; count < 10; i++) {  
    if (i % 3 == 0) {  
        System.out.println(i);  
        count++;  
    }  
}
```

3. 식에 연산된 복잡한 수식이 들어갈 수 있다.

```
for(int count=1;sum +count <=100;count++) {  
  
    for(int count=1;sum +count <=100;count=count+3) {
```

## 문제

1. 1부터 차례대로 더한 총합이 최초 100을 넘기 직전까지 누적한 결과값을 순서대로 출력하시오. 1:1 2:3 3:6 4:10 5:15 6:21 ... 100이전의 총합들을 다음과 같이 출력되면 된다. 힌트 : 조건식에 복잡한 수식이 들어 갈수 도 있다.
2. 문자열과 숫자를 입력받아 입력받은 문자열을 숫자만큼 출력하는 프로그램을 만들어 보자.

3. 입력 받은 두수 사이의 숫자들의 합을 구하는 프로그램을 만들어 보자.
4. 두 수를 입력 받아 첫 번째수 부터 시작하여 하나씩 카운트하여 두번째수 만큼 화면에 출력 하는 프로그램을 만든다. 5 6를 입력 받으면 5부터 6개 5 6 7 8 9 10 이 화면에 출력 된다.
5. 사용자가 “종료”를 입력할때 까지 무한 반복하는 프로그램을 3가지 방법으로 구현하시오.
6. 100의 모든 약수를 구하시오. 힌트) 0으로 나눌수 없듯이 0의 나머지는 구할수 없다.  
ex) 어떤 수에 대해서 나누어 떨어지는 수를 약수라고 한다. 100을 2로 나눈 나머지는 0이 된다. 100은 2로 나누어 떨어 지므로, 2는 100의 약수이다. 3은 100으로 나뉘 떨어지지 않으므로 약수가 아니다. 100의 모든 약수를 구하려면 100보다 크면 더 이상 나눌 수 없으므로 1~100사이의 수들로 하나씩 100을 나뉘서 나누어 떨어 지는지 판별하여 나누어 떨어지는 수들이 100의 약수가 된다. 1,2,4,5,..100 이 100의 약수에 해당한다.
8. 두 수를 입력받아 두수의 공통된 약수를 모두 출력해 보자. 0부터 하나씩 증가시키며 두수 모두 나뉘지는 수를 출력하면 된다. 출력된 수중에서 가장 큰수를 입력받은 두수의 최대공약수라고 한다.
9. 두수를 입력받아 두수의 최대 공약수를 구하는 프로그램을 만들어 보자.
10. 사용자에게 숫자를 하나 입력받아 입력한 숫자들의 합이 100이 될때까지 계속입력을 받다가 100이 넘으면 최종 합산한 값을 출력하는 프로그램을 구현해 보자.
11. 사용자에게 계속해서 숫자를 입력받아 1~10사이의 숫자를 3번 입력 할 때 까지 반복한다. 입력이 끝나면 잘못 입력한 회수와 제대로 입력한 회수를 출력하고 사용자가 제대로 입력한 총합을 출력하는 프로그램을 만들어 보자.
12. 0부터 시작하는 4의 배수 10개를 출력하시오.  
ex) 어떤수에 0,1,2,3,4.. 를 곱해서 나온 수를 어떤수의 배수라 한다.
13. 두수를 입력 받아 최소 공배수를 출력하는 프로그램을 만들어 보자.  
ex)2의 배수도 되고 3의 배수도 되는 공배수는 6,12,18,24 등이 있다. 이중에서 가장 작은 수 6를 최소 공배수 라고 한다. 2에 어떤수를 곱해서 나온 결과가 2의 배수 이므로 어떤 수를 2로 나누어 0이 되면 2의 배수이다.  
1 부터 하나씩 증가시켜 입력 받은 두 수가 모두 0으로 나누어 떨어지는 수들은 공배수 이고 이중 가장 먼저 찾은 수가 가장 작은 수이므로 처음 찾은수가 최소 공배수이다. 2와 5의 최소 공배수는 5이다.
14. 3개의 수를 입력 받아 가장 작은 수와 가장 큰수를 출력하는 프로그램을 만들어 보자.
15. 100이하의 수학과목 점수를 5번 입력받아 60이하 점수가 몇 개인지 출력하는

프로그램을 구현해 보자.

16. 소수는 1과 자기 자신만으로 나뉘지는 수이다. 2,3,5같은 경우 1과 본인 자신만으로 나뉘지므로 소수이다. 수를 하나 입력받아 소수인지 아닌지 판별하는 프로그램을 구현하시오.

ex) 2부터 본인 보다 하나 작은 숫자를 차례대로 나눠서 나누어 떨어지는 수가 없으며 1과 자기 자신만 나뉘지는 수이므로 소수에 해당한다.

17. 숫자를 하나 입력받아 1~1000사이에 입력받은 숫자의 배수가 몇 개인지 출력하는 프로그램을 만들어 보자.

18. 사용자에게 열의 데이터 개수를 입력받아 열에 맞춰서 1부터 100까지 순서대로 출력하는 프로그램을 구현해 보자.

ex) 3을 입력하면 한줄에 3개씩 출력 하면 된다.

19. 컴퓨터가 던진 동전이 앞면 인지 뒷면 인지 맞추는 프로그램을 구현해 보자. 맞춘 회수와 틀린 회수를 기록해서 보여주고 3번 맞추면 프로그램이 종료되게 만들자.

20. 컴퓨터가 던진 주사위 수를 맞추는 프로그램을 구현해 보자. 10회 입력받아 맞춘 회수와 틀린 회수를 기록해서 보여 주자.

답안

```
int count=1;

int sum=0;
while(sum +count <=100) {
    sum = sum+count;
//    if(sum<=100) {
//        break;
//    }
    System.out.println(count+":"+sum);
    count++;
}

//    for(int count=1,sum=0;;)

int sum=0;
for(int count=1;sum +count <=100;count++) {
    sum = sum+count;
    System.out.println(count+":"+sum);
}
```

```
}
```

```
int a= new java.util.Scanner(System.in).nextInt();  
int b= new java.util.Scanner(System.in).nextInt();  
int sum=0;
```

```
//b가 작으면 a랑 교환
```

```
//  
if(a>b) {  
//    int t=a;  
//    a=b;  
//    b=t;  
//}
```

```
//b가 더 큰것을 가정
```

```
if(a>b) {  
    for(int i=a;i>=b;i--) {  
        sum=sum+i;  
    }  
}else {  
    for(int i=a;i<=b;i++) {  
        sum=sum+i;  
    }  
}  
System.out.println(sum);
```

```
Scanner scanner=new Scanner(System.in);
```

```
// 11. 사용자에게 계속해서 숫자를 입력받아 1~10사이의 숫자를 3번 입력 할 때 까지  
// 반복한다. 입력이 끝나면 잘못 입력한 회수와 제대로 입력한 회수를 출력하고 사용자가  
// 제대로 입력한 총합을 출력하는 프로그램을 만들어 보자.
```

```
int wrongCount = 0;  
int correctCount = 0;  
int totalSum = 0;  
while(correctCount<3) {  
    System.out.println("숫자를 입력하세요(1~10):");  
    int number=Integer.parseInt(scanner.nextLine());  
    if(number>=1 && number<=10) {  
        correctCount++;  
        totalSum=totalSum+number;  
    }  
}
```

```

        }else {
System.out.println("잘못 입력했습니다. 1~10사이 숫자를 입력해주세요");
        wrongCount++;
        }
    }
System.out.println("입력이 종료되었습니다.");
System.out.println("제대로 입력한 회수: " + correctCount);
System.out.println("잘못 입력한 회수: " + wrongCount);
System.out.println("제대로 입력한 숫자들의 총합: " + totalSum);
    }
}

```

12.

```

//0,4,8,12,.....
//0부터 4씩 10번 더하면 된다.
int sum=0;
for(int i=0;i<10;i++) {
    System.out.println(sum);
    sum=sum+4;
}
//0*4,1*4,2*4,3*4
int num=4;
for(int i=0;i<10;i++) {
    System.out.println(num*i);
}

```

13.

```

//2에 배수 와 3에 배수 출력
for(int x=1;x<20;x++) {
    if(x%2==0) {
        System.out.print("2의 배수:"+x+" ");
    }
    if(x%3==0) {
        System.out.print("3의 배수:"+x);
    }
    System.out.println("");
}
//2와 3의 최소공배수 구하기
for(int x=1;;x++) {
    if(x%2==0&& x%3==0) {
        System.out.print("2와 3의 최소 공배수:"+x+" ");
        break;
    }
    System.out.println("");
}
//사용자 입력 처리
int num1=2;

```

```

    int num2=3;
    for(int x=1;;x++) {
        if(x%num1==0&& x%num2==0) {
            System.out.print(num1+"와"+num2+"의최소공배수:"+x+" ");
            break;
        }
    }
    // System.out.println("");
}

```

16.//소수 구하는 방법

// a라는 숫자가 2~a-1 사이 숫자로 나뉘서 떨어지는 숫자가 있으면 소수가 아니다.

// 없으면 소수

```
java.util.Scanner scanner =new java.util.Scanner(System.in);
```

```
System.out.println("소수 입력");
```

```
int number=Integer.valueOf(scanner.nextLine());
```

//number가 소수 인지 아닌지 확인하는 방법

//2~number-1사이 수중 number를 나눠 나머지가 0이되는 수가 있으면 소수가 아니다.

//없으면 소수이다.

//number가 6이면 2,3,4,5를 반복해서 number를 나눠서 나머지가 0인 값이 있는지 없는지 판별

```

boolean isFlag=true;
for(int i=2;i<number;i++) {
    if(number%i==0) {//소수가 아님
        isFlag=false;
        break;
    }else {//소수임
    }
}
if(isFlag) {
    System.out.println(number+"는 소수입니다.");
}else {
    System.out.println(number+"는 소수가 아닙니다.");
}

```

18.

```

int userLine=3;
for(int i=1;i<=100;i++) {
    System.out.print(i+" \t");
    if(i%userLine==0) {
        System.out.println();
    }
}

```

20.

//컴퓨터가 던진 주사위 수를 맞추는 프로그램을 구현해 보자.

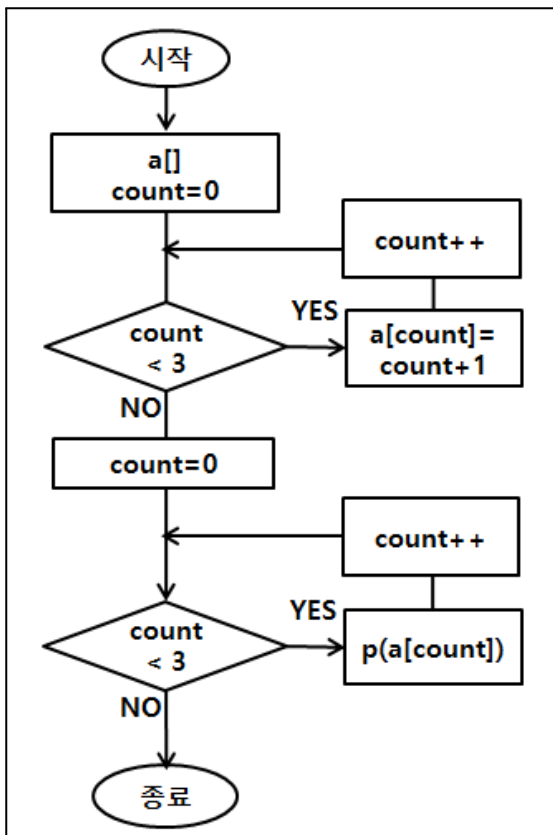


```
//10회 맞추어 맞춘 회수와 틀린 회수를 기록해서 보여 주자.  
int rCount=0;//맞은회수  
int wCount=0;//틀린회수  
  
for(int i=0;i<10;i++) {  
    int comNumber=(int)(Math.random()*6 +1);//1~6  
    System.out.println("컴퓨터의 예상 주사의 숫자를 입력하세요?");  
    int userNumber=Integer.parseInt(  
        new java.util.Scanner(System.in).nextLine());  
  
    if(comNumber==userNumber) {  
        rCount++;  
        System.out.println("맞았습니다.");  
    }else {  
        wCount++;  
        System.out.println("틀렸습니다.");  
    }  
}  
System.out.println("rCount:"+rCount);  
System.out.println("wCount:"+wCount);
```

## > 10. 반복문을 사용한 배열 접근

배열의 인덱스는 0부터 하나씩 증가 하므로 반복문을 이용해서 접근할 수 있다.

왼쪽 순서도는 크기 3인 배열 a에 1,2,3를 넣은 다음 배열에 들어 있는 내용들을 모두 화면에 찍는 순서도 이다. 반복문을 통해서 배열에 접근하는 방법을 확인할 수 있다. 한줄 한줄 따라가 보면서 배열 a와 변수 count와 화면에 출력되는 내용을 확인해 보자.



```
int a[]=new int[3];
for(int count=0;count<3;count++){
    a[count]=count+1;
}
for(int count=0;count<3;count++){
    System.out.println(a[count]);
}
```

변수를 이용해서 메모리 공간을 할당할 수 도 있다.

```
int num=10;
```

```
int a[]=new int[num];
```

아래는 이전에 만든 야구게임 전광판이다.  
반복문을 이용해서 업그레이드 해 보자.

스코어보드										2019.07.29 02
	1	2	3	4	5	6	7	8	9	R
탬파베이	0	0	0	0	1	3	2	3	1	10
토론토	0	2	2	0	4	1	0	0	0	9

team1TotalScoure=team1[0]+team1[1]+team1[2]+team1[3]+team1[4]+team1[5]+team1[6]+team1[7]+team1[8]; 이렇게 하면 팀1의 득점한 총 점수를 얻을 수 있다. 이 처럼 일일이 써서 넣을 수 있지만 다음과 같이 여러 줄로 처리할 수 있다.

```
team1TotalScoure=team1TotalScoure+team1[0]
```

```

team1TotalScoure=team1TotalScoure+team1[1]
team1TotalScoure=team1TotalScoure+team1[2]
team1TotalScoure=team1TotalScoure+team1[3]
team1TotalScoure=team1TotalScoure+team1[4]
team1TotalScoure=team1TotalScoure+team1[5]
team1TotalScoure=team1TotalScoure+team1[6]
team1TotalScoure=team1TotalScoure+team1[7]
team1TotalScoure=team1TotalScoure+team1[8]
team1TotalScoure=team1TotalScoure+team1[9]

```

여기에서 인덱스를 변수로 사용할 수 있다. 변수를 사용해서 다음과 같이 변경해 보자.

```

i=0
team1TotalScoure=team1TotalScoure+team1[i]
i++
team1TotalScoure=team1TotalScoure+team1[i]
i++
team1TotalScoure=team1TotalScoure+team1[i]
i++
...

```

이렇게 변수를 사용하면 처음과 동일한 결과가 나올 것이다. 이렇게 변경해서 사용하는 이유는 다음과 같이 반복할 수 있기 때문이다. `team1TotalScoure = team1TotalScoure + team1[i]; i++;` 이 부분을 반복 시키면 더욱 간단한 형태로 코드를 구현할 수 있는데 이부분을 반복문으로 변경해 보자.

이제 야구 경기를 진행할 때 필요한 모든 데이터가 선언된 것 같다. 야구 경기가 진행될 때 표시 되는 전광판 점수 처럼 순서대로 출력해 보자. 최종 출력 결과물은 `0,0,0,2,0,2,0,0,...1,0,10,9` 과 같이 출력될 것이다. 이 출력 부분을 반복 문으로 만들어 보자. 상위 이미지 같은 전광판을 만드는 프로그램을 구현해 보자.

**for**-each 루프의 문법은 다음과 같다.

```

for (elementType element : array) {
    // 반복할 코드
}

```

elementType: 배열에 저장된 요소의 데이터 타입을 나타냅니다.

element: 현재 반복 중인 요소에 대한 변수입니다.

array: 반복하고자 하는 배열입니다.

```

int[] numbersArray = {1, 2, 3, 4, 5};
// Enhanced for Loop를 사용하여 배열 순회
for (int num : numbersArray) {
    System.out.println(num);
}

```

다음 예제를 확인해 보자.

1. 배열의 요소들을 모두 출력하는 예제

```
int[] arr = {1, 2, 3, 4, 5};
for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
}
System.out.println("1부터 100까지의 홀수들의 합: " + sum);
```

2. 배열의 요소들 중에서 짝수만 출력하는 예제

```
int[] myArray = {1, 12, 53, 14, 25, 16, 57};
```

```
// 짝수만 출력
```

```
for (int num : myArray) {
    if (num % 2 == 0) {
        System.out.println(num);
    }
}
```

3. // 배열 선언 및 초기화

```
int[] numbers = {10, 20, 30, 40, 50};
```

```
// 배열 요소 합계 계산
```

```
int sum = 0;
for (int i = 0; i < numbers.length; i++) {
    sum += numbers[i];
}
```

4. // 배열 요소 평균 계산

```
double average = (double) sum / numbers.length;
```

```
// 결과 출력
```

```
System.out.println("합계: " + sum);
System.out.println("평균: " + average);
```

5.

배열에서 최댓값과 최솟값을 구하는 예제

```
int[] arr = {10, 5, 8, 2, 7};
int max = arr[0];
int min = arr[0];
for (int i = 1; i < arr.length; i++) {
    if (arr[i] > max) {
        max = arr[i];
    }
    if (arr[i] < min) {
        min = arr[i];
    }
}
```

```
System.out.println("배열에서 최댓값: " + max);  
System.out.println("배열에서 최솟값: " + min);
```

다음 문제를 반복문으로 구현해 보자.

문제 1) 배열 a에 1,2,3 을 넣은 후 배열 내의 모든 값에 2를 더한 값인 3,4,5로 변경한 다음에 배열의 내용을 화면에 출력하는 코드를 구현하여 보자.

문제 2) 배열 a[10]에 3의 배수를 넣은 다음에 배열의 내용을 출력하는 코드를 만들어 보자.

문제 3) 배열 a[100]에 1부터 100까지의 숫자를 순서대로 넣은 다음 배열의 인덱스가 짝수인 배열에 들어 있는 값만 출력하는 코드를 만들어 보자.

문제 4) 배열 a[]={12,1,5,3,6,8,5,3}를 만든 다음에 배열의 모든 내용을 더한 값을 sum에 저장하여 출력하는 코드를 만들어 보자.

문제 5) a[]={12,1,5,3,6,8,5,3}의 a배열에서 배열 안의 숫자가 짝수인 12,6,8 의 값을 더한 결과 값을 출력하는 프로그램을 작성해 보자.

문제 6) a[]={12,1,51,3,6,8,5}의 a배열에서 가장 큰 값과 가장 작은 값을 더하는 프로그램을 작성해 보자.

{12,1,51,3,6,8,5}에서 가장 큰수를 찾으려면 일단 가장 큰수를 저장할 변수 max에 12를 넣는다. max에 12를 넣은 이유는 비교 대상 수중 하나를 사용하기 위해서이다. 100과 같이 아무수나 입력하게 되면 제시한 수중에 100이 가장 큼으로 배열안에 가장 큰수를 찾는데 문제가 되어서 비교 대상중 하나의 숫자를 넣었다. 그 다음 {12,1,51,3,6,8,5}에서 처음 부터 순서대로 비교해서 더 큰 숫자가 나오면 max값에 더 큰 값을 넣는다. 첫번째 값 12는 현재 max값 12와 같으므로 패스한다. 두 번째 값 1은 max값보다 작으므로 패스한다. 세 번째 값 51은 max값보다 크므로 max값에 51를 넣는다. 네 번째 값 3은 max값 51 보다 작으므로 패스한다. 다섯 번째 값 6은 max값 51 보다 작으므로 패스한다. 여섯 번째 값 8은 max값 51 보다 작으므로 패스한다. 일곱 번째 값 5는 max값 51 보다 작으므로 패스한다. 이렇게 모든 숫자를 max 와 비교하는 작업이 끝나면 제시된 모든 숫자에서 max가 가장 큰 수가 된다. 여기서는 max값 51이 모든 숫자들 중에 가장 큰 숫자이다

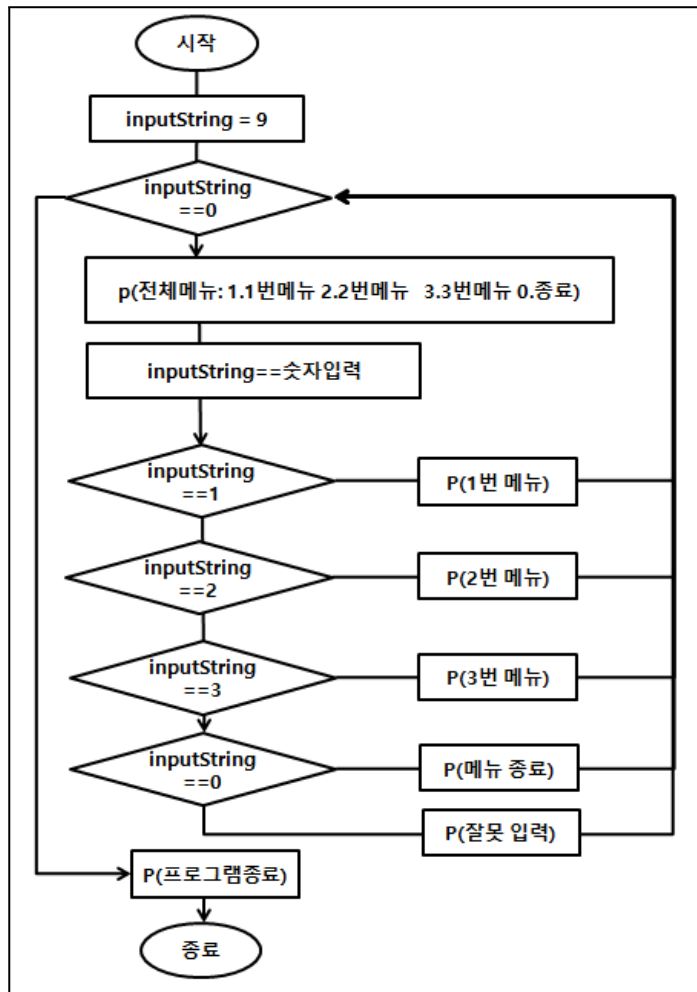
순서대로 상위 사항을 구현할 때 어떻게 해야 하는지 살펴보자. 배열의 가장 큰 값을 변수 max에 찾아서 넣는 방법은 max를 선언한 다음 배열 안의 임의의 수를 배열에 넣고(max=a[0]) 배열을 인덱스 0부터 끝까지 순회 하면서 해당 인덱스의 배열 안의 수와 max를 비교해서 (a[i]>max) 배열[인덱스]의 값이 max보다 크면(a[i]>max 이 true이면) 배열[인덱스] 값이 max 값 보다 더 크다는 이야기 이므로 max에 더 큰수 a[i]를 넣어(max=a[i]) 기존 max 값보다 더 큰 수를 max에 넣는다. 배열 안의 모든 수를 max값 보다 큰수가 나올 때마다 max에 넣으면 모든 배열의 데이터를 순회 하고 나면 max에는 배열안의 수중 가장 큰 수가 남게 된다.

- 7.배열에 5개의 숫자를 입력받아 넣은 다음 숫자 하나를 입력받아 해당 숫자가 몇번째 인덱스에 들어 있는지 출력하는 프로그램을 만들어 보자.
- 8.배열에 5개의 문자열을 입력받아 넣은 다음 문자열 하나를 입력받아 해당 문자가 몇번째 인덱스에 들어 있는지 출력하는 프로그램을 만들어 보자.
- 9.배열 크기를 입력 받아 배열 크기 만큼 1,2,3,1,2,3,1,2,3 ... 숫자를 순서대로 배열안에 넣은 다음 배열의 내용을 출력해 보자.
- 10.배열에 들어있는 내용을 정순과 역순으로 저장하는 새로운 배열을 만들어 출력해 보자.
- ex)배열에 1,2,3이 들어 있으면 1,2,3,3,2,1 이 들어 있는 배열
- ex)배열에 7,9,21,15,23 이 들어 있으면 23,15,21,9,7,7,9,21,15,23 이 들어 있는 배열
- 11.컴퓨터에서 주사위 던지는 프로그램을 구현하였을때 던져서 나온수의 빈도(회수)를 구하는 프로그램을 작성해 보자.

10.

```
int arr1[] = {1,2,3,5};
int arr2[] = new int[arr1.length*2];
// arr2[0]=arr1[0];
// arr2[1]=arr1[1];
// arr2[2]=arr1[2];
for(int i=0;i<arr1.length;i++) {
    arr2[i]=arr1[i];
}
// arr2[3]=arr1[2];
// arr2[4]=arr1[1];
// arr2[5]=arr1[0];
for(int i=0;i<arr1.length;i++) {
    arr2[arr1.length+i]=arr1[arr1.length-1-i];
}
System.out.println(Arrays.toString(arr2));
```

## > 11. 사용자 메뉴 만들기



프로그램에서 메뉴에 대해서 생각해 보자.

요즘 음식점에 가면 pos 기계라 부르는 사용자가 메뉴를 직접 선정해서 결제하도록 되어 있는 기계를 볼수 있는데 이 기계의 메뉴는 사용자가 원하는 메뉴를 선택하면 해당 메뉴의 작업이 실행되고 이 작업이 마무리되면 다음 사용자가 다시 전체 메뉴를 사용할 수 있도록 지속적으로 반복하는 형태로 되어 있다.

처음에 화면에 ‘전체메뉴: 1.1번메뉴 2.2번메뉴 3.3번메뉴 0.종료’ 과 같은 선택 메뉴가 나오고 사용자가 원하는 메뉴를 입력 받아 해당 메뉴 작업을 진행하고 다시 처음 화면 전체 메뉴로 돌아오는 순서도를 왼쪽 이미지 처럼 만들어 보았다

다음 왼쪽 순서도를 확인해 보자. inputString=9에서 9는 프로그램에 영향을 주지 않는 임의의 숫자를 넣은 것이다.

‘전체메뉴: 1.1번메뉴 2.2번메뉴 3.3번메뉴 0.종료’ 와 같은 문자열 메뉴가 화면에 나오고 사용자가 숫자를 입력 받아 사용자가 선택한 번호의 메뉴가 화면에 나온다. 1번을 선택하면 ‘1번 메뉴입니다.’ 출력하고 2번을 선택하면 ‘2번 메뉴입니다.’가 출력 되고 0를 입력하면 프로그램이 종료 되고 1~3번의 경우는 해당 번호의 메뉴를 출력하고 다시 전체메뉴가 출력되어 사용자 입력을 기다린다. p(n번메뉴)와 같은 부분에서 실제 n번 메뉴를 실행하는 내용을 복잡하게 만들수 있는데 여기서는 생략하고 몇번 메뉴가 실행된다는 문자열만 출력 하였다.

다음은 ifelse코드로 만든 것이니 확인해 보자.

```
String inputString = "9";
while (!inputString.equals("0")) {
    System.out.println("전체메뉴: 1.1번메뉴 2.2번메뉴 3.3번메뉴 0.종료");
    inputString = new java.util.Scanner(System.in).nextLine();
    if(inputString.equals("1")) {
```

```

        System.out.println("1번 메뉴");
    }else {
        if(inputString.equals("2")) {
            System.out.println("2번 메뉴");
        }else {
            if(inputString.equals("3")) {
                System.out.println("3번 메뉴");
            }else {
                if(inputString.equals("0")) {
                    System.out.println("메뉴 종료");
                }else {
                    System.out.println("잘못 입력");
                }
            }
        }
    }
}

```

System.out.println("프로그램종료");

상위 if문 들은 출력 대상중 반드시 하나만 출력된다는 특징이 있다. 이런 특징을 가진 if문 집단은 elseif문이나 switch로 변경할 수 있다.

상위 if문을 다음과 같이 elseif문으로 바꿀 수 있다.

```

if (inputString.equals("1")) {
    System.out.println("1번 메뉴");
} else if (inputString.equals("2")) {
    System.out.println("2번 메뉴");
} else if (inputString.equals("3")) {
    System.out.println("3번 메뉴");
} else if (inputString.equals("0")) {
    System.out.println("메뉴 종료");
} else {
    System.out.println("잘못 입력");
}

```

상위 if문을 다음과 같이 switch문으로 바꿀 수 있다. 비교 값이 1:1 매칭 이므로 switch문으로 구현하는 것이 일반적이다.

처음 공부하는 사람은 else if문이나 switch문을 어려워 하는 사람이 많다. 어디 적어 놓고 필요할때 마다 보고 따라 하다 보면 저절로 외워질 것이다. 문법이니 외워서 구현하자.

```

String inputString = "2"; // 사용자로부터 입력받은 문자열
switch (inputString) {
    case "1":
        System.out.println("1번 메뉴");
        break;
    case "2":

```



```

        System.out.println("2번 메뉴");
        break;
    case "3":
        System.out.println("3번 메뉴");
        break;
    case "0":
        System.out.println("메뉴 종료");
        break;
    default:
        System.out.println("잘못 입력");
        break;
}

```

```

package com.the.ex;
public class Java11 {
    public static void main(String[] args) {
        String inputString="9";
        while(!inputString.equals("0")) {
            System.out.println("전체메뉴:1.메뉴 2. 메뉴 3메뉴 0.종료");
            inputString=new java.util.Scanner(System.in).nextLine();
            switch(inputString) {
                case "1":
                    System.out.println("메뉴1번 관련작업");
                    break;
                case "2":
                    System.out.println("메뉴2번 관련작업");
                    break;
                case "3":
                    System.out.println("메뉴3번 관련작업");
                    break;
                case "0":
                    System.out.println("메뉴 종료");
                    break;
                default:
                    System.out.println("잘못된 입력");
            }
        }
        System.out.println("프로그램 종료");
    }
}

```

다음은 중괄호 안에 선언된 변수는 중괄호 안에서만 사용할 수 있다는 예제이다. 잘 확인해 보고 배열 변수도 마찬가지이다. 되도록 사용 할 변수를 바깥쪽보다는 안쪽에 기술하는 것이 좋다.

```
{
    int a=1;

    {
        int b=2;
        a=10;//안쪽 블록에서는 바깥쪽 블록에 접근 가능
    }
    b=10;//바깥쪽 블록에서는 안쪽 블록에 접근 불가능
    {
        int c=2;
        a=10;//안쪽 블록에서는 바깥쪽 블록에 접근 가능
        b=10;//바깥쪽 블록에서는 안쪽 블록에 접근 불가능
    }
    c=10;//바깥쪽 블록에서는 안쪽 블록에 접근 불가능
}
```

다음 문제를 풀어보자.

1. 50명의 학생의 출석 사항을 처리하는 프로그램을 만들어 보자. 출석자는 true이고 결석자는 false 데이터를 가진다.

boolean student[]=new boolean[50]; 배열을 사용해서 아래와 같은 메뉴를 가진 프로그램을 만들어 보자.

메뉴:

1.전체 출석사항 2.결석자 추가 3.출석자 추가 6.프로그램 종료

설명

1.전체 출석사항

배열에 true가 들어 있으면 출석, false가 들어 있으면 결석이다.

인덱스 + 1 를 학생 번호라 할때 모든 학생별 상황을 다음과 같이 출력 한다.

"1번 학생의 출석 사항은 출석입니다."

"2번 학생의 출석 사항은 결석입니다."

## 2. 결석자 추가

결석한 학생의 번호를 입력 받아 입력 받은 숫자에 해당하는 배열의 인덱스에 false를 넣으면 된다.

## 3. 출석자 추가

출석한 학생의 번호를 입력 받아 입력 받은 숫자에 해당하는 배열의 인덱스에 true를 넣으면 된다.

## 4. 프로그램 종료

진행중인 프로그램을 종료시키면 된다.

2. 은행 프로그램을 순서도와 프로그래밍 언어로 만들어 보자.

은행 프로그램의 메뉴는 다음과 같다. ‘전체메뉴: 1.입금 2.출금 3.조회 0.종료’

account 변수에 처음에 0으로 세팅하고 입금 출금을 통해 원하는 액수를 더하거나 빼준다. 결국 account 변수에 있는 숫자가 은행계좌에 남은 돈이 된다.

입금 메뉴를 통해 입금 액을 받아 account 변수에 추가 할 수 있다.

출금 메뉴를 통해 출금 액을 account 변수에서 뺄 수 있다.

조회 메뉴를 통해 account에 입금 액이 얼마나 남아 있나 확인 할 수 있다.

조심해야 할 부분은 운영중 account의 값이 유지될수 있도록 적절한 위치에 선언해야 운영중에 값 손실이 생기지 않는다.

실행화면

메뉴: 1.입금 2.출금 3.조회 0.종료

3

현재 잔액은 0원 입니다.

메뉴: 1.입금 2.출금 3.조회 0.종료

1

입금액을 입력하세요.

100000

메뉴: 1.입금 2.출금 3.조회 0.종료

2

출금액을 입력하세요

35000

메뉴: 1.입금 2.출금 3.조회 0.종료

3

현재 잔액은 65000원 입니다.

메뉴: 1.입금 2.출금 3.조회 0.종료

0

종료합니다.

3. 금액을 입력받아 천원, 오백원, 백원, 오십원, 십원 짜리 잔돈으로 거슬러주는 프로그램과 순서도를 구현해 보자. 예제 금액을 1000으로 나눈 몫은 돈을 거슬러 주었을때 천원짜리 개수 이고 1000으로 나눈 나머지는 천원짜리로 환산하고 남은 잔돈이다. 10원 이하는 입력받지 않는다. 최종 결과물은 다음과 같이 될 것이다. 5820원을 잔돈으로 바꾸면 천원짜리 5개, 오백원짜리 1개, 백원짜리 3개, 오십원짜리 0개, 십원짜리 2개

선언할 변수:change1000,change500,change100,change50,change10,moneyInput

출력값:p(“천원짜리 “+change1000+”개, 오백원짜리 “+change500+”개, 백원짜리”+change100+”개, 오십원짜리 “+change50+”개, 십원짜리 “+change10개”)

4. 실존하는 자판기와 동일하게 만들어 보자. 문자열 변수를 처리하려면 int 가 아닌 String 자료형을 사용해서 선언해야 한다.

String 사용방법은 다음과 같다.

```
String str="";    str=str+“사이다”; str=str+“콜라”;
```

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

String 변수 str를 출력하면 사이다콜라가 출력된다.

메뉴:

남은돈 0원.

1.사이다 700 2.콜라 500 3.환타 350 4.100투입 5.500투입 6.반환

음료반환구에 사이다 콜라 가 있음

메뉴를 보여주고 사용자가 선택을 하면 선택 사항을 반영하여 선택후 항상 메뉴를 보여

준다.

시나리오 :

천원짜리 세장 넣고 사이다1개 콜라2개 환타1개 를 구매한다.

최종 출력값:

사이다 콜라 콜라 환타 천원짜리 0개 오백원짜리 1개 백원짜리 4개 오십원 1개

5. 369게임 해답지를 만들어 보자.

1부터 순서대로 숫자를 출력 하다가 숫자에 3,6,9중에 하나라도 들어가면

박수를 치면 된다.

1,2,짝,4,5,짝,7,8,짝,10,11,12,짝,14.....

그런데 만약 33, 36같이 두개가 들어가면 박수를 두번 친다.

27,28,짝(29),짝(30),짝(31),짝(32),짝짝(33)

1000이하의 정답지를 출력하는 프로그램을 만들어 보자.

6. 랜덤하게 덧셈, 뺄셈, 곱셈, 나눗셈 문제를 내서 사용자가 맞추는 프로그램을 구현해 보자. 레벨이 4단계로 되어 있어 처음에는 더하기 문제만 나오다가 레벨이 업되면 뺄셈,곱셈,나눗셈을 차례대로 추가하여 문제가 나오도록 해보자. 2자리 이하 정수 계산만 하자.

#### ● 이용권

종류	대인	청소년	소인/경로
주간권(1일권)	46,000원	39,000원	36,000원
오후권(오후3시~)	38,000원	32,000원	29,000원
윈터야간권(오후5시~)	20,000원	17,000원	16,000원
2일권	74,000원	62,000원	58,000원

(출처 : 에버랜드 공식 홈페이지)

7.상위 표를 이용해서 요금 계산하는 프로그램을 구현해보자. “1.대인 2.청소년 3.소인/경로 4.종료” 와 같은 메인 메뉴에서 구매하는 형태로 만들어 보자.

8. 컴퓨터가 임의로 정한 하나의 숫자를 맞추는 프로그램을 만들어 보자. 사용자가 입력한 숫자가 임의의 숫자랑 같지 않다면 사용자가 숫자를 맞출 수 있도록 큰지 작은지 힌트를 줘서 사용자가 맞출수 있도록 해준다. 정답을 맞추면 정답을 맞췄다는 메시지와 함께 총 몇회에 도전끝에 맞췄는지 출력해 보자.

9) 문제1)에서 구현한 은행 프로그램을 배열을 이용해서 100명의 사용자 계정을 관리하는 프로그램으로 업그레이드 해보자.

메뉴 변경 : “메뉴: 1.입금 2.출금 3.조회 0.종료”와 같은 기존 메뉴에서 다음과 같은 메뉴로 변경해서 구현해 보자. “메뉴: 1.입금 2.출금 3.조회 4.0부터99사이의 숫자를 입력받아 작업계정 선택 5. 전체 계정 출력 0.종료”

계정저장 : 한명만 저장하던 `int account = 0;`에서 100명을 저장 할 수 있는 배열 형태로 변경한다. `double account[] = new double[100];`

현재 작업중인 계정 인덱스 저장을 위해서 `int nowUserIndex=0;` 를 선언한다. 작업할 사용자 인덱스로 사용된다.

다음은 문제 풀이 이다.

1.

```
package com.human.menu;
public class AttendanceBook {
    public static void main(String[] args) {
        //1. 50 명의 학생의 출석정보를 저장할 수 있는 변수 선언
        //boolean 배열로 50개 선언하고 true이면 출석 false이면 결석
        boolean student[]=new boolean[50];
        //2. 기본값을 true로 설정한다.
        //반복문을 이용해서 배열의 모든 내용을 true로 변경
        for(int i=0;i<student.length;i++) {
            student[i]=true;
        }
        //3. 메뉴 제작 1.전체 출석사항 2.결석자 추가 3.출석자 추가 6.프로그램 종료
        //a.반복 입력 변수 선언
        //b.while문 제작
        //c.메뉴 출력
        //d.사용자 입력
        //e.메뉴 처리 switch문 제작
        String input="";//a.반복 입력 변수 선언
        while(!input.equals("6")) {//6이면 빠져나가게 구현함//b.while문 제작
            System.out.println("1.출석사항 2.결석자추가 3.출석자추가 6.종료");//c.메뉴 출력
            input=new java.util.Scanner(System.in).nextLine();
            switch(input) {
                case "1"://배열에 true이면 출석 false이면 결석
                    System.out.println("출석사항");
                    for(int i=0;i<student.length;i++) {
                        String str=i+"번 학생의 출석 사항은";
                        if(student[i]==true) {
                            str=str+"출석입니다.";
                        }else {
```

```

        str=str+"결석입니다.";
    }
    System.out.println(str);
}
break;
case "2":
    System.out.println("결석한 학생번호를 입력하세요 0~49");
int index=Integer.parseInt(new java.util.Scanner(System.in).nextLine());
    student[index]=false;
    break;
case "3":
    System.out.println("출석한 학생번호를 입력하세요 0~49");
index=Integer.parseInt(new java.util.Scanner(System.in).nextLine());
    student[index]=true;
    break;
case "6":
    System.out.println("프로그램을 종료하였습니다.");
    break;
default:
    System.out.println("입력을 잘못하였습니다.");
}
}
}
}
}

```

2.

```

int account = 0;
boolean flag=true;
Scanner sc = new Scanner(System.in);
while (flag) {
    System.out.println("메뉴: 1.입금 2.출금 3.조회 0.종료");
    int choice = Integer.parseInt(sc.nextLine());
    switch (choice) {
        case 1:
            System.out.println("입금액을 입력하세요.");
            int deposit = Integer.parseInt(sc.nextLine());
            account += deposit;
            break;
        case 2:
            System.out.println("출금액을 입력하세요");
            int withdrawal = Integer.parseInt(sc.nextLine());
            account -= withdrawal;
            break;
        case 3:
            System.out.println("현재 잔액은 " + account + "원 입니다.");

```

```

        break;
    case 0:
        flag=false;
        System.out.println("종료합니다.");
        return;
    default:
        System.out.println("잘못된 입력입니다. 다시 선택해주세요");
    }
}
4-1

```

```

int oriNum = 234;
String str = "";
int num = oriNum;
if(num%10%3==0&&num%10!=0) {
    str=str+"짝";
}
num=num/10;//291
if(num%10%3==0&&num%10!=0) {
    str=str+"짝";
}
num=num/10;//29
if(num%10%3==0&&num%10!=0) {
    str=str+"짝";
}
num=num/10;//2
if(num%10%3==0&&num%10!=0) {
    str=str+"짝";
}
num=num/10;//0a
if (str.equals("")) {
    System.out.println(oriNum);
} else {
    System.out.println(str + "(" + oriNum + ")");
}

```

4-2

```

for (int i = 1; i <= 1000; i++) {
    int oriNum = i;
    String str = "";
    int num = oriNum;
    while (num != 0) {
        if (num % 10 % 3 == 0 && num % 10 != 0) {
            str = str + "짝";
        }
        num = num / 10;// 291
    }
}

```



```

        if (str.equals("")) {
            System.out.println(oriNum);
        } else {
            System.out.println(str + "(" + oriNum + ")");
        }
    }
}
4-3

```

```

for(int j=1;j<=1000;j++) {
    int num =j;
    String str="";

    String strNum="" +num;
    for(int i=0;i<strNum.length();i++) {
        int charNum=strNum.charAt(i)-'0';
        if(charNum%3==0&&charNum!=0) {
            str=str+"짝";
        }
    }
    if(str.equals("")) {
        System.out.println(num);
    }else {
        System.out.println(str+"(" +num+")");
    }
}
}

```

8.

```

int nowUserIndex = 0;
double account[] = new double[100];
boolean flag = true;
java.util.Scanner sc = new java.util.Scanner(System.in);
System.out.println("작업계정을 입력하세요. (0~99)");
nowUserIndex = Integer.parseInt(sc.nextLine());
System.out.println();
while (flag) {
    System.out.println("메뉴:1.입금 2.출금 3.조회 4.계정선택 5.전체 계정 출력 0.종료");
    int choice = Integer.parseInt(sc.nextLine());
    switch (choice) {
        case 1:
            System.out.println("입금액을 입력하세요.");
            double deposit = Double.parseDouble(sc.nextLine());
            account[nowUserIndex] += deposit;
            System.out.println("["+nowUserIndex+"]계정에 "+deposit+"원이 입금되었습니다.");
            break;
        case 2:
            System.out.println("출금액을 입력하세요");

```

```

        double withdrawal = Double.parseDouble(sc.nextLine());
        account[nowUserIndex] -= withdrawal;
        System.out.println "["+nowUserIndex+"]계정에서 "+withdrawal+"원이 출금됨");
        break;
    case 3:
        System.out.println "["+nowUserIndex+"]계정 잔액"+ "\n"+account[nowUserIndex]);
        break;
    case 4:
        System.out.println("어떤 계정을 선택하시겠습니까? (0~99)");
        nowUserIndex = Integer.parseInt(sc.nextLine());
        System.out.println "[" + nowUserIndex + "]계정으로 변경되었습니다.");
        break;
    case 5:
        for (int i = 0; i < account.length; i++) {
            System.out.println "["+i+"]님의 잔액 : " + account[i] + "원");
        }
    case 0:
        flag = false;
        System.out.println("종료합니다.");
        return;
    default:
        System.out.println("잘못된 입력입니다. 다시 선택해주세요");
    }
}

```

---

## > 13. 상수를 이용한 출석 관리

---

다음 코드는 상수가 포함된 출석 프로그램을 업그레이드한 코드이다 확인해 보자.

이전 코드는 단순히 지각 결석 정보만 저장 할 수 있었는데 지금은 좀더 다양한 출석 정보를 넣어서 프로그램을 구현하다 보니 문자열 상수를 이용해서 프로그램을 구현 하였다.

```
package com.human.menu;
public class AttendanceBookEx {
    // final 키워드를 해당 변수 선언에 사용하면 해당 변수는 상수가 되어 변경이
    // 불가능해진다. 상수는 한번 선언해서 값을 넣으면 도중에 변경이 불가능하다.
    // 관용적으로 상수는 모두 대문자를 사용하고 새로운 의미가 나올때마다 밑줄을
    // 사용한다.
    // 여기서는 상태를 숫자로 사용하면 의미를 파악하기 어려워 상수를 사용하여
    // 상태 표현 하는 용도로 사용 하였다.
    public static final int ATTENDANCE = 0; //출석
    public static final int ABSENCE = 1;    //결석
    public static final int LATE = 2;        //지각
    public static final int EARLY_LEAVE = 3; //조퇴
    public static final int OUTING = 4;      //외출
    public static final int SICK_LEAVE = 5;  //병결
    public static final int PUBLIC_LEAVE = 6; //공결

    public static void main(String args[]) {
        //기존 출결 프로그램은 출결 표시만 가능하다. 프로그램을 업그레이드 해서
        //출석 결석 지각 조퇴 외출 병결 공결 정보를 처리할 수 있는 프로그램을 구현하시오.

        // boolean 배열을 int배열로 변경
        // 각각의 상태 번호를 부여
        // 상태번호를 상수로 변경

        // 이후 코드를 확인해서 출석관리 프로그램을 구현해 보자.

        //출석 표현
        int student1=0;//student1의 의미를 파악하기 어려워 다음과 같이
        int student2=AttendanceBookEx.ATTENDANCE;//student2는 출석상태이다.
        student1=3;//EARLY_LEAVE는 3 숫자보다 문자열이 이해하기 더 쉽다.
        System.out.println(student1+":"+student2);
        switch (student1) {
            case AttendanceBookEx.ATTENDANCE:
                System.out.println("출석");
                break;
            case AttendanceBookEx.ABSENCE:
                System.out.println("결석");
                break;
```

```
        case AttendanceBookEx.LATE:
            System.out.println("지각");
            break;
        case AttendanceBookEx.EARLY_LEAVE:
            System.out.println("조퇴");
            break;
        case AttendanceBookEx.OUTING:
            System.out.println("외출");
            break;
        case AttendanceBookEx.SICK_LEAVE:
            System.out.println("병결");
            break;
        case AttendanceBookEx.PUBLIC_LEAVE:
            System.out.println("공결");
            break;
        default:
            System.out.println("알 수 없는 상태");
            break;
    }
}
```

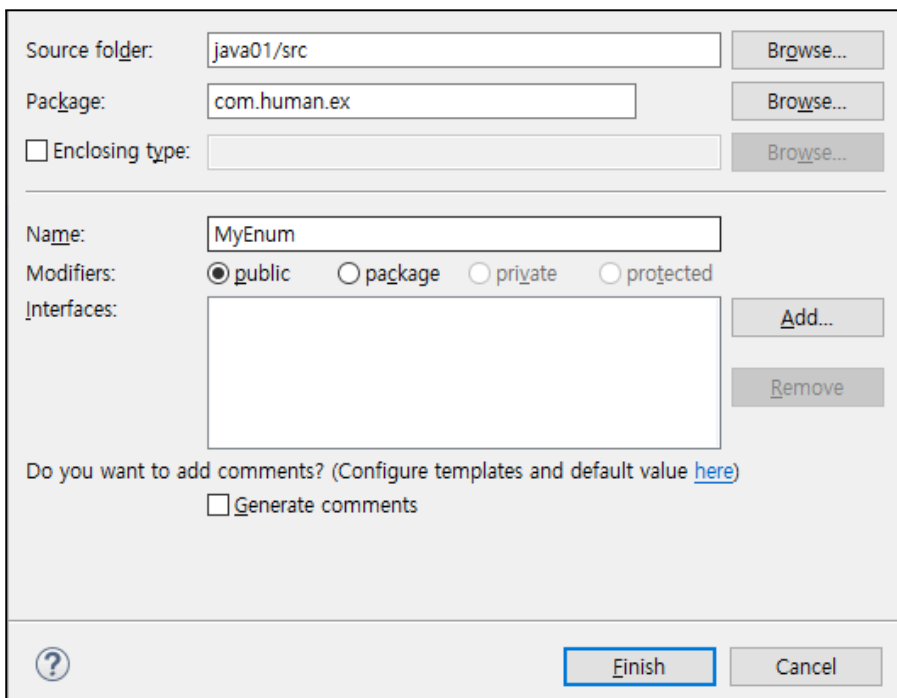
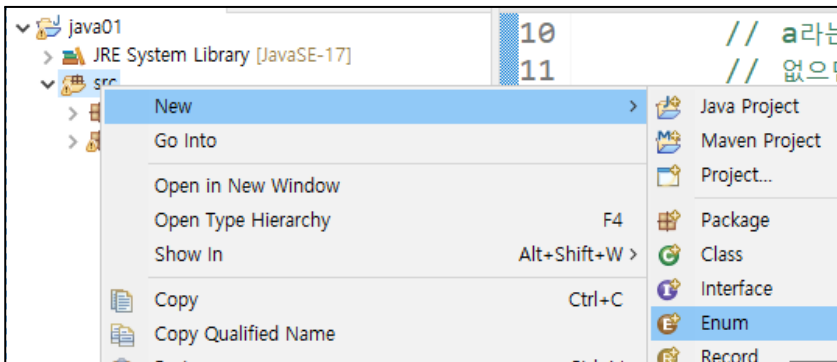
상위 예제를 바탕으로 학생 출석관리 프로그램을 완성해 보자.

## > 13. enum

enum 은 관련있는 상수들을 묶어서 그룹을 만든 것이다.

다음은 MyEnum이라는 enum 만드는 방법이다.

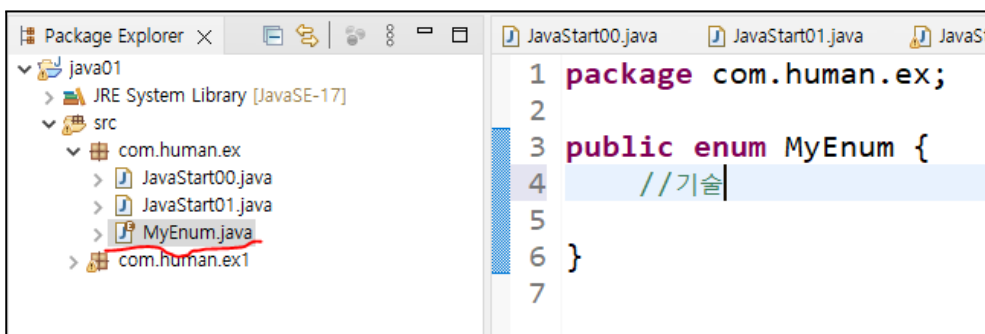
1. 다음 이미지 처럼 프로젝트 소스 폴더에서 오른쪽 클릭 new >> Enum를 선택한다.
2. 존재하지 않으면 other를 이용해서 enum를 찾아 선택한다.



3. 다음과 같은 창이 뜨면 아래와 같이 패키지과 enum이름을 입력한다.

com.human.ex, MyEnum

4. 제대로 만들었으면 다음과 같은 창이 뜬다. 기술 부분에 원하는 내용을 기술한다.



5. 상위 enum 만드는 방법을 참고해서 다음을 확인하여 enum Season 예제를 만들어 실행해 보자.

*//Season.java 파일의 내용*

```
public enum Season {  
    SPRING, SUMMER, AUTUMN, WINTER  
}
```

6. Season enum를 사용하는 다음과 같은 실행 클래스를 만들어 실행해 보자.

*//SeasonalActivities.java 파일의 내용*

```
public class SeasonalActivities {  
    public static void main(String[] args) {  
        Season currentSeason = Season.AUTUMN; //enum 선언 및 할당  
        switch (currentSeason) {  
            case SPRING:  
                System.out.println("꽃 구경하기");  
                break;  
            case Season.SUMMER:  
                System.out.println("해수욕");  
                break;  
            case Season.AUTUMN:  
                System.out.println("단풍 구경하기");  
                break;  
            case Season.WINTER:  
                System.out.println("눈싸움");  
                break;  
        }  
    }  
}
```

enum은 "enumerated type"의 약어로, 프로그래밍에서 관련 있는 상수들의 집합을 나타내는 자료형입니다. Java에서 enum은 다양한 상수 값을 나타내기 위해 사용되며, 코드의 가독성과 유지 보수성을 향상 시키는 데 도움을 줍니다.

1. 클래스 처럼 파일 이름은 enum이름과 동일해야 한다.
2. 파일이름과 enum이름 이 동일하게 다음과 같이 선언해야 한다.

```
public enum enum이름 {  
    VALUE1, //상수 목록  
    VALUE2,  
    // ...  
}
```

3. enum를 사용하기 위해서는 변수처럼 선언하고 'enum이름.원하는값' 을 기술해서 값을 넣어 사용 할 수 있다.

```
enum이름 value = enum이름.VALUE1;
```

4. enum를 이용해서 switch문을 사용할 수있다.

```
switch (value) {  
    case VALUE1:  
        // 처리  
        break;  
    case VALUE2:  
        // 처리  
        break;  
    // ...  
}
```

5. public으로 만들어진 enum은 클래스 처럼 하나의 파일에 하나만 만들어야 한다.

6. 같은 패키지 안에서 접근할 때에는 enum명을 생략할 수 있다. Season.SUMMER를 SUMMER로 생략가능하다. 되도록 생략하지 말자

```
Season mySeason = SUMMER; // 패키지 내에서 정의된 Season Enum 사용
```

```
Season mySeason = Season.SUMMER;
```

7. switch 의 case문에서는 Season.SPRING이라고 사용하면 안되고 enum값을 직접 SPRING이라고 써야 한다.

8. Enum 상수끼리는 == 연산자를 사용하여 비교할 수 있습니다.

```
Mood myMood = Mood.HAPPY;
```

```
if (myMood == Mood.HAPPY) {
```

9. values()를 이용해서 enum이 가지고 있는 모든 값들을 배열로 생성해서 받을 수 있다.

```
for (Fruit fruit : Fruit.values()) { //enum Fruit배열이 됨
```

10. valueOf를 이용해서 문자열을 Enum으로 변환할 수 있다.

```
String genderStr = "MALE";
```

```
Gender parsedGender = Gender.valueOf(genderStr); // 문자열을 Enum으로 변환
```

```
System.out.println("Parsed gender: " + parsedGender);
```

다음 다양한 예제를 확인해 보자.

1.

```
enum Direction {  
    NORTH,  
    SOUTH,  
    EAST,  
    WEST  
}  
  
public class DirectionExample {  
    public static void main(String[] args) {  
        Direction myDirection = Direction.EAST;  
        System.out.println("I am facing " + myDirection);  
    }  
}
```

2.

```
enum DayOfWeek {  
    SUNDAY,  
    MONDAY,  
    TUESDAY,  
    WEDNESDAY,  
    THURSDAY,  
    FRIDAY,  
    SATURDAY  
}  
  
public class DayOfWeekExample {  
    public static void main(String[] args) {  
        DayOfWeek today = DayOfWeek.WEDNESDAY;  
        System.out.println("Today is " + today);  
    }  
}
```

3.

```
enum Grade {  
    A,    B,    C,    D,    F  
}  
  
public class GradeExample {  
    public static void main(String[] args) {  
        Grade myGrade = Grade.B;  
        System.out.println("My grade is " + myGrade);  
    }  
}
```

4.

```
enum Mood {  
    HAPPY,    SAD  
}
```



```

public class EnumComparisonExample {
    public static void main(String[] args) {
        Mood myMood = Mood.HAPPY;
        if (myMood == Mood.HAPPY) {
            System.out.println("I'm happy!");
        } else {
            System.out.println("I'm not happy.");
        }
    }
}
5.
enum Gender {
    MALE, FEMALE
}
public class EnumMethodsExample {
    public static void main(String[] args) {
        Gender[] genders = Gender.values(); // Enum의 상수 배열을 가져옴
        for (Gender gender : genders) {
            System.out.println(gender);
        }
        String genderStr = "MALE";
        Gender parsedGender = Gender.valueOf(genderStr); // 문자열을 Enum으로 변환
        System.out.println("Parsed gender: " + parsedGender);
    }
}
6.
//enum를 배열로 사용하는 방법
//Season[] seasons = new Season[4];
// seasons[0] = Season.SPRING;
// seasons[1] = Season.SUMMER;
// seasons[2] = Season.AUTUMN;
// seasons[3] = Season.WINTER;
Season[] seasons = Season.values(); // enum의 모든 값들을 배열로
얻어옴

Season currentSeason = Season.AUTUMN;
for (Season season : seasons) {
    if (season == currentSeason) {
        switch (season) {
            case SPRING:
                System.out.println("꽃 구경하기");
                break;
            case SUMMER:
                System.out.println("해수욕");
                break;
            case AUTUMN:
                System.out.println("단풍 구경하기");
                break;
        }
    }
}

```

```
        case WINTER:  
            System.out.println("눈싸움");  
            break;  
    }  
}  
}
```