1. 조건문

개발자가 작성한 코드를 **조건에 따라 코드의 실행 흐름을 다르게 동작하도록 제어하는 것**이라고 할 수 있다.

조건문에는 if문과 switch문이 있다. 그리고 if문은 if문, if else문, else if문으로 다시 구분할 수 있다.

```
if(조건식) {   //조건식이 true일 때 아래 실행문 동작, false면 미실행
실행문;
실행문;
...
}
```

<--- if문은 이런식으로 정의가 된다.

Ex)

```java
package week4;

public class Main {
    public static void main(String[] args) {
        int a = 6;
        if(a%2==0){ // true
            System.out.println(a+"는 2의 배수");        // 실행
            if(a%3==0){ // true
                System.out.println(a+"는 3의 배수");   // 실행
            }
            if(a%4==0){ // false
                System.out.println(a+"는 4의 배수");
            }
        }
    }
}
```

→ 이 예제는 첫 번째 if문이 true 값을 갖기 때문에 중괄호 안의 실행문들이 동작한다. 이때 if문 내부의 if절의 조건식이 true이면 실행되고 false면 해당 조건문이 실행되지 않는 if문이다.

다음은 if-else문이다.

if문은 true만 실행된다. False 일 때 다르게 실행시키기 위해서는 if-else문을 알아야한다.

if(조건식) {

실행문; // 조건식이 true일 경우 실행

} else {

실행문; // 조건식이 false일 경우 실행

} → if-else문을 정의하는 방법은 이와같다.

```
1  package week4;
2
3  public class Main {
4
5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7          int a = 10;
8          if(a%2==0){  // true
9              System.out.println(a+"는 짝수");  // 실행
10         }else{
11             System.out.println(a+"는 홀수");
12         }
13     }
14 }
15
```

→조건식이 true이면 if절의 실행문을 실행시키고, false면 else 절의 실행문이 실행된다.

다음은 else if문이다.

Else if문을 사용하면 2개 이상의 조건식을 두고 흐름을 제어할 때 더 자유롭고 편리하게 코드를 작성할 수 있다.

if(조건식1) {

실행문1; //조건식 1이 true일 때 실행

} else if(조건식2) {

실행문2; //조건식 1이 false이고 조건식 2가 true일 때 실행

} else {

실행문3; //조건식 1과 2가 false일 때 실행

} → else if문을 정의하는 방법은 이와같다

```java
package week4;

public class Main {

    public static void main(String[] args) {
        int age = 22;
        if(age>=20 && age<30){  // 조건식 1
            // 조건식 1이 true일 때 실행
            System.out.println("20대");
        }else if(age<20){   // 조건식 2
            // 조건식 1이 false이면서 조건식 2가 true일 때 실행
            System.out.println("10대 이하");
        }else{
            // 조건식 1과 2가 false일 때 실행
            System.out.println("30대 이상");
        }
    }
}
```

→else if절은 여러 번 정의할 수 있고, else절은 생략이 가능하다. 단, **if문이 처음에 나오고 그다음에 else if문이 제일 마지막에 else문이 나와야 한다.**

다음은 switch문이다.

switch문은 if문으로 다 대체 가능하지만 때때로 장점을 비교하고자 하는 조건이 많은 경우 사용하면 보기에 더 편한 경우가 있다.

```
Switch (변수) {
Case값: // 변수와 값이 일치하면 해당 case 실행문을 작동시킨다.
    실행문;
    Break; // break는 조건에 해당하는 실행문을 작동시키고 switch문을 종료하기 위해 사용된다.
Default: // 변수와 값이 불일치하면 default 실행문을 작동시킨다.
    실행문;
    Break;
}
```

→ switch문을 정의하는 방법은 이와 같다.

```java
package week4;

public class Main {
    public static void main(String[] args) {
        int n = 2;
        switch (n) {     // 조건
        case 1:          // 값 불일치(미실행)
            System.out.println("1");
            break;
        case 2:          // 값 일치
            System.out.println("2"); // 실행
            break;
        case 3:
            System.out.println("3"); // 실행
            break;
        default:
            System.out.println("4이상");
        }
    }
}
```

위의 예제처럼 조건(2)과 일치하는 값(2)에서만 실행문이 작동된다

2. **반복문.**

반복문에는 for문과 while문이 있다.

먼저 for문을 보면

```
For(int i = 0 ; i<10 ; i++) {
//조건이 참일 경우 for문 내부 실행
}
```

→ for문을 정의하는 방법은 이와 같다.

여기서 int i=0 ←//나를 도와주는 변수 정의, i<10←//조건, i++ ←//나를 도와주는 변수 업데이트 를 표시한다.

for문은 조건 검사 후 참일경우 for문 내부를 실행하고 다시 돌아와서 조건을 검사한다. 조건이 거

짓이 될때까지 반복해준다.

For문 안에서 정의된 변수는 for문이 끝나면 메모리에서 사라진다.

```
1  package week4;
2
3  public class Main {
4      public static void main(String[] args) {
5
6          for(int i=0 ; i<10 ; i++) {
7              System.out.println("i는: "+i);
8          }
9      }
10 }
```

Tip) for문 무한루프

조건을 true로 바꾸거나, 모두 비워주면 된다. 즉 조건만 참이 만족되면 무한루프가 된다.

```
For(int i = 0 ; true ; i++) {
        System.out.println("i는: "+i);
}
```

또,

```
For( ; ; ) {
        System.out.println("hi");
}
```

이 무한루프를 탈출하려면, if문으로 특정조건을 걸어주면 된다.

Break; 를 해주면 루프문을 종료시켜준다.

다음은 while문이다.

for문이랑 거의 비슷하지만 문법이 살짝 다르다.

```
int i=0 //도와주는 변수 선언
while(i<10){ //괄호 안에 조건 넣어주기
   i++ //도와주는 변수 업데이트
} →while 문을 정의하는 방법은 이와 같다.
```

또한 이 while문을 for문으로 바꾸면

`For(int i = 0 ; i<10 ; i++) {}` ←이거와 같다.

for문과 while문의 차이점.

for문은 도와주는 변수가 선언 후 for문이 끝나면 도와주는 변수가 사라지는 반면에 while문은 while문 위에 도와주는 변수를 선언하기 때문에, while문이 끝나도 도와주는 변수가 사라지지 않는다.

do while문

```
do{
        System .out.println("안녕하세요"); //실행할 구문
}while(false); //조건 검사
```

→do while문을 정의하는 방법은 이와 같다.

do while문은 조건이 false여도 한 번은 실행한다.

실행하고 난 후, 조건 검사!

If문 문제풀기

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int score = sc.nextInt();

        if (score >= 90) {
            System.out.println('A');
        } else if (score >= 80) {
            System.out.println('B');
        } else if (score >= 70) {
            System.out.println('C');
        } else if (score >= 60) {
            System.out.println('D');
        } else {
            System.out.println('F');
        }
    }
}
```

(**시험 성적**을 표현한 문제.)

→시험 점수에 따라 등급이 나오는 문제.

for문 문제풀기

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        for (int i = n; i > 0; --i) {
            System.out.println(i);
        }
    }
}
```

(N찍기 문제.)

→N부터 1까지 출력하는 문제.

While문 문제 풀기

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        while(true) {

            int A = sc.nextInt();
            int B = sc.nextInt();

            if(A == 0 && B == 0) {
                break;
            }
            System.out.println(A+B);
        }
    }
}
```

(A+B -5 문제)

→A,B를 입력하여 A+B를 출력하는 프로그램 A,B둘다 0을 입력받으면 종료된다.