

제어문 (조건문과 반복문)

융합프로그래밍

학습 목표

- 조건문의 동작을 이해한다.
- 다양한 반복구문을 이해한다.

제어문

기본적으로 Java에서의 실행은 위에서 아래로 순차적으로 실행이 된다.

```
import java.util.Scanner;

public class SignOperation {
    public static void main(String...args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Input number?");
        int num = sc.nextInt();
        int minus_num = -num;

        System.out.printf("%d, %d\n",
                           num, minus_num);
    }
}
```

제어문

기본적으로 Java에서의 실행은 위에서 아래로 순차적으로 실행이 된다.

```
import java.util.Scanner;
```

```
public class SignOperation {  
    public static void main(String...args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.println("Input number?");  
        int num = sc.nextInt();  
        int minus_num = -num;  
        System.out.printf("%d, %d\n",  
                           num, minus_num);  
    }  
}
```

main 함수로 자바 프로그램을 실행

키보드 입력을 위한 Scanner 생성

화면에 Input number? 출력

키보드에서 int 타입을 입력 받음

minus_num 에 -num 을 입력

minus_num과 num을 화면에 출력



제어문

- 제어문은 실행의 흐름을 바꾸는 방법
- 조건문과 반복문이 있음.

실습 #1 - VoteAge.java

```
import java.util.Scanner;

public class VoteAge {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Input your age? ");
        int age = sc.nextInt();
        if (age >= 19) {
            System.out.println("You can vote!");
        } else {
            System.out.println("You can't vote!");
        }
    }
}
```

if - 해당 조건이 true 일 때만 수행된다.

```
if (boolean 조건) {  
    위의 조건이 참 일때만 실행됨.  
}
```

else - 연결된 위의 조건이 모두 false 일 때 실행

```
if (boolean 조건) {  
    조건이 참 일때만 실행됨.  
} else {  
    조건이 만족하지 않으면 실행  
}
```


실습 #2 - VoteAge2.java

```
import java.util.Scanner;

public class VoteAge2 {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Input your age? ");
        int age = sc.nextInt();
        if (age >= 19)
            System.out.println("You can vote!");
        else
            System.out.println("You can't vote!");
    }
}
```

if, else 의 경우 실행될 문장이 한줄이면 {, }를 안 써도 됨

if (boolean 조건)

위의 조건이 참 일때만 한줄 만 실행됨.

if, else 의 경우 실행될 문장이 한줄이면 {, }를 안 써도 됨

if (boolean 조건)

a = a + 1; <- 참일 때만 실행

b = b + 1; <- 항상 실행

실습 #3 - VoteAge3.java

```
import java.util.Scanner;

public class VoteAge3 {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Input your age? ");
        int age = sc.nextInt();
        if (age > 19) {
            System.out.println("You can vote!");
        } else if (age == 19) {
            System.out.println("Check Internaltional Age!");
        }
    }
}
```

else if - else 상황에서 다시 if 조건을

```
if (boolean 조건1) {  
    위의 조건이 참 일때만 실행됨.  
} else if (boolean 조건2) {  
    조건1이 거짓이고 조건2가 참일때 실행  
}
```

실습 #4 VoteAge4.java

```
import java.util.Scanner;

public class VoteAge4 {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Input your age? ");
        int age = sc.nextInt();
        if (age > 19) {
            System.out.println("You can vote!");
        } else if (age == 19) {
            System.out.println("Check Internaltional Age!");
        } else {
            System.out.println("You can't vote!");
        }
    }
}
```

else if 가 있을 때의 else 는?

```
if (boolean 조건1) {  
    위의 조건이 참 일때만 실행됨.  
} else if (boolean 조건2) {  
    조건1이 거짓이고 조건2가 참일때 실행  
} else {  
    위의 조건들이 모두 거짓일때 실행  
}
```

실습 #5 - Score.java

- 다음 조건의 프로그램을 완성하시오.
- 0~100까지의 숫자를 입력받아서 숫자값에 따라서 다음 결과를 출력하시오.

숫자값	출력값
90~100	A
80~89	B
70~79	C
60~69	D
그 외의 모든값	F

실습 #6 - ScoreSwitch.java

```
import java.util.Scanner;

public class ScoreSwitch {
    public static void main(String...args) {
        Scanner sc = new Scanner(System.in);
        int score = sc.nextInt() / 10;

        switch(score) {
            case 10:
            case 9:
                System.out.println('A');
                break;
            case 8:
                System.out.println('B');
                break;
            case 7:
                System.out.println('C');
                break;
            case 6:
                System.out.println('D');
                break;
            default:
                System.out.println('F');
                break;
        }
    }
}
```

switch #1

- switch 문에는 변수가 들어가고 해당 변수와 동일한 case 문이 실행된다.
- 실행되는 문장은 break를 만나기 전까지 계속 실행된다.

switch #2

```
switch(변수) {  
  case 변수1:  
    일치하면 실행  
    break;  
  case 변수2:  
    .....  
}
```

switch #3

```
switch(변수) {  
    case 2018:  
        일치하면 실행  
        break;  
    case 2019:  
        .....  
}
```

변수가 2018 이면

일치하면 실행

여기까지 실행

switch #4

- break 가 없으면 아래로 계속 실행된다.

```
switch(변수) {  
  case 1:  
    일치하면 실행  
  case 2:  
    .....  
  case 3:  
    .....  
    break;  
}
```



switch #5 - default

```
switch(변수) {  
  case 변수1:  
    일치하면 실행  
    break;  
  default:  
    .....  
}
```

switch #6 - default

```
switch(변수) {
```

```
case 변수1:
```

```
    일치하면 실행
```

```
    break;
```

```
default:
```

switch 문과 case 문의 변수 조건이 하나도 맞지 않으면 실행

```
.....
```

```
}
```

switch #7

- 문자열도 가능함.

```
switch(변수) {  
  case "hello":  
    일치하면 실행  
  case "2":  
    .....  
  case "3":  
    .....  
    break;  
}
```



실습 #7 - While.java

```
public class While {  
    public static void main(String...args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.println("i => " + i);  
            i++;  
        }  
    }  
}
```

while (조건)

```
while (조건) {  
    조건이 참일 동안 계속 아래 조건이 반복적으로 실행  
}
```

while (조건)

while (조건) {
 조건이 참일 동안 계속 아래 조건이 반복적으로 실행
}

boolean 형식만 가능

반복문을 적용한 코드의 비교

```
public class Print {  
    public static void main(String...args)  
{  
        System.out.println("i => 1");  
        System.out.println("i => 2");  
        System.out.println("i => 3");  
        System.out.println("i => 4");  
        System.out.println("i => 5");  
    }  
}
```

```
public class While {  
    public static void main(String...args)  
{  
        int i = 0;  
        while (i < 5) {  
            System.out.println("i => " + i);  
            i++;  
        }  
}
```

실습 #8 - 1 부터 100 까지 더하는 프로그램

- 1 부터 100까지의 합을 출력하는 프로그램을 while 문을 통해서 구현하시오.

무한 루프

```
public class InfiniteLoop {  
    public static void main(String...args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.println("i => " + i);  
            i++;  
        }  
    }  
}
```

**이 코드가 없으면 어떻게
동작하게 될까요?**

실습 #9 구구단 출력 프로그램을 작성

- 1단 부터 9단 까지 구구단을 출력하는 프로그램을 while 문으로 작성합니다.
- 힌트 : while 문을 두 개 사용하십시오.

실습 #10 - DoWhile.java

```
import java.util.Scanner;

public class DoWhile {
    public static void main(String...args) {
        Scanner sc = new Scanner(System.in);
        int num;
        do {
            System.out.println("Input Number(0 is quit) => ");
            num = sc.nextInt();
            System.out.println("number is " + num);
        }while(num != 0);
    }
}
```


Do ~ while (조건);

do {

반복 코드

} while (조건);

무조건 최초 한번은 실행됨
그 뒤에 조건을 평가함



조건이 참일 동안 계속 아래 조건이 반복적으로 실행

Do ~ while (조건);

```
do {  
    반복 코드 ①  
} while (조건);  
        ②
```

1 수행 뒤에 2 조건을 판단,
조건이 참이면 다시 1 -> 2
순서로 실행

일부러 무한 루프를 만들어야 할 경우

- 계속 동작해야 하는 프로그램의 경우에는 일부러 종료하지 않도록 만들 필요가 있다.
- 실습 #10 과 같은 경우

실습 #11 - 두 수를 입력받아서 더 하는 프로그램을 작성하시오.

- 두 수가 모두 음수가 되기 전까지는 계속적으로 두 수를 입력받고 그 합을 출력하시오.
- input > 1
- input > 2
- sum > 3
- input > 3
- input > 5
- sum > 8

실습 #12 - For.java

```
public class For {  
    public static void main(String...args) {  
        int i = 0;  
        for (i = 0; i < 5; i++) {  
            System.out.println("i => " + i);  
        }  
    }  
}
```

for

- for 는 세 개의 파트로 구성
 - 해당 파트는 ; 로 구분

```
for (초기 값 설정; 조건; 증감 식) {  
    반복 내용  
}
```

최초 한번만 실행



for

- for 는 세 개의 파트로 구성
 - 해당 파트는 ; 로 구분

```
for (초기 값 설정 ; 조건 ; 증감 식) {  
    반복 내용  
}
```

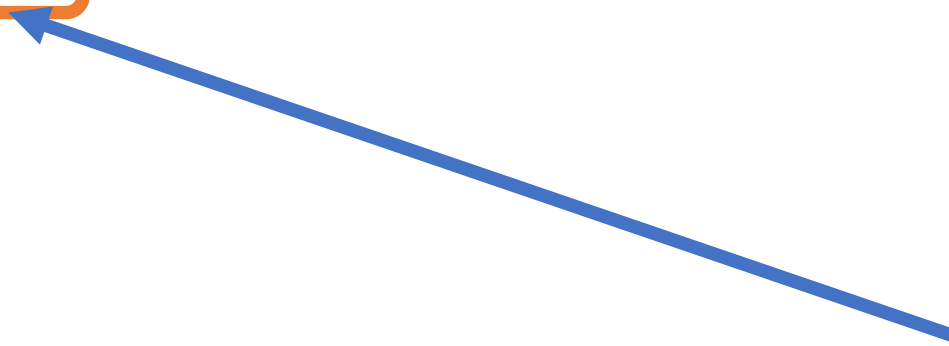
조건이 true 면
반복 내용이 실행됨



for

- for 는 세 개의 파트로 구성
 - 해당 파트는 ; 로 구분

```
for (초기 값 설정 ; 조건 ; 증감 식) {  
    반복 내용  
}
```



조건이 false 면
for 문이 끝나고 그
아래로 이동

for

- for 는 세 개의 파트로 구성
 - 해당 파트는 ; 로 구분

```
for (초기 값 설정 ; 조건 ; 증감 식) {  
    반복 내용  
}
```

한번 반복 내용이 실행될 때 마다
실행되는 증감 식



for

- for 는 세 개의 파트로 구성
 - 해당 파트는 ; 로 구분

```
for (초기 값 설정; 조건; 증감 식) {  
    반복 내용  
}
```

조건 2가 참이면 계속
2 -> 3 -> 4 순으로 반복됨

for 와 while의 비교

- for, while은 표현 방식만 다름

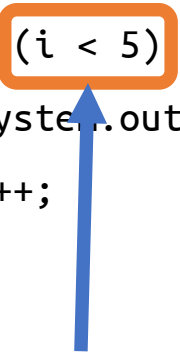
```
public class While {  
    public static void main(String...args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.println("i => " + i);  
            i++;  
        }  
    }  
}
```

```
public class For {  
    public static void main(String...args) {  
        int i = 0;  
        for (i = 0 ; i < 5; i++) {  
            System.out.println("i => " + i);  
        }  
    }  
}
```

for 와 while의 비교

- for, while은 표현 방식만 다름

```
public class While {  
    public static void main(String...args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.println("i => " + i);  
            i++;  
        }  
    }  
}
```



조건 자체에 집중

```
public class For {  
    public static void main(String...args) {  
        int i = 0;  
        for (i = 0 ; i < 5; i++) {  
            System.out.println("i => " + i);  
        }  
    }  
}
```

for 와 while의 비교

- for, while은 표현 방식만 다름

```
public class While {  
    public static void main(String...args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.println("i => " + i);  
            i++;  
        }  
    }  
}
```

```
public class For {  
    public static void main(String...args) {  
        int i = 0;  
        for (i = 0 ; i < 5; i++) {  
            System.out.println("i => " + i);  
        }  
    }  
}
```



조건과 증감식에 집중

실습 #13-1 구구단 출력 프로그램을 작성

- 1단 부터 9단 까지 구구단을 출력하는 프로그램을 for 문으로 작성합니다.
- 힌트 : for 문을 두 개 사용하십시오.

실습 #13-2 구구단 출력 프로그램을 작성 2

- 숫자를 1~9까지 입력받고, 입력 받은 단 부터 +2 단까지 출력하는 프로그램을 작성하시오. 다만 최대 출력하는 단은 9단까지만...
- 1을 입력받으면 1~3단, 8을 입력받으면 8~9, 9를 입력받으면 9단만 출력

while, for 도 한줄만 실행 할 때 {} 생략 가능

```
public class While {  
    public static void main(String...args) {  
        int i = 0;  
        while (i++ < 5)  
            System.out.println("i => " + i);  
    }  
}
```

```
public class For {  
    public static void main(String...args) {  
        int i = 0;  
        for (i = 0 ; i < 5; i++)  
            System.out.println("i => " + i);  
    }  
}
```


실습 #14 - 어떤 결과가 나올까요?

```
public class While {  
    public static void main(String...args) {  
        int i = 0;  
        if (i > 0);  
            System.out.println ("i > 0");  
        while (i++ < 5)  
            System.out.println("i => " + i);  
    }  
}
```

조건문 반복문 등에서 흔히 하는 실수

```
public class While {  
    public static void main(String...args) {  
        int i = 0;  
        if (i > 0);  
            System.out.println ("i > 0");  
        while (i++ < 5;  
            System.out.println("i => " + i);  
        }  
    }  
}
```

```
public class For {  
    public static void main(String...args) {  
        int i = 0;  
        if (i > 0);  
            System.out.println ("i > 0");  
        for (i = 0 ; i < 5; i++;  
            System.out.println("i => " + i);  
        }  
    }  
}
```

조건문 반복문 등에서 흔히 하는 실수

```
public class While {  
    public static void main(String...args) {  
        int i = 0;  
        if (i > 0);  
            System.out.println ("i > 0");  
        while (i++ < 5;  
            System.out.println("i => " + i);  
        }  
    }  
}
```

```
public class For {  
    public static void main(String...args) {  
        int i = 0;  
        if (i > 0);  
            System.out.println ("i > 0");  
        for (i = 0 ; i < 5; i++;  
            System.out.println("i => " + i);  
        }  
    }  
}
```

; 이 있으면 루프가 저기서 끝나버림

실습 #15 - Break

```
public class Break {  
    public static void main(String...args) {  
        for (int i = 0; i < 10; i++) {  
            System.out.println("i => " + i);  
            if (i > 5) {  
                break;  
            }  
        }  
    }  
}
```

break

- 반복문에서 break 문을 만나면 반복문을 종료하고 빠져나온다.

실습 #16 - Break2

```
public class Break2 {  
    public static void main(String...args) {  
        for (int i = 0; i < 5; i++) {  
            for (int j = 0; j < 10; j++) {  
                System.out.println("i,j => " + i + " " + j);  
                if (j == 0) {  
                    break;  
                }  
            }  
            System.out.println("i => " + i);  
        }  
    }  
}
```

중첩 루프에서의 break

```
public class Break2 {  
    public static void main(String...args) {  
        for (int i = 0; i < 5; i++) {  
            for (int j = 0; j < 10; j++) {  
                System.out.println("i,j => " + i + " " + j);  
                if (i == 0) {  
                    break;  
                }  
            }  
            System.out.println("i => " + i);  
        }  
    }  
}
```

**break를 만나면 어디로 이동하
게 될까요?**

중첩 루프에서의 break

```
public class Break2 {  
    public static void main(String...args) {  
        for (int i = 0; i < 5; i++) {  
            for (int j = 0; j < 10; j++) {  
                System.out.println("i,j => " + i + " " + j);  
                if (j == 0) {  
                    break;  
                }  
            }  
            System.out.println("i => " + i);  
        }  
    }  
}
```

break가 속한 반복문에서만 빠져나온다.

실습 #17 - continue

```
public class Continue {  
    public static void main(String...args) {  
        int sum = 0;  
        for (int i = 0; i < 100; i++) {  
            if (i % 2 == 0) {  
                continue;  
            }  
  
            sum += i;  
        }  
  
        System.out.println("sum = " + sum);  
    }  
}
```

continue

- 반복문에서 continue를 만나면 해당 반복문의 처음으로 이동한다.

```
for (초기 값 설정; 조건; 증감 식) {  
    반복 내용1  
    continue;  
    반복 내용2  
}
```

조건 2가 참이면 계속
2 -> 3 -> 5 순으로 반복됨
4는 실행되지 않음