

반복문

▶ 반복문

프로그램 수행 흐름을 바꾸는 역할을 하는 제어문 중 하나로 특정 문장들을 반복해서 수행하도록 함

✓ 예시

// "Happy" 라는 문장 5번 출력하기

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

▶ 반복문

프로그램 수행 흐름을 바꾸는 역할을 하는 제어문 중 하나로 특정 문장들을 반복해서 수행하도록 함

✓ 반복문의 종류

for문

```
for(초기식; 조건식; 증감식)
{
    수행될 문장;
}
```

while문

```
while(조건식) {
    수행될 문장;
    [증감식 or 분기문];
}
```

▶ for문

✓ for

```
for(초기식; 조건식; 증감식) {  
    수행될 문장;  
}
```

1회전: 초기식 확인 후 조건식 확인

조건식이 true면 문장 수행

조건식이 false면 수행하지 않음

2회전: 증감식 연산 후 조건식 확인

조건식이 true면 문장 수행

조건식이 false면 수행하지 않음

* 2회전 이상부터는 모두 **2회전과 동일**하고
조건식이 false가 나올 때까지 문장 수행

✓ for문 예시

```
for(int i = 1; i <= 10; i++) {  
    System.out.println(i + " 출력");  
}
```

✓ 실행 결과

```
1 출력  
2 출력  
...  
9 출력  
10 출력
```

for문

```
for(int i = ①; i < ②; i++ ④) {  
    System.out.println(. . .); ③  
}
```

첫 번째 루프의 흐름 [i=0]

① → ② → ③ → ④

두 번째 루프의 흐름 [i=1]

② → ③ → ④

세 번째 루프의 흐름 [i=2]

② → ③ → ④

네 번째 루프의 흐름 [i=3]

② i=3이므로 탈출!

▶ while문

✓ while

```
while(조건식) {  
    수행될 문장;  
    [증감식 or 분기문];  
}
```

조건식이 true일 때 문장 수행
문장 수행이 끝나면 조건식
다시 확인 후 true면 수행,
조건식이 false가 될 때까지 수행
조건식이 false가 되면 반복문 종료

* {} 안에 조건을 벗어나게 할 연산(증감식, 분기문) 필요

✓ while문 예시

```
int i = 1;  
while(i <= 10) {  
    System.out.println(i + " 출력");  
    i++;  
}
```

✓ 실행 결과

```
1 출력  
2 출력  
...  
9 출력  
10 출력
```

▶ while문

✓ do ~ while

```
do {
    수행될 문장;
    [증감식 or 분기문];
} while(조건식);
```

do 안의 내용 먼저 실행
조건식 확인 후 true면 문장 수행,
false면 종료
while 뒤에 ; 꼭 필요

* while과 do~while의 차이점 :

do~while은 조건문이 true가 아니더라도
무조건 한 번 이상 수행

* {} 안에 조건을 벗어나게 할 연산(증감식, 분기문) 필요

✓ do ~ while문 예시

```
int i = 1;
do {
    System.out.println(i + "출력");
    i++;
} while(i <= 10);
```

✓ 실행 결과

1 출력
2 출력
...
9 출력
10 출력

```
for(...;...;...) {  
    for(...;...;...) {  
        . . .  
    }  
}
```

```
while(...) {  
    for(...;...;...) {  
        . . .  
    }  
}
```

```
do {  
    for(...;...;...) {  
        . . .  
    }  
} while(...);
```

```
for(...;...;...) {  
    while(...) {  
        . . .  
    }  
}
```

```
while(...) {  
    while(...) {  
        . . .  
    }  
}
```

```
do {  
    while(...) {  
        . . .  
    }  
} while(...);
```

```
for(...;...;...) {  
    do {  
        . . .  
    } while(...);  
}
```

```
while(...) {  
    do {  
        . . .  
    } while(...);  
}
```

```
do {  
    do {  
        . . .  
    } while(...);  
} while(...);
```

생각해 볼 수 있
는 반복문의 중
첩의 형태

▶ 중첩 반복문


✓ 표현식

```
for(초기값1; 조건식1; 증감식1) {  
    수행될 문장1;  
    for(초기값2; 조건식2; 증감식2) {  
        수행될 문장2;  
    }  
    수행될 문장3;  
}
```

for문에 진입하면 수행될 문장1을 먼저 수행하고 두 번째 for문에 진입하면 조건식2가 false가 될 때까지 수행될 문장2를 수행 후 나오면 수행될 문장3을 수행하고 조건식1로 돌아와 true면 다시 반복

◆ ForInFor.java

```
1. class ForInFor {
2.     public static void main(String[] args) {
3.         for(int i = 0; i < 3; i++) {      // 바깥쪽 for문
4.             System.out.println("-----");
5.             for(int j = 0; j < 3; j++) {    // 안쪽 for문
6.                 System.out.print "[" + i + ", " + j + " ] ");
7.             }
8.             System.out.print('\n');
9.         }
10.    }
11. }
```



명령 프롬프트

C:\JavaStudy>java ForInFor

[0, 0] [0, 1] [0, 2]

[1, 0] [1, 1] [1, 2]

[2, 0] [2, 1] [2, 2]

C:\JavaStudy>

for문 중첩의 예

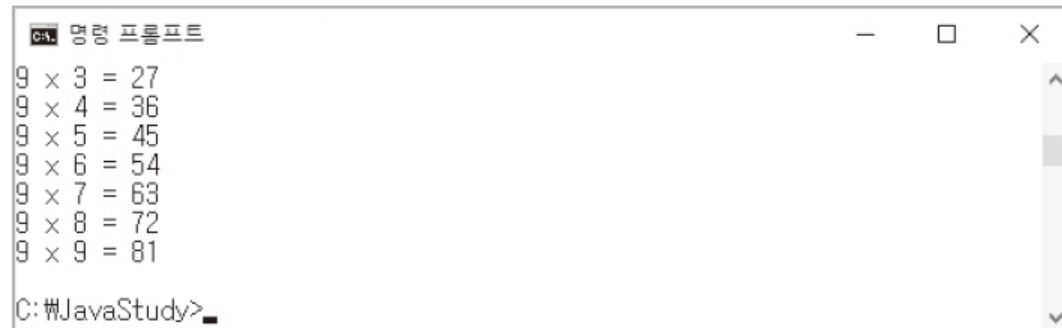
바깥쪽 for문 담당 →

안쪽 for문
담당 ↓

2 × 1 = 2	3 × 1 = 3	4 × ..	5 ..	6 .	7 .	8 .	9 × 1 = 9
2 × 2 = 4	3 × 2 = 6	4 × ..	5 ..	6 .	7 .	8 .	9 × 2 = 18
2 × 3 = 6	3 × 3 = 9	4 × ..	5 ..	6 .	7 .	8 .	9 × 3 = 27
2 × 4 = 8	3 × 4 = 12	4 × ..	5 ..	6 .	7 .	8 .	9 × 4 = 36
2 × 5 = 10	3 × 5 = 15	4 × ..	5 ..	6 .	7 .	8 .	9 × 5 = 45
2 × 6 = 12	3 × 6 = 18	4 × ..	5 ..	6 .	7 .	8 .	9 × 6 = 54
2 × 7 = 14	3 × 7 = 21	4 × ..	5 ..	6 .	7 .	8 .	9 × 7 = 63
2 × 8 = 16	3 × 8 = 24	4 × ..	5 ..	6 .	7 .	8 .	9 × 8 = 72
2 × 9 = 18	3 × 9 = 27	4 × ..	5 ..	6 .	7 .	8 .	9 × 9 = 81

구구단 전체 출력을 위한 관찰

```
for(int i = 2; i < 10; i++) {    // 2단부터 9단까지 진행 위한 바깥쪽 for문
    for(int j = 1; j < 10; j++)    // 1부터 9까지의 곱을 위한 안쪽 for문
        System.out.println(i + " x " + j + " = " + (i * j));
}
```



```
C:\#JavaStudy>
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```

구구단 출력 예제

▶ 분기문

✓ break

반복문에서는 break문 자신이 포함된 가장 가까운 반복문을 빠져나가는 구문

✓ break문 예시

```
for(int i = 1;; i++) {  
    System.out.println(i + " 출력");  
  
    if(i >= 10) {  
        break;  
    }  
}
```

▶ 분기문

✓ continue

반복문 내에서만 사용 가능하며 반복문 실행 시 continue 아래 부분은 실행하지 않고 반복문 다시 실행

for문의 경우 증감식으로 이동,

while(do~while)문의 경우 조건식으로 이동

전체 반복 중에 특정 조건을 만족하는 경우를 제외하고자 할 때 유용

✓ continue문 예시

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