



반복문과 배열 그리고 예외 처리

# 반복문(Loop)

2

- 반복문은 특정 코드를 여러 번 반복해서 실행하고 싶을 때 사용하는 프로그래밍 구조입니다.

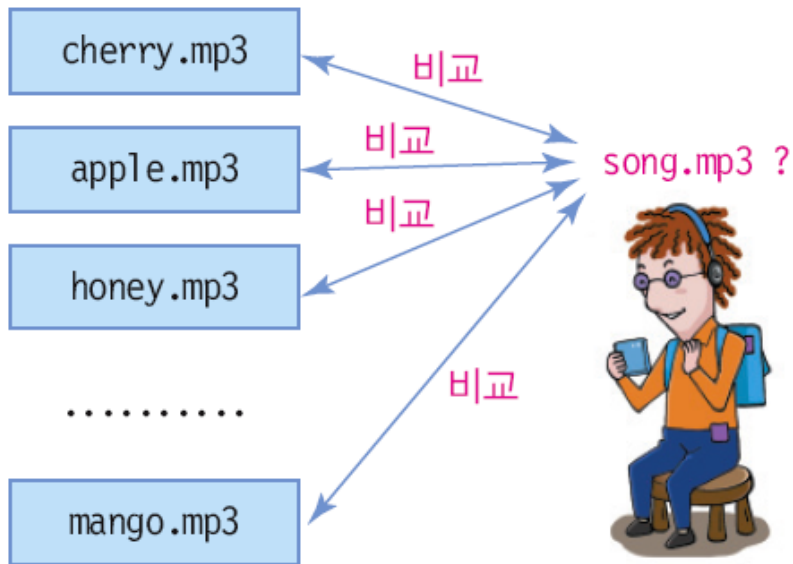


# 반복문

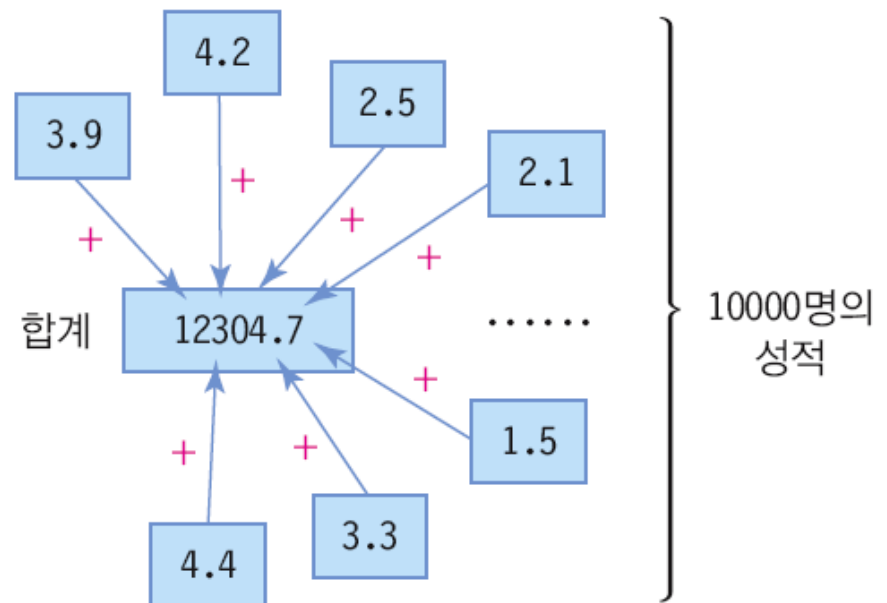
3

## □ 자바 반복문의 종류

- ▣ for 문
- ▣ while 문



Mp3에 있는 목록을 조회할 때



N명의 성적을 계산할 때

# for문

4

- for 문은 반복 횟수가 정해져 있을 때 주로 사용
- for문은 4가지 순서를 가짐

for (①초기식; ②조건식; ④증감식)

{

③ 조건식의 결과가 참인 동안 반복적으로 실행하고자 하는 명령문;

...

};

- ①
  - for 문이 실행한 후 오직 한번만 실행되는 초기화 작업
  - 콤마(',')로 구분하여 여러 문장 나열 가능
- ②
  - 논리형 변수나 논리 연산만 가능
  - 반복 조건이 true이면 반복 계속, false이면 반복 종료
  - 반복 조건이 true 상수인 경우, 무한 반복
- ④
  - 반복 작업 문장들의 실행 후 처리 작업
  - 콤마(',')로 구분하여 여러 문장 나열 가능

# for문의 예시

5

- 0에서 9까지 정수 출력

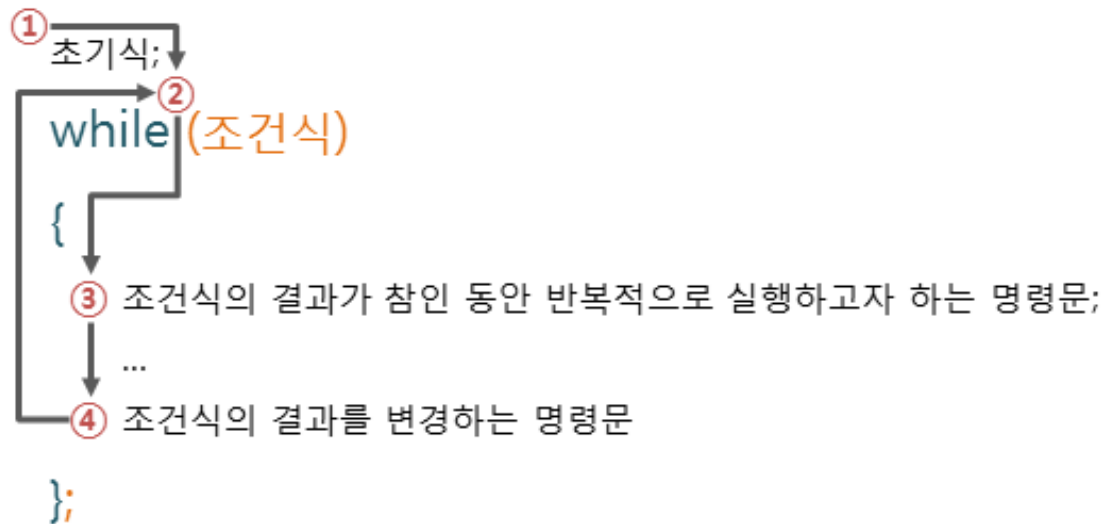
```
for (int i = 1; i <= 9; i++) {  
    System.out.println(i + "번째 반복");  
}
```

- 0에서 100까지의 합 구하기

```
int sum = 0;  
for (int i = 0; i <= 100; i++) {  
    sum += i;  
}
```

# while 문의 구성

6



- 반복 조건이 true이면 반복, false이면 반복 종료
- 반복 조건이 없으면 컴파일 오류
- 처음부터 반복조건을 통과한 후 작업문 수행

# 중첩 반복

7

## □ 중첩 반복

- 반복문이 다른 반복문을 내포하는 구조
- 이론적으로는 몇 번이고 중첩 반복 가능
- 너무 많은 중첩 반복은 프로그램 구조를 복잡하게 하므로 2중 또는 3중 반복이 적당

```
for(i=0; i<100; i++) { // 100 개의 학교 성적을 모두 더한다.
```

```
.....
```

```
    for(j=0; j<10000; j++) { // 10000 명의 학생 성적을 모두 더한다.
```

```
        .....
```

```
        .....
```

```
    }
```

```
.....
```

```
}
```

10000명의 학생이 있는 100개 대학의 모든 학생 성적의 합을 구할 때,  
for 문을 이용한 이중 중첩 구조

# 예제 : 구구단

8

2중 중첩된 for문을 사용하여 구구단을 출력하는 프로그램을 작성하시오.  
한 줄에 한 단씩 출력한다.

```
public class NestedLoop {  
    public static void main (String[] args) {  
        int i, j;  
  
        for (i = 1; i < 10; i++, System.out.println()) {  
            for (j = 1; j < 10; j++, System.out.print("\t")) {  
                System.out.print(i + "*" + j + "=" + i*j);  
            }  
        }  
    }  
}
```

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



# continue문

9

## □ continue 문

- 반복문을 빠져 나가지 않으면서
- 반복문 실행 도중 다음 반복을 진행

```
for (초기문; 조건식; 반복후작업) {  
    .....  
    continue;  
    .....  
}
```

분기

```
while (조건식) {  
    .....  
    continue;  
    .....  
}
```

조건식으로  
분기


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

조건식으로  
분기

# 예제 : 1부터 100까지 짝수의 합

10

for와 continue문을 사용하여 1부터 100까지 짝수의 합을 구해보자.

```
public class ContinueExample {  
    public static void main (String[] args) {  
        int sum = 0;  
        for (int i = 1; i <= 100; i++) {  
            if (i%2 == 1)  
                continue;   
            else  
                sum += i;  
        }  
        System.out.println("1부터 100까지 짝수의 합은 " + sum);  
    }  
}
```

1부터 100까지 짝수의 합은 2550

# break문

11

## □ break 문

- ▣ 반복문 하나를 완전히 빠져 나갈 때 사용
- ▣ break문은 하나의 반복문만 벗어남
  - 중첩 반복의 경우 안쪽 반복문의 break 문이 실행되면 안쪽 반복문만 벗어남