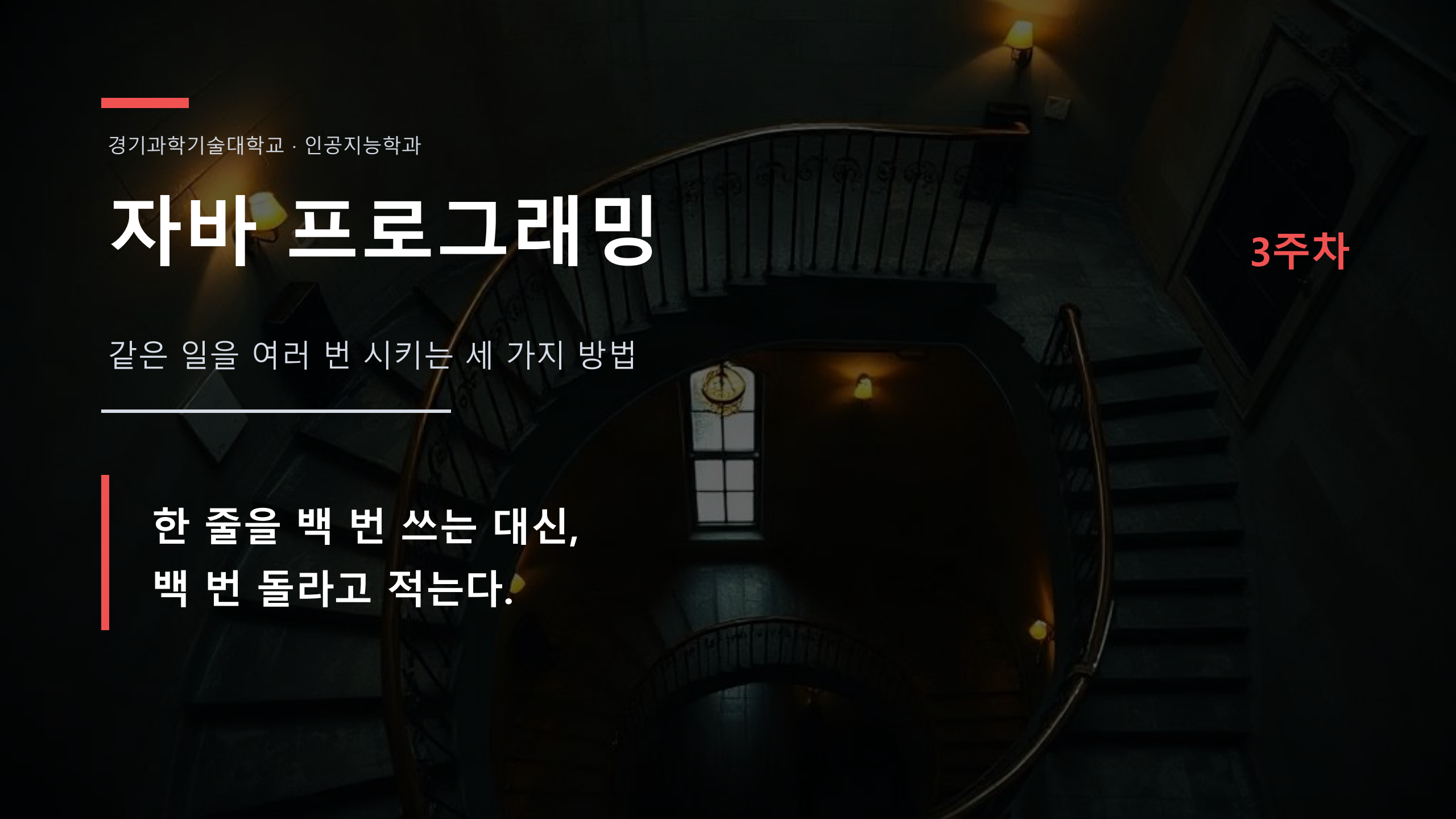




경기과학기술대학교 · 인공지능학과

자바 프로그래밍

3주차

같은 일을 여러 번 시키는 세 가지 방법

한 줄을 백 번 쓰는 대신,
백 번 돌라고 적는다.

오늘의 강의

1 교시 for 문, 2 교시 while 과 do-while, 3 교시 중첩 반복과 흐름 바꾸기

1교시



for 문

구조와 도는 순서
합을 쌓는 변수

2교시



while 과 do-while

횟수를 모를 때
무엇을 쓸지 고르는 기준

3교시



중첩과 흐름

구구단과 별 찍기
continue 와 break

1교시

for 문

세 자리만 정하면 나머지는 컴퓨터가 센다.

구조와 순서

조건식의 경계

쌓는 변수

지난주에서 이어받는 것

오늘 새로 배우는 것은 몇 번 되풀이할지 정하는 틀뿐이다.

증감 연산자 `i++`

반복 후 작업

비교·논리 연산자

조건식

`if / if-else`

반복 안의 분기

Scanner

2교시 입력 반복

오늘부터 우리 실습실

지난주 Programiz

들어가기

주소만 열면 된다

코드 이름

클래스는 늘 Main

한글 출력

시작 템플릿 한 줄이 필요

교수 코드

없음

오늘부터 실습실

학번으로 로그인

파일 이름 = 클래스 이름

그대로 나온다

주차별코드 폴더에 있다

QR로 실습실 들어가기



javalearning.monday9.com

실습실 PC 브라우저로 연다

로그인 화면

경기과학기술대학교 인공지능학과
자바프로그래밍 과목 실습실

학번과 비밀번호로 들어주세요.

학번

비밀번호

들어가기

학번과 비밀번호로 들어간다



처음 로그인하기

1

학번 칸

학번 칸에 학번 입력

2

처음 비밀번호
규칙

처음 비밀번호는
학번 뒤에 이름
(예 : 20260101홍길동)

3

첫 로그인 때
변경

첫 로그인 때
비밀번호를 바꾼다

4

준비 화면

준비 화면이
끝날 때까지
기다린다

실습실 첫 화면

경기과학기술대학교 인공지능학과 자바프로그래밍 과목 실습실

테스트학생 [→]

workspace

주차별코드

1주차

2주차

3주차

Ex01_ForFirst.java

Ex02_ForSample.java

Ex03_OddSum.java

Ex04_MySum.java

Ex06_WhileFirst.java

Ex07_WhileSample.j...

Ex08_SentinelSum.j...

Ex09_DoWhileSam...

Ex10_MultipleOf3Fo...

Ex11_MultipleOf3W...

Ex12_StarRectangl...

Ex13_StarTriangle.j...

Ex14_NestedLoop.j...

Ex15_ContinueExa...

Ex16_RockPaperSci...

out

Main.java

실행

저장

초기화

제출

힌트 (38)

주차별코드/3주차/Ex01_ForFirst.java 교수님 자료 · 읽기 전용

```
1 public class Ex01_ForFirst {
2     public static void main(String[] args) {
3         for (int i = 0; i < 10; i++) {
4             System.out.print(i);
5         }
6     }
7 }
8
```

터미널

실행 결과

▶ 실행을 누르면 결과가 여기에 나와요

출력은 하단 독의 실행 결과 탭

실습실 화면 구성



파일 트리



편집기



하단 독 (터미널 / 실행 결과)



실행 버튼



힌트 버튼

주차별코드 폴더

workspace

주차별코드

1주차

2주차

3주차

Ex01_ForFirst.java

Ex02_ForSample.java

Ex03_OddSum.java

Ex04_MySum.java

Ex06_WhileFirst.java

Ex07_WhileSample.j...

Ex08_SentinelSum.j...

Ex09_DoWhileSam...

Ex10_MultipleOf3Fo...

Ex11_MultipleOf3W...

Ex12_StarRectangl...

Ex13_StarTriangle.j...

Ex14_NestedLoop.j...

Ex15_ContinueExa...

Ex16_RockPaperSci...

out

Main.java

실행

저장

초기화

제출

힌트 (38)

주차별코드/3주차/Ex01_ForFirst.java 교수님 자료 · 읽기 전용

```
1 public class Ex01_ForFirst {
2     public static void main(String[] args) {
3         for (int i = 0; i < 10; i++) {
4             System.out.print(i);
5         }
6     }
7 }
8
```

터미널

실행 결과

▶ 실행을 누르면 결과가 여기에 나와요

파일 번호가 슬라이드 순서다

주차별코드와 내 작업공간

1

읽기 전용

주차별코드/3주차 파일은
자물쇠, 읽기만

2

복사

고치려면 내 작업공간으로 복사

3

새 파일 버튼

새로 칠 땐 파일 트리 위쪽
새 파일 버튼을 누르고 이름 입력

4

파일 이름 =
클래스 이름

파일 이름 = public 클래스 이름
Ex01_ForFirst.java ↔
public class Ex01_ForFirst

실행 중인 화면

workspace

주차별코드

1주차

2주차

3주차

Ex01_ForFirst.java

Ex02_ForSample.java

Ex03_OddSum.java

Ex04_MySum.java

Ex06_WhileFirst.java

Ex07_WhileSample.j...

Ex08_SentinelSum.j...

Ex09_DoWhileSam...

Ex10_MultipleOf3Fo...

Ex11_MultipleOf3W...

Ex12_StarRectangl...

Ex13_StarTriangle.j...

Ex14_NestedLoop.j...

Ex15_ContinueExa...

Ex16_RockPaperSci...

out

Ex05_WhileError.java

Main.java

중지

저장

초기화

제출

힌트 (38)

주차별코드/3주차/Ex07_WhileSample.java 교수님 자료 · 읽기 전용

```
1 import java.util.Scanner;
2
3 public class Ex07_WhileSample {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         int count = 0, n = 0;
7         double sum = 0;
8
9         System.out.println("정수를 입력하고 마지막에 0을 입력하세요.");
10        while ((n = scanner.nextInt()) != 0) {
11            sum = sum + n;
12            count++;
13        }
14        System.out.print("수의 개수는 " + count + "개이며 ");
15        System.out.println("평균은 " + sum / count + "입니다.");
16        scanner.close();
17    }
18 }
19
```

터미널

실행 결과

컴파일 중...

실행: 주차별코드/3주차/Ex07_WhileSample.java

정수를 입력하고 마지막에 0을 입력하세요.

10

30

-20

40

0

멈추고 싶으면 같은 자리의 ■ 중지

실행하고 입력 넣기

- 1 파일 트리에서 파일을 연다
- 2 ▶ 실행 (저장은 자동)
- 3 하단 독 실행 결과 탭에 출력이 나온다
- 4 입력을 기다리면 커서 자리에 치고 Enter
- 5 실행이 끝났어요 가 뜨면 끝

막히면 화면 문장부터

컴파일에서 막혀 실행하지 못했어요

→ 오류 줄 번호로 간다

실행이 오류로 끝났어요

→ 예외 이름 줄을 읽는다

실행이 너무 오래 걸려 멈췄어요

→ 조건식을 본다

실습 환경을 준비하고 있어요

→ 기다린다

같은 줄을 100번 쓰면

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("반가워요");  
        System.out.println("반가워요");  
        : (97번 더 복사)  
        System.out.println("반가워요");  
    }  
}
```

한 곳을 고치려면
백 곳을 고친다

문구 하나만 바뀌도
100군데를 손으로 고쳐야 한다

반복이 필요한 세 가지 상황



합계 구하기

1에서 100까지 더하기



평균 구하기

학생 수만큼 점수 더해 평균 내기



이름 찾기

이름이 나올 때까지 목록을 비교하기

세 상황 모두 같은 일을 여러 번 하는 것이다.

자바의 반복문은 셋



for 문

가장 많이 쓴다



while 문

그다음으로 흔하다



do-while 문

쓸 일이 드물다

오늘 이 세 가지를 모두 다룬다.

for 문의 구조

</>

for (초기문 ; 조건식 ; 반복 후 작업)

1

초기문
첫째 자리

2

조건식
둘째 자리

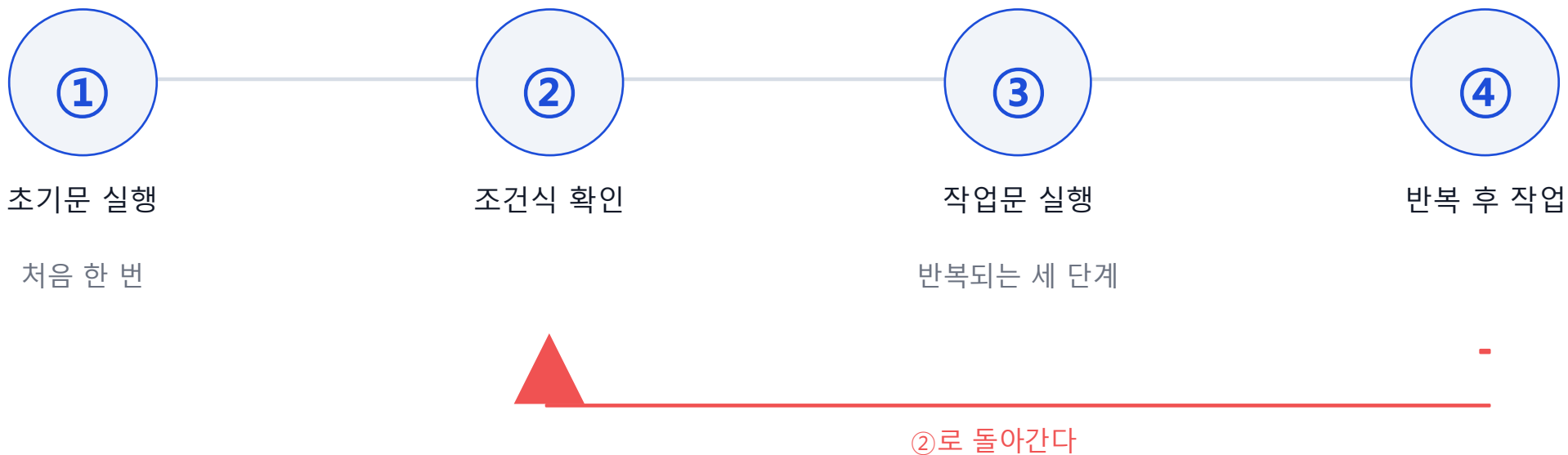
3

반복 후 작업
셋째 자리

{ }

중괄호 안은 되풀이되는 작업문이다

for 문이 도는 순서



i 가 어떻게 변하나

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

i	조건과 결과
0	조건 참, 0 출력
1~8	같은 방식으로 반복, 1부터 8까지 출력
9	조건 참, 9 출력
10	조건 거짓, 반복 종료

초기문

for (초기문 ; 조건식 ; 변경문)

초기문 · $i = 0$

조건식 · $i < 10$

변경문 · $i++$

초기문이 하는 일

- 반복 변수의 시작 값을 정한다
- 딱 한 번만 실행된다
- 반복될 때 다시 실행되지 않는다
- 여러 문장은 콤마로 나눈다

조건식

초기문

;

조건식

;

반복후작업



참(true)

작업문 실행 후 다시 조건식을 본다



거짓(false)

false면 반복문을 빠져나간다

반복 후 작업

초기문



실행 위치

작업문이 끝난
뒤에 실행된다

조건식



바뀌는 값

반복 변수를
여기서 바꾼다

반복 후 작업



흔한 형태

`i++`, `i--`,
`i+=2` 가 흔하다

작업문과 중괄호



중괄호 생략

```
for (int i = 0; i < 3; i++)  
    System.out.println("Hi");  
    System.out.println("Bye");
```

둘째 줄은 반복 밖에서 한 번만 실행된다



중괄호 사용

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

두 줄 모두 반복 안에서 실행된다

문장이 하나면 중괄호를 생략할 수 있다. 쓰는 편이 안전하다.

0에서 9까지 출력해 보기

```
public class Ex01_ForFirst {  
    public static void main(String[] args) {  
  
        for (int i = 0; i < 10; i++) {  
            System.out.print(i);  
        }  
    }  
}
```

0에서 9까지 실행 결과

```
1 public class Ex01_ForFirst {  
2     public static void main(String[] args) {  
3         for (int i = 0; i < 10; i++) {  
4             System.out.print(i);  
5         }  
6     }  
7 }  
8
```

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex01_ForFirst.java

0123456789

실행이 끝났어요

숫자가 한 줄로 이어 붙어 나온다

어디까지 출력될까

```
for (int i = 0; i < 10; i++)  
    System.out.print(i);
```

```
for (int i = 0; i <= 10; i++)  
    System.out.print(i);
```

답: 경계가 한 번을 가른다

$i < 10$

10번

0 1 2 3 4 5 6 7 8 9

마지막 값 9

VS

$i \leq 10$

11번

0 1 2 3 4 5 6 7 8 9 10

마지막 값 10

부등호 한 글자가 반복 횟수와 마지막 값을 함께 바꾼다.

한 칸이 남는 자리

$i < 10$

0~9 → 10번

$i \leq 10$

0~10 → 11번



그 한 칸

나중엔 고치기 어렵다

거꾸로 세기

i-- 만 바꾸면

```
for (int i = 0; i < 10; i--)  
{  
    System.out.print(i);  
}
```



세 자리를 모두 바꾸면

```
for (int i = 9; i >= 0; i--)  
{  
    System.out.print(i);  
}
```

실행 결과: 9876543210

두 칸씩 건너뛰기

```
for (int i = 0; i < 10; i += 2) {  
    System.out.println(i);  
}
```

$i += 2$ 는 $i = i + 2$ 와 같다

시작값을 1로 두면 홀수만 나온다

실행 결과

0 2 4 6 8

짝수 다섯 개 출력

더해 두는 변수

반복 밖에서 선언

```
int sum = 0;
for (int i=1;i<=10;i++) {
    sum += i;
}
```

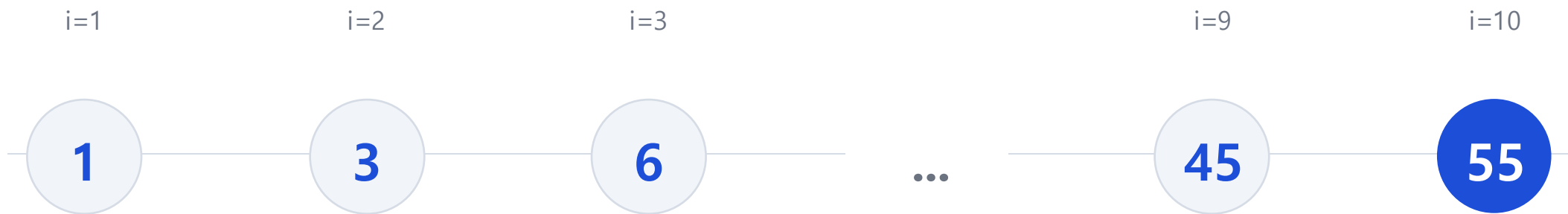
반복 전에 한 번만 만든다

반복 안에서 선언

```
for (int i=1;i<=10;i++) {
    int sum = 0;
    sum += i;
}
```

매번 새로 만들어져 0이 된다

sum 이 쌓이는 과정



$\text{sum} = \text{sum} + i$

예제 3-1 ForSample

```
public class Ex02_ForSample {  
    public static void main(String[] args) {  
        int i, sum = 0;  
  
        for (i = 1; i <= 10; i++) {
```



```
            sum += i;  
            System.out.print(i);  
            if (i <= 9)  
                System.out.print(" + ");  
            else {  
                System.out.print(" = ");  
                System.out.print(sum);  
            }  
        }  
    }  
}
```

예제 3-1 실행 결과

터미널

실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex02_ForSample.java

1+2+3+4+5+6+7+8+9+10=55

실행이 끝났어요

더하는 식과 합이 한 줄로 나온다

왜 `if(i<=9)` 인가

`if(i<=9)`

```
System.out.print(i+" ");
```

`else`

```
System.out.print(i+"="+sum);
```

i 가 1~9 이면 참, 10 이면 거짓

if 조건문은 지난주에 배운 문법 그대로다.

`i = 1 ~ 9`

뒤에 + 를 붙인다



`i = 10`

뒤에 = 와 합계를 붙인다

같은 묶음을 다시 돌린다

같은 묶음을 값만 바꿔 다시 돌린다

되풀이라는 생각은 자바보다 먼저 있었다.

for

while

do-while

오늘 이 셋을 순서대로 다 본다.

반복 후 작업에 두 문장

for 문의 반복 후 작업

```
for (int i = 0; i < 10;  
    i++, sum += i) {  
    ...  
}
```

콤마로 잇는다

- 콤마로 문장을 잇는다
- 왼쪽부터 차례로 실행된다
 - i++ 먼저, sum+=i 나중
- 초기문도 같은 방식을 쓴다

초기문에서 변수 만들기

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

i의 생존 구간: for 블록 안

유효 범위

- int i를 for 안에서 선언한다
- for를 벗어나면 i는 사라진다
- 밖에서 쓰려면 밖에 선언한다

for 밖에서 i 를 쓰면

 코드

```
for (int i = 0; i < 10; i++) {  
    // 반복 작업  
}
```

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



컴파일 오류

Main.java:5: error:
cannot find symbol
System.out.println(i);
symbol: variable i
location: class Main

일부러 무한 반복



조건식 자리에 true

```
for(int i=0; true; i++)
```



조건식을 통째로 비움

```
for(;;)
```

끝내려면 안에서 break 를 쓴다

실수로 만드는 무한 반복

실행해 보면

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

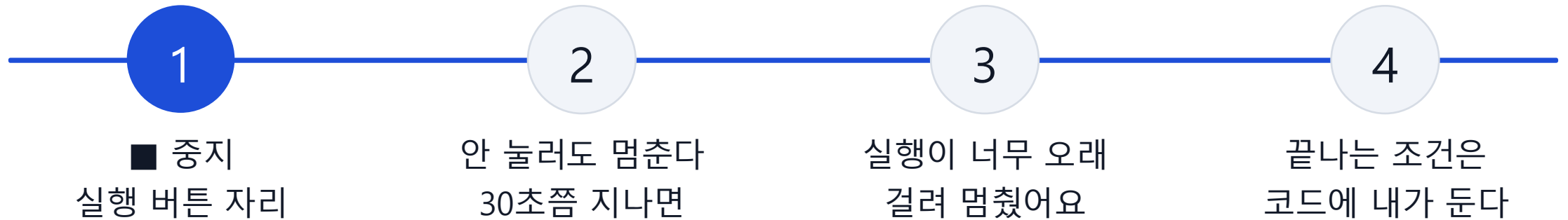
↑ 반복 후 작업이 비어 있다

무한 반복이 되는 이유

- 조건에 쓰는 값이 변하지 않는다
- 문법 오류가 아니라 컴파일된다
- 출력이 끝없이 이어진다

멈추는 법

실행을 중단하고 조건식부터 다시 본다



무한 반복은 3교시 break 로 제대로 다룬다

홀수만 더하기

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



체크포인트

sum은 반복 밖에서 선언한다
첫 번째 줄 · $i+=2$
두 번째 줄 · $\text{if}(i\%2==1)$
합은 반복 후에 출력한다

홀수만 더하기 결과

```
1 public class Ex03_OddSum {  
2     public static void main(String[] args) {  
3         int sum = 0;  
4  
5         for (int i = 1; i <= 100; i += 2) {  
6             sum += i;  
7         }  
8         System.out.println("홀수의 합은 " + sum);  
9     }  
10 }  
11
```

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex03_OddSum.java

홀수의 합은 2500

실행이 끝났어요

홀수의 합은 2500

내 학번 끝자리까지 더하기

학번 끝자리가 0이면 10으로 한다

```
...
public class Ex04_MySum {
    public static void main(String[] args) {
        int last = 7; // 내 학번 끝자리
        int sum = 0;
        for (int i = 1; i <= last; i++) {
            sum += i;
        }
        System.out.println("1부터 " + last + "까지의 합은 " + sum);
    }
}
```

내 학번까지 더하기 결과

```
1 public class Ex04_MySum {  
2     public static void main(String[] args) {  
3         int last = 7; // 내 학번 끝자리  
4         int sum = 0;  
5         for (int i = 1; i <= last; i++) {  
6             sum += i;  
7         }  
8         System.out.println("1부터 " + last + "까지의 합은 " + sum);  
9     }  
10 }  
11
```

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex04_MySum.java

1부터 7까지의 합은 28

실행이 끝났어요

1부터 7까지의 합은 28

for 문에서 오늘 본 것

for 는 네 자리를 채우면 나머지를 컴퓨터가 센다.



초기문

초기문은 한 번
만 실행된다



조건식

조건식은 매 바
퀴 시작에서 분
다



반복 후 작업

반복 후 작업은
작업문 뒤에 실행
된다



작업문

작업문은 중괄
호 안에 둔다

1교시 실습 살펴보기



실습 ①

0부터 세서
10번 반복



실습 ②

$i+=2$ 와 if
두 갈래 방식



실습 ③

학번만 바뀌도
코드 틀 그대로

세 실습에서 값만 바뀌고 반복의 틀은 같았다.

2교시

while 과 do-while

몇 번 돌지 모를 때 쓰는 반복문.

조건으로 끝내기

한 번은 반드시

고르는 기준

while 문의 구조

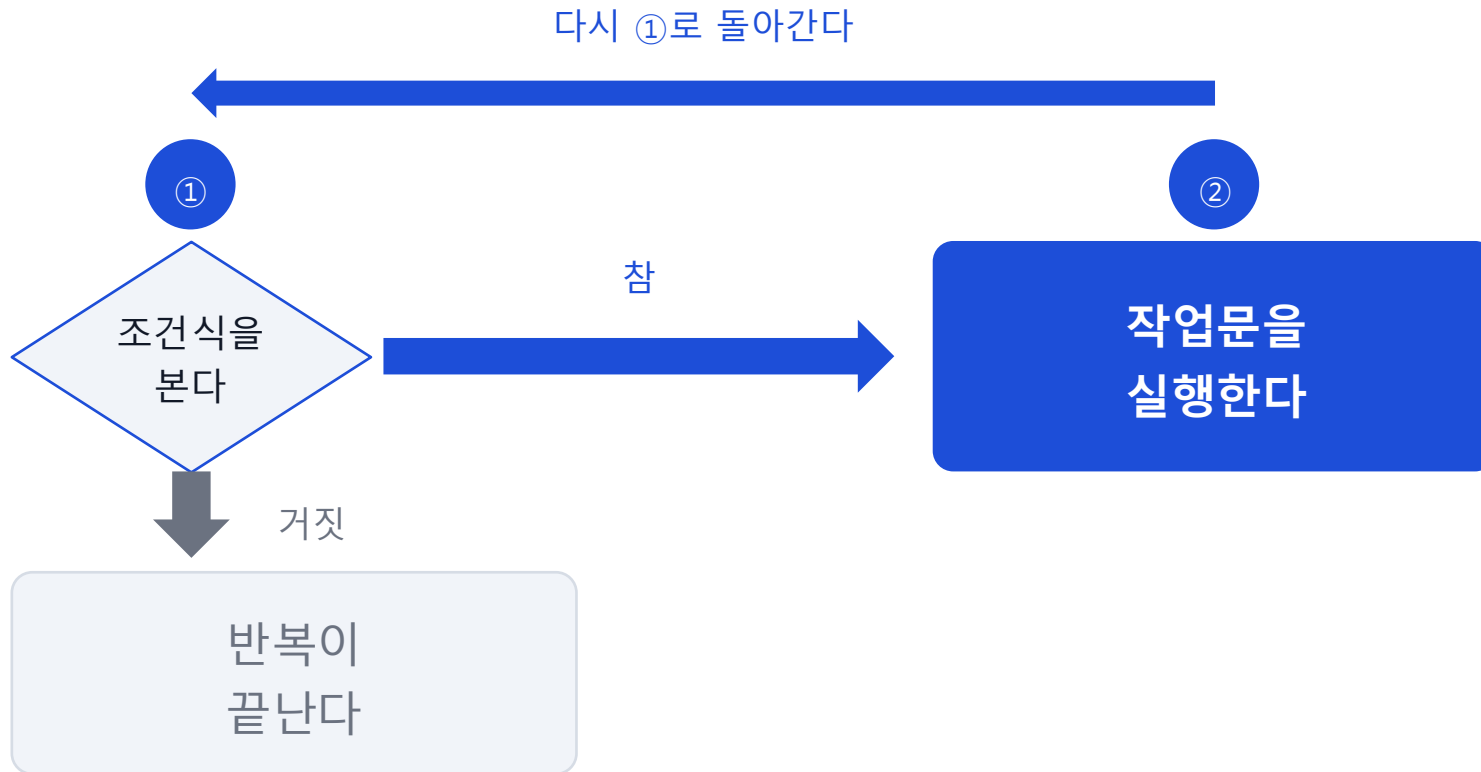
구조

```
while (조건식) {  
    작업문  
}
```

핵심

- 조건식은 하나만 온다
- 초기문·반복 후 작업 자리가 없다
- 중괄호 안이 되풀이된다

while 이 도는 순서



같은 일을 두 문법으로

for 문

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

while 문

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

출력은 0123456789 로 같다

생김새만 다르다

세 자리가 흠어진다



조건식을 빼면

...

```
while ( ) {
```

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

```
    i++;
```

```
}
```

...

for 와 while 의 차이

- for(;;)는 비워도 된다
- while()는 오류가 난다
- 무한 반복은 while(true)

조건식 없는 while 의 오류

```
1 public class Ex05_WhileError {  
2     public static void main(String[] args) {  
3         int i = 0;  
4         while ( ) {  
5             System.out.print(i);  
6             i++;  
7         }  
8     }  
9 }  
10
```

터미널 실행 결과

▶ 컴파일 중...

Ex05_WhileError.java:4: error: illegal start of expression

```
    while ( ) {  
           ^
```

1 error

컴파일에서 막혀 실행하지 못했어요. 위의 오류를 먼저 고쳐 주세요.

조건식이 비면 실행 전에 멈춘다

0에서 9까지 while 로

```
public class Ex06_WhileFirst {  
    public static void main(String[] args) {
```

```
        int i = 0;  
        while (i < 10) {  
            System.out.print(i);  
            i++;  
        }  
    }  
}
```

while 로 0~9 결과

```
1 public class Ex06_WhileFirst {  
2     public static void main(String[] args) {  
3         int i = 0;  
4         while (i < 10) {  
5             System.out.print(i);  
6             i++;  
7         }  
8     }  
9 }  
10
```

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex06_WhileFirst.java

0123456789

실행이 끝났어요

for 로 만든 것과 같은 출력

카운터를 안 늘리면

```
1 int i = 0;
2 while (i < 10) {
3     System.out.print(i);
4 }
```

몇 번 돌지 모를 때



이름 찾기

이름이 나올 때까지
목록을 본다



값 입력받기

0이 들어올 때까지
입력을 받는다



직접 확인

몇 번 돌지는
실행해봐야 안다

끝나는 조건을 먼저 정한다

지난주 Scanner 다시

```
Scanner scanner = new Scanner(System.in);  
int n = scanner.nextInt();
```

- scanner.nextInt() 로 정수를 읽는다
- import java.util.Scanner; 를 첫 줄에 쓴다
- 입력은 실행 결과 탭 입력 칸에 한 줄씩

예제 3-2 WhileSample

```
import java.util.Scanner;

public class Ex07_WhileSample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int count = 0, n = 0;
        double sum = 0;

        System.out.println("정수를 입력하고 마지막에 0을 입력하세요.");
        while ((n = scanner.nextInt()) != 0) {
            sum = sum + n;
            count++;
        }
        System.out.print("수의 개수는 " + count + "개이며 ");
        System.out.println("평균은 " + sum / count + "입니다.");
        scanner.close();
    }
}
```

예제 3-2 실행 결과

터미널

실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex07_WhileSample.java

정수를 입력하고 마지막에 0을 입력하세요.

10

30

-20

40

0

수의 개수는 4개이며 평균은 15.0입니다.

실행이 끝났어요

0 을 넣자 반복이 끝나고 개수와 평균이 나온다

한 줄에 두 가지 일

```
while((n = scanner.nextInt()) != 0)
```

1

`scanner.nextInt()`

괄호 안에서
입력값을 읽는다

2

`n = ...`

n 에 대입한다
대입도 값이 된다

3

`!= 0`

그 값이
0 인지 비교한다

끝내는 값을 정한다

처음 생각

0 을 넣으면 반복을 끝낸다
약속한 값은 계산에서 뺀다

그런데

0 도 실제 점수일 수 있다
음수도 다루면 0 도 못 쓴다

끝내는 값은 데이터에 나올 수 없는 값으로 고른다.

끝내는 값 바꿔 보기

예제 3-2에서 조건식만 `!= -1` 로 바꾼다

```
...
public class Ex08_SentinelSum {
    public static void main(String[] args) {
        ...
        while ((n = scanner.nextInt()) != -1) {
            sum += n;
        }
        System.out.println("합은 " + sum);
        scanner.close();
    }
}
```

확인할 입력값

-20 입력
개수 포함 여부 확인

-1 입력
반복 종료 확인

끝내는 값 바꾼 결과

터미널

실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex08_SentinelSum.java

정수를 입력하고 마지막에 -1을 입력하세요.

10

0

-30

40

-1

합은 20

실행이 끝났어요

0 은 더해지고 -1 에서 끝난다

가지 말고 돌아라

// Go To 문은 해롭다고 여겨진다

에츠퍼드 다익스트라가 1968년 학술지에 보낸 짧은 편지의 제목이다.

당시엔 goto 로 원하는 줄로 건너뛰었다. 사람이 흐름을 따라갈 수 없다는 지적이었다.

에츠퍼드 다익스트라 · 1968



조건식 한 곳에서만

조건식의 위치

```
while (count <= 9) {  
    System.out.println(count);  
    count++;  
}
```

조건식은 이 자리 하나뿐이다

좋은 점

- 조건은 한 곳에 있다
- 시작과 끝이 중괄호로 보인다
- 끝나는 자리를 안 헤맨다
- 라벨 분기도 이유가 같다

do-while 문의 구조

```
do {
```

```
    작업문
```

```
} while(조건식);
```

do 뒤에 중괄호가 먼저 온다

반복해서 실행할 문장

조건식은 while() 괄호 안에 있다

끝에 세미콜론을 붙인다

do-while 이 도는 순서



참이면

다시 ①로 돌아간다

거짓이면

반복이 끝난다

한 번은 반드시 실행된다

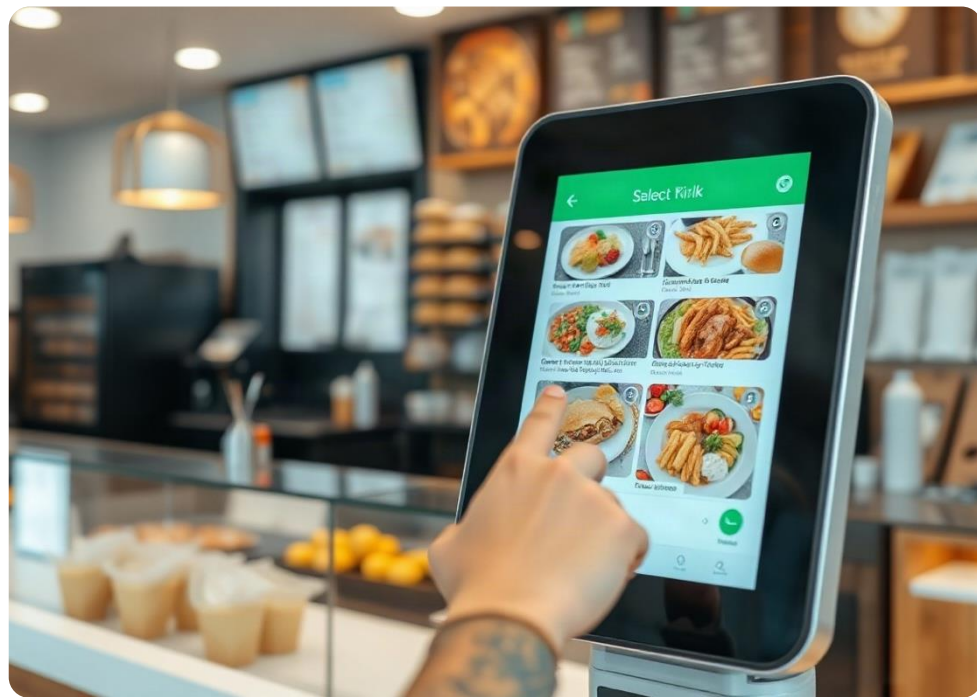
작업문 실행



조건 검사

조건과 상관없이 먼저 실행된다

- while과 다른 점은 이 실행 순서뿐이다
- 메뉴를 보여주고 고르게 할 때 쓴다



조건이 처음부터 거짓이면

for

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

0번

while

```
i = 10; while (i < 10) { }
```

0번

do-while

```
i = 10;
```

```
do { ... }
```

```
while (i < 10);
```

실행 횟수

1번

차이는 조건을 보는 때에서 온다

몇 번 출력될까

```
int i=10;  
do {  
    System.out.print(i);  
    i++;  
} while(i<10);
```

답: 한 번은 나온다

몸통이 먼저 돈다

1. `i=10` 대입
2. 출력문 실행 → 10 출력
3. `i++` 후 조건 확인 → 거짓

출력: 10 하나

while 로 바꾸면

1. `i=10` 대입
2. 조건 먼저 확인 → 거짓
3. 몸통 실행 안 됨

출력: 없음

예제 3-3 DoWhileSample

Ex09_DoWhileSample.java

```
public class Ex09_DoWhileSample {  
    public static void main(String[] args) {  
        char a = 'a';  
  
        do {  
            System.out.print(a);  
            a = (char) (a + 1);  
        } while (a <= 'z');  
    }  
}
```

예제 3-3 실행 결과

```
1 public class Ex09_DoWhileSample {  
2     public static void main(String[] args) {  
3         char a = 'a';  
4  
5         do {  
6             System.out.print(a);  
7             a = (char) (a + 1);  
8         } while (a <= 'z');  
9     }  
10 }  
11
```

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex09_DoWhileSample.java

abcdefghijklmnopqrstuvwxyz

실행이 끝났어요

a 부터 z 까지 한 줄로 나온다

문자에 1 을 더하면

예제 3-3 핵심 한 줄

```
char a = 'a';
```

```
a = (char)(a + 1);
```

코드 값에 1 을 더하면
다음 문자가 된다

이렇게 되는 이유

- char 는 문자 코드 값을 담는다
- a+1 은 int 가 되어 형변환이 필요하다
- (char) 를 빼면 컴파일 오류

세 반복문 비교표

구분	for	while	do-while
조건을 언제 보나	반복 전	반복 전	반복 후
최소 몇 번 도나	0번	0번	1번
조건식을 빼도 되나	된다 (무한 반복)	안 된다 (컴파일 오류)	안 된다 (컴파일 오류)

무엇을 쓸지 고르는 기준

횟수를 안다

반복 횟수나 범위가 명확하면 for 문.

횟수를 모른다

횟수를 처음부터 알 수 없고 반복하면서 조건이 달라지면 while 문이나 do-while 문.

세 상황에 적용해 보기

1부터 100까지
더하기

for

횟수가 정해져 있다

0이 나올 때까지
입력받기

while

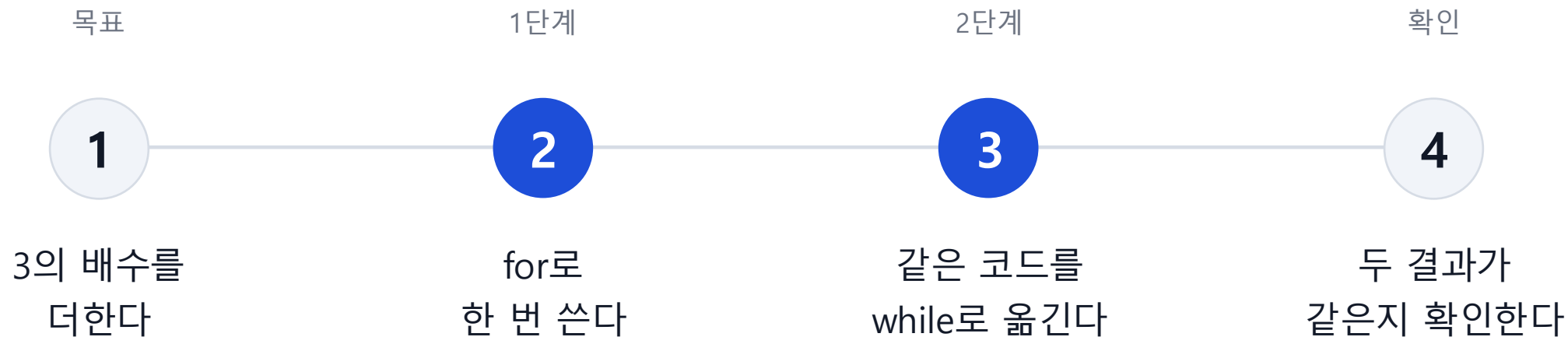
횟수를 모른다

메뉴를 보여주고
고르게 하기

do-while

한 번은 보여준다

같은 과제를 두 문법으로



for 로 쓴 코드

```
public class Ex10_MultipleOf3For {  
    public static void main(String[] args) {  
        int sum = 0;  
        for (int i = 3; i <= 50; i += 3) {  
            sum += i;  
        }  
        System.out.println("3의 배수의 합은 " + sum);  
    }  
}
```

while 로 쓴 코드

```
...
public class Ex11_MultipleOf3While {
    public static void main(String[] args) {
        int sum = 0;
        int i = 3;
        while (i <= 50) {
            sum += i;
            i += 3;
        }
        System.out.println("3의 배수의 합은 " + sum);
    }
}
```

두 코드의 실행 결과

```
1 public class Ex11_MultipleOf3While {  
2     public static void main(String[] args) {  
3         int sum = 0;  
4         int i = 3;  
5         while (i <= 50) {  
6             sum += i;  
7             i += 3;  
8         }  
9         System.out.println("3의 배수의 합은 " + sum);  
10    }  
11 }  
12
```

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex11_MultipleOf3While.java

3의 배수의 합은 408

실행이 끝났어요

3의 배수의 합은 408

어느 쪽이 읽기 쉬운가

for 문

- 세 자리가 한 줄에 모여 있다
- i 를 빠뜨릴 자리가 적다

vs

while 문

- 세 자리가 위아래로 흩어진다
- 조건이 복잡할 때 읽기 쉽다

무엇을 쓸지는 상황에 따라 갈린다.

while 에서 자주 나는 실수

1

```
while(i<10) {  
  print(i);  
}
```

2

```
while(i = 10) {  
  
}
```

3

```
while(i<10);  
{  
  print(i);  
  i++;  
}
```

전 세계가 같은 날 멈췄다

2008년 12월 31일 · Zune 30

days 가 366일 때 바깥 조건은 참이고 안쪽 조건은 거짓이다.
그래서 아무것도 바꾸지 않고 다시 같은 조건을 본다.

```
while (days > 365) {  
  if (윤년이면) {  
    if (days > 366) { days -= 366; year += 1; }  
  } else {  
    days -= 365; year += 1;  
  }  
}
```

조건은 참인데 아무것도 안 바뀐다

조건에 쓰이는 값을 몸통이 바꿔야 반복이 끝난다



카운터를 안 늘림

증감문 자체가 없다



다른 값만 바꿈

조건 변수는 그대로



조건 자체가 상수

바뀌어도 항상 참

조건식의 경계 다시

```
int i = 1;  
while (i < 10) {
```

9번 실행

+1

```
int i = 1;  
while (i <= 10) {
```

10번 실행

시작 값이 1이면 $i \leq n$ 이 n 번
시작 값이 0이면 $i < n$ 이 n 번

2교시 실습 살펴보기

실습 ④



카운터 증가

증가식을 몸통에 직접
적는다.

실습 ⑤



종료값 주의

끝내는 값이 데이터와
겹치지 않게 한다.

실습 ⑥



세 자리 순서

옮길 때 세 자리를 하
나씩 확인한다.

while 은 반복 변수를 사람이 챙겨야 한다.

3교시

중첩 반복과 흐름 바꾸기

반복 안의 반복, 그리고 건너뛰기와 빠져나오기.

구구단과 별 찍기

continue

break

반복문 안의 반복문

- 작업문 자리에 반복문을 넣는다
- 세 반복문 모두 넣을 수 있다
- while 안에도 셋 다 들어간다
- 이것을 중첩 반복이라 한다

바깥 반복문



안쪽 반복문

작업문 자리에 놓인다

100개 대학, 10000명 학생

대학 100개

바깥은 1바퀴만 돈다

학생 10000명

안쪽은 10000바퀴 돈다

**바깥이 한 번 돌 때
안쪽이 끝까지 돈다**

- 대학 하나에서 학생 10000명을 다 훑는다
- 다 훑어야 다음 대학으로 넘어간다

안쪽은 몇 번 도나

바깥 반복

100

×

안쪽 반복 (1회당)

10,000

=

전체 실행 횟수

1,000,000

안쪽 작업문은 두 횟수를 곱한 만큼, 백만 번 실행된다.

몇 겹까지 쓰나



2~3중

적당한 중첩 깊이



- 문법상 겹 수 제한은 없다
- 겹칠수록 실행 횟수가 는다
- 겹칠수록 읽기도 어려워진다

별로 직사각형 그리기

```
public class Ex12_StarRectangle {  
    public static void main(String[] args) {  
        for (int i = 0; i < 5; i++) {  
            for (int j = 0; j < 5; j++) {  
                System.out.print("*");  
            }  
            System.out.println();  
        }  
    }  
}
```

직사각형 실행 결과

터미널

실행 결과

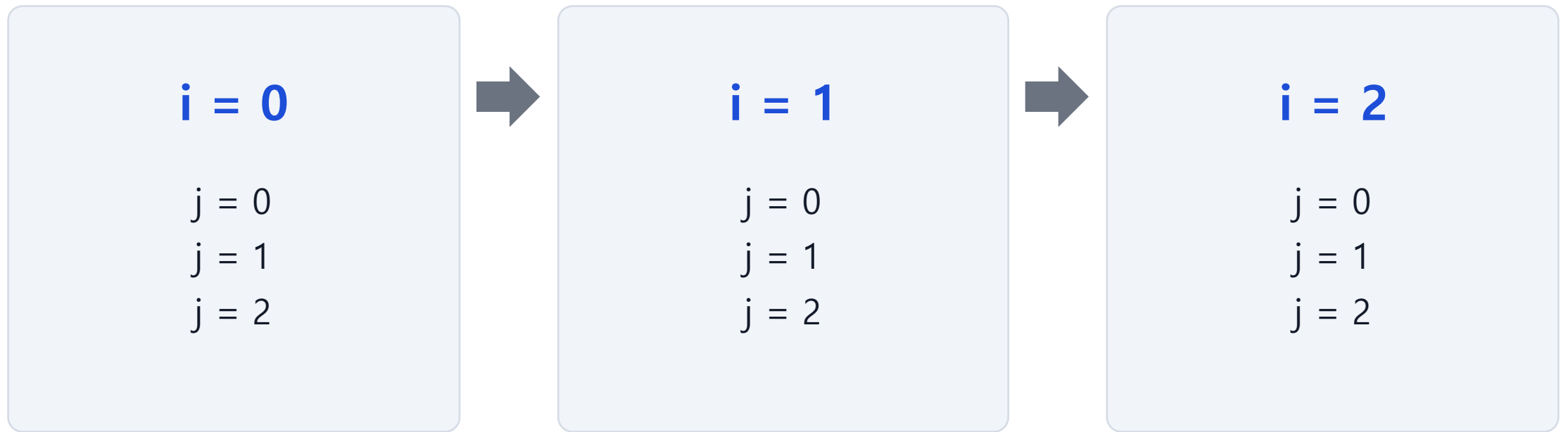
▶ 컴파일 중...

실행: 주차별코드/3주차/Ex12_StarRectangle.java

실행이 끝났어요

별이 직사각형으로 찍힌다

i 와 j 가 함께 변한다



작업문은 모두 9번 실행된다.

한 줄씩 늘려 삼각형

```
...  
    for (int i = 0; i < 5; i++) {  
        for (int j = 0; j <= i; j++) {  
            System.out.print("*");  
        }  
        System.out.println();  
    }  
...
```

안쪽 조건을 바깥에 맞춘다

- 안쪽 조건을 $j \leq i$ 로 바꾼다
- 줄마다 별 개수가 달라진다

삼각형 실행 결과

터미널

실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex13_StarTriangle.java

*

**

실행이 끝났어요

한 줄에 하나씩 늘어 삼각형이 된다

예제 3-4 NestedLoop

```
public class Ex14_NestedLoop {  
    public static void main(String[] args) {  
        for (int i = 1; i < 10; i++) {  
            for (int j = 1; j < 10; j++) {  
                System.out.print(i + "*" + j + "=" + i * j);  
                System.out.print('\t');  
            }  
            System.out.println();  
        }  
    }  
}
```

구구단 실행 결과

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex14_NestedLoop.java

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

실행이 끝났어요

9단까지 차례로, 한 단이 한 줄

칸을 맞추는 \wt

정렬 대조

스페이스 여러 칸

사과 1200

바나나 500

시작 위치 다름

탭 한 번

사과 1200

바나나 500

시작 위치 같음

예시 코드

```
System.out.println("이름\wt점수");  
System.out.println("김철수\wt95");  
System.out.println("박영희\wt100");
```

줄은 누가 바꾸나

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

→ 그 줄이 없으면

한 줄로 이어진다

↩ 안에 넣으면

한 칸마다 줄이 바뀐다

무엇을 건너뛰고 무엇에서 빠져나올까

무엇을 안 할지 정하는 것도 프로그램의 일이다.



생략의 기준

다 처리하지
않아도 된다



continue

반복 안에서
건너뛴다



break

반복에서
빠져나온다

흐름을 바꾸는 두 문장



continue

continue 는 다음 반복으로 간다.



break

break 는 반복문을 벗어난다.

둘 다 반복 안에서만 쓴다 · 대개 if 와 함께 쓴다

continue

반복문을 빠져나가지 않고 다음 반복으로 간다.



건너뛴다

이번 바퀴의 남은 문장만 넘어간다.



계속된다

정해진 횟수만큼 반복은 계속된다.

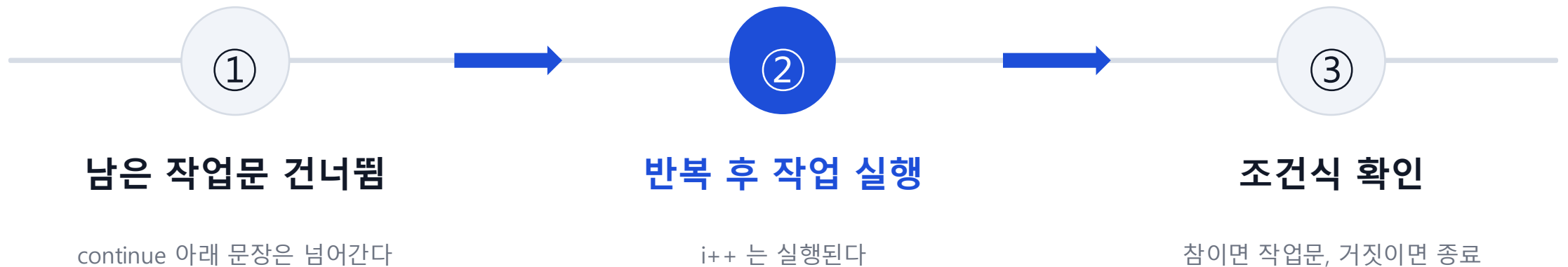


이럴 때

특정 값만 빼고 처리하고 싶을 때 쓴다.

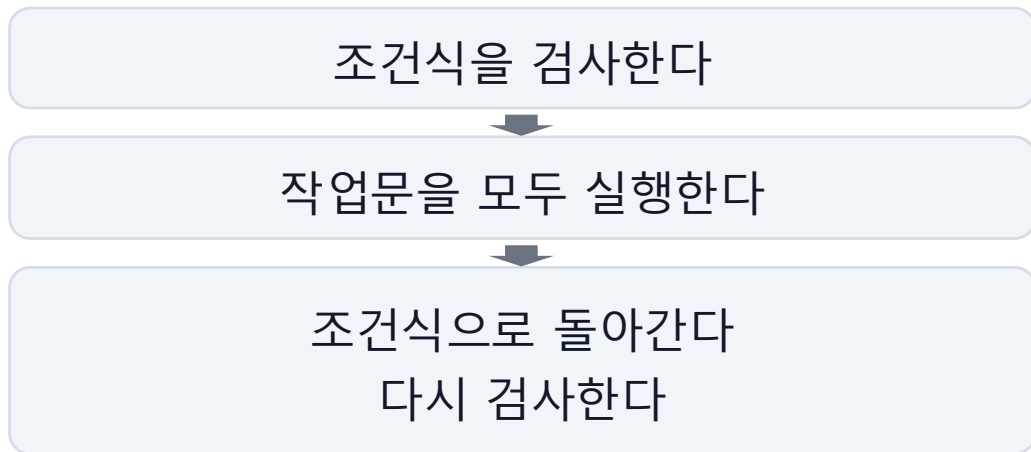
for 에서 continue 는 어디로

작업문 안에서 continue 를 만나면

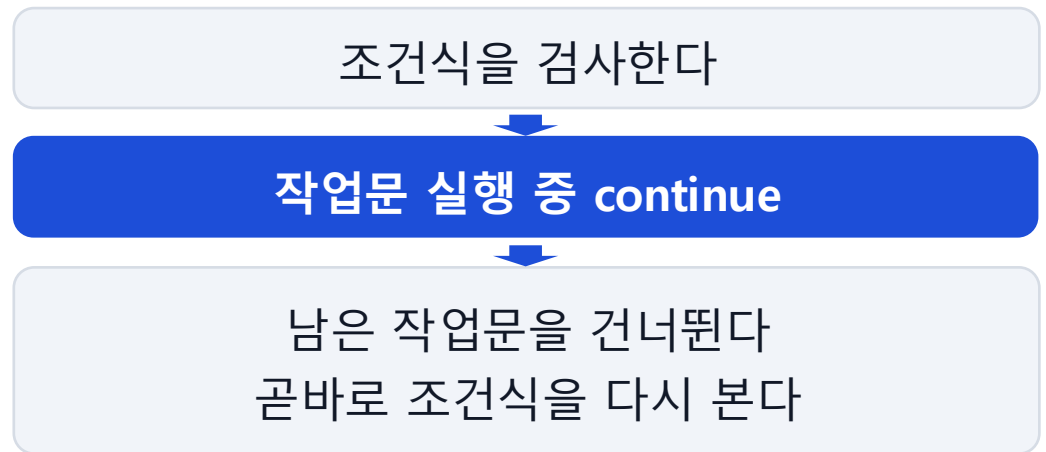


while 에서 continue 는 어디로

continue 없이



continue 만나면



i++ 도 작업문이라 함께 건너뛰는다.
값이 그대로인 채 조건식만 다시 검사한다.

예제 3-5 ContinueExample

```
import java.util.Scanner;

public class Ex15_ContinueExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("정수를 5개 입력하세요.");
```

```
        int sum = 0;
        for (int i = 0; i < 5; i++) {
            int n = scanner.nextInt();
            if (n <= 0) continue;
            else sum += n;
        }
        System.out.println("양수의 합은 " + sum);

        scanner.close();
    }
}
```

예제 3-5 실행 결과

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex15_ContinueExample.java

정수를 5개 입력하세요.

5

-2

6

10

-4

양수의 합은 21

실행이 끝났어요

음수는 건너뛰고 양수만 더해 21

건너뛰다는 것

continue 로 건너뛰기

```
if (num < 0)
    continue;
sum += num;
```

=

if 로 걸러도 같다

```
if (num >= 0) {
    sum += num;
}
```

5

더함

3

더함

-2

건너뛴

7

더함

4

더함

while 에서 continue 를 쓸 때

```
int i = 0;
while (i < 10) {
    if (i % 2 == 0)
        continue;
    i++;
}
```

break



반복문을
즉시 벗어난다

찾는 것을 찾았을 때 쓴다

버려지는 것들

- 남은 작업문도 남은 반복도 버린다
- 조건식을 다시 보지 않는다
- 반복문 다음 문장으로 간다

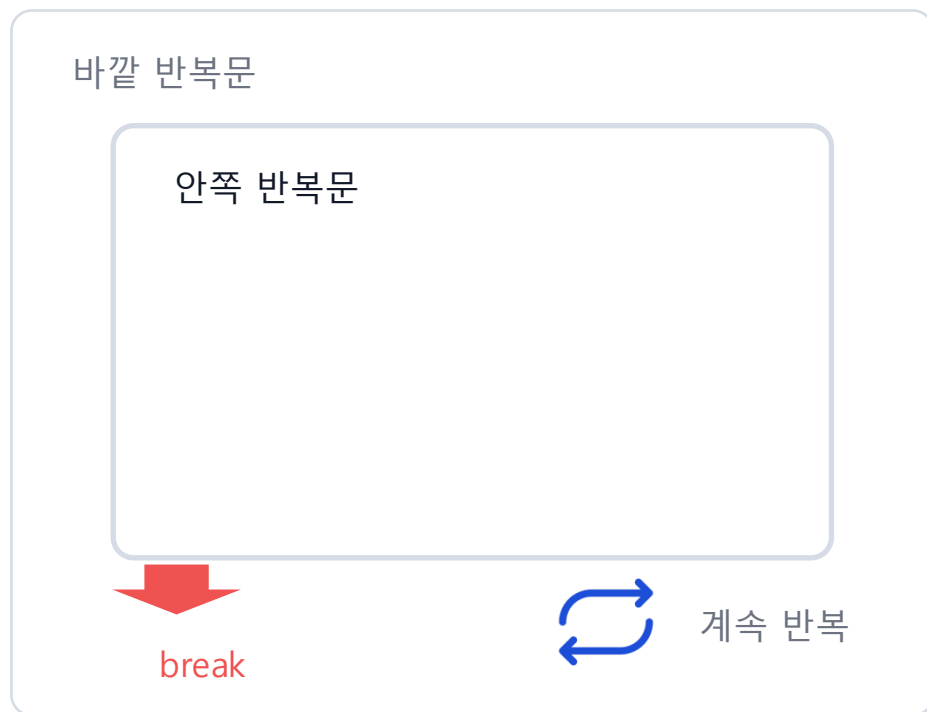
반복문 하나를 벗어난다

```
while (조건) {  
    break;  
}
```

다음 문장부터 실행

- 1 break 를 만난다
- 2 그 반복문을 빠져나온다
- 3 조건식은 다시 보지 않는다

중첩에서는 안쪽만



[→] 안쪽 반복 종료

↺↻ 바깥 반복 유지

어디까지 출력될까

```
for (int i = 0; i < 3; i++) {  
    for (int j = 0; j < 3; j++) {  
        if (j == 1) break;  
        System.out.println(i + ", " + j);  
    }  
}
```

답: 바깥은 계속 돈다

$i = 0$

$j=0$ 실행 → 출력 → break

출력 1개

$i = 1$

$j=0$ 실행 → 출력 → break

출력 1개

$i = 2$

$j=0$ 실행 → 출력 → break

출력 1개

세 번 돌아 출력은 총 3개

건너뛰기와 빠져나오기

1부터 10까지 도는 중, i가 5일 때

 **continue**

그 값만 건너뛰고 계속 돈다

```
if (i == 5) continue;
```

출력: 1 2 3 4 6 7 8 9 10

 **break**

그 값에서 멈추고 빠져나온다

```
if (i == 5) break;
```

출력: 1 2 3 4

라벨을 붙여 빠져나오기

</> 문법

```
OUTER:  
for (...) {  
    for (...) {  
        break OUTER;  
    }  
}
```

⚠ 권장하지 않는 이유

- 정해진 라벨로 한 번에 나간다
- 구조적 프로그래밍에 반한다
- 부작용이 있어 잘 쓰지 않는다

문제: 이 코드의 목적과 결과

```
int n=0;
while(true) {
    n++;
    if(n>=20) break;
    if(n%3 == 0) continue;
    System.out.print(n+" ");
}
```

예측해 보기

이 코드의 목적은 무엇인가

실행 결과는 무엇인가

문제 풀이

1

n 1 증가

2

20 이상
break

3

3배수
continue

4

나머지
출력

결과

1 2 4 5 7 8 10 11 13 14 16 17 19

break 판단이 continue 보다 먼저라 20은 출력되지 않는다

끝내는 문장을 어디에 두나



종료의 어려움

도는 조건보다 끝내는 조건이 더 어렵다.



위치의 차이

break 를 두는 위치가 결과를 가른다.



다음 실습

가위바위보를 여러 판 돌리며 직접 정한다.

간단한 가위바위보 예시

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        int sum = 0;

        System.out.print("가위바위보 중 하나를 내세요. 가위 = 1, 바위 = 2, 보 = 0 : ");

        Scanner sc = new Scanner(System.in);
        int my = sc.nextInt();
        int com = (int)(Math.random() * 3);
        String 결과 = "모름";

        if (my == com) 결과 = "비김";
        else
            if (my == 0 ) 결과 = "이김";
        else
            if (my == 1 ) 결과 = "짐";
        System.out.println(결과);

    }
}
```

한 판을 하고 나면 프로그램이 끝난다.

여러 판으로 만들려면



반복 안

매 판마다 실행

- 가위바위보 입력받기
- 컴퓨터 수 뽑고 승패 판정



반복 밖

처음 한 번만 실행

- 이긴 횟수 변수 선언
- 마지막 결과 출력

가위바위보 여러 판

```
public class Ex16_RockPaperScissors {  
    public static void main(String[] args) {  
        ...  
        Scanner scanner = new Scanner(System.in);  
        int win = 0, play = 0;  
        while (true) {  
            System.out.print("가위 바위 보 (그만하려면 q)  
>> ");  
            String me = scanner.next();
```

```
            if (me.equals("q")) break;  
            play++;  
            if (me.equals("가위")) win++;  
            System.out.println(play + "판째, 이긴 횟수 " +  
win);  
        }  
        System.out.println("총 " + play + "판 중 " + win + "  
판 승");  
        scanner.close();  
    }  
}
```

가위바위보 여러 판 결과

터미널 실행 결과

▶ 컴파일 중...

실행: 주차별코드/3주차/Ex16_RockPaperScissors.java

가위 바위 보 (그만하려면 q) >> 가위

1판째, 이긴 횟수 1

가위 바위 보 (그만하려면 q) >> 바위

2판째, 이긴 횟수 1

가위 바위 보 (그만하려면 q) >> 가위

3판째, 이긴 횟수 2

가위 바위 보 (그만하려면 q) >> q

총 3판 중 2판 승

실행이 끝났어요

q 를 넣을 때까지 반복하고 마지막에 집계

그만두는 방법

순서

1. 끝낼 입력을 하나 정한다
2. if 로 그 값인지 본다
3. 맞으면 break 로 반복 끝낸다

2교시 while 끝내는 값과 같은 생각

예시 코드

```
int hand = sc.nextInt();  
if (hand == 0) {  
    break;  
}
```

0 은 가위바위보 값과 겹치지 않는다

몇 판 이겼나

```
int win = 0, lose = 0;
while (true) {
    ...
    if (이겼다) {
        win++;
    } else if (졌다) {
        lose++;
    }
}
System.out.println(win + "승 " + lose + "패");
```

이긴 판마다 변수를 하나씩 늘린다

- win 을 반복 밖에서 선언
- 이기면 win++;
- 반복 후 win 출력
- lose 도 같은 방식

반복문을 쓸 때 볼 세 가지

01



끝나는 조건이 있나

무한 반복으로 이어진다

02



그 값이 반복 안에서
변하나

안 늘리면 여기서 걸린다

03



몇 번 도는지 말할 수
있나

경계를 놓치면 걸린다

오늘 나온 실수는 전부 이 셋 중 하나였다.

3교시 실습 살펴보기



⑦ 출력

println은 안쪽 for 밖에 놓는다.



⑧ 조건

안쪽 조건에서 바깥 변수를 쓴다.



⑨ 집계

집계 변수는 반복 밖에 둔다.

중첩과 흐름 바꾸기는 자리가 결과를 바꾼다.

반복문으로 바꿔 보기

미션 조건

- 지난주 가위바위보를 여러 판으로 걸쳐 제출
- 끝내는 방법과 승패 집계를 넣을 것
- 제출: 코드와 실행 화면

반복문을 쓸 때 볼 세 가지

끝나는 조건이 있나
그 조건에 쓰이는 값이 반복 안에서 변하나
몇 번 도는지 말할 수 있나

안 될 때 여기부터

틀림

- 카운터를 안 늘림
- 조건식에 = 를 하나만
- 집계 변수를 반복 안에 선언

맞음

- ✓ 반복 후 작업이나 작업문 끝에서 값을 바꾼다
- ✓ 비교는 == 두 개
- ✓ 쌓는 변수는 반복 밖에 만든다

멈추지 않으면 실행을 중단하고 조건식부터 본다

오늘 공부한 내용

반복문은 같은 묶음을 값만 바꿔 다시 돌리는 것

for 는 세 자리, **while** 은 조건 하나, **do-while** 은 한 번은 반드시

조건식의 경계가 반복 횟수를 가른다

중첩은 바깥 한 번에 안쪽이 끝까지

continue 는 건너뛰기, **break** 는 빠져나오기

다음 주엔 배열

오늘 만든 반복문으로 배열을 처음부터 끝까지 훑습니다



다음 주 예고

- 배열 선언과 생성
- for-each 문