3. 제어문

조건문 (`if`, `if-else`, `switch`)

✅ 1. `if` 문

🔷 기본 구조

```
if (조건식) {
    // 조건식이 true일 때 실행
}
```

🔷 예시

```
int score = 90;

if (score >= 90) {
    System.out.println("A 학점");
}
```

> 조건식에는 반드시 `boolean` **타입**의 결과가 나와야 해.
> 정수나 문자열을 조건으로 사용할 수는 없어! (C와 다름)

✅ 2. `if-else` 문

🔷 구조

```
if (조건식) {
    // 조건식이 true일 때
} else {
    // 조건식이 false일 때
}
```

🔷 예시

```
int score = 75;

if (score >= 80) {
    System.out.println("합격");
} else {
    System.out.println("불합격");
}
```

✅ 3. `if - else if - else` 문

> 여러 조건 중 하나를 선택할 때 사용

```
if (조건1) {
    // 조건1이 참일 때
} else if (조건2) {
    // 조건2가 참일 때
} else {
    // 위 모든 조건이 거짓일 때
}
```

🔷 예시

```
int score = 85;

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

✅ 4. 중첩 조건문 (Nested if)

> 조건문 안에 또 다른 조건문이 있는 구조

```
int score = 85;
boolean pass = true;

if (pass) {
    if (score >= 80) {
        System.out.println("우수 통과");
    } else {
        System.out.println("보통 통과");
    }
}
```

✅ 5. 삼항 조건 연산자 (`? :`)

> 조건문을 간단히 표현할 수 있는 연산자

```
String result = (score >= 80) ? "합격" : "불합격";
System.out.println(result);
```

✅ 6. `switch` 문

🔷 개요

- 하나의 **변수(또는 표현식)** 값을 기준으로 여러 `case` 중 하나를 선택
- `break`를 반드시 명시하지 않으면 **fall-through(다음 case로 넘어감)** 발생함
- Java 7부터는 `String`도 지원함
- Java 14 이상에서는 **Switch 표현식 (→ 화살표 문법)** 지원됨

🔷 기본 구조

```
switch (값) {
    case 값1:
        // 코드
        break;
    case 값2:
        // 코드
        break;
    default:
        // 어느 case에도 해당하지 않을 때
}
```

🔷 예시

```
int day = 3;
switch (day) {
    case 1:
        System.out.println("월요일");
        break;
    case 2:
        System.out.println("화요일");
        break;
    case 3:
        System.out.println("수요일");
        break;
    default:
        System.out.println("주말 또는 오류");
}
```

🔷 `String` 사용 예시 (Java 7+)

```
String lang = "Java";

switch (lang) {
    case "C":
        System.out.println("C언어");
        break;
    case "Java":
        System.out.println("자바");
        break;
    default:
        System.out.println("기타");
}
```

🔷 Java 14+ Switch 표현식 (Arrow `→` 문법)

```
String result = switch (day) {
    case 1 -> "월요일";
    case 2 -> "화요일";
    case 3 -> "수요일";
    default -> "기타";
};

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

> Java 14 이상, **표현식(값 반환)**으로 switch를 사용할 수 있음

✅ 7. 조건문 선택 기준

조건문 종류	사용 조건
`if`	조건이 **범위, 복잡한 비교**인 경우
`if-else if`	**계단식 분기**가 필요한 경우
`switch`	변수의 **값이 고정된 경우** (`int`, `char`, `String`, enum 등)

✅ 8. 예제: 학점 분기

```
int score = 82;

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

✅ 9. 예제: 월별 일수 (switch)

```
int month = 2;
int days;

switch (month) {
    case 1: case 3: case 5: case 7: case 8: case 10: case 12:
        days = 31;
        break;
    case 4: case 6: case 9: case 11:
        days = 30;
        break;
    case 2:
        days = 28;
        break;
    default:
        days = 0;
}

System.out.println("일수: " + days);
```

✅ 요약 비교

항목	`if-else`	`switch`
사용 대상	범위, 논리 판단	고정된 값 (정수, 문자열 등)
조건식	`boolean` 결과	`byte`, `short`, `int`, `char`, `enum`, `String`
실행 흐름	위에서 아래로	case에 일치하면 분기
성능	조건 많으면 느려질 수 있음	일반적으로 빠름

반복문 (`for`, `while`, `do-while`)

✅ 1. 반복문의 종류

반복문	특징
`for`	**반복 횟수가 명확**할 때 적합
`while`	**조건만으로 반복** 제어
`do-while`	최소 1회 **조건 없이 실행 후 반복 조건 판단**

✅ 2. `for` 문

🔷 기본 구조

```
for (초기식; 조건식; 증감식) {
    // 반복할 코드
}
```

> 초기식 → 조건 검사 → 코드 실행 → 증감식 → 조건 검사 ...

🔷 예시: 1부터 5까지 출력

```
for (int i = 1; i <= 5; i++) {
    System.out.println(i);
}
```

🔷 특징

- 반복 횟수가 정해져 있을 때 가장 많이 사용
- 변수 `i`, `j` 등의 **루프 변수(loop variable)** 사용

✅ 3. `while` 문

🔷 기본 구조

```
while (조건식) {
    // 반복할 코드
}
```

- 조건이 **참일 때만 반복**
- 조건이 **처음부터 false이면 한 번도 실행 안 됨**

🔷 예시: 1부터 5까지 출력

```
int i = 1;
while (i <= 5) {
    System.out.println(i);
    i++;
}
```

✅ 4. `do-while` 문

🔷 기본 구조

```
do {
    // 반복할 코드
} while (조건식);
```

- `while`과 유사하지만, **무조건 한 번 실행하고 조건 판단**
- **사용 빈도는 적지만**, 사용자 입력 처리에 자주 쓰임

🔷 예시: 사용자 입력 받기

```
Scanner sc = new Scanner(System.in);
String input;

do {
    System.out.print("입력(q 입력 시 종료): ");
    input = sc.nextLine();
} while (!input.equals("q"));
```

✅ 5. 반복문 중단 및 건너뛰기

🔹 `break`: 반복문 전체 종료

```
for (int i = 1; i <= 10; i++) {
    if (i == 5) break;
    System.out.println(i);  // 1 2 3 4
}
```

🔹 `continue`: 현재 반복 건너뛰고 다음 반복 진행

```
for (int i = 1; i <= 5; i++) {
    if (i == 3) continue;
    System.out.println(i);  // 1 2 4 5
}
```

✅ 6. 중첩 반복문 (Nested Loop)

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

- 2차원 배열, 구구단, 표 그리기 등에 자주 사용

✅ 7. 향상된 `for-each` 문 (배열/컬렉션 전용)

```
int[] nums = {10, 20, 30};
for (int n : nums) {
    System.out.println(n);
}
```

- **배열/컬렉션**에서 요소를 순서대로 탐색할 때 사용
- 인덱스 사용 ❌ → 수정 불가, 읽기만 가능

✅ 8. 무한 루프 만들기

```
while (true) {
    // 조건 없이 반복
    if (조건) break;
}
```

✅ 9. 예제: 구구단 (2단~9단)

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

✅ 10. 반복문 비교 요약

항목	`for`	`while`	`do-while`
반복 횟수 명확성	✅ 명확할 때	❌ 불명확	❌ 불명확
조건 검사 시점	반복 전	반복 전	**반복 후**

항목	`for`	`while`	`do-while`
최소 실행 횟수	0회 가능	0회 가능	**최소 1회**
사용 예	횟수 기반 반복	조건 기반 루프	사용자 입력 유효성 검사 등

✅ 11. 예제 코드 전체

```
public class LoopDemo {
    public static void main(String[] args) {
        // for
        for (int i = 1; i <= 5; i++) {
            System.out.print(i + " ");
        }

        // while
        int j = 1;
        while (j <= 5) {
            System.out.print(j + " ");
            j++;
        }

        // do-while
        int k = 1;
        do {
            System.out.print(k + " ");
            k++;
        } while (k <= 5);
    }
}
```

`break`, `continue`, `label` 사용법

✅ 1. `break`

🔷 개념

`break`는 **반복문 전체를 즉시 종료**하고, 반복문 **바깥으로 탈출**시킨다.

🔷 사용 위치

- `for`, `while`, `do-while`, `switch` 문 안

🔷 기본 예시

```
for (int i = 1; i <= 10; i++) {
    if (i == 5) break;
    System.out.println(i);
}
// 출력: 1 2 3 4
```

- `i == 5`일 때 반복문 전체 종료됨

✅ 2. `continue`

🔷 개념

`continue`는 **현재 반복만 건너뛰고**, 다음 반복으로 진행한다.

🔷 기본 예시

```
for (int i = 1; i <= 5; i++) {
    if (i == 3) continue;
    System.out.println(i);
}
// 출력: 1 2 4 5
```

- `i == 3`일 때는 출력 생략됨

✅ 3. `break` vs `continue`

구문	효과
`break`	반복문 **전체 종료**
`continue`	현재 반복만 **건너뛰고 계속 진행**

✅ 4. `label` (레이블)

🔷 개념

Java는 `label:`을 사용해서 **반복문에 이름을 붙일 수 있음**.
`break label`, `continue label`을 사용하면 **특정 반복문을 직접 지정**해서 제어할 수 있다.

🔷 label 선언 예

```
outer: for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        if (j == 2) break outer;
        System.out.println("i=" + i + ", j=" + j);
    }
}
```

- `break outer`: 바깥쪽 반복문(`outer:`)을 **즉시 종료**

🔷 label with `continue`

```
outer: for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        if (j == 2) continue outer;
        System.out.println("i=" + i + ", j=" + j);
    }
}
```

- `continue outer`: 바깥쪽 루프의 **다음 반복으로 이동**

✅ 5. label 없는 경우 vs 있는 경우 비교

동작	label 없이	label 사용
`break`	가장 가까운 반복문 종료	지정한 label의 반복문 종료
`continue`	가장 가까운 반복문의 다음 반복	지정한 label의 반복문 다음 반복

✅ 6. 실전 예제: 2중 반복문 탈출

```
outer: for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        if (i * j > 4) {
            System.out.println("조건 충족: i=" + i + ", j=" + j);
            break outer;
        }
    }
}
```

- `i=2, j=3`에서 조건 충족 → 바로 `outer` 루프 종료

✅ 7. 실전 예제: 특정 조건만 건너뛰기

```
outer: for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        if (j == 2) continue outer;
        System.out.println("i=" + i + ", j=" + j);
    }
}
```

- `j == 2`일 때는 `outer` 루프의 다음 반복으로 이동

✅ 8. 주의사항

- label은 **반복문 앞에만 쓸 수 있음** (`for`, `while`, `do-while`)
- `switch`에는 label 사용 불가
- label 이름은 식별자 규칙을 따름 (`outer`, `loop1` 등)

✅ 요약 정리

키워드	기능	중첩 루프 제어	사용 예시
`break`	반복문 **전체 종료**	❌ (내부 루프만)	`if (조건) break;`
`continue`	**현재 반복 건너뛰기**	❌ (내부 루프만)	`if (조건) continue;`
`break label`	**지정 루프 종료**	✅	`break outer;`
`continue label`	**지정 루프 반복으로 이동**	✅	`continue outer;`

✅ 전체 실습 코드 예시

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

        // break
        for (int i = 1; i <= 5; i++) {
            if (i == 3) break;
            System.out.println("break: " + i);
        }

        // continue
        for (int i = 1; i <= 5; i++) {
            if (i == 3) continue;
            System.out.println("continue: " + i);
        }

        // label
```

```
        outer: for (int i = 1; i <= 2; i++) {
            for (int j = 1; j <= 2; j++) {
                if (i + j == 3) break outer;
                System.out.println("label: i=" + i + ", j=" + j);
            }
        }
    }
}
```