

객체지향 프로그래밍

4장

제어 흐름

조용주

ycho@smu.ac.kr

표현식(expression)과 명령문(statement)

□ 표현식이란 결과 값을 도출할 수 있는 코드

- 대입 연산자의 오른쪽 피연산자로 있는 변수
- 결과 값을 반환하는 함수 호출 코드
- 연산자를 사용하는 연산식

□ 명령문(statement)

- 구문이라고도 부름 (자바에서는 ~문 형태로 불림)
- 코드를 실행시키는 기본 단위
- 자바 언어에서의 명령문은 세미콜론(';')으로 종료되거나 '{'와 '}'로 감싸진 코드 블록으로 마무리됨

```
long x = 3L;  
int a;  
a = 3;
```

```
int a  
= 3;  
int  
b = 4;
```

들여쓰거나 줄바꿈은
가독성에 장점이 있지만
코드 실행에 영향을
미치지 못함

코드 블록(code block)

- '{'와 '}'안에 있는 명령문들을 넣어서 한 개의 명령문으로 취급하는 경우가 있음
 - "{...}" 형태로 묶인 코드를 블록(또는 코드 블록)
 - 실행할 코드가 없는 경우 완전히 비어 있는 형태("{}")로 만들어지기도 하지만, 보통 한 개 이상의 명령문이 포함됨
 - 블록은 클래스, 함수, 또는 다른 명령문을 구성하는 일부로 사용되는 경우가 많음

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("hello world!");  
    }  
}
```

코드 블록(code block)

- ▣ 블록 안에 또 다른 블록이 들어갈 수 있음

```
if (a == 5) { // 조건문
    for (int i = 0; i < a; i++) { // 반복문
        // 실행할 코드
    }
    System.out.println(a);
}
```

제어 흐름

□ 제어 흐름의 기본은 위에서 아래로

- 제어 흐름(Control Flow 또는 Flow of Control)은 프로그램을 실행시키는 흐름 즉 작업 순서를 제어하는 것
- 프로그램은 순차적(sequential order) 실행이 원칙
- 하지만 조건에 의해 경로가 바뀌거나 반복에 의해 흐름이 바뀌는 경우가 있음

□ 라면 조리 순서

- 물을 500mm 정도 냄비에 붓고 끓인다
- 물이 끓으면 면과 스프를 넣고 약 4분간 더 끓인다
- 식성에 따라 계란, 파 등을 넣고 먹는다

□ 큰 흐름은 작업들을 순서대로 실행

- 세부적으로 들어가면 달라짐

제어 흐름

□ 세분화된 라면 조리 단계

- 물을 500mm 정도 냄비에 붓고 끓인다
 - 냄비에 물을 500mm 정도 넣는다
 - 물을 끓이기 위해 불을 켜다
 - 냄비의 물이 끓는지 확인하면서 작업 순서를 결정한다
 - 물이 끓지 않으면 계속 확인하면서 기다린다
 - 물이 끓으면 다음 단계로 진행한다
- 면과 스프를 넣고 약 4분간 더 끓인다
 - 면을 넣는다
 - 스프를 넣는다
 - 4분간 기다린다
 - 4분이 지날 때까지 반복적으로 시간을 확인
 - 4분이 지나면 다음 단계로 진행

제어 흐름

- 식성에 따라 계란, 파 등을 넣고 먹는다
 - 계란, 파 등을 넣을 것인지 결정한다
 - 필요한 재료를 추가한다
 - 먹는다
- 조건을 확인하거나 반복 작업을 통해 흐름을 조정할 수 있는 **조건문**이나 **반복문**을 제공
- 이 밖에 순차적 흐름을 위반하는 명령어가 **break**나 **continue** 등이 있음

흐름의 제어

▣ 프로그램은 순차, 분기, 반복으로 흐름을 제어

구분	설명	명령문
순차	순서에 따라 작업이 실행되는 기본적인 실행 방법	goto, continue, break 등을 제외한 모든 명령문 goto는 실제 사용되지는 않음
분기	조건 표현식의 값 에 따라 분기하므로 의사 결정이라고 함	if, else, switch
반복	조건에 따라 혹은 일정 횟수 만큼 반복	for, while, do...while

조건문(분기문)

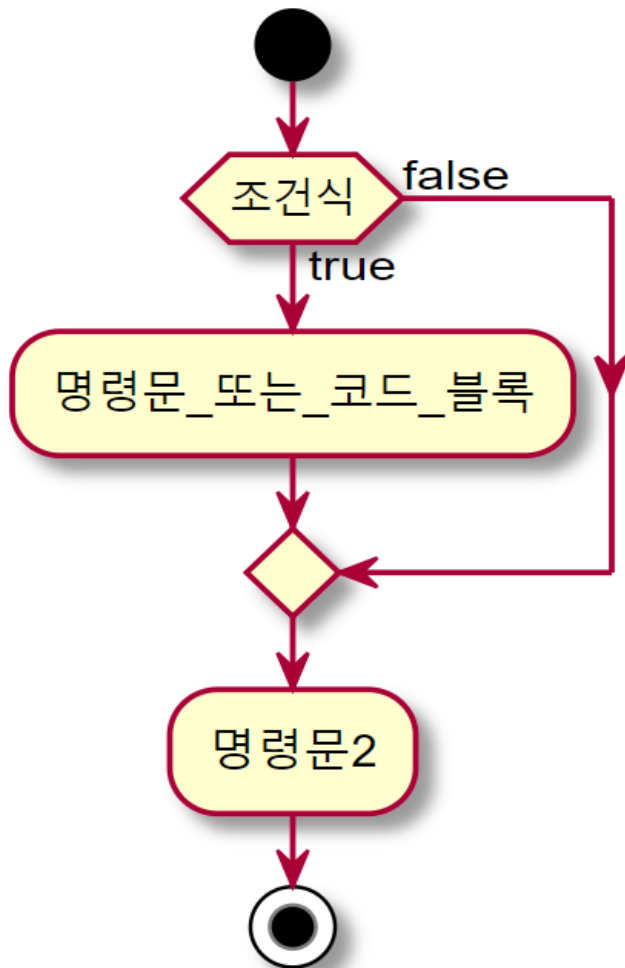
- 자바에서 조건에 따라 다른 일을 실행시키기 위해 조건문(conditional statement)을 사용
 - "더우면 냉방을 추우면 난방을 켜다"
 - 프로그래밍에서는 좀 더 구체적이어야 함
 - if문, if...else, if...else if else 등과 switch문
- if 문
 - 가장 기본적인 조건문
 - "if"라는 키워드로 시작됨
 - 두 가지 사용법

if (조건식)
명령문

if (조건식)
블록

조건문

□ if문 실행 방법



조건문

▣ 예제 코드

```
boolean b1 = true;
boolean b2 = false;
if (b1) // a 값이 5이면 true
    System.out.println("화면에 출력된다");
if (b2) { // a 값이 5가 아니면 true
    System.out.println("화면에 출력되지 않는다");
    System.out.println("a는 5가 아닙니다");
}
```

▣ 들여쓰기 주의

```
if (false)
    System.out.println("화면에 출력된다");
    System.out.println("화면에 또 출력된다");
System.out.println("조건과 관계없이 출력된다");
```

비교 연산과 논리 연산

- 조건식의 결과 값은 불린형으로 나옴
- 자바 언어에서 불린값을 생성하는 연산자
 - 비교 연산자
 - 같다, 다르다, 크다, 작다 등을 비교
 - 논리 연산자
 - 불린형 표현식을 조합해서 또 다른 불린값을 생성

비교 연산자

연산자	설명	사용 예
>	왼쪽 피연산자의 값이 오른쪽 값보다 크면 참, 작거나 같으면 거짓	$a > b$
>=	왼쪽 피연산자의 값이 오른쪽 값보다 크거나 같으면 참, 작으면 거짓	$a \geq b$
<=	왼쪽 피연산자의 값이 오른쪽 값보다 작거나 같으면 참, 크면 거짓	$a \leq b$
<	왼쪽 피연산자 값이 오른쪽 값보다 작으면 참, 크거나 같으면 거짓	$a < b$
==	왼쪽과 오른쪽 피연산자 값들이 같으면 참, 같지 않으면 거짓	$a == b$
!=	왼쪽과 오른쪽 피연산자 값들이 같지 않으면 참, 같으면 거짓	$a != b$

비교 연산자

```
int i = 1;
int j = 2;
int k = 2;
if (i > j) // false, 출력 안됨
    System.out.println("i(1)이 j(2)보다 큼");
if (i < j) // true, 출력됨
    System.out.println("i(1)이 j(2)보다 큼");
if (k >= j) // true, 출력됨
    System.out.println("k(2)가 j(2)보다 크거나 같음");
if (k <= j) // true, 출력됨
    System.out.println("k(2)가 j(2)보다 작거나 같음");
if (k == j) // true, 출력됨
    System.out.println("k(2)가 j(2)와 같음");
if (k != j) // false, 출력 안됨
    System.out.println("i(1)가 j(2)와 같지 않음");
```

비교 연산자

- System.out.println() 함수는 불린형 인식

```
jshell> int i = 1;
```

```
jshell> System.out.println(i > 0);  
true
```

문자열 비교

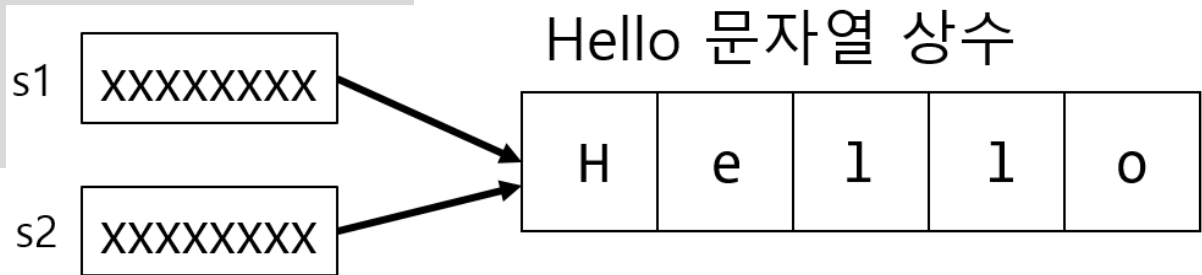
□ 문자열 비교

- 기본형과는 달리 주의해야 함
- 문자열 같은 참조형 변수에 비교 연산자를 사용하면 메모리 주소값을 비교

```
jshell> String s1 = "Hello";  
s1 ==> "Hello"
```

```
jshell> String s2 = "Hello";  
s2 ==> "Hello"
```

```
jshell> s1 == s2;  
$6 ==> true
```



문자열 비교

```
jshell> String s3 = new String("Hello");  
s3 ==> "Hello"
```

```
jshell> String s4 = new String("Hello");  
s4 ==> "Hello"
```

```
jshell> s3 == s4;  
$9 ==> false
```

Hello 문자열 상수

H	e	l	l	o
---	---	---	---	---

Hello 문자열 객체

H	e	l	l	o
---	---	---	---	---

s3 XXXXXXXXX



s4 XXXXXXXXX



Hello 문자열 객체

H	e	l	l	o
---	---	---	---	---

논리 연산자

- 논리 연산자를 사용하는 연산식의 결과는 참 또는 거짓

연산자	종류	설명
&&	AND	두 개 피연산자 값이 모두 true일 때 true
	OR	두 개 피연산자 값이 false일 때에만 false
!	NOT	true를 false로 false를 true로 바꿈

- AND 연산

연산	결과
true && true	true
true && false	false
false && true	false
false && false	false

논리 연산자

□ OR 연산

연산	결과
true true	true
true false	true
false true	true
false false	false

□ NOT 연산

연산	결과
!true	false
!false	true

논리 연산자

```
GPA >= 3.5 && incomeBracket <= 3  
!(GPA < 3.5 || incomeBracket > 3)  
temperature >= 25 || humidity <= 0.5  
!(temperature < 25) || humidity <= 0.5
```

□ 논리 연산자의 평가 순서와 단축 평가

- 논리 연산자의 값 확인 순서는 왼쪽에서 오른쪽 방향
- 자바에서는 논리 연산자의 왼쪽과 오른쪽에 있는 값들을 평가해서 최종 결과 값을 도출하는 과정에서 단축 평가 방법(short-circuit evaluation)을 사용함

□ 단축 평가

- 논리 연산자의 왼쪽 피연산자 값만으로 전체 논리 연산의 결과 값을 유추할 수 있다면 오른쪽 피연산자 값을 평가하지 않는 것

단축 평가

□ 단축 평가의 장점

- 효율적
- 예외 상황을 미리 막을 수 있음

```
int d = 0;
int n = 4;
if (d != 0 && n / d > 0) {
    System.out.println(n / d);
}
```

실습 문제 1: 장학생 선발하기

□ 문제

- 어떤 학생이 이번 학기에 장학생 후보가 되는지 확인하는 프로그램 작성
- 학교에서 정한 후보는 학기 평점 3.5이상과 소득분위 5 이하

이름	평점	소득분위
김규상	4.1	3
김민재	3.71	5
김용하	3.93	7

□ 요구사항

- 학생 이름, 평점, 소득분위를 데이터로 가지고 있고, 이름, 평점, 소득분위 등을 결과 값으로 반환하는 메소드를 포함하는 클래스 구성하고 표에 있는 자료로 초기값 지정

조건이 만족될 때와 만족되지 않을 때 다른 코드 실행하기

- 조건이 만족될 때와 만족되지 않을 때 서로 다른 코드가 실행되어야 하는 경우가 있음
 - 동전을 던져서 어떤 면이 나오는지 화면에 출력

```
// 동전을 던지는 것에 대해서는 어떻게 처리하는지
// 배우지 않았으므로, 임의의 값을 미리 지정
char coin = '앞';
if (coin == '앞') // ----- (1)
    System.out.println("앞면");
if (coin == '뒤') // ----- (2)
    System.out.println("뒷면");
```

조건이 만족될 때와 만족되지 않을 때 다른 코드 실행하기

□ if-else 구문 사용법

```
if (조건식)          // 방법 1
    명령문;          // (1)번 코드
else
    명령문;          // (2)번 코드

if (조건식) {        // 방법 2
    명령문;          // (1)번 블록
    명령문;
}
else {
    명령문;          // (2)번 블록
    명령문;
}
```


조건이 만족될 때와 만족되지 않을 때 다른 코드 실행하기

▣ 동전 던지는 예제 코드

```
// 동전을 던지는 것에 대해서는 어떻게 처리하는지
// 배우지 않았으므로, 임의의 값을 미리 지정
char coin = '앞';
if (coin == '앞') // ----- (1)
    System.out.println("앞면");
else             // ----- (2)
    System.out.println("뒷면");
```

실습 문제 2: 약수인지 확인하고 출력하기

□ 문제

- 사용자로부터 정수 2개를 입력 받고, 첫 번째 숫자가 두 번째 숫자의 약수인지 확인해서 출력

□ 요구사항

- 사용자는 정수만을 입력할 것이라고 가정

여러 가지 조건 중 한 가지만 선택하기

▣ 여러 조건 중 한 개만을 선택해야 하면 다중 분기문인 **if-else if-else**문을 사용

- 2016년 이후 극장에서는 좌석의 위치와 요일을 분할해서 다르게 요금을 받음

```
if (조건식1)           // 방법 1
    명령문;           // (1)번 코드
else if (조건식2)
    명령문;           // (2)번 코드

...                   // 추가 else if 문

else                   // 생략 가능
    명령문;
```

여러 가지 조건 중 한 가지만 선택하기

```
if (조건식) {           // 방법 2
    명령문;             // (1)번 블록
    명령문;
}
else if (조건식 2) {
    명령문;             // (2)번 블록
    명령문;
}

...                     // 추가 else if 문

else {                  // 생략 가능
    명령문;
}
```

여러 가지 조건 중 한 가지만 선택하기

■ 영화 관람 좌석에 대한 코드 작성

- 코드의 단순화를 위해 요일에 상관없이 좌석에 따라 세 가지 등급으로 분류

좌석 영역	가격
이코노미존(economy zone)	9000
스탠다드존(standard zone)	10000
프라임존(prime zone)	11000

- 코드에서는 zone이라는 문자열 변수를 만들어서 좌석 영역 별로 첫 번째 단어를 저장
- 그리고 좌석 등급에 따라 화면에 관람 가격을 출력

여러 가지 조건 중 한 가지만 선택하기

```
String zone = "prime"; // 프라임존으로 지정

if (zone.compareTo("prime") == 0) {
    System.out.println("프라임존 표는 11000원.");
}
else if (zone.compareTo("standard") == 0) {
    System.out.println("스탠다드존 표는 10000원.");
}
else if (zone.compareTo("economy") == 0) {
    System.out.println("이코노미존 표는 9000원.");
}
```

여러 가지 조건 중 한 가지만 선택하기

```
String zone = "prime"; // 프라임존으로 지정

if (zone.compareTo("prime") == 0) {
    System.out.println("프라임존 표는 11000원.");
}
else if (zone.compareTo("standard") == 0) {
    System.out.println("스탠다드존 표는 10000원.");
}
else { // prime이나 standard가 아니면
    System.out.println("이코노미존 표는 9000원.");
}
```

실습 문제 3: 극장 표 값 알아보기

□ 문제

- 사용자로부터 원하는 종류의 극장 좌석을 입력 받고, 해당 영역의 가격을 화면에 출력하는 프로그램을 작성

□ 요구사항

- 사용자로부터의 입력은 키보드로 받음
- 사용자는 "prime", "standard", "economy" 또는 다른 단어 입력 가능
- 다른 단어 입력했을 경우 "좌석 종류를 잘못 입력했습니다."를 출력

중첩 조건문

■ 중첩 조건문

- 조건문에서 실행시키는 명령문에 또 다른 조건문이 들어 있음
- 중첩이 반복되는 경우도 중첩 조건문

```
if (조건식 1) {  
    if (조건식 2) {  
        명령문; // (1)  
    }  
}
```

```
if (조건식 1) {  
    if (조건식 2) {  
        if (조건식 3) {  
            명령문;  
        }  
    }  
}
```

중첩 조건문

- 장학생 후보 확인 코드를 전에는 &&를 이용해서 처리

- 중첩 반복문을 사용하면

```
if (KimGyuSangGPA >= 3.5)
    if (KimGyuSangIncomeBracket <= 5)
        System.out.println("김규상은 장학생 후보");
```

- 중첩 조건문을 &&형태로 무조건 바꿀 수 있는 것은 아님

```
if (KimGyuSangGPA >= 3.5) {
    System.out.println("김규상 평점은 3.5이상");
    if (KimGyuSangIncomeBracket <= 5)
        System.out.println("김규상은 장학생 후보");
}
```

중첩 조건문

- 중첩 조건문은 if, if-else, if-else if, if-else if-else 등에서 모두 사용 가능

```
if (KimGyuSangGPA >= 3.5) {  
    System.out.println("김규상 평점은 3.5이상");  
    if (KimGyuSangIncomeBracket <= 5)  
        System.out.println("김규상은 장학생 후보");  
    else  
        System.out.println(  
            "안타깝게도 김규상은 장학생 후보가 아님");  
}
```

삼항 조건 연산자(ternary conditional operator)

- 삼항 조건 연산자는 ?과 :로 구성된 연산자
 - 조건식과 표현식에 해당되는 피연산자 3개
 - 특수한 경우의 if-else 구문을 대체할 수 있음
 - 결과 값이 존재하며 조건식이 아닌 피연산자와 같은 자료 형이 도출됨
- 사용 방법
 - 조건식 ? 표현식1 : 표현식2;

삼항 조건 연산자(ternary conditional operator)

□ 주 사용 예

- 조건에 따라 변수에 다른 값을 저장

```
변수 = 조건식 ? 표현식1 : 표현식2; // 방법 1
```

```
if (조건식)                                // 방법 2
    변수 = 표현식1;
else
    변수 = 표현식2;
```

- 조건에 따라 함수에서 다른 값을 반환

```
return (조건식) ? 표현식1 : 표현식2; // 방법 1
```

```
if (조건식)                                // 방법 2
    return 표현식1;
else
    return 표현식2;
```

삼항 조건 연산자(ternary conditional operator)

- ?: 연산자를 사용해서 학생이 절대 평가 교과목에서 다음 단계로 넘어갈 수 있는지 확인하는 코드 작성

```
int score = 70;  
boolean pass = (score >= 60) ? true : false;  
System.out.println("pass or fail: " + pass);
```

- 삼항 조건 연산자를 쓰지 않는다면?

```
int score = 70;  
boolean pass;  
if (score >= 60)  
    pass = true;  
else  
    pass = false;  
System.out.println("pass or fail: " + pass);
```

삼항 조건 연산자(ternary conditional operator)

▣ 정수값을 절대값으로 만드는 코드

■ if문

```
int number = -10;
int absNumber;

if (number < 0)
    absNumber = -number;
else
    absNumber = number;
System.out.println("number의 절대값 = " + absNumber);
```

■ 삼항 조건 연산자

```
int number = -10;
int absNumber = (number < 0) ? -number : number;
System.out.println("number의 절대값 = " + absNumber);
```

반복

□ 반복문

- 같은 작업 또는 비슷한 작업을 반복시키는 것을 처리하는 방법
- while, do...while, for문 등이 있음

□ 컴퓨터에서 프로그램을 작성하는 이유 중 가장 큰 것은 사람이 하기 싫어하는 단순하고 반복적인 작업을 시키기 위해서일 듯

□ 예

- 주사위를 100번 굴려 확률 계산
- 자바 수업에서 시험을 보고 평균을 구하는 문제

반복문의 구성

□ 종료 조건

- 반복문을 종료시키는 조건
- 프로그래밍에서 **반복**은 대부분 특정 조건이 만족될 때까지 지속되는 것
- 반복을 종료시키는 조건은 true또는 false값이 나오는 조건식으로 표현됨

□ 반복 명령어

- 반복문을 구성하는 명령어 (while, do...while, for 키워드)
- 세 가지 키워드를 사용하는 반복문들은 서로 바꿔 사용 가능
- 특화된 용도가 있음

반복문의 구성

▣ 자바의 반복문은 조건 만족형

명령어	종료 조건 확인 시기	설명
while	반복 작업 전	조건이 만족될 때 반복하고 싶을 때 주로 사용됨 조건이 먼저 확인되므로 반복작업이 시작 안될 수 있음
do...while	반복 작업 후	조건이 만족될 때 반복하고 싶을 때 주로 사용됨 조건이 나중에 확인되므로 반복작업을 최소한 한 번 실행
for	반복 작업 전	주로 정해진 횟수만큼 반복할 때 사용됨

while 문 살펴보기

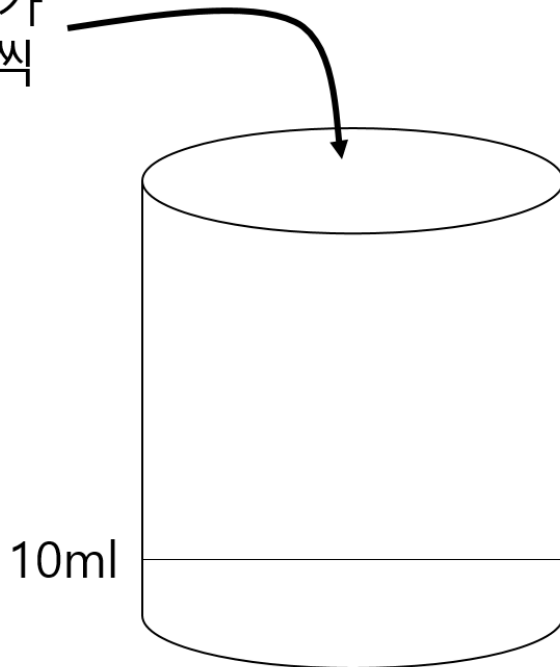
□ if문과 while문 비교

If문	While문
if (조건식) 명령문;	while (조건식) 명령문;
if (조건식) { 명령문; }	while (조건식) 명령문;
if (조건식) { 명령문; }	while (조건식) { 명령문; }
if (조건식) { 명령문; }	while (조건식) { 명령문; }

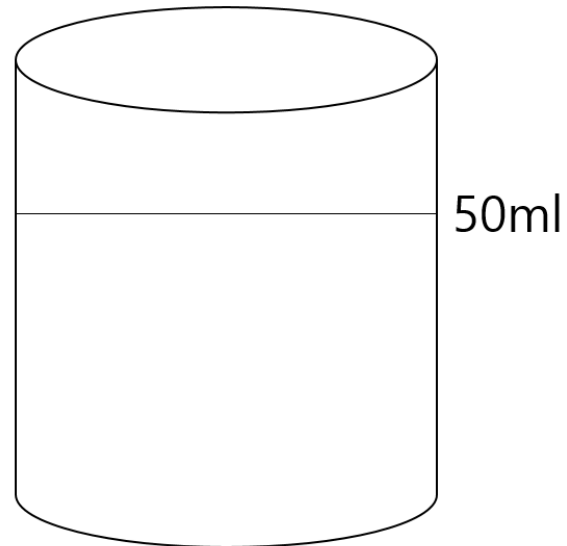
while문을 사용하는 예

- Cup1에 물을 한 번에 10ml씩 추가해서 Cup2에 있는 물의 양과 같거나 더 많게 만들려고 함

물 추가
10ml씩



Cup1



Cup2

while문을 사용하는 예

□ 해결 방법

1. 현재 컵에 있는 물의 양을 확인 (Cup1 = 10ml, Cup2 = 50ml)
2. Cup1의 물의 양이 Cup2에 있는 물의 양보다 적으면 3번을 실행. 만약 Cup1의 물의 양이 Cup2에 있는 물의 양보다 많거나 같다면 4번으로 넘어감
3. Cup1에 10ml 물을 추가
4. 물을 추가하는 작업을 멈춤

while문을 사용하는 예

□ 코드로 작성

- Cup1과 Cup2에 있는 물의 양을 기록

```
int Cup1 = 10;  
int Cup2 = 50;
```

- 물의 양을 비교

```
Cup1 < Cup2
```

- Cup1에 물 10ml 추가

```
Cup1 = cup1 + 10; // 또는 Cup1 += 10;
```

- Cup1의 물의 양이 Cup2의 물의 양보다 적다면 Cup1에 10ml의 물을 추가
 - 반복 (if → while)

```
if (Cup1 < Cup2) {  
    Cup1 += 10;  
}
```

```
while (Cup1 < Cup2) {  
    Cup1 += 10;  
}
```

while문을 사용하는 예

▣ 전체 코드

```
// Cup1Cup2While.java
public class Cup1Cup2While {
    public static void main(String[] args) {
        int Cup1 = 10;
        int Cup2 = 50;

        while (Cup1 < Cup2) {
            Cup1 = Cup1 + 10;
        }

        System.out.println("Cup1 = " + Cup1);
        System.out.println("Cup2 = " + Cup2);
    }
}
```

```
1 C:\Code\java\04>java Cup1Cup2While
2 Cup1 = 50
3 Cup2 = 50
```

실습 문제 4: 단어 입력 받고 화면에 출력

□ 문제

- 사용자로부터 단어를 입력 받고 화면에 출력하는 프로그램 작성
- 사용자가 "quit"이라는 단어를 입력하면 프로그램 종료

□ 요구사항

- "quit"을 입력하면 화면에 출력하지 않고 프로그램 종료
- 사용자가 숫자를 입력해도 문자열로 취급

do...while 살펴보기

□ do...while문

- while문과 유사하지만, 반복 코드를 한 번 실행한 이후에 종료 조건을 확인

```
do {  
    명령문;  
} while (조건식);
```

□ while과 do...while 비교

	while문	do...while문
반복 종료 조건 값	true일때 반복	true일때 반복
종료 조건 확인 시점	반복 코드 실행 전	반복 코드 실행 후
최소 코드 실행 횟수	0 (한 번도 실행 안될 수 있음)	1 (최소 한 번 이상은 실행 됨)

do...while 사용 예

□ 컵에 물 붓는 문제 다시 풀기

- Cup1의 물의 양이 Cup2의 물의 양에 비해 적을 때 시작하기 때문에, 무조건 물을 10ml 붓고 조건을 따진다고 해도 문제되지는 않음
- while문을 do...while문으로 바꾼 코드

```
do {  
    Cup1 += 10;  
while (Cup1 < Cup2);
```

- 이 경우에는 while문과 do...while문이 동일하게 동작함

do...while 사용 예

▣ 전체 코드

```
// Cup1Cup2DoWhile.java
public class Cup1Cup2DoWhile {
    public static void main(String[] args) {
        int Cup1 = 10;
        int Cup2 = 50;

        do {
            Cup1 += 10;
        } while (Cup1 < Cup2);

        System.out.println("Cup1 = " + Cup1);
        System.out.println("Cup2 = " + Cup2);
    }
}
```

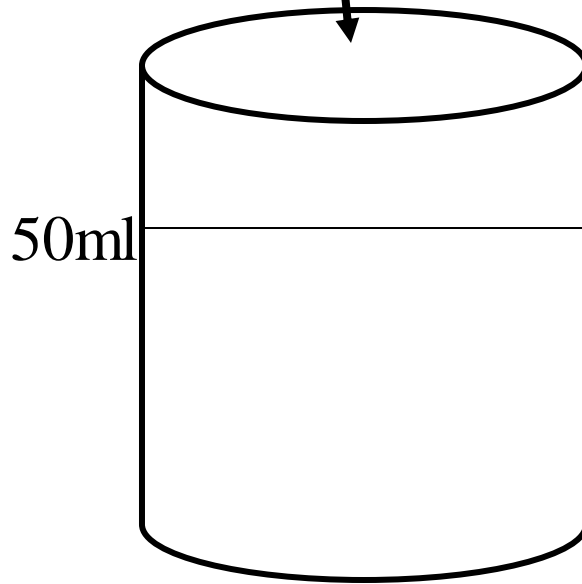
```
1 C:\Code\java\04>java Cup1Cup2DoWhile
2 Cup1 = 50
3 Cup2 = 50
```

do...while 사용 예

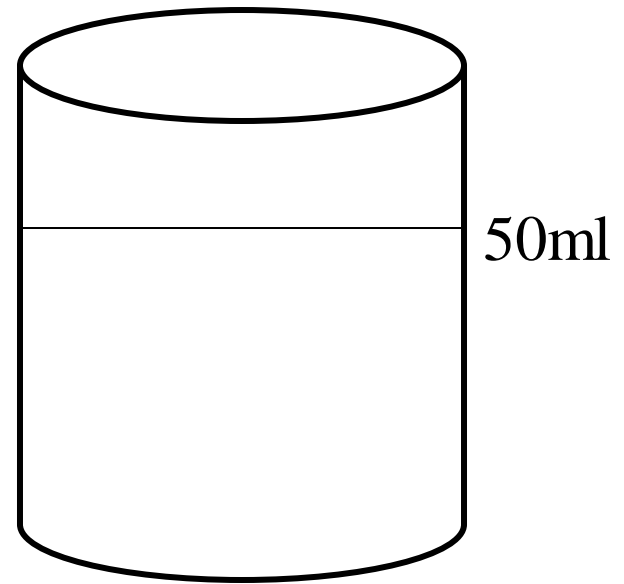
□ 항상 while문을 do...while문으로 바꿀 수 있는 것은
아님

□ 예

물 추가
10ml씩



Cup1



Cup2

실습 문제 5: 사용자가 0을 입력할 때까지 지속적으로 정수를 입력 받고 화면에 출력

□ 문제

- 사용자로부터 지속적으로 정수를 한 개씩 입력 받아 출력하는 프로그램을 작성
- 사용자가 0을 입력하면 프로그램 종료

□ 요구사항

- 0을 입력하면, 그 값을 화면에 출력하고 프로그램이 종료됨

for 반복문

- 자바의 for 반복문은 세 가지 중에서 가장 활용도가 높을 것 같음
 - while문이나 do...while문을 대체해서 쓰는 것도 가능 (특히 while문)
 - 주로 정해진 횟수만큼 반복하거나, 배열이나 자료 구조 처럼 한정된 데이터가 포함된 자료구조들과 자주 사용됨
- 정해진 횟수만큼 반복하는 내용을 while문으로 작성
 - 5번 "hello" 출력

```
int i = 0;           // 횟수 초기화
while (i < 5) {      // 횟수 비교
    System.out.println("hello"); // 반복될 코드
    i++;             // 횟수 세기
}
```

for 반복문

□ 정해진 횟수만큼 반복하는 내용을 for문으로 작성

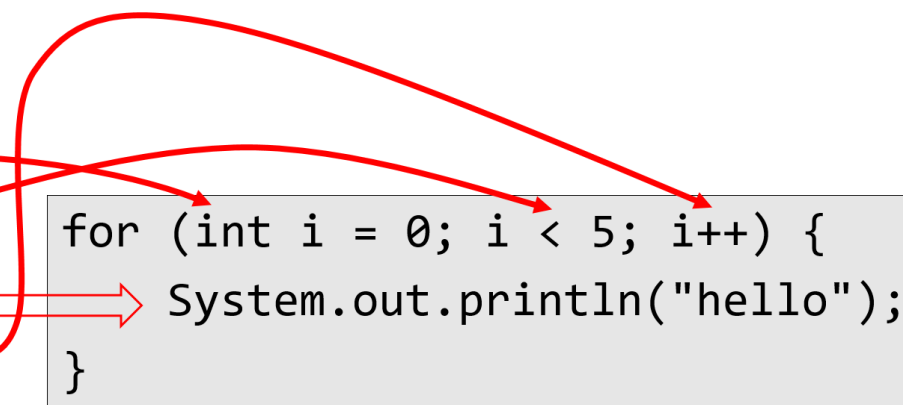
■ 5번 "hello" 출력

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

□ while문과 for문 비교

```
int i = 0;           // 횟수 초기화  
while (i < 5) {      // 횟수 비교  
    System.out.println("hello");  
    i++;             // 횟수 세기  
}
```

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



for 반복문

□ for 반복문 사용법

```
for (명령문1; 명령문2; 명령문3) {  
    반복될_명령문;  
}
```

□ 명령문1

- 반복될_명령문 실행 전에 수행됨
- 반복될_명령문 내부에서 사용될 변수를 생성하고 초기화

□ 명령문2

- 종료 조건을 확인하는 코드. 명령문2의 값이 false가 되면 반복될_명령문 실행 중지
- 명령문1이 실행된 후, 반복될 명령문 실행 전에 수행됨

□ 명령문3

- 반복될_명령문 실행 후 수행됨

for 반복문

▣ 실행 순서

1 2,5,8,... 4,7,10,...

```
for (명령문1; 명령문2; 명령문3) {  
    반복될_명령문;3,6,9,...  
}
```

- ▣ 명령문 → 명령문2 → 반복될_명령문 → 명령문3 →
명령문2 → 반복될_명령문 → 명령문3 -> ...

for 반복문

- for문은 while문을 완벽히 대체할 수 있음

```
for ( ; 종료 조건 확인; ) {  
    반복될_명령문;  
}
```

- 하지만 주로 정해진 횟수만큼 반복할 때 혹은 정해진 개수의 자료들을 처리할 때 주로 사용됨
- 다음 형태로 많이 사용됨

```
for ( 변수 초기화; 종료_조건_확인; 변수_증감_연산 ) {  
    반복될_명령문;  
}
```

for 반복문

□ for문 사용 예

- $1 + 2 + \dots + 9 + 10$ 의 합을 구하는 프로그램

```
int sum = 0; // 합계를 저장할 변수를 초기화
// 1부터 10까지 i 변수값을 증가시키며 반복
for (int i = 1; i <= 10; i++) {
    sum += i;    // sum에 i값을 더한 후에 다시 저장
}
```

- 정수형 배열을 생성하고 해당 배열의 요소를 모두 화면에 출력

```
// 초기화 형태로 요소 세 개짜리 배열 생성
int[] arr = { 1, 2, 3 };
for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
}
```

반복문 사용 예

- ▣ 0부터 9까지의 숫자를 연속적으로 출력하는 프로그램 작성
 1. 변수를 한 개 만들어서 0으로 초기화
 2. 변수 값을 화면에 출력
 3. 변수 값을 1만큼 증가
 4. 증가된 변수 값이 10보다 작으면 2번부터 반복
- ▣ 조건 검사를 마지막에 하므로, do...while문이 제일 적합해 보임

```
int i = 0;
do {

    System.out.println(i);
    i++;
} while (i < 10);
```

반복문 사용 예

□ while문으로 수정

```
int i = 0;           // 초기화 코드
while (i < 10) {     // 종료 조건
    확인 코드
    System.out.println(i);
    i++;             // 증감 연산
}
```

□ while문을 for문으로 변환

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

반복문 사용 예

- 반복문은 감소하는 형태로 사용될 수도 있음
 - 9부터 0까지 거꾸로 출력하는 프로그램을 작성

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

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

반복문 사용 예

- 반복문의 증감 연산은 1이 아닐 수도 있음

```
// 반복할 때마다 2씩 증가됨
for (int i = 1; i <= 15; i+=2) {
    System.out.println(i);
}
```

중첩 반복문

□ 반복문 안에 반복문이 들어가는 것

- 10-59까지 숫자를 출력하는 프로그램을 단일 반복문과 중첩 반복문을 이용해서 구현함
- 단일 반복문은 10-59까지 1씩 증가시키며 출력

```
// Output10_59_1.java
public class Output10_59_1 {
    public static void main(String[] args) {
        for (int i = 10; i <= 59; i++) {
            System.out.print(i);
            System.out.print("\t");
        }
    }
}
```

C:\Code\java\04>java Output10_59_1

```
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
```


중첩 반복문

- 중첩 반복문 사용
 - 바깥쪽 반복문
 - 변수 i 를 10-50까지 10씩 증가
 - 안쪽 반복문
 - 변수 j 를 0-9까지 1씩 증가
 - $i + j$ 를 출력

중첩 반복문

```
// Output10_59_2.java
public class Output10_59_2 {
    public static void main(String[] args) {
        for (int i = 10; i <= 50; i+=10) {
            // i가 10, 20, ..., 50일 때
            // 다음 반복문이 실행됨
            for (int j = 0; j <= 9; j++) {
                System.out.print(i + j);
                System.out.print(" ");
            }
        }
    }
}
```

중첩 반복문

- 중첩 반복문
 - 십의 자리와 일의 자리를 분리시킴
 - 십의 자리는 1~5까지 1씩 증가
 - 일의 자리는 1~9까지 1씩 증가
 - 십의 자리 숫자와 일의 자리 숫자를 문자열로 변환한 후에 연결시켜 화면에 출력

```
jshell> int tens = 3;  
tens ==> 3
```

```
jshell> int units = 5;  
units ==> 5
```

```
jshell> System.out.println("" + tens + units);  
35
```

중첩 반복문

```
// Output10_59_3.java
public class Output10_59_3 {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            // i가 1, 2, 3, 4, 5일 때 반복문 실행
            for (int j = 0; j <= 9; j++) {
                System.out.print("" + i + j);
                System.out.print(" ");
            }
        }
    }
}
```

중첩 반복문

- 중첩 반복문
 - ▣ for문과 while문을 중첩시킴

```
// Output10_59_4.java
public class Output10_59_4 {
    public static void main(String[] args) {
        for (int i = 10; i <= 50; i+=10) {
            int j = 0;
            while (j <= 9) {
                System.out.print(i + j);
                System.out.print(" ");
                j++;
            }
        }
    }
}
```