

객체지향 프로그래밍

4장

제어 흐름

조용주

ycho@smu.ac.kr

제어 흐름

- 제어 흐름(Control Flow 또는 Flow of Control)은 프로그램을 실행시키는 흐름 즉 작업 순서를 제어하는 것
- 제어 흐름의 기본은 위에서 아래로 실행
 - 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)

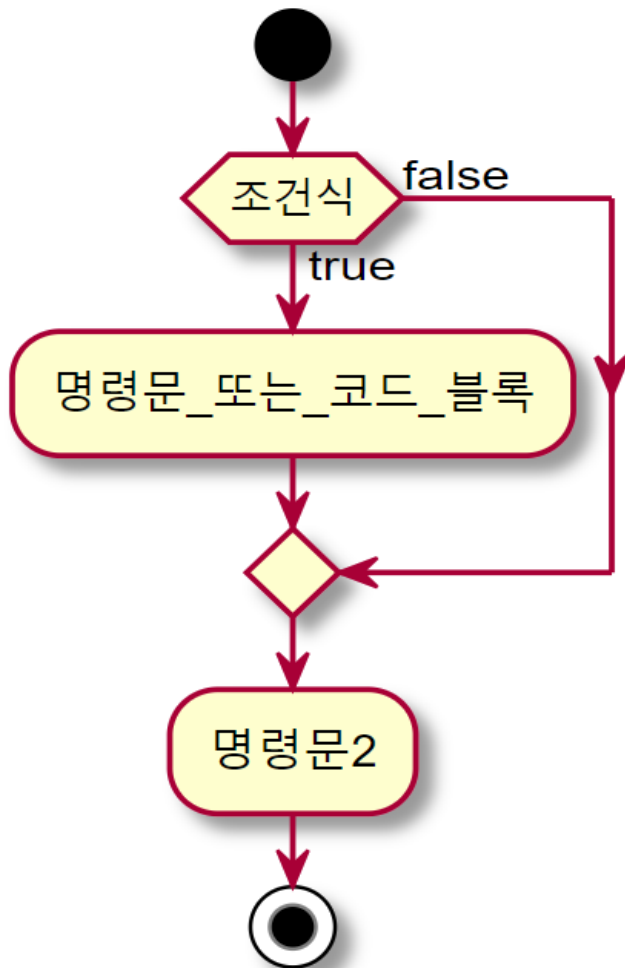
- 자바에서 조건에 따라 다른 일을 실행시키기 위해 조건문(conditional statement)을 사용
 - "더우면 냉방을 추우면 난방을 켜다"
 - 프로그래밍에서는 좀 더 구체적이어야 함
 - if문, if...else, if...else if else 등과 switch문
- if 문
 - 가장 기본적인 조건문
 - "if"라는 키워드로 시작됨
 - 두 가지 사용법

if (조건식)
명령문

if (조건식)
블록

조건문

□ if문 실행 방법



조건문

▣ 예제 코드

```
boolean b1 = true;
boolean b2 = false;
if (b1)
    System.out.println("화면에 출력된다");
if (b2) {
    System.out.println("화면에 출력되지 않는다");
}
```

▣ 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

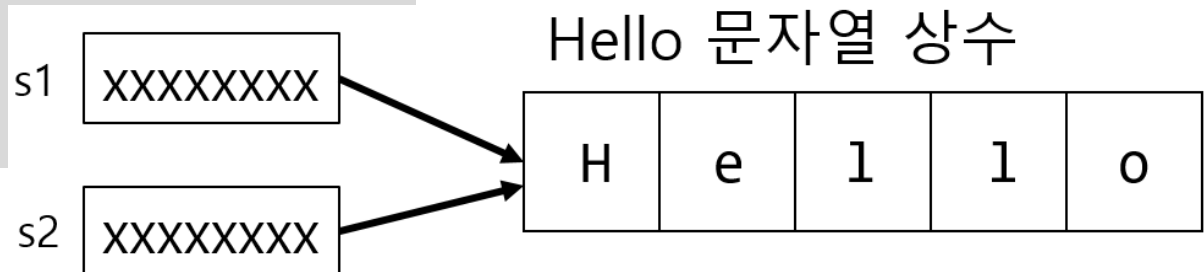
□ 문자열 비교

- 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
 - 값을 반환하는 간단한 형태의 if-else 구문 대체 가능
 - 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)

□ 삼항 조건 연산자를 쓰지 않는다면?

```
int score = 70;  
String status = "pass";  
if (score >= 60)  
    status = "pass";  
else  
    status = "fail";  
System.out.println("status: " + status);
```

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

```
int score = 70;  
String status = (score >= 60) ? "pass" : "fail";  
System.out.println("status: " + status);
```

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 때까지 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

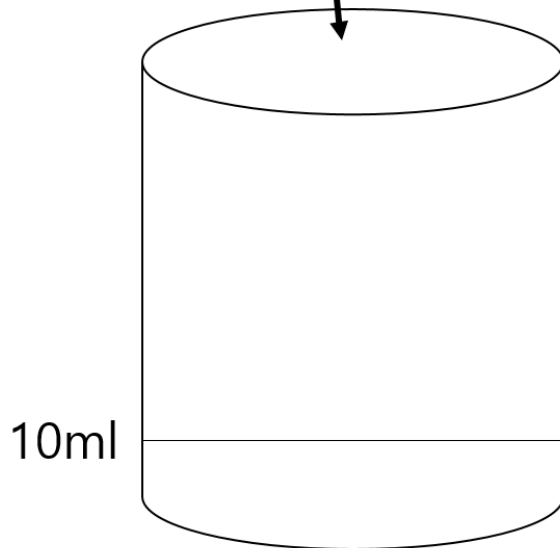
□ if문과 while문 비교

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; }

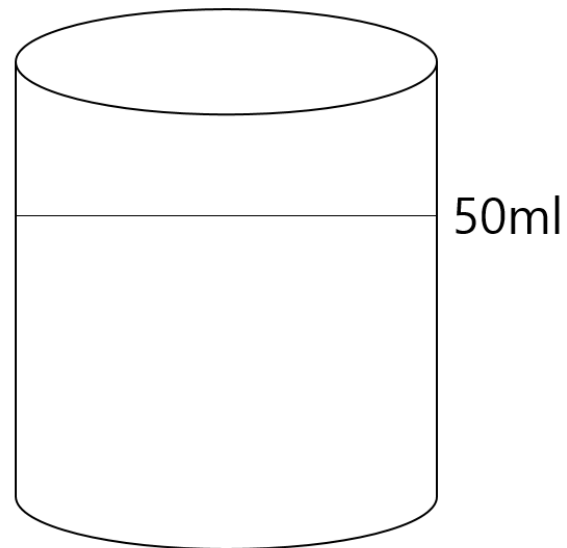
while문을 사용하는 예

- 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 단계 3번을 실행. 만약 Cup1의 물의 양이 Cup2에 있는 물의 양보다 많거나 같다면 단계 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);
    }
}
```

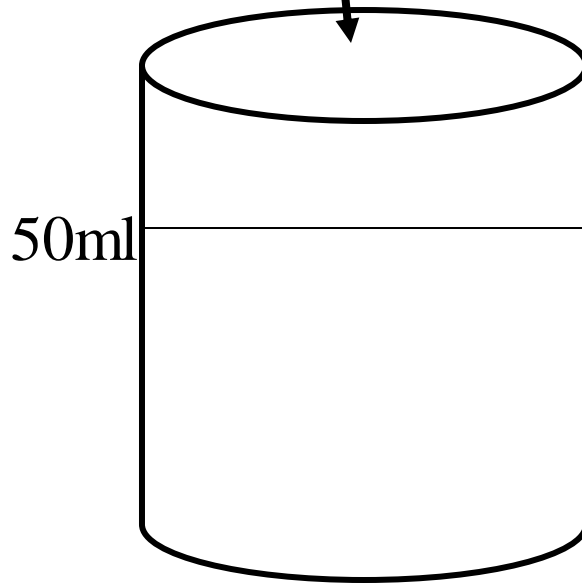
```
1 C:\Code\java\04>java Cup1Cup2DoWhile
2 Cup1 = 50
3 Cup2 = 50
```


do...while 사용 예

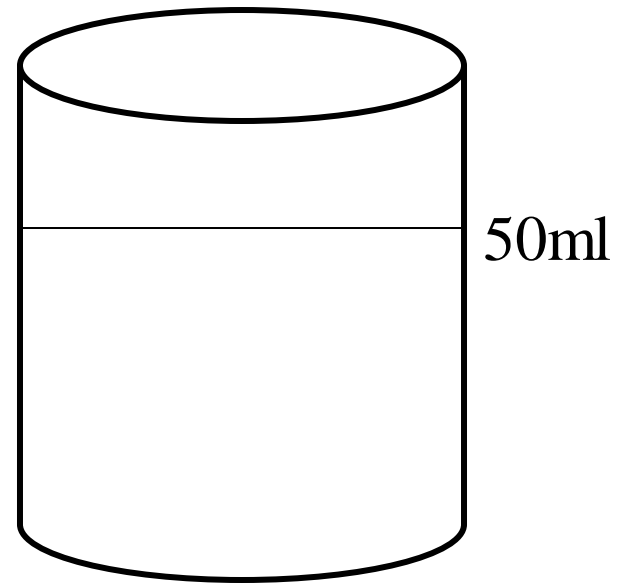
□ 항상 while문을 do...while문으로 바꿀 수 있는 것은
아님

□ 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
- 사용자가 정수만을 입력하는 것을 가정

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

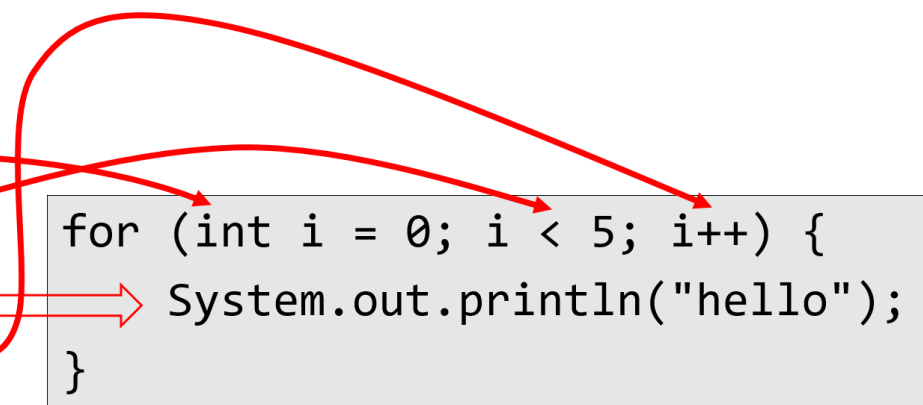
■ 5번 "hello" 출력

```
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 + ... + 9 + 10의 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(" ");
        }
    }
}
```

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 0~9까지 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++;
            }
        }
    }
}
```

break를 이용해서 반복 끝내기

- 사용자로부터 글자 한 개를 지속적으로 입력 받아 화면에 출력하는 프로그램 작성
- 반복하기 전에 한 글자를 입력 받고 화면에 출력하는 프로그램을 작성

```
jshell> import java.util.Scanner;
```

```
jshell> Scanner scanner = new  
Scanner(System.in);
```

```
jshell> char ch = scanner.next().charAt(0);  
// 단어를 읽어서 첫 번째 글자만 추출  
hello
```

```
jshell> System.out.println(ch);  
h
```

break를 이용해서 반복 끝내기

- 이번에는 'n'이라는 글자가 들어올 때까지 반복하는 코드 작성하기 전에 입력한 글자가 'n'일 때 화면에 출력하지 않도록 하는 코드

```
jshell> ch = scanner.next().charAt(0);  
  
jshell> if (ch != 'n') {  
            System.out.println(ch);  
        }
```

- 반복하는 코드로 다시 수정

```
ch = scanner.next().charAt(0);  
while (ch != 'n') {  
    System.out.println(ch);  
    ch = scanner.next().charAt(0);  
}
```

break를 이용해서 반복 끝내기

□ do...while로 변경

```
do {  
    ch = scanner.next().charAt(0);  
    if (ch != 'n') {  
        System.out.println(ch);  
    }  
} while (ch != 'n');
```

□ 'n'인지 확인하는 코드가 아직 중복됨

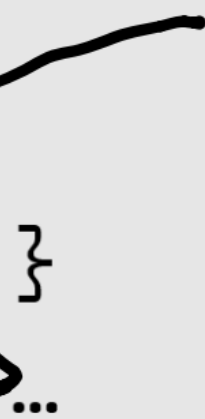
□ 이를 해결할 수 있는 것이 break문

- break는 반복문 또는 switch문 내부에서만 사용 가능하며 해당 구문 실행 종료

break를 이용해서 반복 끝내기

□ break 동작 방법

```
반복문 {  
    ...           // (1)  
    break;        // (2)  
    ...           // (3)  
}  
...              // (4)
```



break를 이용해서 반복 끝내기

▣ break를 이용해서 수정한 코드

```
import java.util.Scanner;

char ch;
Scanner scanner = new
Scanner(System.in);

do {
    ch = scanner.next().charAt(0);
    if (ch != 'n') {
        System.out.println(ch);
    }
    else {
        break;
    }
} while (true);
```

중첩 반복문에서 break 실행

- break가 중첩 반복문에서 사용되면, break문이 속한 반복문만 종료됨

```
반복문 {  
    ...  
    반복문 {  
        ...  
        반복문 { // (1)  
            ...  
            break;  
            ...  
        }  
    }  
    ...  
} // (2)
```


실습 문제 6: 정수 입력 받고 화면에 출력하기

□ 문제


- 사용자로부터 정수 한 개를 입력 받아서 출력하는 프로그램을 작성
- 사용자가 0을 입력하면 프로그램은 종료됨

□ 요구사항

- 0이 입력되면 화면에 출력하지 않고 프로그램 종료
- main() 함수에서 System.exit()이나 return 등을 이용해서 강제로 종료시키지 않고 break를 사용해서 반복문을 빠져나가도록 할 것
- 사용자가 정수만을 입력하는 것을 가정

continue를 이용한 반복 되돌리기

- continue문은 반복문 내부에서 사용되며, continue 다음에 있는 반복문 코드는 실행되지 않고, 다시 반복문의 첫 부분부터 실행



```
반복문 { // (1)
    ... // (2)
    continue; // (3)
    ... // (4)
}
// (5)
```

continue를 이용한 반복 되돌리기

- continue가 실행되면, 반복문의 종류에 관계없이 종료 조건이 만족되어야 다시 실행
- for문에서는 종료 조건을 다시 확인하기 전에 증감 연산이 먼저 실행됨

```
while (조건식) {  
    ... ③  
    continue; ①  
    ...  
}
```

```
do {  
    ... ③  
    continue; ①  
    ...  
} while (조건식); ②
```

continue를 이용한 반복 되돌리기

```
for (초기화; 조건식③; 증감연산②) {  
    ...④  
    continue;①  
    ...  
}
```

continue를 이용한 반!

```
// ContinueInWhile
```

```
int i = 3;
```

```
while (i < 7) {
```

```
    System.out.printf("if 전: i = %d\n", i);
```

```
    if (i < 7) {
```

```
        i += 2;
```

```
        System.out.printf("continue 전: i = %d\n", i);
```

```
        continue;
```

```
        //System.out.printf("continue 후 코드: i =  
%d\n", i);
```

```
    }  
    System.out.println("이건 출력되면 안됨!");
```

```
}  
System.out.printf("i = %d, 이건 출력됨", i);
```

```
C:\Code\java\04>java ContinueInWhile
```

```
if 전: i = 3
```

```
continue 전: i = 5
```

```
if 전: i = 5
```

```
continue 전: i = 7
```

```
i = 7, 이건 출력됨
```

continue를 이용한

```
// ContinueInDoWhile
```

```
int i = 3;
```

```
do {
```

```
    System.out.printf("if 전: i = %d\n", i);
```

```
    if (i < 7) {
```

```
        i += 2;
```

```
        System.out.printf("continue 전: i = %d\n", i);
```

```
        continue;
```

```
        //System.out.printf("continue 실행 후 코드: i =  
%d\n", i);
```

```
    }  
    System.out.println("이건 출력되면 안됨!");
```

```
} while (i < 7);
```

```
System.out.printf("i = %d, 이걸 출력됨", i);
```

```
C:\Code\java\04>java ContinueInDoWhile
```

```
if 전: i = 3
```

```
continue 전: i = 5
```

```
if 전: i = 5
```

```
continue 전: i = 7
```

```
i = 7, 이걸 출력됨
```

continue를 이용한 반복

```
// ContinueInFor
int i;
for (i = 3; i < 7; i++) {
    System.out.printf("if 전: i = %d\n", i);
    if (i < 7) {
        i += 2;
        System.out.printf("continue 전: i = %d\n", i);
        continue;
        //System.out.printf("continue 실행 후 코드: i = %d\n", i);
    }
    System.out.println("이건 출력되면 안됨!");
}
System.out.printf("i = %d, 이건 출력됨", i);
```

C:\Code\java\04>java ContinueInFor

if 전: i = 3

continue 전: i = 5


if 전: i = 6

continue 전: i = 8

i = 9, 이건 출력됨

중첩 반복문과 continue문

```
반복문 {  
    ...  
    반복문 {  
        ...  
        반복문 { // (1)  
            ...  
            continue;  
            ... // (2)  
        }  
        ... // (3)  
    }  
    ...  
}
```



실습 문제 7: 정수 입력 받고 화면에 출력하되, 0이면 다시 입력 받고, 100이상이면 종료

□ 문제

- 사용자로부터 정수 한 개를 입력 받고 화면에 출력
- 0이 입력되면 값을 출력하지 않고 다시 입력 받음
- 100이상의 정수가 입력되면 프로그램 종료

□ 요구사항

- 사용자가 정수만을 입력하는 것으로 가정

반복문과 배열

□ 실습 문제 8 배열과 반복문 사용1

■ 문제

- 배열을 1.1, 2.3, 4.7, 7.5로 초기화시켜 생성하고, 배열의 크기 및 각 요소를 화면에 출력

■ 요구사항

- 반복문 사용

```
public class Array6 {  
    public static void main(String[] args) {  
        double a[] = { 1.1, 2.3, 4.7, 7.5 };  
        System.out.printf("크기: %d\n", a.length);  
        System.out.printf("a[0] = %f\n", a[0]);  
        System.out.printf("a[1] = %f\n", a[1]);  
        System.out.printf("a[2] = %f\n", a[2]);  
        System.out.printf("a[3] = %f\n", a[3]);  
    }  
}
```

반복문과 배열

□ 실습 문제 9: 배열과 반복문 사용 2

■ 문제

- double 값을 저장할 수 있는 4개짜리 배열을 먼저 생성하고, 각 요소에 1.1, 2.2, 3.3, 4.4를 저장
- 배열의 크기와 각 요소를 화면에 출력

■ 요구사항

- 값을 출력할 때 소수점 첫 째 자리까지 출력

for each문과 배열

- for each문은 배열 또는 자료구조와 반복문을 사용할 때 코드를 단순화
 - 키워드는 for 사용

- 사용법

```
for (자료형 변수_이름 : 배열_이름) {  
    명령문;  
}
```

- 배열의 요소 출력

```
double arr[] = { 1.1, 2.3, 4.7, 7.5 };
```

for each문과 배열

□ for 반복문 사용

```
jshell> for (int i = 0; i < arr.length; i++) {  
    System.out.printf("%.1f ", arr[i]);  
}  
1.1 2.3 4.7 7.5
```

□ for each 사용

```
jshell> for (double a : arr) {  
    System.out.printf("%.1f ", a);  
}  
1.1 2.3 4.7 7.5
```

for each문과 배열

□ for each문이 유용한 경우

- 인덱스 번호가 필요 없음
- 배열의 각 요소에 반복적인 작업을 함
- 배열 요소의 값을 변경하지 않고 사용만 함

□ 배열을 인덱스와 함께 출력한다면?

```
for (int i = 0; i < arr.length; index++) {  
    System.out.printf("arr[%d] = %f\n", i, arr[i]);  
}
```

□ for each

```
int index = 0;  
for (double a : arr) {  
    System.out.printf("arr[%d] = %f\n", index, a);  
    index++;  
}
```

for each문과 배열

▣ 배열 요소 수정

■ for

```
jshell> double arr[] = { 0, 0, 0, 0 };  
arr ==> double[4] { 0.0, 0.0, 0.0, 0.0 }  
  
jshell> for (int i = 0; i < arr.length; i++) {  
    arr[i] = 1.1 * (i + 1);  
}  
  
jshell> arr  
arr ==> double[4] { 1.1, 2.2, 3.3000000000000003,  
4.4 }
```

for each문과 배열

▣ 배열 요소 수정

■ for each문

```
jshell> double arr[] = { 0, 0, 0, 0 };
```

```
jshell> int index = 0;
```

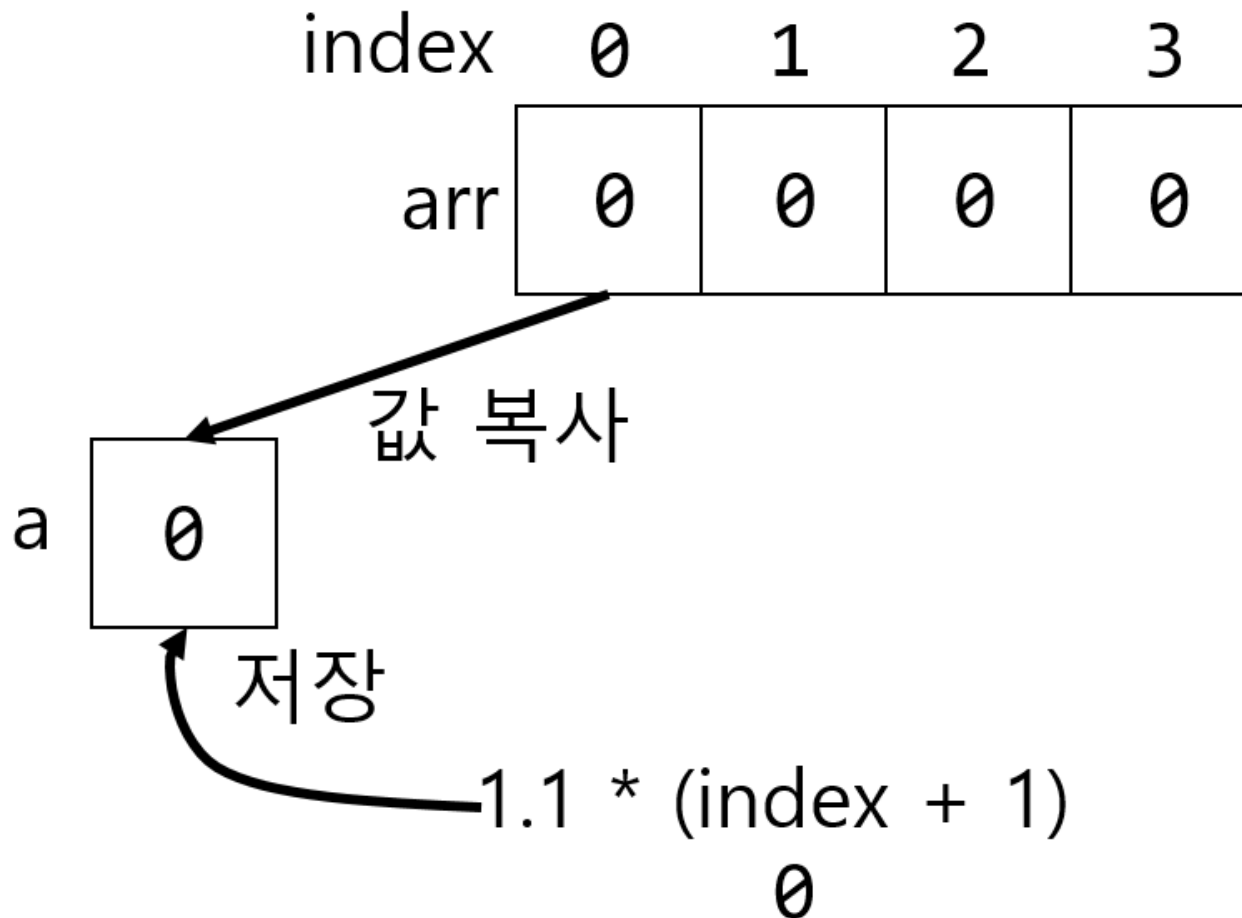
```
jshell> for (double a : arr) {  
    a = 1.1 * (index + 1);  
    index++;  
}
```

```
jshell> arr  
arr ==> double[4] { 0.0, 0.0, 0.0, 0.0 }
```


for each문과 배열

배열 요소 수정

for each문



for each문과 배열

▣ for each문을 객체 배열에 사용

```
class Hello {
    String toWhom = "world";
    Hello() {
    }
    void setWhom(String whom) {
        toWhom = whom;
    }
    void sayHello() {
        System.out.println("hello " + toWhom);
    }
}
Hello arr[] = new Hello[3];
for (int i = 0; i < 3; i++) {
    arr[i] = new Hello();
}
```

for each문과 배열

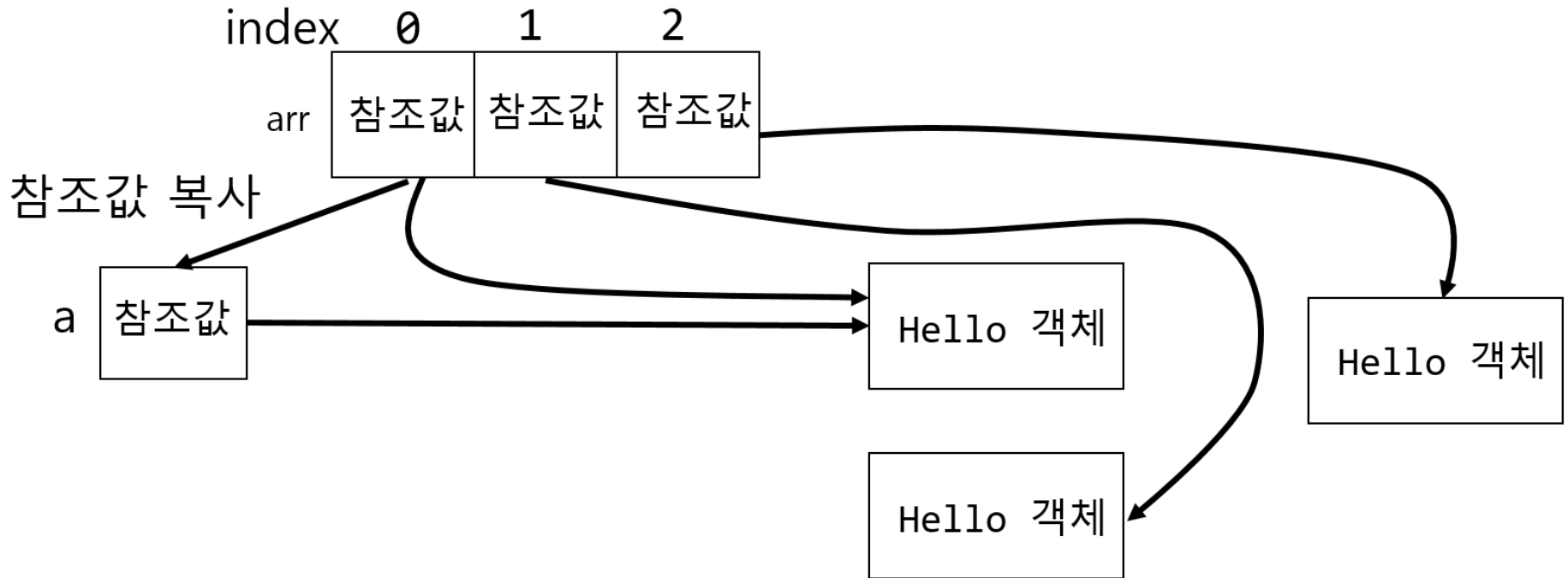
```
// 배열 요소의 클래스에 대상을 다시 지정
String whom[] = { "ycho", "jsl", "everyone" };
int index = 0;
for (Hello a : arr) {
    a.setWhom(whom[index]);
    index++;
}
```

```
// 화면에 출력
for (Hello a : arr) {
    a.sayHello();
}
```

```
// 화면에 출력
for (Hello a : arr) {
    a.sayHello();
}
hello ycho
hello jsl
hello everyone
```

for each문과 배열

■ 어떻게 객체 값이 바뀌었을까?



for each문 사용 예

```
class ForEach1 {  
    public static void main(String[] args) {  
        int arr[] = { 1, 2, 3, 4, 5 };  
  
        for (int n : arr) {  
            System.out.printf("%d ", n);  
        }  
    }  
}
```

```
C:\Code\java\04>java ForEach1  
1 2 3 4 5
```

for each문 사용 예

```
class ForEach2 {  
    public static void main(String[] args) {  
        String cities[] = { "Seoul", "New  
York", "Sydney" };  
  
        for (String city : cities) {  
            System.out.println(city);  
        }  
    }  
}
```

```
C:\Code\java\04>java ForEach2  
Seoul  
New York  
Sydney
```

실습 문제 10: 배열에서 문자열 검색

□ 문제

- "New York", "Beijing", "Seoul"을 요소로 포함한 문자열 배열을 생성하고, 그 중에서 "Seoul"이라는 단어를 가진 요소가 있으면 화면에 출력

□ 요구사항

- for, for each문을 사용
- "Seoul"이 있으면 "index번호 문자열" 형태로 출력
- 문자열이 배열에 없으면, 해당 문자열을 찾을 수 없다고 출력
- 문자열이 검색되면 더 이상 반복을 멈추고 종료

switch

- if-else if문과 비슷함
- switch문은 if-else if문으로 구현 가능
 - 반대는 안될 수도 있음
- 일반적으로 switch문이 if-else if문보다 효율적
- if-else if문에서는 다양한 종류의 조건식 사용 가능
 - 비교연산 및 논리 연산 사용 가능
 - 문자, 정수, 실수 등과 같은 기본형 뿐만 아니라 문자열, 객체를 비교하는 작업에서도 사용 가능
- switch문은 값을 비교해서 같은 경우(==연산자)에 대해 처리하는 것만 가능
 - 같음을 비교하는 조건식들이 ||로 묶이면 처리 가능
 - 문자, 정수형, enum, 문자열(자바 7이후) 가능

switch문

□ 사용 문법

```
switch (비교값) {  
    case 값1:  
        실행코드1;  
        break;  
  
    case 값2:  
    case 값3:  
        실행코드2;  
        break;  
  
    case 값4:  
        실행코드3;  
  
    case 값5:  
        실행코드4;  
        break;           // (1)  
  
    default:  
        실행코드5;  
        break;  
}
```

switch문

□ if-else if로 다시 작성

```
if (비교값 == 값1) {  
    실행코드1;  
}  
else if (비교값 == 값2 || 비교값 == 값3) {  
    실행코드2;  
}  
else if (비교값 == 값4) {  
    실행코드3;  
    실행코드4;  
}  
else if (비교값 == 값5) {  
    실행코드4;  
}  
else {  
    실행코드5;  
}
```

switch문

□ switch 문 사용 코드

- 정수값을 한 개 정해서 변수에 넣고 변수 값에 해당되는 요일을 문자열 변수에 저장 후 화면에 출력

요일 (한국어)	요일 (영어)	정수값
일요일	Sunday	1
월요일	Monday	2
화요일	Tuesday	3
수요일	Wednesday	4
목요일	Thursday	5
금요일	Friday	6
토요일	Saturday	7

switch문

□ if-else if문 사용

```
int day = 2;
String dayStr;

if (day == 1) { dayStr = "Sunday"; }
else if (day == 2) { dayStr = "Monday"; }
else if (day == 3) { dayStr = "Tuesday"; }
else if (day == 4) { dayStr = "Wednesday"; }
else if (day == 5) { dayStr = "Thursday"; }
else if (day == 6) { dayStr = "Friday"; }
else if (day == 7) { dayStr = "Saturday"; }
System.out.println(dayStr);
```

switch문

```
int day = 2;    //switch to case 2
String dayStr;
switch (day) {
    case 1:
        dayStr = "Sunday"; break;
    case 2:
        dayStr = "Monday"; break;
    case 3:
        dayStr = "Tuesday"; break;
    case 4:
        dayStr = "Wednesday"; break;
    case 5:
        dayStr = "Thursday"; break;
    case 6:
        dayStr = "Friday"; break;
    case 7:
        dayStr = "Saturday"; break;
}
System.out.println(dayStr);
```

실습 문제 11: 극장 표 값 알아보기

- 사용자로부터 원하는 종류의 극장 좌석("prime", "standard", "economy" 중 한 가지)을 입력 받고, 해당 영역의 가격을 화면에 표시
- 요구사항
 - switch문을 사용
 - 사용자는 문자열을 단어 형태로 입력
 - 세 종류 좌석 말고 다른 단어도 입력할 수 있다고 가정

enum (열거형, 상수 집합)

□ enum

- 자바 5부터 제공되기 시작한 특별한 형태의 클래스 자료형
- 열거형(enumeration type)이라고도 부름
- 연관된 상수들을 멤버로 포함
- 메소드도 포함 가능

□ enum의 용도

- 프로그래머가 기억하기 쉬운 이름을 이용해서 코드 작성
- **존재하지 않는 선택으로 인한 오류를 막을 수 있음**
- **잘못 기억하고 사용하는 오류를 줄일 수 있음**

enum 선언

- enum은 클래스의 일종이어서, 별도의 자바 파일로 저장할 수도 있고, 클래스 안 또는 밖에서 만들 수 있음

- 클래스나 변수처럼 접근 제어자 지정 가능

- enum 사용법

```
enum 열거형_이름 { 상수1, 상수2, ..., 상수N }
```

- 열거형_이름

- 연관된 상수들을 집합으로 묶고 그 내용을 대표하는 이름
- 상수1, 상수2, ..., 상수N은 상수값
 - enum 이름은 프로그래머가 자유롭게 지정
 - 영문 대문자로 표기

enum 선언

- 요일을 enum으로 표현

```
enum Day {SUNDAY, MONDAY, TUESDAY,  
          WENDESDAY, THURSDAY, FRIDAY, SATURDAY }
```

- 상수 사용은 열거형_이름.상수

```
Day.MONDAY
```

- enum 상수를 문자열로 변환은 열거형_이름.상수.
.toString()

- enum 상수를 화면에 출력하는 것은
System.out.print() 종류 함수 사용

- System.out.printf()에서는 문자열 서식인 %s를 사용

enum과 문자열, 화면 출력

```
jshell> enum Day { SUNDAY, MONDAY, TUESDAY,  
                  WEDNESDAY, THURSDAY, FRIDAY, SATURDAY }
```

```
jshell> String monday = Day.MONDAY.toString();
```

```
jshell> monday;  
monday ==> "MONDAY"
```

```
jshell> System.out.println(Day.MONDAY);  
MONDAY
```

```
jshell> System.out.printf("Monday 값 = %s\n",  
                          Day.MONDAY);
```

```
Monday 값 = MONDAY
```

enum 변수

□ enum 변수 선언

```
열거형_이름 변수_이름;
```

□ 사용 예

```
Day day = Day.SUNDAY;
```

```
jshell> enum Day { SUNDAY, MONDAY, TUESDAY,  
                  WEDNESDAY, THURSDAY, FRIDAY, SATURDAY }
```

```
jshell> void printWhatDay(Day day) {  
    System.out.println("Day: " + day);  
}
```

```
jshell> printWhatDay(Day.SUNDAY);  
Day: SUNDAY
```

enum 변수

▣ 다른 자료형 값을 함수에 전달하면 오류 발생

```
jshell> printWhatDay(1);  
|   Error:  
|   incompatible types: int cannot be converted  
to Day  
|   printWhatDay(1);  
|                   ^
```

```
jshell> enum Month { JAN, FEB, MAR, APR, MAY,  
                  JUN, JUL, AUG, SEP, OCT, NOV, DEC }
```

```
jshell> printWhatDay(Month.JAN);  
|   Error:  
|   incompatible types: Month cannot be converted  
to Day  
|   printWhatDay(Month.JAN);  
|                   ^-----^
```

enum과 비교 연산

- enum 상수는 "같다" 또는 "다르다" 비교가 가능

```
jsshell> Day today = Day.SUNDAY;

jsshell> if (today == Day.MONDAY) {
        System.out.println("Today is Monday");
    }
    if (today != Day.TUESDAY) {
        System.out.println("Today is Tuesday");
    }
    else {
        System.out.println("Today is " + today);
    }
Today is SUNDAY
```

switch와 같이 쓰이는 enum

- enum은 switch문에서 case 값을 지정할 때 사용 가능
 - enum자료형이 switch문의 "비교값"에서 사용되면 case값에 상수만 사용
 - 열거형이름.상수 형태로 사용하면 오류 발생
- 사용 예

```
jshell> Day day = Day.MONDAY;  
jshell> String msg = null;  
jshell> switch (day) {  
    case MONDAY:  
    case TUESDAY:  
    case WEDNESDAY:  
    case THURSDAY:  
    case FRIDAY:  
        msg = "schooling";  
        break;
```

switch와 같이 쓰이는 enum

```
case SATURDAY:
    case SUNDAY:
        msg = "off";
        break;

    default: // 이걸 절대 발생시킬 수 없음
        msg = "what is today?";
        break;
}
```

```
jshell> System.out.println(msg);
schooling
```

enum과 for each문

- for each문과 enum 자료형의 values() 메소드를 이용하면 각 열거형 값에 대해서 특정 코드를 실행시킬 수 있음
 - 열거형의 values() 메소드는 **열거형_이름.values()** 형태로 사용하며, 해당 enum 자료형에 있는 모든 값이 포함된 배열을 반환

```
jshell> for (Day d : Day.values()) {  
    System.out.println(d);  
}
```

```
SUNDAY  
MONDAY  
TUESDAY  
WEDNESDAY  
THURSDAY  
FRIDAY  
SATURDAY
```


enum과 valueOf() 메소드

- enum의 valueOf() 메소드는 매개변수로 전달된 문자열에 해당하는 상수를 반환
- valueOf() 메소드는 입력 받은 문자열에 해당하는 값을 반환
- 사용 예

```
jshell> Day day = Day.valueOf("MONDAY");
```

```
jshell> System.out.println("day = " + day);  
day = MONDAY
```

```
jshell> day = Day.valueOf("Monday");
```

실습 문제 12: 극장 표 값 알아보기

□ 문제

- 실습 문제 11을 enum 자료형을 사용해서 다시 작성
- 사용자로부터 "PRIME", "STANDARD", "ECONOMY" 중 한 가지를 입력 받고 가격을 화면에 출력

□ 요구사항

- enum 사용
- 사용자가 입력한 문자열을 enum 형으로 바꿔서 비교하고 표 값을 출력
- 사용자는 정확하게 세 종류 좌석 중 한 가지를 대문자로 입력

enum에 멤버 변수와 함수 추가하기

- enum에 멤버 변수와 함수를 추가하면, enum 상수를 특정 값과 연동시켜 사용할 수 있음
- 예
 - Month 자료형에는 "JAN", "FEB", ..., "DEC" 등 같은 상수로 구성됨
 - 해당 상수에 대한 숫자 값을 사용하려면 if-else if문이나 switch문을 만들어서 값을 변환해야 함

enum에 멤버 변수와 함수 추가하기

- Month 자료형을 이용해서 월에 해당되는 숫자값을 뽑아주는 switch 문의 일부 코드

```
Month month = Month.FEB;  
int monthNum = 0;  
  
switch (month) {  
    case JAN:  
        monthNum = 1;  
        break;  
  
    case FEB:  
        monthNum = 2;  
        break;  
  
    ...  
  
    case DEC:  
        monthNum = 12;  
        break;  
}
```

enum에 멤버 변수와 함수 추가하기

- enum 자료형의 상수에 멤버 변수와 함수를 포함시킬 수 있는 방법을 제공
- enum 자료형을 선언할 때 생성자와 멤버 함수들을 포함시키면, 이들은 enum에 있는 상수의 멤버 변수와 함수가 됨

```
enum Month {  
    JAN(1), FEB(2), MAR(3), APR(4), MAY(5),  
    JUN(6), JUL(7), AUG(8), SEP(9), OCT(10),  
    NOV(11), DEC(12);  
  
    int month;  
  
    Month(int month) { this.month = month;}  
    int getMonth() { return month; }  
}
```

enum에 멤버 변수와 함수 추가하기

- Month의 상수에 포함된 메소드는
열거형_이름.상수.메소드() 형태로 호출
- Month.SEP이 나타내는 월의 숫자값을 화면에 출력하는 코드를 보임

```
System.out.printf("Month.SEP = %d\n",  
                  Month.SEP.getMonth());
```

enum에 멤버 변수와 함수 추가하기

▣ Month에 숫자값과 이름을 함께 추가

```
enum Month {  
    JAN(1, "January"), FEB(2, "February"),  
    MAR(3, "March"), APR(4, "April"),  
    MAY(5, "May"), JUN(6, "June"),  
    JUL(7, "July"), AUG(8, "August"),  
    SEP(9, "September"), OCT(10, "October"),  
    NOV(11, "November"), DEC(12, "December");  
}
```

```
int month;  
String name;
```

```
Month(int month, String name) {  
    this.month = month;  
    this.name = name;  
}
```

```
int getMonth() {
```

enum에 멤버 변수와 함수 추가하기

- Month에 숫자값과 이름을 함께 추가

```
int getMonth() {  
    return month;  
}  
  
String getName() {  
    return name;  
}  
}
```

- for each문을 이용해서 각 월에 해당되는 숫자값과 이름을 화면에 출력하는 코드를 작성

enum에 멤버 변수와 함수 추가하기

```
for (Month month : Month.values()) {  
    System.out.printf("Value: %s, Num: %d, Name:  
%s\n", month, month.getMonth(), month.getName());  
}
```

Value: JAN, Num: 1, Name: January

Value: FEB, Num: 2, Name: February

Value: MAR, Num: 3, Name: March

Value: APR, Num: 4, Name: April

Value: MAY, Num: 5, Name: May

Value: JUN, Num: 6, Name: June

Value: JUL, Num: 7, Name: July

Value: AUG, Num: 8, Name: August

Value: SEP, Num: 9, Name: September

Value: OCT, Num: 10, Name: October

Value: NOV, Num: 11, Name: November

Value: DEC, Num: 12, Name: December

실습 문제 13: 교통 신호등을 enum 클래스로 표현하기

□ 문제

- 세 가지 색상(빨강(RED), 주황(AMBER), 초록(GREEN))에 해당되는 교통 신호등을 나타내는 enum을 작성할 것
- 해당 색상들이 켜져서 유지되는 시간을 각 색상 정보와 함께 저장
- 클래스를 생성한 후에는 각 색상의 이름과 켜져 있는 시간 정보를 각 요소별로 화면에 출력

□ 요구사항

- 각 색상별 신호등이 켜져 있는 시간은 정수로 표현(초 단위)
- 빨강색과 초록색은 30초
- 주황색은 10초간 유지