

COMP217

Java Programming

Spring 2018

Week 2

Expressions, Decisions, Loops

Goals

- 이번주에 배우게 될 내용 :
 - 각종 연산자(Operators)
 - 수식의 계산(Arithmetic expressions)
 - 조건문
 - `if`, `if-else`, `nested if`, `if-else if-else`, `switch-case`
 - 반복문: 조건에 따라서 여러 개의 같은 처리를 반복
 - `while`, `for`, `do...while`
 - 분기문: 지정된 영역으로 실행을 이동
 - `break`, `continue`
- 파워 자바 5장 (변수, 연산자, 수식)
- 파워 자바 6장 (선택과 반복)

Expression

Operator / Operand

Arithmetic operators / expressions

Relational operators / expressions

operator	precedence
in/decrement, postfix	expr++ expr--
unary	++expr --expr +expr ~ !
multiplicative	* / %
addition/subtraction	+ -
bit shift	<< >>> >>
relational	< > <= >= instanceof
equivalence test	== !=
bitwise AND	&
bitwise XOR	^
bitwise OR	
logical AND	&&
logical OR	
conditional	? :
assignment	= += -= *= /= %= &= ^= = <<= >>= >>>=

수식

- 수식

- 프로그래밍 언어에서 연산자와 피연산자의 조합으로 구성된 연산식
- 표현식은 항상 하나의 결과 값이 있음

- 연산자와 피연산자

- 연산자(operator)
 - $+$, $-$, $*$ 기호와 같이 이미 정의된 연산을 수행하는 문자 또는 문자 조합 기호
- 피연산자(operand)
 - 연산(operation)에 참여하는 변수나 상수

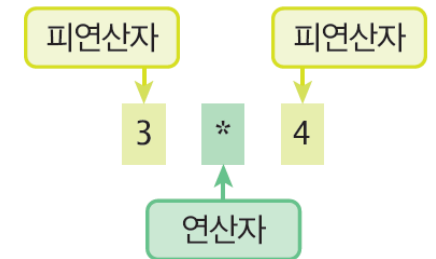


그림 3-1 • 연산자와 피연산자

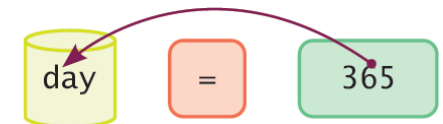
- 피연산자의 수에 따라

- 단항(unary operator), 이항(binary operator), 삼항 연산자(ternary operator)

대입연산자, 산술연산자

- 대입 연산자(assignment operator)
 - 연산자의 오른쪽 값을 왼쪽 변수에 저장하는 연산자
 - 대입 연산자의 왼쪽은 반드시 값을 저장할 수 있는 변수
 - 대입 연산자는 할당 또는 치환 연산자라고도 부름

```
int day = 365;
```



- 산술연산자 $+$, $-$, $*$, $/$, $\%$

구분	연산자	연산자 의미	예	결과	비고
부호 연산자	+	양수 부호	+3	3	
	-	음수 부호	-7	-7	
산술 연산자	+	더하기	3 + 7	10	문자열 연결 연산자 "java" + "lang"
	-	빼기	7 - 3	4	
	*	곱하기	7 * 3	21	별표 * 기호 ^{asterisk}
	/	나누기	7 / 2	3	정수와의 나누기는 결과도 정수
	%	나머지	7 % 2	1	백분율이 아님

산술 연산자(Arithmetic operators)

```
1 public class ArithmeticOperator {  
2     public static void main (String args[] ) {  
3         System.out.println(3+2);  
4         System.out.println(3-2);  
5         System.out.println(3*2);  
6         System.out.println(3/2);  
7         System.out.println(3%2);  
8         System.out.println(3.0/2.0);  
9         System.out.println(3.0/2);  
10        System.out.println(3/2.0);  
11        System.out.println(3.5%2);  
12    }  
13 }  
14
```

Problem: Converting Temperatures

- Write a program that converts a Fahrenheit degree to Celsius using the formula:

$$celsius = (\frac{5}{9})(fahrenheit - 32)$$


- Be careful with integer division

```
/* wrong */  
celsius = (5 / 9) * (fahrenheit - 32);  
// is equivalent to 0 * (fahrenheit-32);
```


```
/* correct */  
celsius = (5.0 / 9) * (fahrenheit - 32);
```

Increment and Decrement Operators, cont.

```
int i = 10;  
int newNum = 10 * i++;
```

Same effect as 

```
int i = 10;  
int newNum = 10 * (++i);
```

Same effect as 

- Example) Today is Saturday. What day is in 10 days? You can find that day is Tuesday using the following expression:

단항 연산자(Unary operators)

연산자	설명
+x	no operation
-x	reverses the sign
++x	increment x first, then use it
x++	use the value of x, and increment x
--x	decrement x first, then use it
x--	use the value of x, and decrement x

```
1 public class UnaryOperator {
2     public static void main (String args[] ) {
3         int x = 1;
4         int y = -1;
5         int z;
6
7         z = +x;      //z에 x대입 (+연산자는 의미 없음)
8         System.out.println(z);
9         z = +y;      //z에 y대입 (+연산자는 의미 없음)
10        System.out.println(z);
11        z = -x;      //z에 x의 부호를 바꿔 대입
12        System.out.println(z);
13        z = -y;      //z에 y의 부호를 바꿔 대입
14        System.out.println(z);
15        z = ++x;     //x를 하나 증가시킨 후 z에 대입
16        System.out.println(z);
17        z = y++;     //y를 z에 대입시킨 후 y 하나 증가시킴
18        System.out.println(z);
19        z = y;       //z에 y대입
20        System.out.println(z);
21    }
22 }
```

- ++
 - 변수의 값을 1 증가
- --
 - 변수의 값을 1 감소

그 외의 연산자 타입

복합대입연산자	의미
	$x = x + y$
	$x = x - y$
	$x = x * y$
	$x = x / y$
	$x = x \% y$

관계 연산자	의미
$x == y$	true if the values in x and y are equal
$x != y$	true if the values in x and y are NOT equal
$x > y$	true if x is greater than y
$x < y$	true if x is less than y
$x >= y$	true if x is greater than or equal to y
$x <= y$	true if x is less than or equal to y

- 관계 연산자
 - 결과 값은 boolean 값인 true 또는 false
- 우선 순위
 - 단항 연산자-> 산술 연산자-> 관계 연산자-> 대입연산자

관계 연산자(Relational Operator)

```
1 public class ComparisonOperator {  
2     public static void main (String args[] ) {  
3         int x, y;  
4  
5         x = 3; y = 4;  
6         System.out.println(x == y); //false  
7         System.out.println(x != y); //true  
8         System.out.println(x > y);  //false  
9         System.out.println(x < y);  //true  
10        System.out.println(x <= y); //true  
11  
12        x = 3; y = -4;  
13        System.out.println(x == y); //false  
14        System.out.println(x != y); //true  
15        System.out.println(x > y);  //true  
16        System.out.println(x < y);  //false  
17        System.out.println(x <= y); //false  
18    }  
19 }
```

논리 연산자(Logical Operators)

논리 연산자	의미
<code>x && y</code>	Logical AND: x ,y 모두 참일때 true
<code>x y</code>	Logical OR: x 또는 y가 참일때 true
<code>!x</code>	true ↔ false

```
1 public class LogicalOperator {
2     public static void main (String args[] ) {
3         int x, y;
4
5         x = 3; y = 4;
6
7         System.out.println("실행 결과 : ");
8         System.out.println((x==3) && (y==4));
9         System.out.println((x==3) && (y==7));
10        System.out.println((x==3) || (y==4));
11        System.out.println((x==5) || (y==4));
12        System.out.println(!(x==3)); // (x==3) 이 true인데 그것을 부정하여 false
13        System.out.println(!(x==5)); // (x==5) 이 false인데 그것을 부정하여 true
14    }
15 }
16
```

우선 순위

```
result = x * y % z - a / b
```

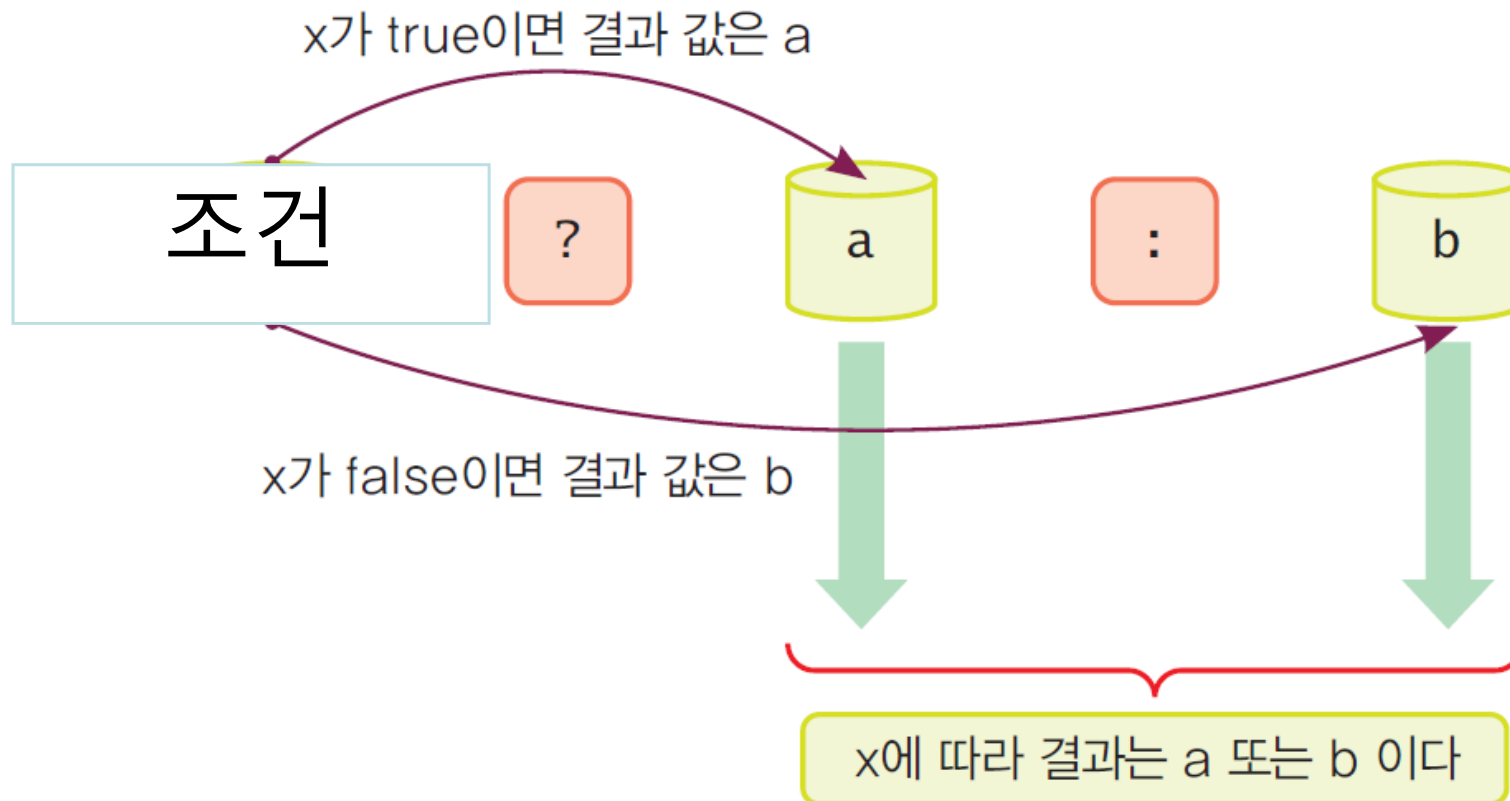
```
m = ( x + y + z ) / 3
```

```
result = x % y * z;
```

```
x = y = w = z ;
```

삼항 연산자(Ternary operator)

- 조건 연산자(conditional operator)
 - 조건의 논리 값에 따라 2개의 피연산자 중 하나가 결과 값
 - 유일한 삼항 연산자



삼항 연산자-Example

```
1 import java.util.Scanner;
2 public class TernaryOperator{
3     public static void main(String args []){
4         int a, b;
5         Scanner sc = new Scanner(System.in);
6         System.out.println("정수 두개를 입력하세요.");
7         a = sc.nextInt();
8         b = sc.nextInt();
9
10        System.out.println("max : "+((a>b)?a:b));
11
12
13
14    }
15
16 }
```

L:\USB\2015.10.20\COMP250\강의자료\셋째주\Week3>java TernaryOperator
정수 두개를 입력하세요.

5 1

max : 5

입력하신 두 수는 5, 1이며, 최대는 앞의 수입니다.

L:\USB\2015.10.20\COMP250\강의자료\셋째주\Week3>java TernaryOperator
정수 두개를 입력하세요.

2 4

max : 4

입력하신 두 수는 2, 4이며, 최대는 뒤의 수입니다.

L:\USB\2015.10.20\COMP250\강의자료\셋째주\Week3>java TernaryOperator
정수 두개를 입력하세요.

1 1

max : 1

입력하신 두 수는 1, 1이며, 두 수는 동일합니다.

비트 연산자(&, |, ^, ~)

피연산자는 int형

비트연산자	의미	Examples
~x	bitwise negation	~(0x0FFF) == 0xFFFFF000 [Q] why not 0xF000?
x & y	bitwise AND	(0x0FFF & 0xFFF0) == 0x0FF0 [Q] why not 0xFFFF0FF0?
x ^ y	bitwise XOR	(0x0FFF ^ 0xFFF0) == 0xF00F [Q] Where are first 16 bits?
x y	bitwise OR	(0x0FFF 0xFFF0) == 0xFFFF [Q] why not 0xFFFFFFFF?
x << n	n bits shift to left	0x0FFF << 4 == 0xFFF0
x >> n	n bits shift to right	0xFFF0 >> 4 == 0x0FFF

```

public class BitOperator {
    public static void main (String args[] ) {
        int x, y;

        x = 0x0fff; y = 0xfff0;
        System.out.println("실행결과");
        System.out.printf("x \t: %8x\n", x); //x : 전체 폭을 8칸으로 두고 16진수로 표현
        System.out.printf("y \t: %8x\n", y);
        System.out.printf("(x & y) : %8x\n", (x & y)); //AND
        System.out.printf("(x | y) : %8x\n", (x | y)); //OR
        System.out.printf("(x ^ y) : %8x\n", (x ^ y)); //Xor, 두 값이 같으면 0, 다르면 1
        System.out.printf("~x\t: %8x\n", ~x); //NOT
        // \t는 탭만큼 간격두기
        System.out.printf("(x << 4) : %8x\n", (x << 4)); //왼쪽으로 4비트씩 이동
        System.out.printf("(x >> 4) : %8x\n", (x >> 4)); //오른쪽으로 4비트씩 이동
    }
}

```

실행결과

```

x      :      fff
y      :     fff0
(x & y) :     ff0
(x | y) :     ffff
(x ^ y) :     f00f
~x     : fffff000
(x << 4) :     fff0
(x >> 4) :      ff

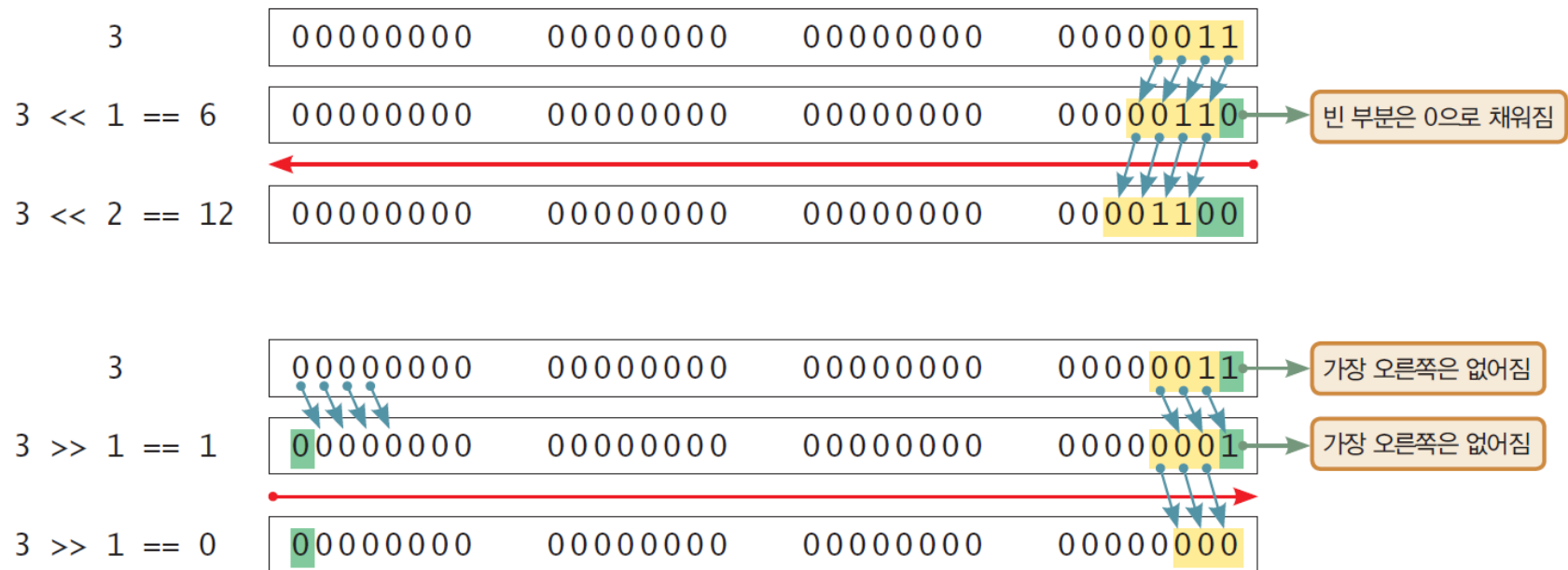
```


비트 연산자(shift)

- 비트 이동 연산자(bit shift operators)

– \gg , \ll , \ggg

연산자	이름	사용	연산 방법	새로 채워지는 비트
\gg	Signed left shift	$op1 \gg op2$	op1을 오른쪽으로 op2 비트만큼 이동	가장 왼쪽 비트인 부호 비트는 원래의 비트로
\ll	Signed right shift	$op1 \ll op2$	op1을 왼쪽으로 op2 비트만큼 이동	가장 오른쪽 비트를 모두 0으로 채움
\ggg	Unsigned right shift	$op1 \ggg op2$	op1을 오른쪽으로 op2 비트만큼 이동	가장 왼쪽 비트인 부호 비트는 모두 0으로 채워짐



중간 점검 문제

1. 다음의 각 변수의 값을 적어보라.

```
int x = 1;
int y = 1;
int a = ++x * 2; // a의 값은 _____
int b = y++ * 2; // b의 값은 _____
```

2. 다음 수식의 값을 쓰시오.

```
12/5 - 3
5 + 19%3
```

3. 다음의 수식에서 연산의 순서를 적으시오.

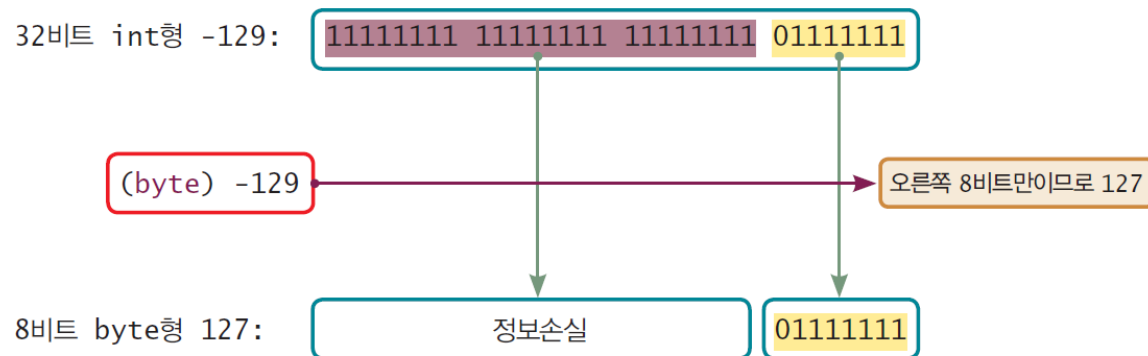
```
(1) x = y = 3 / 5 * 2 % 6;
(2) y = a * x * x + b * x + c;
```

4. 변수 y, z, a, b의 값은?

```
int x = 0xff0f;
int y = x << 4;
int z = x >> 4;
int a = x & 0xf0ff;
int b = x | 0xf0ff;
```

형변환(casting) 연산자

- 명시적 형변환
 - 실수를 정수로 변환하거나
 - 범위가 큰 정수형에서 더 작은 정수형으로 변환하려면 명시적 형변환(explicit type cast)이 필요



```
byte bt = (byte) -129;
```

- double에서 int로
 - 자동 변환이 안되므로 오류발생
 - 명시적 형변환이 필요

```
int n = 5.0 / 4.0;
```

Type mismatch: Cannot convert from double to int

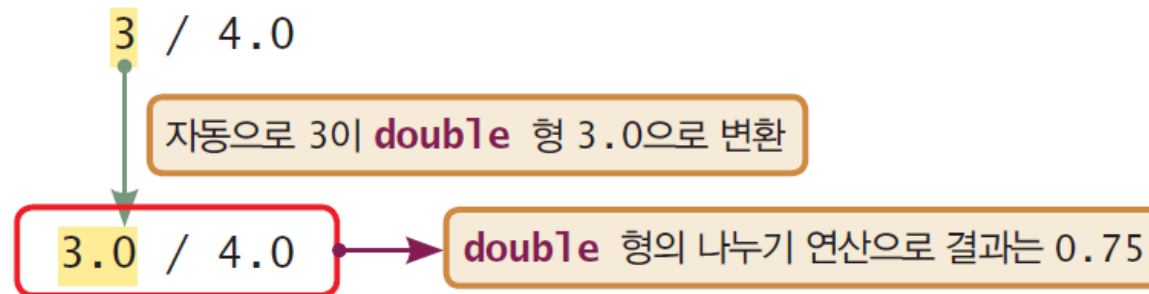
자료형 불일치 오류 발생 !

```
int n = (int) (5.0 / 4.0);
```

변수 n에 1이 저장

형변환 연산자

- 자동 형변환
 - 자바 연산은 동일한 형의 피연산자로 연산을 수행
 - 표현식 $3 / 4.0$
 - int 형 3이 자동으로 4.0인 double으로 변환
 - 표현 범위가 넓은 자료형으로 변환



형변환-Example

```
1 public class TypeCast {
2     public static void main (String args[] ) {
3         int i;
4         double f;
5         System.out.println("실행결과");
6         System.out.println("5/4 \t\t\t: "+(5/4));
7         |
8         f = 5/4;
9         System.out.println("f=5/4 \t\t\t: "+f);
10
11         /*서로 다른 자료형이 나올 경우 가장 큰 타입으로 자동 형변환*/
12
13         f = (double)5/4;
14         System.out.println("(double)5/4 \t\t: "+f);
15         f = 5/(double)4;
16         System.out.println("5/(double)4 \t\t: "+f);
17         f = (double)5/(double)4;
18         System.out.println("(double)5/(double)4 \t\t: "+f);
19         i = (int) 1.3 + (int) 1.8; //1+1
20         System.out.println("(int) 1.3 + (int) 1.8 \t: "+i);
21     }
22 }
```

실행결과

5/4	: 1
f=5/4	: 1.0
<double>5/4	: 1.25
5/<double>4	: 1.25
<double>5/<double>4	: 1.25
<int> 1.3 + <int> 1.8	: 2

산술연산 주의점

- ArithmeticException이라는 예외
 - 정수를 0으로 나누면 실행 중에 발생
- Infinity
 - 실수인 0.0으로 나누면 무한대를 의미
- 표현식 0.0/0.0
 - NaN(Not a Number)가 출력
- 자료형 byte와 short의 산술 연산
 - 모두 int로 변환되어 연산을 수행

DevideByZero.java

```
01 package operator;
02
03 public class DevideByZero {
04     public static void main(String[] args) {
05         short data1 = 32766;
06         short data2 = 1;
07         //short data3 = data1 + data2; //오류발생
08         short data3 = (short)(data1 + data2);
09         short data4 = 32766 + 1;
10         System.out.println(data3 + " " + data4);
11
12         System.out.println(0.0 / 0.0); //NaN
13         System.out.println(3 / 0.0);   //Infinity
14         System.out.println(3 / 0);     //예외발생
15     }
16 }
```

더한 결과가 short의 범주인 -32768에서 32767 사이면 오류가 발생하지 않음.

결과

32767 32767

NaN

Infinity

Exception in thread "main" java.lang.ArithmeticException: / by zero
at operator.DevideByZero.main(DevideByZero.java:14)

APPENDIX

Examples of Common Errors

Common Errors and Pitfalls

- ☞ Common Error 1: Undeclared/Uninitialized Variables and Unused Variables
- ☞ Common Error 2: Integer Overflow
- ☞ Common Error 3: Round-off Errors
- ☞ Common Error 4: Unintended Integer Division
- ☞ Common Error 5: Redundant Input Objects

- ☞ Common Pitfall 1: Redundant Input Objects

Common Error 1: Undeclared/Uninitialized Variables and Unused Variables

```
double interestRate = 0.05;
```

```
double interest = interestrate * 45;
```

Common Error 2: Integer Overflow

```
int value = 2147483647 + 1;  
// value will actually be -2147483648
```

Common Error 3: Round-off Errors

```
System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
```

```
System.out.println(1.0 - 0.9);
```

Common Error 4: Unintended Integer Division

```
int number1 = 1;  
int number2 = 2;  
double average = (number1 + number2) / 2;  
System.out.println(average);
```

(a)

```
int number1 = 1;  
int number2 = 2;  
double average = (number1 + number2) / 2.0;  
System.out.println(average);
```

(b)

Common Pitfall 1: Redundant Input Objects

```
Scanner input = new Scanner(System.in);  
System.out.print("Enter an integer: ");  
int v1 = input.nextInt();
```

```
Scanner input1 = new Scanner(System.in);  
System.out.print("Enter a double value: ");  
double v2 = input1.nextDouble();
```

LAB

- 어떤 도시가 메트로폴리스(거대도시)가 되려면 다음과 같은 2가지 조건 중의 하나를 만족하여야 한다고 가정하자.
 - 한 나라의 수도이고 인구가 100만 이상이어야 한다.
 - 연 소득이 1억 이상인 인구가 50만 이상이어야 한다.



실행결과

```
수도입니까?(수도: 1 수도아님: 0)1  
인구(단위: 백만)200  
부자의 수(단위: 백만)100  
메트로폴리스 여부: true
```

SOURCE

Hint

```
import java.util.Scanner;

public class Metropolis {
    public static void main(String[] args) {
        boolean isCapital;
        int citizens;
        int riches;

        Scanner sc = new Scanner(System.in);
        System.out.print("수도입니까?(수도: 1 수도아님: 0)");
        isCapital = (sc.nextInt() == 1) ? true : false
        System.out.print("인구(단위: 백만)");
        citizens = sc.nextInt();
        System.out.print("부자의 수(단위: 백만)");
        riches = _____();

        boolean isMetro = (isCapital && citizens >= 100)
                           || (_____);

        System.out.println("메트로폴리스 여부: " + isMetro);
    }
}
```

Code for the exercise

```
/**
 * Check if a city is a metropolitan city
 */
import java.util.Scanner;          // Built-in Scanner class

public class Metropolis {
    public static void main (String args[] ) {
        boolean isCapital, isMetropolis;
        int citizen;
        int bourgeois;

        Scanner sc = new Scanner(System.in);
        System.out.print("It the city a capital? (capital:1 non-capital:0) ");
        isCapital = (sc.nextInt() == 1);
        System.out.print("Population? (in thousands) ");
        citizen = sc.nextInt();
        System.out.print("Bourgeois? (in thousands) ");
        bourgeois = sc.nextInt();

        isMetropolis = (isCapital && citizen >= 1000)
            || (bourgeois >= 500);

        System.out.println("Metropolis: " + isMetropolis);
    }
}

/* execution example
$ javac Metropolis.java
$ java Metropolis
It the city a capital? (capital:1 non-capital:0) 1
Population? (in thousands) 2000
Bourgeois? (in thousands) 1000
Metropolis: true
$ java Metropolis
It the city a capital? (capital:1 non-capital:0) 0
Population? (in thousands) 1000
Bourgeois? (in thousands) 500
Metropolis: true
*/
```


원 넓이 구하는 클래스 작성

- 클래스명 CircleArea
- 사용자로부터 반지름(실수) 입력받기
 - Scanner클래스의 nextDouble() 메서드 사용
- 예)

```
Scanner sc = new Scanner(System.in);  
double d = sc.nextDouble();  
int i = sc.nextInt();
```
- 원 넓이(실수) 계산 : $\pi \times r^2$
- 원 넓이 출력하기
 - 소수점 아래 둘째 자리까지
 - printf 메서드 사용

원 넓이 구하는 클래스

```
1 import java.util.Scanner;    //Scanner클래스를 사용할 것을 알림
2                               //사용자로부터 값을 입력받을 수 있도록
                               //도와주는 클래스
3 public class CircleArea{     //클래스 헤더 : 클래스 이름 알림
4                               //클래스 이름은 파일명과 일치시킴
5     public static void main(String args [] ){ //메인메서드 헤더
6         final float PI = 3.141592f;    //상수 (final 키워드가
        없으면 변수라서 PI=PI*2;와 같이 값을 바꿀 수 있지만
        상수라서 바꿀수 없음)
7         double r; //8바이트 실수형으로 반지름 선언
8         double area;    //넓이를 저장할 변수 선언
9         Scanner s = new Scanner(System.in); //스캐너 클래스 생성
10        System.out.print("반지름 : ");
11        r = s.nextDouble();    //사용자가 입력한 값을 반지름
        변수에 대입
12        area = PI * r * r;    //넓이 계산
13
14        System.out.printf("넓이 : %.2f cm^2\n", area); //넓이를
        출력하는데 소수점 아래 2째 자리까지만 출력
15    }
16 }
```

Decisions and Loops

* 난수 생성

- Math class: `java.lang.Math`
 - `random()`: 0이상 1미만의 임의의 double형 실수를 반환
 - 정수형 난수 생성은??
- Random class: `java.util.Random`
 - `nextDouble()`
 - `nextInt()`
 - `nextInt(int bound)`

점검 문제

1. 변수 n의 값이 100보다 크거나 같으면 "large", 100보다 작으면 "small"을 출력하는 if-else 문을 작성하라.

2. k의 값이 각각 3, 0, -1인 경우에 다음의 코드에 의하여 생성되는 출력은 무엇인가?

```
if( k == 0 )  
    System.out.println("A");  
else if( k > 3 )  
    System.out.println("B");  
else  
    System.out.println("C");
```

3. 컵의 사이즈를 받아서 100ml미만은 small, 100ml이상 200ml미만은 medium, 200ml 이상은 large라고 출력하는 연속적인 if-else 문을 작성하시오.

숫자 추측 게임

- 사용자가 숫자(dap)을 맞추면 성공하는 게임
 - 사용자로부터 숫자 입력 받음
 - int userInput
 - 사용자가 입력한 값이 dap이면 "성공" 출력
 - 단, dap은 1이상 100이하의 임의의 정수(난수)
 - 아니라면 dap과 userInput 의 대소 비교 결과 출력

```
C:\Users\helloworld\Desktop>
맞춰 보세요 : 50
정답보다 작네요. 정답 : 99

C:\Users\helloworld\Desktop>
맞춰 보세요 : 100
정답보다 크네요. 정답 : 99

C:\Users\helloworld\Desktop>
맞춰 보세요 : 99
99 : 정답입니다!!
```

입력한 달에 존재하는 날의 수 출력 코드

- 31일
 - 1, 3, 5, 7, 8, 10, 12월
- 30일
 - 4, 6, 9, 11월
- 2월은??
 - 윤년 여부에 따라 다름
 - 윤년 : 29일
 - 년도가 4의 배수 but 100의 배수는 아님
 - 년도가 400의 배수
 - 평년 : 28일

```
int year, month;
scanf("%d %d", &year, &month);
if (month < 1 || month > 12)
    return 0;
if (month == 2)
    if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0)
        printf("29\n");
    else
        printf("28\n");
else
    printf("%d\n", days[month]);
```

2017년 3월은 31일까지 있습니다.

최대 공약수 예제

- 사용자로부터 두 정수 입력 받음
- 유클리드 알고리즘 사용하여 최대 공약수 계산

```
① 두 수 가운데 큰 수를 x, 작은 수를 y라 한다.  
② y가 0이면 공약수는 x와 같다.  
③  $r \leftarrow x \% y$   
④  $x \leftarrow y$   
⑤  $y \leftarrow r$   
⑥ 단계 ②로 되돌아간다.
```

– 입력 받은 두 수 중

- 큰 수 : a
- 작은 수 : b

```
정수 1 : 15  
정수 2 : 12  
최대공약수는 3입니다.
```


숫자 추측 게임 예제

0~100까지의 정수를 입력하여 답을 맞춰보세요: 78
정답은 좀 더 큰 수 입니다.
0~100까지의 정수를 입력하여 답을 맞춰보세요: 90
정답은 좀 더 작은 수 입니다.
0~100까지의 정수를 입력하여 답을 맞춰보세요: 100
정답은 좀 더 작은 수 입니다.
0~100까지의 정수를 입력하여 답을 맞춰보세요: 79
축하합니다. 79은 정답입니다. 4번만에 맞추셨습니다.

- 추측할 숫자 고정
 - `int jungdap = 79`
- 사용자가 숫자(`jungdap`)을 맞추면 성공하는 게임
 - 사용자로부터 숫자 입력 받음
 - `int userInput`
 - 사용자가 입력한 값이 `jungdap`이면 "성공" 출력
 - 맞출때까지의 시도 횟수 출력
 - `int tries`
 - 아니라면 `userInput`과 `jungdap`의 대소를 비교하여 출력
 - 사용자가 `jungdap`을 맞출때까지 반복
 - 일단 한번은 무조건 실행해야 하므로
 - `while`문 대신
 - `do-while` 사용

기본 반복문

- 1~10까지 출력하기
- 구구단 출력하기
- 1~10까지의 합 구하기
- n! 구하기(overflow 발생시점 파악)
- 약수 출력하기
- 패턴 출력

직각 삼각형을 출력합니다.
단 : 5

```

*
**
***
****
*****
  
```

직각 삼각형을 출력합니다.
단 : 10

```

      *
     **
    ***
   ****
  *****
 *****
*****
*****
*****
*****
*****
  
```

직각 삼각형을 출력합니다.
단 : 10

```

*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
  
```

피라미드를 출력합니다.
단 : 10

```

      *
     ***
    *****
   *****
  *****
 *****
*****
*****
*****
*****
*****
  
```

직각 삼각형을 출력합니다.
단 : 10

```

*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
  
```

C:\Users\Administr
출력할 단 : 3

```

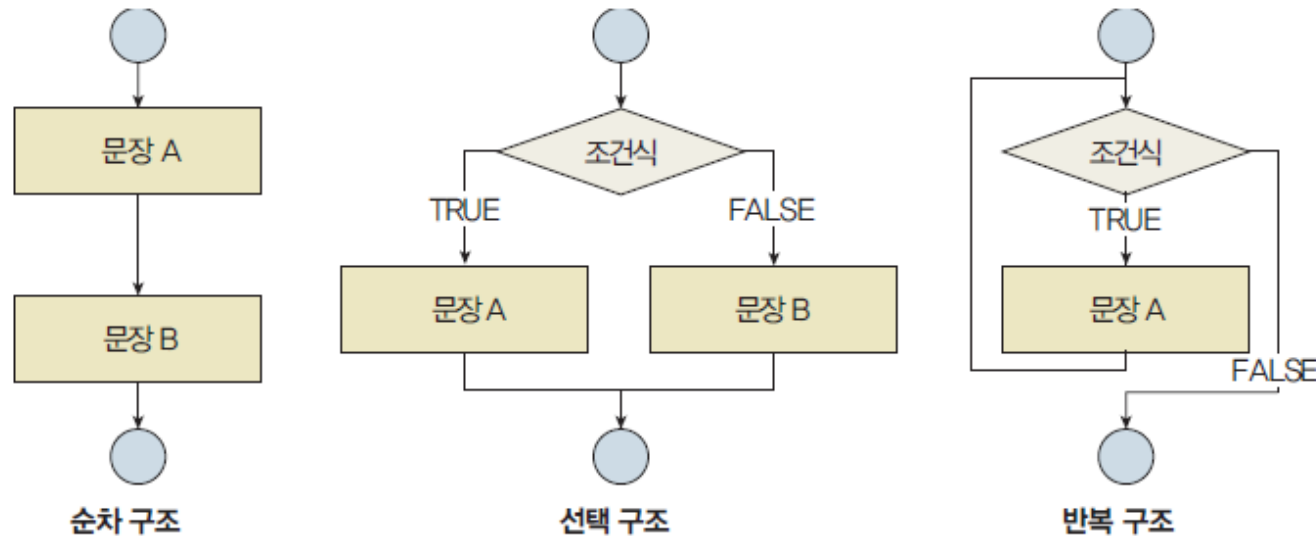
----3단----
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
  
```

```

1의 약수는 1
2의 약수는 1 2
3의 약수는 1 3
4의 약수는 1 2 4
5의 약수는 1 5
6의 약수는 1 2 3 6
7의 약수는 1 7
8의 약수는 1 2 4 8
9의 약수는 1 3 9
10의 약수는 1 2 5 10
  
```

Decisions

프로그램 문장의 실행 구조



- 제어문 : 문장이 실행되는 순서에 영향을 주는 문장
- 조건문 : 조건에 따라서 여러 개의 실행 경로 가운데 하나를 선택
 - if, if-else, if-else if, nested if, switch-case
- 반복문 : 조건에 따라서 여러 개의 같은 처리를 반복
 - for, while, do-while
- 분기처리 : 지정된 영역으로 실행을 이동
 - continue, break

조건문(간단한 if 문)

– 기온이 30도 이상이면 날씨가 더운 편이다.

- if(조건식) 문장; or {문장1; ... 문장n;}

```
int degree = 30;
```

```
if (30 <= degree)
    System.out.println("날씨가 덥습니다.");
```

날씨가 덥습니다.
날씨 예보였습니다.

```
System.out.println("날씨 예보였습니다.");
```

– 점수가 60점 이상이면 합격이고 그렇지 않으면 불합격이다.

- if(조건식) 문장; or {문장1;..문장n;}
- else 문장'; or {문장1';..문장n';}

```
if (grade >= 60)
    System.out.println("합격");
else
    System.out.println("불합격");
```

```
if ( grade >= 60 )
{
    System.out.println("합격입니다.");
    System.out.println("장학금도 받을 수 있습니다.");
}
else {
    System.out.println("불합격입니다.");
    System.out.println("장학금을 받을 수 없습니다.");
}
```

숫자 추측 게임

- 사용자가 숫자(dap)을 맞추면 성공하는 게임
 - 사용자로부터 숫자 입력 받음
 - `int userInput`
 - 사용자가 입력한 값이 dap(99로 고정)이면 "성공" 출력
 - 아니면 dap과 userInput 의 대소 비교 결과 출력

```
C:\Users\helloworld\Desktop>
맞춰 보세요 : 50
정답보다 작네요. 정답 : 99

C:\Users\helloworld\Desktop>
맞춰 보세요 : 100
정답보다 크네요. 정답 : 99

C:\Users\helloworld\Desktop>
맞춰 보세요 : 99
99 : 정답입니다!!
```

숫자 추측 게임 소스??

```

1  import java.util.Scanner;
2
3  public class NG{
4      public static void main(String args []){
5
6          int dap = 99;
7          int userInput;
8          Scanner s = new Scanner(System.in);
9
10         System.out.print("맞춰 보세요 : ");
11         userInput = s.nextInt();
12         if(userInput > dap)
13             System.out.println("정답보다 크네요. 정답 : "+dap);
14         if(userInput < dap)
15             System.out.println("정답보다 작네요. 정답 : "+dap);
16         if(userInput == dap)      System.out.println(userInput+" :
정답입니다!!");
17
18     }
19 }

```

숫자 추측 게임 소스??

```
1 import java.util.Scanner;
2
3 public class NG{
4     public static void main(String args []){
5
6         int dap = 99;
7         int userInput;
8         Scanner s = new Scanner(System.in);
9
10        System.out.print("맞춰 보세요 : ");
11        userInput = s.nextInt();
12        if(userInput > dap)
13            System.out.println("정답보다 크네요. 정답 : "+dap);
14        else {
15            if(userInput < dap)
16                System.out.println("정답보다 작네요. 정답 : "+dap);
17            else
18                System.out.println(userInput+" : 정답입니다!!");
19        }
20    }
```


숫자 추측 게임 소스??

```
1  import java.util.Scanner;
2
3  public class NG{
4      public static void main(String args []){
5
6          int dap = 99;
7          int userInput;
8          Scanner s = new Scanner(System.in);
9
10         System.out.print("맞춰 보세요 : ");
11         userInput = s.nextInt();
12         if(userInput > dap)
13             System.out.println("정답보다 크네요. 정답 : "+dap);
14         else
15             if(userInput < dap)
16                 System.out.println("정답보다 작네요. 정답 : "+dap);
17             else System.out.println(userInput+" : 정답입니다!!");
18         }
19     }
20 }
```

숫자 추측 게임 소스

```
1  import java.util.Scanner;
2
3  public class NG{
4      public static void main(String args []){
5          int dap = 99;
6          int userInput;
7          Scanner s = new Scanner(System.in);
8
9          System.out.print("맞춰 보세요 : ");
10         userInput = s.nextInt();
11         if(userInput > dap)
12             System.out.println("정답보다 크네요. 정답 : "+dap);
13         else if(userInput < dap)
14             System.out.println("정답보다 작네요. 정답 : "+dap);
15         else      System.out.println(userInput+" : 정답입니다!!");
16     }
17 }
```

Note

```
if i > 0 {  
    System.out.println("i is positive");  
}
```

(a) Wrong

```
if (i > 0) {  
    System.out.println("i is positive");  
}
```

(b) Correct

```
if (i > 0) {  
    System.out.println("i is positive");  
}
```

(a)

Equivalent

```
if (i > 0)  
    System.out.println("i is positive");
```

(b)

Compound Statements

- Grouping single statements with braces ('{ }')
 - Also called a 'block'
 - Replaceable with a single statement

```
if ( grade >= 60 )
{
    System.out.println("Passed");
    System.out.println("Scholarship");
}
else
{
    System.out.println("Failed");
    System.out.println("No Scholarship");
}
```

[Q] What if no braces?

```
if ( grade >= 60 )
    System.out.println("Passed");
    System.out.println("Scholarship");
else
    System.out.println("Failed");
    System.out.println(
        "No Scholarship");
```

➔ Error!!

임금 계산 예제

- 8시간까지는 시간당 5,000원
- 8시간 초과분에 대해서는 1.5배 지급

```
일한 시간 : 10
임금 : 55000
C:\Users\Adminis
일한 시간 : 8
임금 : 40000
.....
```

```
1 public class Pay{
2     public static void main(String args[]){
3         final int RATE = 5_000; //상수라서 변수명 모두 대문자(Not mandatory)
4         int time;           //일한 시간
5         int pay;            //급여 표현
6         java.util.Scanner sc = new java.util.Scanner(System.in);
7         //사용자로 부터 값을 입력받을 수 있는 Scanner 클래스 사용할 것이기 때문에
8         Scanner 클래스 선언
9         //import java.util.Scanner; 를 쓰지 않는다면 위와 같이 쓸수도 있음
10        System.out.print("일한 시간 : ");
11        time = sc.nextInt();           //사용자가 입력한 값을 일한 시간에 대입
12
13        if(time>8) pay = 8*RATE + (int)((time-8)*RATE*1.5);
14        //일한 시간이 8 시간을 초과한 경우 pay 계산식
15        else pay = time*RATE;
16        //일한 시간이 8 시간을 초과하지 않는 경우 pay 계산식
17        System.out.print("임금 : " + pay);
18        //여기에서 +는 연결해서 출력하라는 의미로 사용됨
19    }
}
```

Nested if

- if 문에 다시 if 문이 포함
- 예제 : 자동차 면허 필기 합격 여부
 - 1급의 경우 : 70점 이상
 - 2급의 경우 : 60점 이상

```
import java.util.Scanner;
```

```
public class NestedIf {
```

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

```
        Scanner in = new Scanner(System.in);
```

```
        System.out.print("면허시험 종류선택 (1[1종] 또는 2[2종] 입력) >> ");
```

```
        int type = in.nextInt();
```

```
        System.out.print("필기 면허시험 점수 입력 >> ");
```

```
        int score = in.nextInt();
```

```
        if (type == 1) {
```

```
            if (score >= 70)
```

```
                System.out.println("1종 면허 시험 합격");
```

```
            else
```

```
                System.out.println("1종 면허 시험 불합격");
```

```
        } else if (type == 2) {
```

```
            if (score >= 60)
```

```
                System.out.println("2종 면허 시험 합격");
```

```
            else
```

```
                System.out.println("2종 면허 시험 불합격");
```

```
        }
```

```
    }
```

```
}
```

결과 1

면허시험 종류선택 (1[1종] 또는 2[2종] 입력) >> 1

필기 면허시험 점수 입력 >> 70

1종 면허 시험 합격

결과 2

면허시험 종류선택 (1[1종] 또는 2[2종] 입력) >> 2

필기 면허시험 점수 입력 >> 50

2종 면허 시험 불합격

외부 if에서 면허 종류를 먼저 구분한다.

내부에서 점수로 합격 판정을 수행한다.

Nested if(cont.)

- 만약 c1이 참이라면 statement 실행
- 만약 c1이 참인데, c2도 참이라면 statement2 실행
- 만약 c1이 참이고, c2도 참이라면 statement2 실행하고
c1은 참이지만, c2는 거짓이라면 statement3 실행

Cascaded if-else Statements

```
if (score >= 90.0)
    System.out.print("A");
else
    if (score >= 80.0)
        System.out.print("B");
    else
        if (score >= 70.0)
            System.out.print("C");
        else
            if (score >= 60.0)
                System.out.print("D");
            else
                System.out.print("F");
```

(a)

Equivalent

```
if (score >= 90.0)
    System.out.print("A");
else if (score >= 80.0)
    System.out.print("B");
else if (score >= 70.0)
    System.out.print("C");
else if (score >= 60.0)
    System.out.print("D");
else
    System.out.print("F");
```

This is better

(b)

Trace if-else statement

(1) Suppose score is 70.0

(2) The condition is false

```
if (score >= 90.0)
```

```
    System.out.print("A");
```

```
else if (score >= 80.0)
```

```
    System.out.print("B");
```

```
else if (score >= 70.0)
```

```
    System.out.print("C");
```

```
else if (score >= 60.0)
```

```
    System.out.print("D");
```

```
else
```

```
    System.out.print("F");
```

(3) The condition is false

(4) The condition is **true**

(5) Grade is C

(6) Exit the if statement

If문을 통해 변수 초기화

- 사용자로부터 임의의 정수(int input)를 입력 받아 그 값에 따라 int형 변수 b를 아래와 같이 초기화 하시오.
 - 양수이면 b에 1을,
 - 음수이면 b에 -1을,
 - 0이면 b에 0을 대입.

소득세 구하기 예제

- 근로 소득세 계산 방법
 - 과세 표준 금액 $\leq 1,000$
 - 과세 표준 금액 * 0.09
 - $1,000 < \text{과세 표준 금액} \leq 4,000$
 - 1,000만원까지는 위와 같이 과세 표준 금액 * 0.09
 - 1,000만원 초과분에 대해서는 과세 표준 금액 * 0.18
 - $4,000 < \text{과세 표준 금액} \leq 8,000$
 - 4,000만원까지는 위와 같이 계산
 - 4,000만원 초과분에 대해서는 과세 표준 금액 * 0.27
 - 과세 표준 금액 $> 8,000$
 - 8,000만원까지는 위와 같이 계산
 - 8,000만원 초과분에 대해서는 과세 표준 금액 * 0.36

실행결과

과세 표준 금액을 입력하시오: 3000
소득세는 540입니다.

소득세 구하기 코드

```
1 import java.util.Scanner;    //Scanner 클래스를 사용할 것을 알림
2 public class Tax{           //Tax 클래스 선언
3     public static void main(String args []){
4         Scanner sc = new Scanner(System.in);    //Scanner 클래스 sc 생성
5         int gp;    //과세 표준 금액 정수형으로 선언
6         System.out.print("과세 표준 금액 : ");
7         gp = sc.nextInt(); //사용자로부터 정수입력받아 gp 변수에 저장
8         int tax; //세금도 돈! 실수형은 불가능
9
10        if(gp<=1000) tax = (int)(gp*0.09); //과세 표준이 1000이하인 경우 세금 계산
11        else if(gp<=4000) tax = (int)(1000*.09 + (gp-1000)*.18); //과세 표준이 1000초과
12        //4000이하인 경우 세금 계산
13        else if(gp<=8000) tax = (int)(1000*.09 + 3000*.18 + (gp-4000)*.27); //과세 표준이
14        //4000초과 8000이하인 경우 세금 계산
15        else tax = (int)(1000*.09 + 3000*.18 + 4000*.27 + (gp-8000)*.36);
16        //과세 표준이 8000초과인 경우 세금 계산
17
18        System.out.print("세금은 "+tax+"입니다.\n");
19    }
20 }
```

Note

The else clause matches the most recent if clause in the same block.

```
int i = 1, j = 2, k = 3;
if (i > j)
    if (i > k)
        System.out.println("A");
else
    System.out.println("B");
```

(a)

Equivalent

This is better
with correct
indentation

```
int i = 1, j = 2, k = 3;
if (i > j)
    if (i > k)
        System.out.println("A");
    else
        System.out.println("B");
```

(b)

Note(cont.)

Nothing is printed from the preceding statement. To force the else clause to match the first if clause, you must add a pair of **braces**:

```
int i = 1;
int j = 2;
int k = 3;
if (i > j) {
    if (i > k)
        System.out.println("A");
}
else
    System.out.println("B");
```

This statement prints B.

Common Errors

Adding a semicolon at the end of an if clause is a common mistake.

```
if (radius >= 0);  
{  
    area = radius*radius*PI;  
    System.out.println(  
        "The area for the circle of radius " +  
        radius + " is " + area);  
}
```

Wrong

This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error.

This error often occurs when you use the next-line block style.

TIP

```
if (number % 2 == 0)
    even = true;
else
    even = false;
```

(a)

```
boolean even
    = number % 2 == 0;
```

(b)

```
if (even == true)
    System.out.println(
        "It is even.");
```

(a)

```
if (even)
    System.out.println(
        "It is even.");
```

(b)

주사위 결과 출력

- 주사위를 던진 값(`int` `dice`)이 입력되면, 아래와 같이 출력
 - 6일 경우 : 수
 - 5일 경우 : 우
 - 4일 경우 : 미
 - 3일 경우 : 양
 - 2일 경우 : 가
 - 1일 경우 : 가
- if문??
- switch 문

```
switch(dice) {  
    case 6:  
        System.out.println("수");  
  
    case 5:  
        System.out.println("우");  
  
    case 4:  
        System.out.println("미");  
  
    case 3:  
        System.out.println("양");  
  
    case 1: case 2:  
        System.out.println("가");  
  
    default:  
        System.out.println("주사위 값이 아닙니다. ");  
}
```

switch 문

- 여러 가지 경우 중에서 하나를 선택하는데 사용

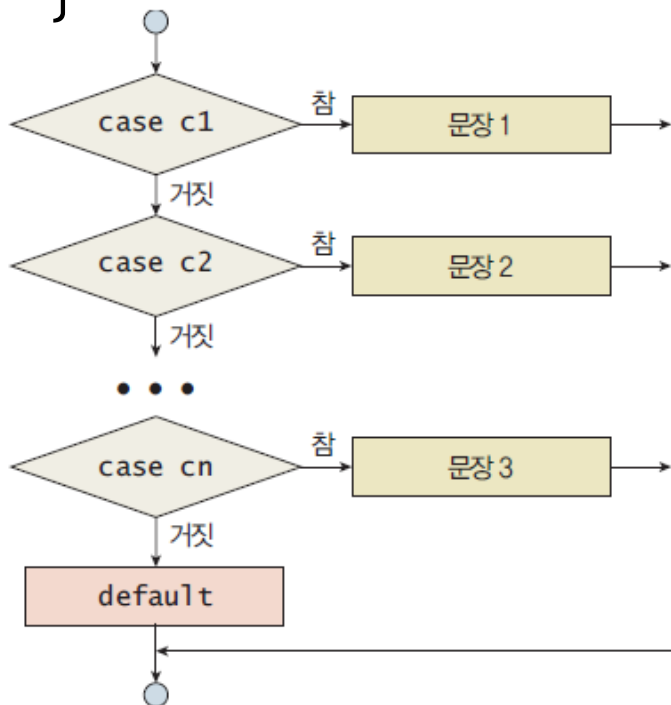
switch(변수) { 정수형변수, char, String

case 1:

...

default:

}



```
import java.util.*;
public class SwitchExample {
    public static void main(String[] args) {
        int number;

        Scanner scan = new Scanner(System.in);
        System.out.print("숫자를 입력하시오: ");
        number = scan.nextInt();
        switch (number) {
            case 0:
                System.out.println("없음");
                break;
            case 1:
                System.out.println("하나");
                break;
            case 2:
                System.out.println("둘");
                break;
            default:
                System.out.println("많음");
                break;
        }
    }
}
```

사용자가 1을 입력하였다고 가정

이 문장이 실행되어서 콘솔에 "하나"가 출력된다.

Switch 예제

StringSwitch.java

```
01 public class StringSwitch {
02     public static void main(String[] args) {
03         String month = "february";
04
05         int monthNumber;
06         switch (month) {
07             case "january":
08                 monthNumber = 1;
09                 break;
10             case "february":
11                 monthNumber = 2;
12                 break;
13             case "march":
14                 monthNumber = 3;
15                 break;
16             default:
17                 monthNumber = 0;
18                 break;
19         }
20         System.out.println(monthNumber);
21     }
22 }
```

JDK 7부터 문자열을 Switch 문에 사용할 수 있다.

실행결과

2

입력한 달에 존재하는 날의 수 출력 코드

- 31일
 - 1, 3, 5, 7, 8, 10, 12월
- 30일
 - 4, 6, 9, 11월
- 2월은??
 - 윤년 여부에 따라 다름
 - 윤년 : 29일
 - 년도가 4의 배수 but 100의 배수는 아님
 - 년도가 400의 배수
 - 평년 : 28일

```
int year, month;
scanf("%d %d", &year, &month);
if (month < 1 || month > 12)
    return 0;
if (month == 2)
    if (isLeapYear(year))
        printf("29\n");
    else
        printf("28\n");
else
    printf("%d\n", days[month]);
```

2017년 3월은 31일까지 있습니다.

입력한 달에 존재하는 날의 수 출력 코드

```
1  import java.util.Scanner;
2  public class DaysInMonth{
3      public static void main(String [] args){
4          byte month;      //몇월인지 저장할 변수 month 선언 (byte형으로 충분히 표현 가능)
5          int year;        //몇년인지 저장할 변수 선언 (2월의 날자수 계산에 필요)
6          int days=31;     //31일이 있는 달이 많으므로 변수 선언과 함께 31로 초기화
7          Scanner s = new Scanner(System.in);
8
9          System.out.print("몇 년 : ");
10         year = s.nextInt();    //사용자로부터 몇년인지 입력 받음
11         System.out.print("몇 월 : ");
12         month = s.nextByte();  //사용자로부터 몇월인지 입력 받음
13
14         switch(month){
15             case 1: case 3: case 5: case 7: case 8: case 10: case 12:
16                 break; //이 달들은 31일이 있는데, 이미 days를 31으로 정의해두었기 때문에 days는 손
                        //들 필요 없고 break걸어서 switch문 빠져 나가면 됨
17             case 4: case 6: case 9: case 11:
18                 days = 30; //이 달들은 30일이 있으므로 변수 days를 30으로 바꿔주고
19                 break;    //break걸어서 switch문 빠져 나가면 됨
20             case 2: //2월인 경우 윤년인 경우와 평년인 경우로 다시 나뉨
21                 if(((year%4==0)&&(year%100!=0)) || (year%400==0)) days = 29; //윤년인 경우
22                 else days = 28;    //윤년이 아닌 경우
23                 break;
24             default:
25                 System.out.println("그런 달은 없습니다.");    //이외의 달을 입력한 경우
26                 break;
27         }
28         if(month>0&&month<13)    //정상적인 달을 입력한 경우에만 days 출력
29             System.out.print(year + "년 " + month + "월은 " + days + "일까지 있습니다.\n");
30     }
```

중간 점검 문제

1. case 절에서 break 문을 생략하면 어떻게 되는가?
2. 변수 fruit의 값이 각각 1, 2, 5일 때, 다음의 코드의 출력을 쓰시오.

```
switch(fruit) {  
    case 1: System.out.println("사과");  
        break;  
    case 2: System.out.println("배");  
    case 3: System.out.println("바나나");  
        break;  
    default: System.out.println("과일");  
}
```

배수 출력

- 정수 하나 입력 받아 아래와 같이 출력하는 프로그램
 - i 입력한 수는 2와 3의 공배수입니다.
 - ii 입력한 수는 2 또는 3의 배수입니다.
 - iii 입력한 수는 2의 배수이거나 3의 배수인데 두 수 모두의 배수는 아닙니다.
- 단, 입력한 수가 만족하는 문장은 모두 출력할 것.
 - 예) 4는 ii, iii번 문장이 모두 출력되어야 함.

Loops

Motivations

Suppose that you need to print a string (e.g., "Welcome to Java!") a hundred times. It would be tedious to have to write the following statement a hundred times:

```
System.out.println("Welcome to Java!");
```

So, how do you solve this problem?

Opening Problem

Problem: Print a string (e.g., "Welcome to Java!") a hundred times. How do you solve this problem?

100
times

```
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
...  
...  
...  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");
```

점검 문제

1. 다음 코드의 출력을 쓰시오.

```
for(int i = 1; i < 5; i++)  
    System.out.print(2 * i + " ");
```

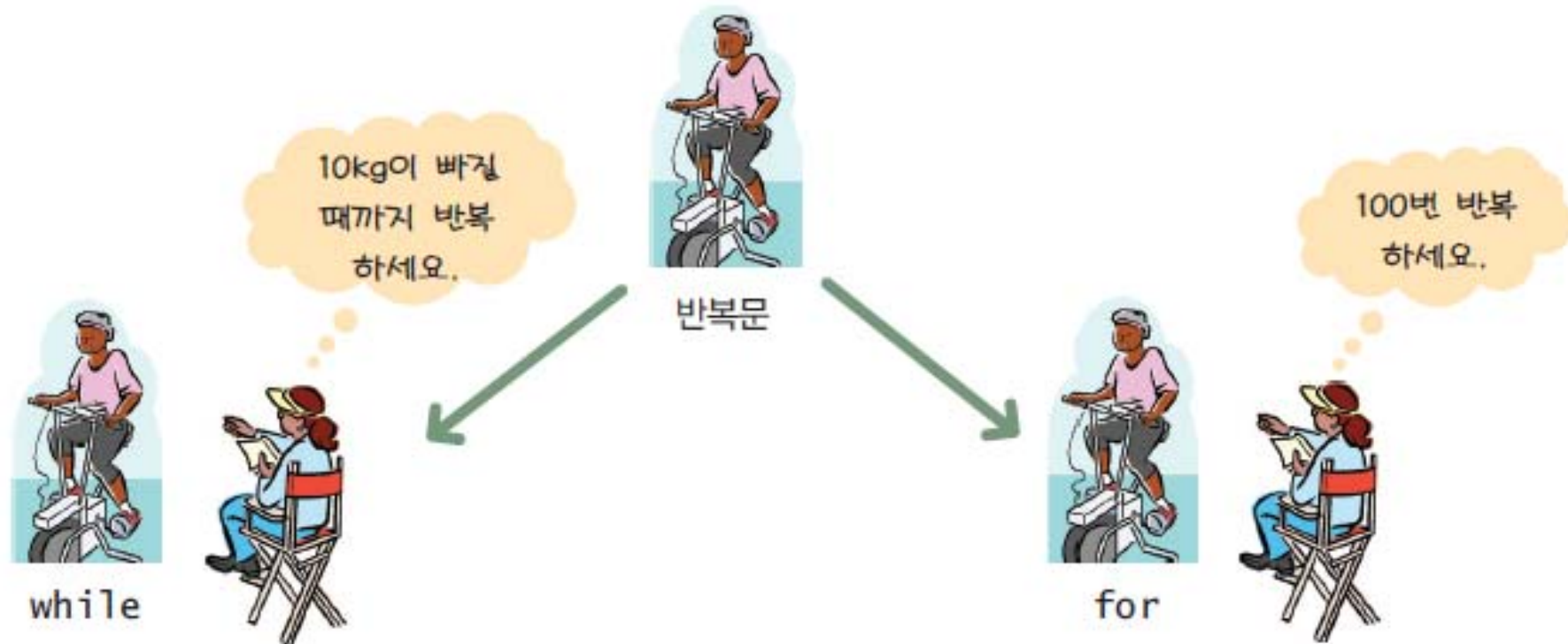
2. 다음 코드의 출력을 쓰시오.

```
for(int i = 10; i > 0; i = i - 2)  
    System.out.println("Student" + i);
```

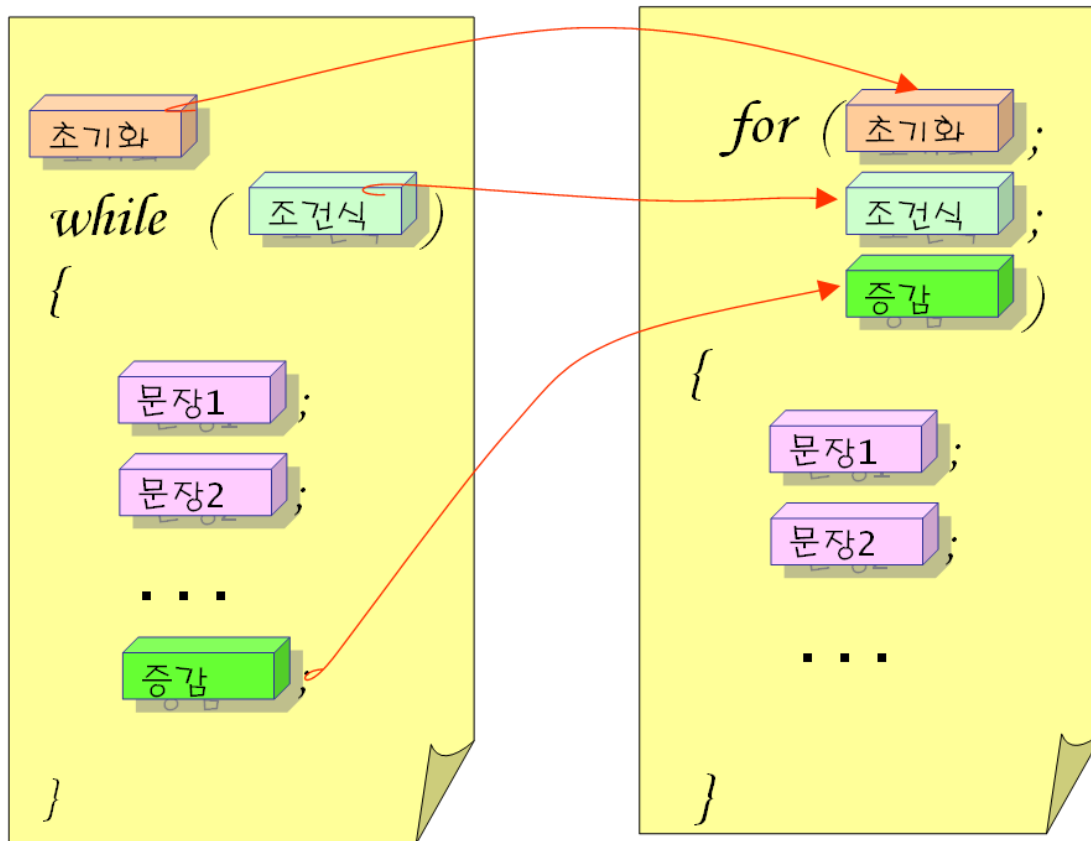
3. 다음 코드의 출력을 쓰시오.

```
for(int i = 1; i < 6; i++)  
    for(int j = 5; j >= 1; j--)  
        System.out.println(i + "곱하기" + j +  
"은 " + i*j);
```

반복문의 종류



while 루프와 for 루프와의 관계



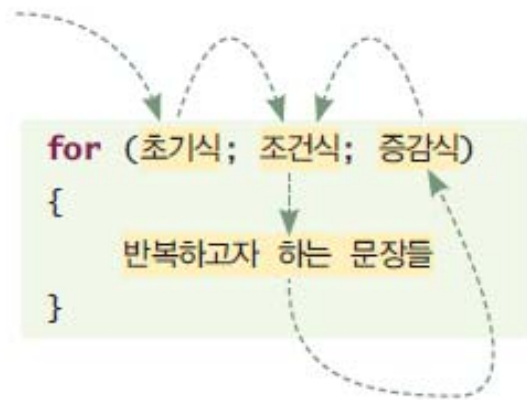
- The `while` loop is the most general repetition structure
- The `for` loop is logically equivalent to the counter-controlled `while` loop

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

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

for 문

- 정해진 횟수만큼 반복하는 구조



- ① 초기화를 실행한다.
- ② 반복 조건을 나타내는 조건식을 계산한다.
- ③ 수식의 값이 거짓이면 for 문의 실행이 종료된다.
- ④ 수식의 값이 참이면 문장이 실행된다.
- ⑤ 증감을 실행하고 ②로 돌아간다.

```
01 public class ForExample {
02     public static void main(String[] args) {
03         for (int i = 0; i < 5; i++) {
04             System.out.println("i의 값은: " + i);
05         }
06     }
07 }
```

i가 0에서 4까지 5
번 반복한다.

실행결과

i의 값은: 0
i의 값은: 1
i의 값은: 2
i의 값은: 3
i의 값은: 4

1~10까지의 합 구하기

Sum.java

```
01 public class Sum {  
02     public static void main(String[] args) {  
03         int sum = 0;  
04  
05         for (int i = 1; i <= 10; i++)  
06             sum += i; // sum = sum + i;와 같음  
07  
08         System.out.printf("1부터 10까지의 정수의 합 = %d\n", sum);  
09  
10     }  
11 }
```

← i가 1부터 10까지 1씩 증가되면서 sum에 더해진다.

실행결과

1부터 10까지의 정수의 합 = 55

Factorial 구하기

F:\WCOMP217\강의자료\5주차\source>java Factorial

정수: 10

10! = 3628800

F:\WCOMP217\강의자료\5주차\source>java Factorial

정수: 20

20! = 2432902008176640000

F:\WCOMP217\강의자료\5주차\source>java Factorial

정수: 30

21!을 구하는 과정에서 오버플로우가 발생했습니다(-4249290049419214848).

20! = 2432902008176640000

```
1  import java.util.Scanner;
2
3  public class Factorial{
4      public static void main(String [] args){
5          Scanner s = new Scanner(System.in);
6          long fact = 1L;
7          long overflowCheck = 1;
8          int n, i;
9
10         System.out.print("정수: ");
11         n = s.nextInt();
12
13         for(i=1; i<=n; i++){
14             fact *= i;
15             if(fact/i!=overflowCheck){
16                 System.out.println(i+"!을 구하는 과정에서 오버플로우가 발생했습니다("+fact+").");
17                 break;
18             }
19             else overflowCheck = fact;
20         }
21
22         System.out.println((i-1)+"! = "+overflowCheck);
23     }
24 }
```

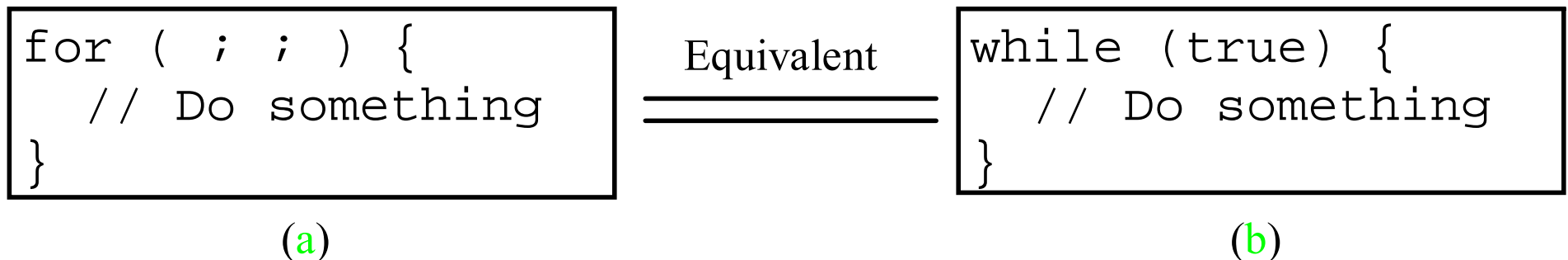

Note

Multiple initial-actions and **multiple** action-after-each-iterations in a for loop and are allowed with comma (,) separator.

```
for (int i = 1; i < 100; System.out.println(i++));
```

```
for (int i = 0, j = 0; (i + j < 10); i++, j++) { /* Do something */ }
```

If the loop-continuation-condition in a for loop is omitted, it is implicitly **true**.




Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:


Logic
Error

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



Caution, cont.

Similarly, the following loop is also wrong:

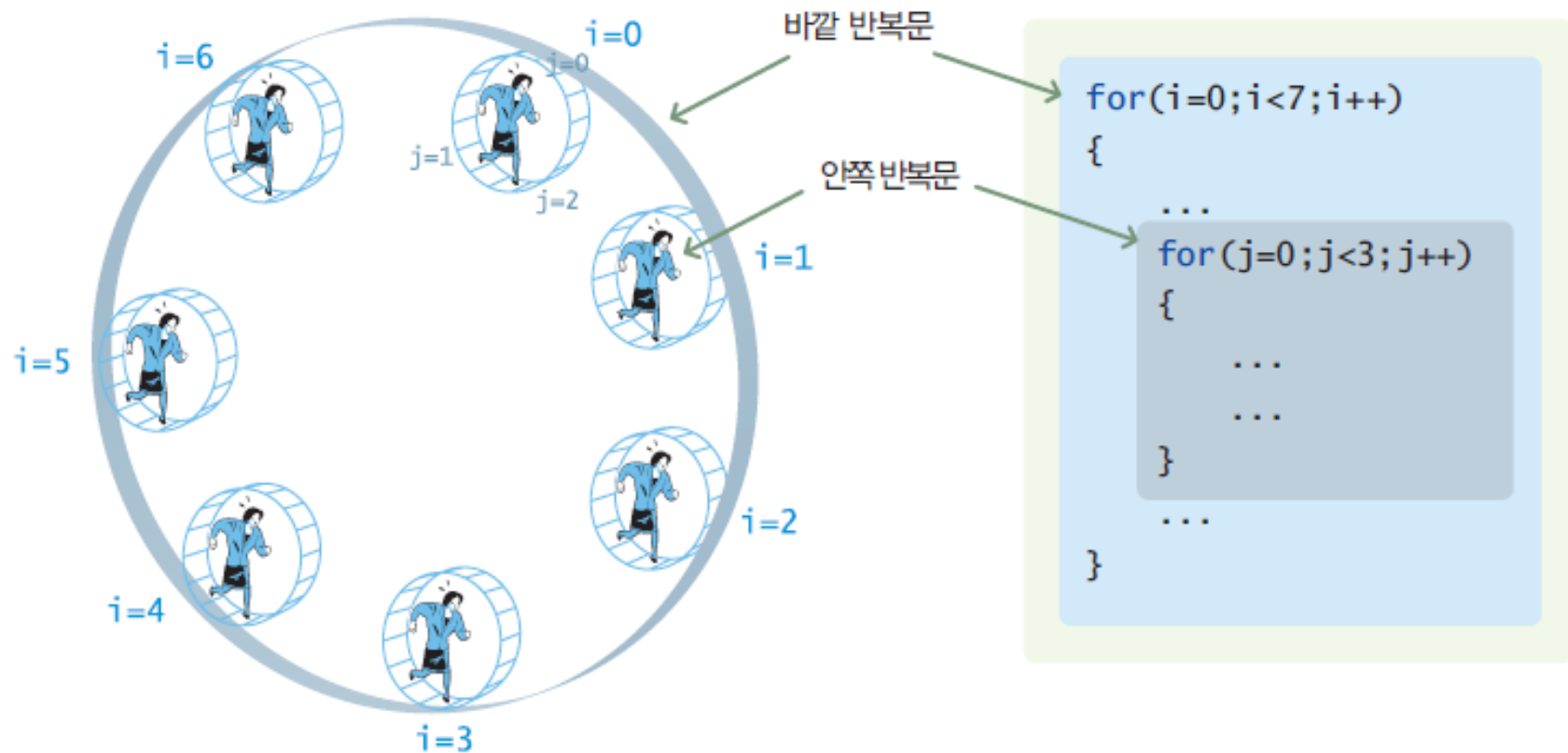
```
int i=0;
while (i < 10);  Logic Error
{
    System.out.println("i is " + i);
    i++;
}
```

In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;
do {
    System.out.println("i is " + i);
    i++;
} while (i<10);
```

중첩 반복문

- 중첩 반복문(nested loop): 반복문 안에 다른 반복문이 위치



Nested Loops

- A nested loop is a loop within a loop
 - Any type and any number of loops can be nested

```
8 public class NestedLoops
9 {
10     public static void main ( String[] args )
11     {
12         for( int i = 1; i <= 10; i++ ) // outer loop
13             for( int j = 1; j <= i; j++ ) // inner loop
14                 if( i % j == 0 )
15                     System.out.println( i + " is divisible by " + j );
16
17     } // end main
18
19 } // end class
```

Figure 4.26 Program code for NestedLoops.java (continued)

별 표시 예제

- * 1개를 사용
 - 가로에 * 10개
 - 세로로 5줄 표시

실행결과

```
*****  
*****  
*****  
*****  
*****
```

```
1 public class NestedLoop{  
2     public static void main(String args []){  
3         for(                ;      ; i++) {    //5줄 표시  
4  
5  
6  
7  
8         }  
9     }  
10 }
```

10까지의 약수 출력

1의 약수	는	1	
2의 약수	는	1	2
3의 약수	는	1	3
4의 약수	는	1	2 4
5의 약수	는	1	5
6의 약수	는	1	2 3 6
7의 약수	는	1	7
8의 약수	는	1	2 4 8
9의 약수	는	1	3 9
10의 약수	는	1	2 5 10

```
3 import java.util.Scanner;
4 public class NL{
5     public static void main(String args []){
6         for(int i=1; i<=10; i++){
7             System.out.printf("%2d의 약수는 ", i);
8             for(int j=1; j<=i; j++){
9                 if(i%j==0) System.out.print(j+" ");
10            }
11            System.out.println();
12        }
13    }
14 }
```

직각삼각형 출력

```
직각 삼각형을 출력합니다.  
단 : 5  
*  
**  
***  
****  
*****
```

```
18 import java.util.Scanner;  
19  
20 public class NL{  
21     public static void main(String args []){  
22         int dan;  
23         Scanner s = new Scanner(System.in);  
24         System.out.print("직각 삼각형을 출력합니다. \n단 : ");  
25         dan = s.nextInt();  
26         for(int i=0; i<dan; i++){  
27             for(int j=0; j<=i; j++){  
28                 System.out.print("*");  
29             }  
30             System.out.println();  
31         }  
32     }  
33 }
```


다른 직각삼각형들과 피라미드

직각 삼각형을 출력합니다.

단 : 10

```
      *
     **
    ***
   ****
  *****
 *****
*****
*****
*****
*****
```

직각 삼각형을 출력합니다.

단 : 10

```
*****
*****
*****
*****
*****
*****
****
***
**
*
```

피라미드를 출력합니다.

단 : 10

```
      *
     ***
    *****
   *******
  *********
 *****
*****
*****
*****
*****
*****
*****
*****
*****
```

직각 삼각형을 출력합니다.

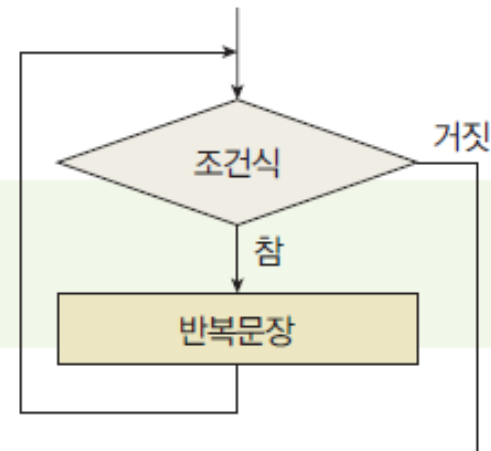
단 : 10

```
*****
*****
*****
*****
*****
*****
****
***
**
*
```

반복문- while

- 같은 처리 과정을 되풀이하는 것이 필요할 때
 - while : 주어진 조건이 만족되는 동안 문장들을 반복 실행
 - for : 정해진 횟수만큼 반복하는 구조
 - do-while : 반복 조건을 루프의 끝에서 검사

```
while (조건식) {  
    반복문장  
}
```



```
public class LoopExample1 {  
    public static void main(String[] args) {  
        int i = 0;  
        while (i < 5) {  
            System.out.println("정수: " + i);  
        }  
    }  
}
```

i가 5보다 작을 때까지
반복한다.

실행결과

정수: 0
정수: 1
정수: 2
정수: 3
정수: 4

Introducing while Loops

```
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java");
    count++;
}
```

```
while ( condition ) {
    action
}
```

- The *condition* is called a *pretest* condition
 - pretest: “test before action”
- loop body: the code between the braces
- 1 iteration: a single execution of the loop body

Flowchart for a `while` Loop

An `if` statement

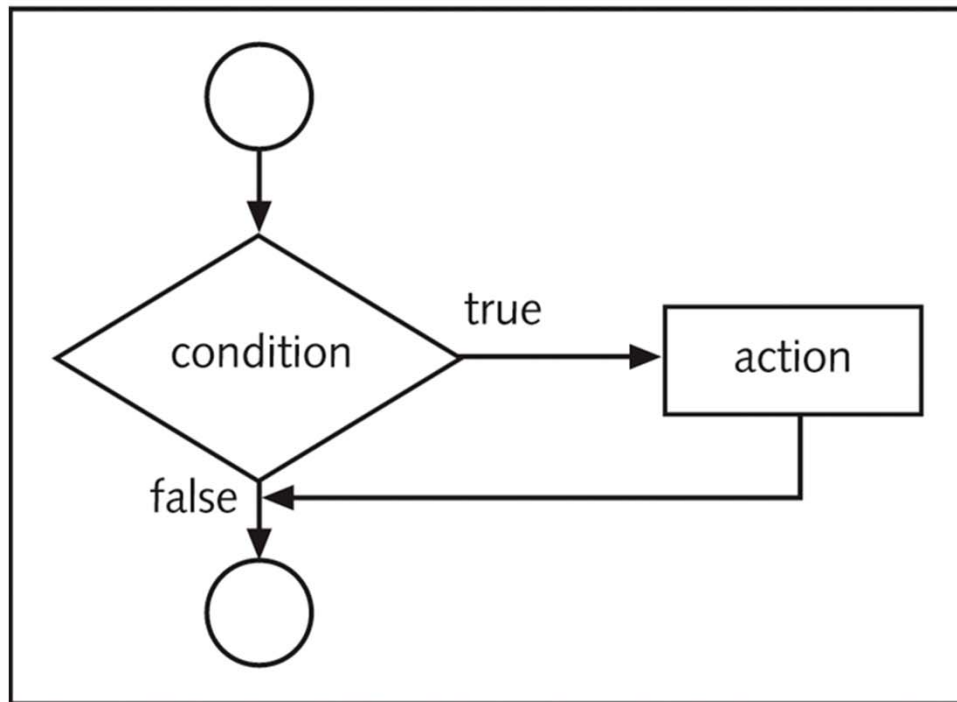


Figure 4.3 Flowchart symbols for the `if` statement (left) and the `while` statement (right)

LoopExample1.java

```
public class LoopExample1 {  
    public static void main(String[] args) {  
        int j=0;  
        while (j < 5) {  
            System.out.println(j + " ");  
            j++;  
        }  
    }  
}
```

/* result:

구구단 출력 예제

- 사용자로부터 출력할 단 입력 받음
- while문을 통해 입력받은 수에 1~9를 차례로 곱함
- 결과 출력 : 곱셈 결과는 전체 2자리로 잡을 것

```
----3단----
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
```

```
1  import java.util.Scanner;
2  public class Gugudan{
3      public static void main(String [] args){
4          int dan;
5          Scanner s = new Scanner(System.in);
6          System.out.print("출력할 단 : ");
7          dan = s.nextInt();
8          System.out.printf("\n----%d단----\n",dan);
9          int i = 1;
10         while(i<=9){
11             System.out.printf("%d X %d = %3d\n", dan, i, dan*i);
12             // System.out.println(dan+"X"+i+"="+dan*i);
13             i++; //++i; ??
14         }
15     }
16 }
```

최대 공약수 예제

- 사용자로부터 두 정수 입력 받음
- 유클리드 알고리즘 사용하여 최대 공약수 계산

```
① 두 수 가운데 큰 수를 x, 작은 수를 y라 한다.  
② y가 0이면 공약수는 x와 같다.  
③  $r \leftarrow x \% y$   
④  $x \leftarrow y$   
⑤  $y \leftarrow r$   
⑥ 단계 ②로 되돌아간다.
```

– 입력 받은 두 수 중

- 큰 수 : a
- 작은 수 : b

```
정수 1 : 15  
정수 2 : 12  
최대공약수는 3입니다.
```

최대 공약수 구하기 코드

```
1  import java.util.Scanner;
2  public class Gcd{
3      public static void main(String [] args){
4          int a, b;
5          Scanner s = new Scanner(System.in);
6          System.out.print("정수 1 : ");
7          a = s.nextInt();
8          System.out.print("정수 2 : ");
9          b = s.nextInt();
10         // 입력 받은 a, b중 큰 수를 a, 작은 수를 b로 두기
11         int tmp;
12         if (a<b){
13             tmp=a;
14             a=b;
15             b=tmp;
16         }
17         //유클리드 알고리즘
18         while (b!=0) {
19             tmp=b;
20             b=a%b;
21             a=tmp;
22         }
23         System.out.println("최대공약수는 "+a+"입니다.");
24     }
25 }
```


Caution

- Don't use floating-point values for equality checking in a loop control.
 - Floating-point values are approximations for real values
 - Ex.) 1.7 may be stored as 1.69999999999999

Not recommended	Good
<pre>double item = 1; double sum = 0; while (item != 0) { // No guarantee item will be 0 sum += item; item -= 0.1; } System.out.println(sum);</pre>	<pre>int counter = 10; double step = 0.1; double sum = 0; while (counter != 0) { // Equality test on integers sum += step * (double) counter; counter -= 1; } System.out.println(sum);</pre>

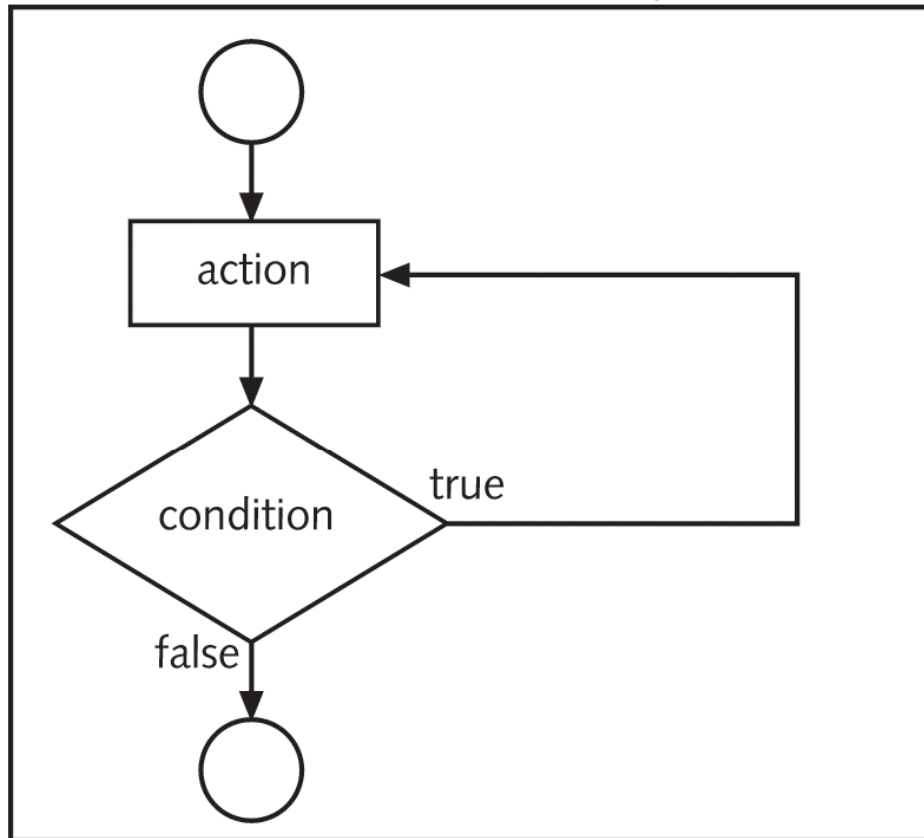
do...while Loop

```
do {  
    action  
} while ( condition );
```

- The loop body must execute at least once
- *posttest*. After the first execution of the loop body, the condition is tested
 - In the case of while loop, the loop body may never executes, because if the condition is false in the beginning, the loop ends

The do...while Flowchart

A do...while loop



A while loop

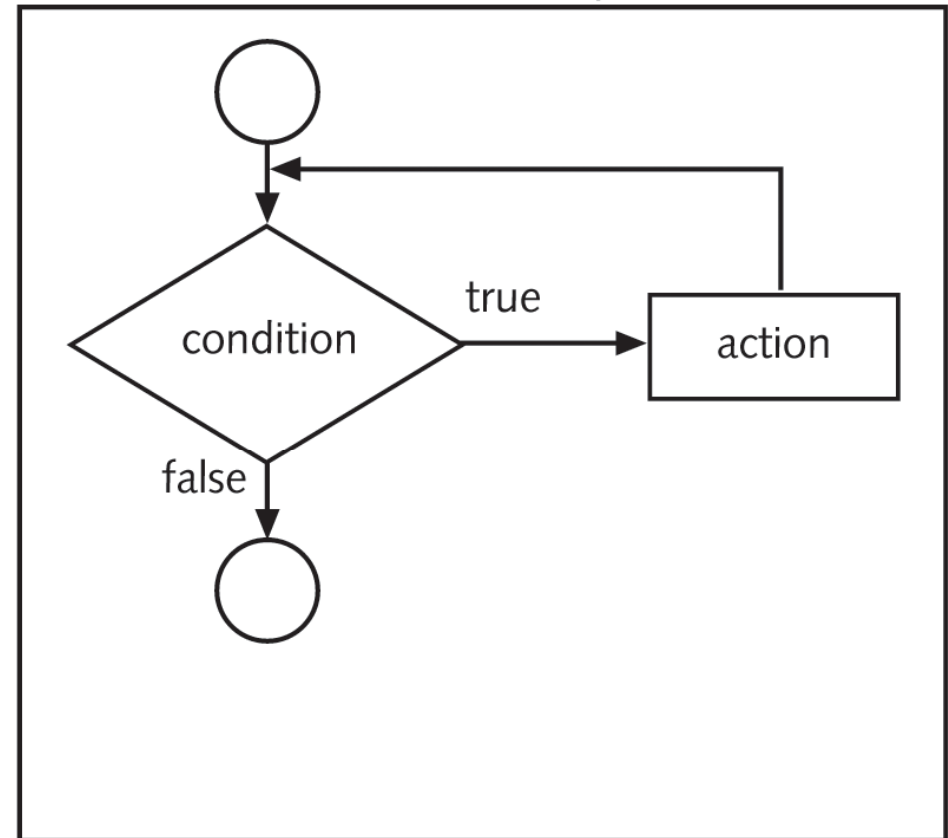


Figure 4.16 Flowcharts for a do...while loop (left) and a while loop (right)

LoopExample3.java

```
import java.util.Scanner;

public class LoopExample3 {
    public static void main(String[] args) {
        int i;
        Scanner sc = new Scanner(System.in);
        System.out.print("Which number? ");
        i = sc.nextInt();
        do {
            System.out.println("i = " + i);
            i++;
        }
        while (i < 9);
    }
}
```

```
/*
$ javac LoopExample3.java
$ java LoopExample3
Which number? 3
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
$ java LoopExample3
Which number? 10
i = 10
$ java LoopExample3
Which number? 20
i = 20
*/
```

숫자 추측 게임 예제

0~100까지의 정수를 입력하여 답을 맞춰보세요: 78
정답은 좀 더 큰 수 입니다.
0~100까지의 정수를 입력하여 답을 맞춰보세요: 90
정답은 좀 더 작은 수 입니다.
0~100까지의 정수를 입력하여 답을 맞춰보세요: 100
정답은 좀 더 작은 수 입니다.
0~100까지의 정수를 입력하여 답을 맞춰보세요: 79
축하합니다. 79은 정답입니다. 4번만에 맞추셨습니다.


- 추측할 숫자 고정
 - `int jungdap = 79`
- 사용자가 숫자(`jungdap`)을 맞추면 성공하는 게임
 - 사용자로부터 숫자 입력 받음
 - `int userInput`
 - 사용자가 입력한 값이 `jungdap`이면 "성공" 출력
 - 맞출때까지의 시도 횟수 출력
 - `int tries`
 - 아니라면 `userInput`과 `jungdap`의 대소를 비교하여 출력
 - 사용자가 `jungdap`을 맞출때까지 반복
 - 일단 한번은 무조건 실행해야 하므로
 - `while`문 대신
 - `do-while` 사용

숫자 추측 게임 코드

```
1  import java.util.Scanner;
2
3  public class Game{
4      public static void main(String args[]){
5          int jungdap = 79;           //미리 정해둔 정답
6          int userInput;              //사용자에게 입력받을 정수를 저장할 변수
7          int tries = 0;              //게임 시도 횟수를 저장할 변수
8
9          Scanner s = new Scanner(System.in); //스캐너 클래스 생성
10         do{
11             System.out.print("0~100까지의 정수를 입력하여 답을 맞춰보세요 : ");
12             //사용자에게 값 입력 요구
13             userInput = s.nextInt(); //사용자가 입력한 값 userInput에 저장
14             if(userInput>jungdap) System.out.println("답은 좀더 작은 수 입니다.");
15             else if(userInput<jungdap) System.out.println("답은 좀더 큰 수 입니다.");
16             tries++;
17         }while(userInput!=jungdap); //사용자가 입력한 값이 정답과 다른 경우 계속 반복
18
19         System.out.println("축하합니다. "+userInput+"은 정답입니다. "+tries+"번만에 맞추셨습니다.");
20         //루프를 빠져나왔다는 건 정답을 맞췄단 말이므로 정답 메시지 출력
21     }
22 }
```

Caution, cont.

Similarly, the following loop is also wrong:

```
int i=0;
while (i < 10);  Logic Error
{
    System.out.println("i is " + i);
    i++;
}
```

In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;
do {
    System.out.println("i is " + i);
    i++;
} while (i<10);
```

중간 점검 문제

1. 다음 코드의 출력을 쓰시오.

```
int n = 10;
while (n > 0) {
    System.out.println(n);
    n = n - 3;
}
```

2. 1번 문제의 반복 구조를 do-while로 변경하면 출력이 어떻게 변화되는가?

The `break` Statement in Loops

- Recall that `break` was used to exit early from a `switch` structure
 - The `break` statement in a `switch` statement ensured that the `switch` exited after a particular case was executed
- A `break` statement in a loop causes that particular loop to terminate
- A `break` statement can be used in any kind of loop (`for`, `while`, `do...while`)

break 예제

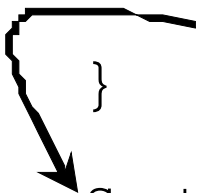
- 사용자로부터 점수 입력 받아 입력 받은 점수의 평균 구하기
- 사용자가 음수를 입력하면 종료하기

수를 입력하세요 : 1
수를 입력하세요 : 2
수를 입력하세요 : 3
수를 입력하세요 : -1
입력하신 수 들의 평균은 2.0

```
4 import java.util.Scanner;
5
6 public class BTest{
7     public static void main(String args[]){
8         double input;
9         double total=0;
10        int n=0;
11        Scanner s = new Scanner(System.in);
12        while(true){
13            System.out.print("수를 입력하세요 : ");
14            input = s.nextDouble();
15            if(input<0) break; //만약 입력이 음수라면 루프 빠져나감
16            total+=input;
17            n++;
18        }
19        System.out.println("입력하신 수 들의 평균은 "+total/n);
20    }
21 }
```

break

```
public class TestBreak {  
    public static void main(String[] args) {  
        int sum = 0;  
        int number = 0;  
  
        while (number < 20) {  
            number++;  
            sum += number;  
            if (sum >= 100)  
                break;  
        }  
        System.out.println("The number is " + number);  
        System.out.println("The sum is " + sum);  
    }  
}
```



continue 예제

- 사용자로부터 문장 입력받고,
- 문장에서 소문자 n의 개수 출력하기

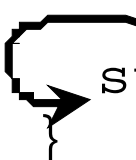
C:\Users\shin\Documents\IDEA\workspace\src\java> java Contest
단어 또는 문장 : No news is good news.
4 번째
17 번째
입력하신 문장에서 'n'의 개수는 2개 입니다.

```
1 import java.util.Scanner;
2 public class Contest{
3     public static void main(String args []){
4         String userMsg;
5         Scanner s = new Scanner(System.in);
6         System.out.print("단어 또는 문장 : ");
7         userMsg = s.nextLine();
8         int count = 0;        //n의 개수 저장할 변수
9
10        for(int i=0; i<userMsg.length(); i++){           //스트링.length() <-해당 스트링의 길이 반환
11            if(userMsg.charAt(i)!='n') continue;         //스트링.charAt(i) <-해당 스트링에서 i번째 문자 반환
12            count++;
13            System.out.printf("%2d번째 %n", i+1);
14        }
15
16        System.out.println("입력하신 문장에서 %n의 개수는 " + count + "개 입니다. ");
17    }
18 }
```

continue

- The `continue` statement causes the current loop iteration to be skipped

```
public class TestContinue {  
    public static void main(String[] args) {  
        int sum = 0;  
        int number = 0;  
  
        while (number < 20) {  
            number++;  
            if (number == 10 || number == 11)  
                continue;  
            sum += number;  
        }  
  
        System.out.println("The sum is " + sum);  
    }  
}
```



A Caveat About `break` and `continue`

- Some programmers prefer to avoid `break` and `continue` because they make the program harder to understand
- Some programmers feel `break` and `continue` are useful in some situations
- Recommendation
 - Try not to use `break` and `continue`,
 - Write a loop to avoid them

중간 점검 문제

1. 다음 코드의 출력을 쓰시오.

```
int n = 12;
while (n > 0) {
    n -= 2;
    if( n == 6 )
        break;
    System.out.println(n);
}
```

2. 1번 문제에서 break를 continue로 변경하면 어떻게 되는가?