



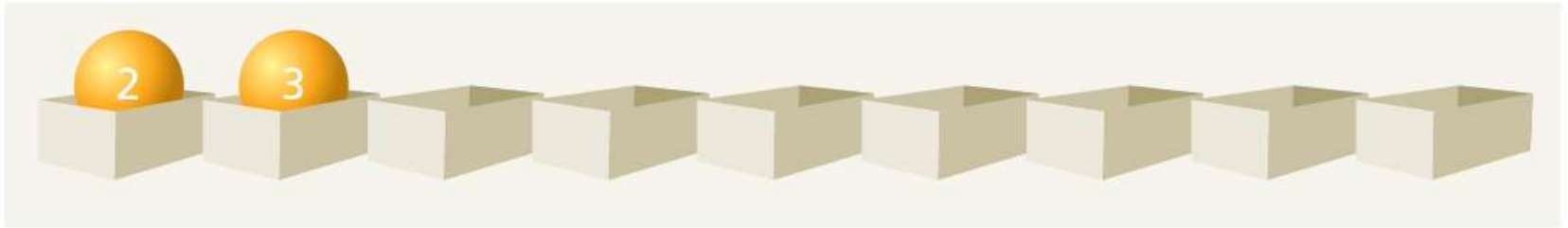
어서와 JAVA는 처음이지!

제2장 자바 프로그래밍 기초



변수

- 변수(variable)는 데이터를 담아두는 상자





변수 선언

전체적인 구조



형식

`int`

`value;`

자료형

변수 이름

value

전체적인 구조



형식

`int`

`value`

`=`

`9`

초기값

value

9



자료형의 종류



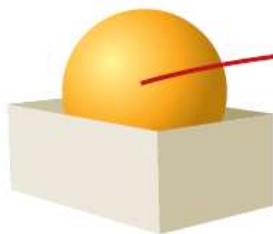
기초형
실제 값이 저장

정수형: byte, short, int, long

실수형: float, double

논리형: boolean

문자형: char



참조형
실제 객체를 가리키는
주소 저장



클래스, 인터페이스, 배열



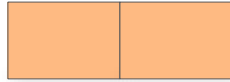
기초 자료형

byte



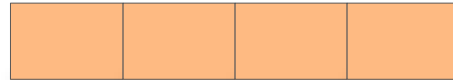
-128부터 127

short



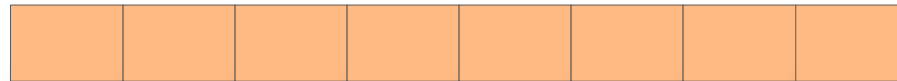
-32768부터 32767

int

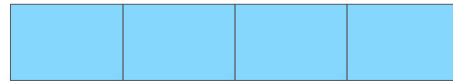


약 -21억부터 21억까지

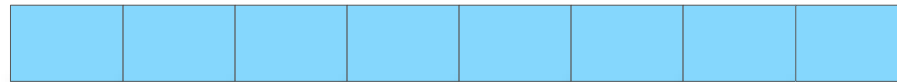
long



float



double



boolean



true, false

char

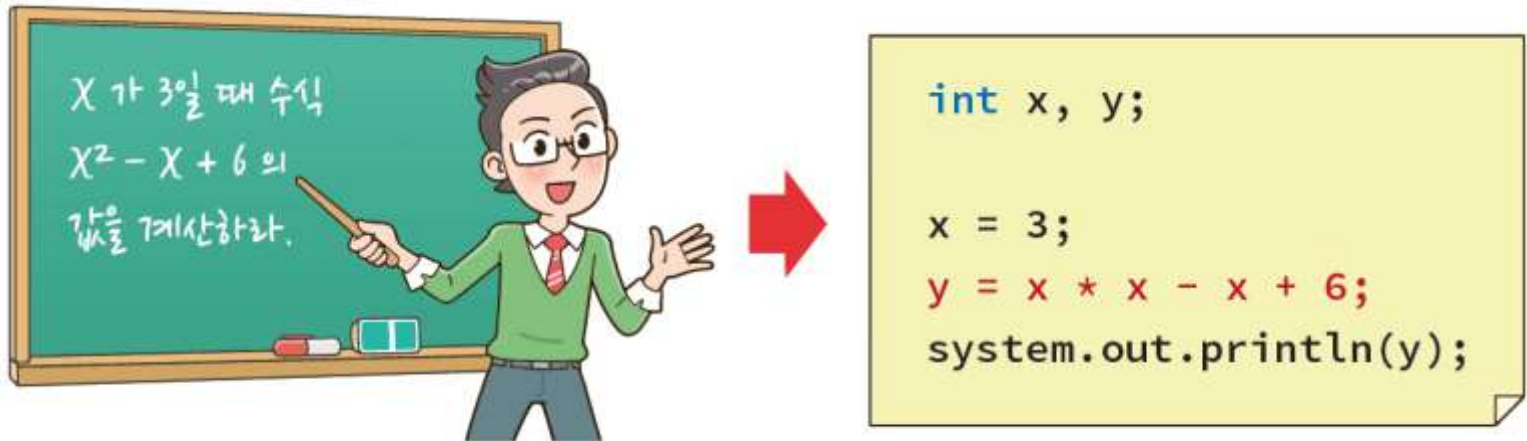


유니코드



수식

- 수식이란 상수나 변수, 함수와 같은 피연산자들과 연산자의 조합



The illustration shows a cartoon teacher with glasses and a green sweater pointing at a chalkboard. The chalkboard contains the text: x 가 3일 때 수식 $x^2 - x + 6$ 의 값을 계산하라. A red arrow points from the chalkboard to a yellow box containing the following Java code:

```
int x, y;  
  
x = 3;  
y = x * x - x + 6;  
system.out.println(y);
```

그림 2-2 • 수식의 예



대입 연산자

- 대입 연산자(=)는 왼쪽 변수에 오른쪽 수식의 값을 계산하여 저장
- 대입 연산자 == 할당 연산자 == 배정 연산자라고도 한다.

`x = 100; // 상수 100을 변수 x에 대입한다.`

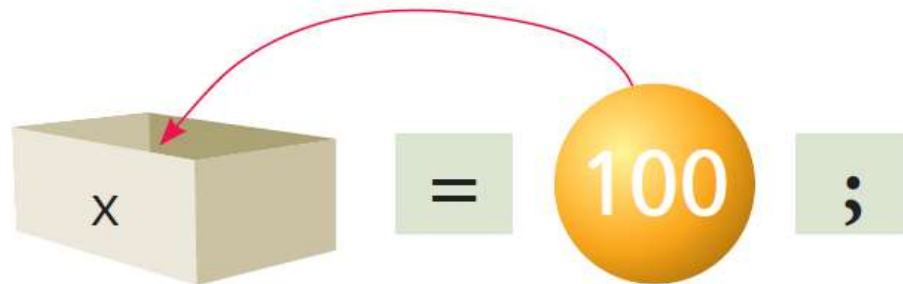


그림 2-3 • 대입 연산자는 변수에 값을 저장하는 연산자이다.



산술 연산자

연산자	기호	의미	예
덧셈	+	x와 y를 더한다	$x+y$
뺄셈	-	x에서 y를 뺀다.	$x-y$
곱셈	*	x와 y를 곱한다.	$x*y$
나눗셈	/	x를 y로 나눈다.	x/y
나머지	%	x를 y로 나눌 때의 나머지값	$x\%y$

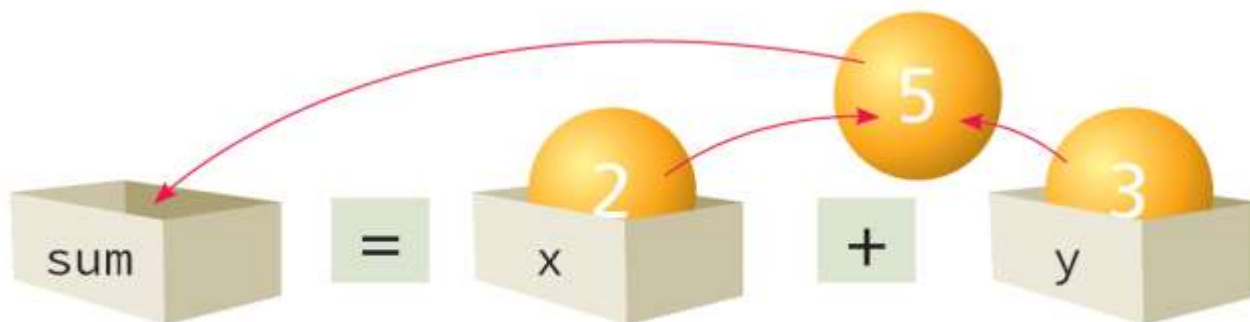


그림 2-4 • 산술 연산의 과정: 먼저 x와 y에서 값을 가져와서 덧셈연산이 수행되고 그 결과값이 sum에 저장된다.



중감 연산자

연산자	의미
$++x$	x 값을 먼저 증가한 후에 다른 연산에 사용한다. 이 수식의 값은 증가된 x 값이다.
$x++$	x 값을 먼저 사용한 후에, 증가한다. 이 수식의 값은 증가되지 않은 원래의 x 값이다.
$--x$	x 값을 먼저 감소한 후에 다른 연산에 사용한다. 이 수식의 값은 감소된 x 값이다.
$x--$	x 값을 먼저 사용한 후에, 감소한다. 이 수식의 값은 감소되지 않은 원래의 x 값이다.



예제: 증감 연산자

직접 입력
하여 확인



UnaryOperator.java

```
01 public class UnaryOperator {
02     public static void main(String[] args) {
03         int x = 1;
04         int y = 1;
05
06         int nextx = ++x; // x의 값이 사용되기 전에 증가된다. nextx는 2가 된다.
07         int nexty = y++; // y의 값이 사용된 후에 증가된다. nexty는 1이 된다.
08         System.out.println(nextx);
09         System.out.println(nexty);
10     }
11 }
```

실행결과



```
2
1
```



관계 연산자

표 2-3 • 관계 연산자

연산자 기호	의미	사용예
==	x와 y가 같은가?	$x == y$
!=	x와 y가 다른가?	$x != y$
>	x가 y보다 큰가?	$x > y$
<	x가 y보다 작은가?	$x < y$
>=	x가 y보다 크거나 같은가?	$x >= y$
<=	x가 y보다 작거나 같은가?	$x <= y$



예제: 관계 연산자

직접 입력
하여 확인



ComparisonOperator.java

```
01 public class ComparisonOperator {  
02  
03     public static void main(String[] args){  
04         int x = 3;  
05         int y = 4;  
06         System.out.print((x == y) + " ");  
07         System.out.print((x != y) + " ");  
08         System.out.print((x > y) + " ");  
09         System.out.print((x < y) + " ");  
10         System.out.print((x <= y) + " ");  
11     }  
12 }
```

실행결과



false true false true true



논리 연산자

연산자 기호	사용예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
	x y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참



예제: 논리 연산자

직접 입력
하여 확인



LogicalOperator.java

```
01 public class LogicalOperator {  
02  
03     public static void main(String[] args){  
04         int x = 3;  
05         int y = 4;  
06         System.out.println((x == 3) && (y == 7)) ;  
07         System.out.println((x == 3 || y == 4)) ;  
08     }  
09 }
```

AND 연산

OR 연산

실행결과



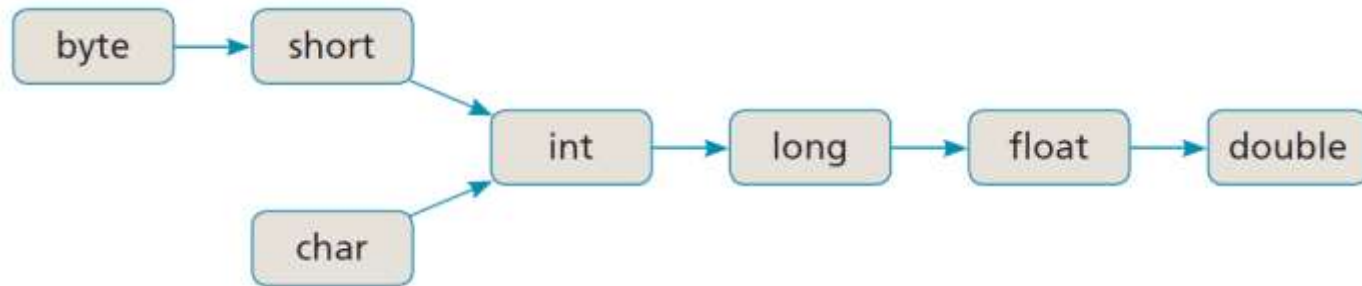
```
false  
true
```



형변환

○ 자동적인 형변환

- ◎ 피연산자 중 하나가 double형이면 다른 피연산자도 double형으로 변환된다.
- ◎ 피연산자 중 하나가 float형이면 다른 피연산자도 float형으로 변환된다.
- ◎ 피연산자 중 하나가 long형이면 다른 피연산자도 long형으로 변환된다.
- ◎ 그렇지 않으면 모든 피연산자는 int형으로 변환된다.





강제적인 형변환

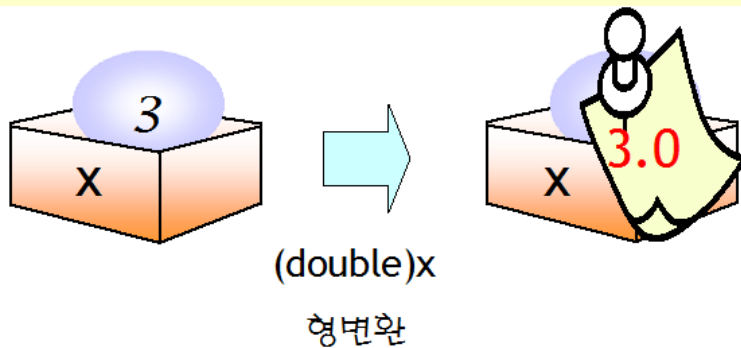
전체적인 구조



형식

(새로운 자료형) 수식;

$y = (\text{double}) x;$





예제: 형변환

직접 입력
하여 확인



LogicalOperator.java

```
01 public class TypeConversion {
02     public static void main(String args[]) {
03         int i;
04         double f;
05
06         f = 5 / 4;
07         System.out.println(f);
08
09         f = (double) 5 / 4;
10         System.out.println(f);
11
12         i = (int) 1.3 + (int) 1.8;
13         System.out.println(i);
14     }
15 }
```

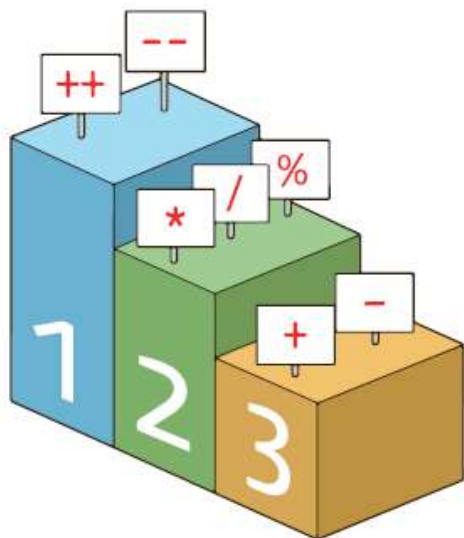
5 / 4는 피연산자가 정수이므로 정수 연산으로 계산되어서 1이 된다. 이것이 double 형 변수로 대입되므로 올림 변환이 발생하여 1.0이 1에 저장된다.

(double)5 / 4에서는 먼저 형변환 연산자가 우선순위가 높기 때문에 먼저 실행되어서 정수 5가 부동소수점수 5.0으로 변환된다. 5.0 / 4는 피연산자중 하나가 double형이므로 4도 double형으로 자동 형변환되고 5.0 / 4.0으로 계산되어서 1.25가 수식의 결과값이 된다.

수식 (int)1.3 + (int)1.8에서는 1.3과 1.8이 모두 1로 변환되므로 변수 i에는 1 + 1하여 2가 저장된다.



연산자의 우선순위



우선 순위는 어떤 연산자를
먼저 계산할지를 나타내는
순위입니다.



그림 2-5 • 산술 연산자의 우선 순위

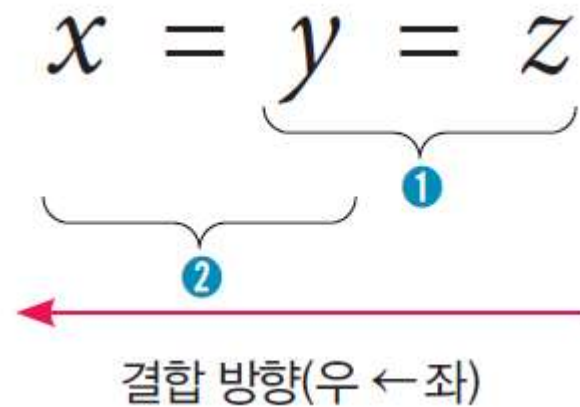
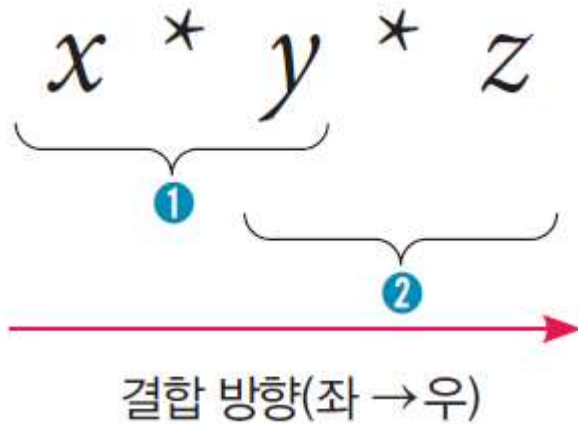
$$x + y * z$$

The diagram shows the expression $x + y * z$ with curly braces indicating the order of operations. A brace labeled '1' is under the $y * z$ part, and a larger brace labeled '2' is under the entire $x + y * z$ expression.



결합 규칙

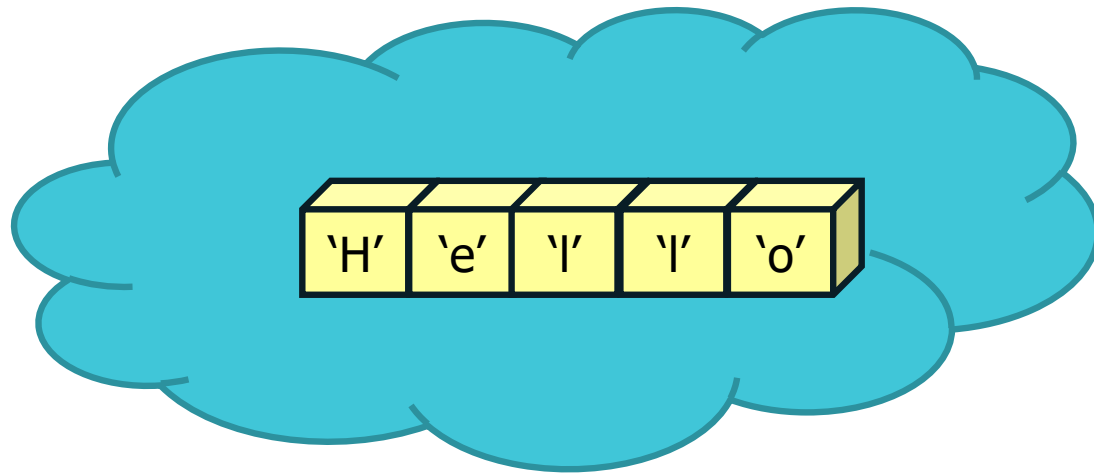
- 동일한 우선 순위의 연산이 있는 경우에 무엇을 먼저 수행하느냐에 대한 규칙





문자열

- 자바에서 문자열(**string**)은 문자들의 모임이다. 예를 들어서 문자열 “Hello” 는 H, e, l, l, o 등의 5개의 유니코드 문자로 구성되어 있다.
- String 클래스가 제공된다.





예제: 문자열 프로그램

+ 연산자로 문자열을 합칠 수 있다.

직접 입력
하여 확인



StringTest.java

```
01 public class StringTest {  
02     public static void main(String args[]) {  
03         String s1 = "Hello World!";  
04         String s2 = "I'm a new Java programmer!";  
05  
06         System.out.println(s1 + "\n" + s2);  
07     }  
08 }
```

실행결과



```
Hello World!  
I'm a new Java programmer!
```

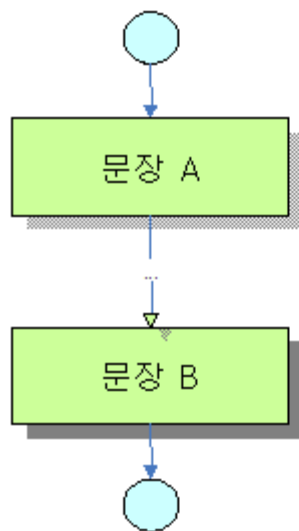


어서와 JAVA는 처음이지!

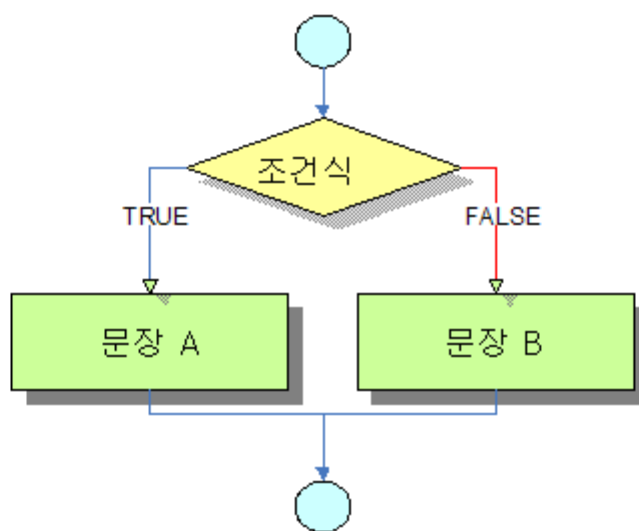
제3장 선택과 반복



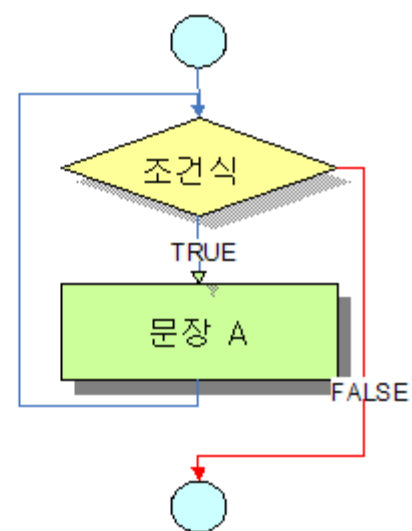
3가지의 제어 구조



순차 구조



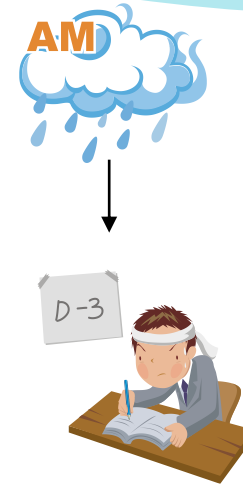
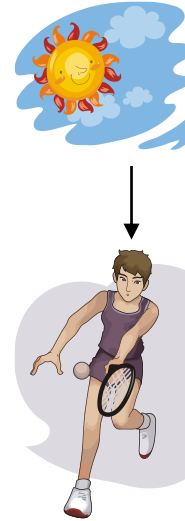
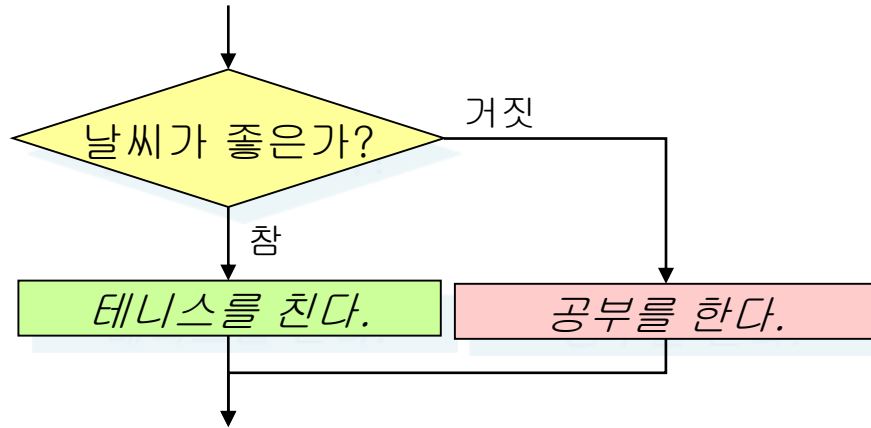
선택 구조



반복 구조



IF-ELSE 문



if(조건식)

문장1;

else

문장2;

조건식이 참이면 실행된다.

조건식이 거짓이면 실행된다.



IF-ELSE 선택 구조

전체적인 구조



형식

조건식

```
if( number > 0 )  
{  
    System.out.println("양수입니다.");  
}  
else  
{  
    System.out.println("0이거나 음수입니다.");  
}
```

조건식이 참일 때 실행되는
문장, then 절이라고 한다.

조건식이 참이 아닐 때 실행되는
문장, else 절이라고 한다.



예제: 짝수, 홀수 구별하기

키보드에서 입력받은 정수가 홀수인지 짝수인지를 말해주는 프로그램을 작성하여 보자.



정수를 입력하시오: 2
입력된 정수는 짝수입니다.
프로그램이 종료되었습니다.



직접 입력
하여 확인



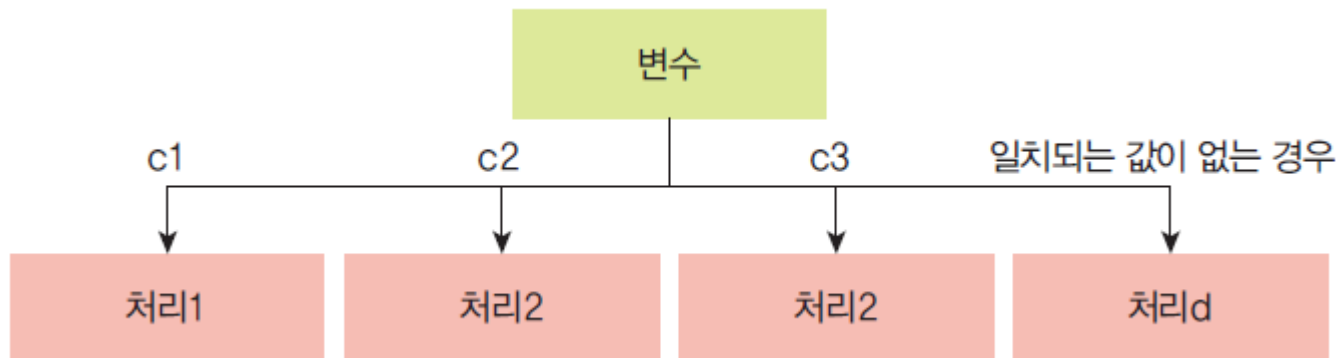
EvenOdd.java

```
01 import java.util.Scanner;
02
03 public class EvenOdd {
04     public static void main(String[] args) {
05         // if 문을 사용하여 홀수와 짝수를 구별하는 프로그램
06         int number;
07         Scanner input = new Scanner(System.in);
08         System.out.print("정수를 입력하시오: ");
09         number = input.nextInt();
11         if (number % 2 == 0) {
12             System.out.println("입력된 정수는 짝수입니다.");
13         } else {
14             System.out.println("입력된 정수는 홀수입니다.");
15         }
16
17         System.out.println("프로그램이 종료되었습니다. ");
18     }
19 }
```



SWITCH 문

- 만약 가능한 실행 경로가 여러 개인 경우에는 switch 문을 사용





SWITCH 선택 구조

전체적인 구조



형식

```
switch(변수)
{
    case c1:
        처리문장1;
        break;
    case c2:
        처리문장2;
        break;
    ...
    default:
        처리문장d;
        break;
}
```

변수와 일치하는 값을
가진 case 절이 실행됩니다.

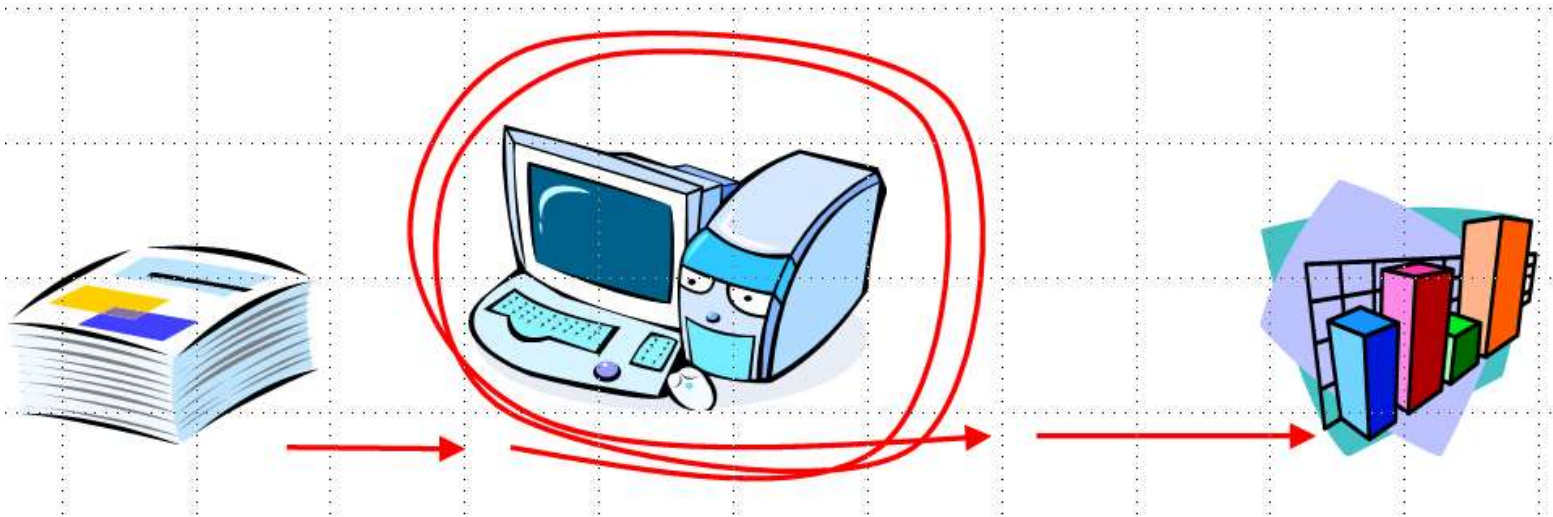




반복문

Q) 반복 구조는 왜 필요한가?

A) 같은 처리 과정을 되풀이하는 것이 필요하기 때문이다. 학생 30명의 평균 성적을 구하려면 같은 과정을 30번 반복하여야 한다.

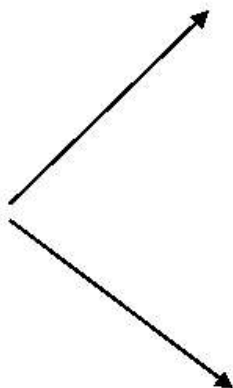




반복문의 종류



반복문



while



10kg이 빠질
때까지 반복
하세요



for



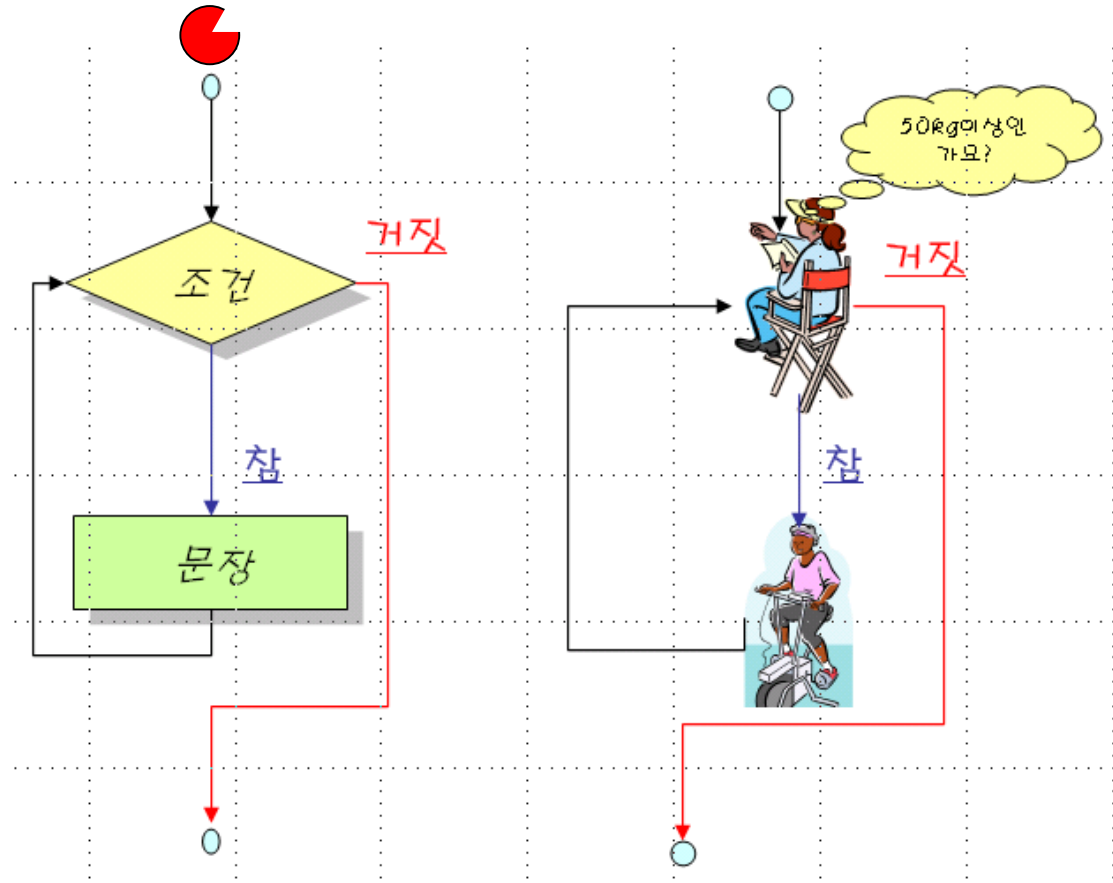
100번 반복
하세요,



WHILE 문

- 주어진 조건이 만족되는 동안 문장들을 반복 실행한다.

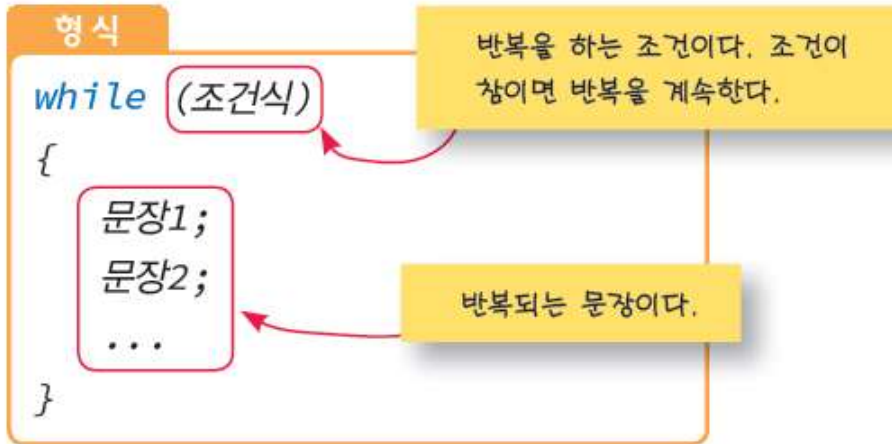
while(조건식)
문장;





WHILE 문의 구조

전체적인 구조



조건이 참이면
반복하는 구조입니다.





예제: 구구단 출력

실행결과



구구단 중에서 출력하고 싶은 단을 입력하시오: 9

$$9 \times 1 = 9$$

$$9 \times 2 = 18$$

$$9 \times 3 = 27$$

...



직접 입력
하여 확인



LoopExample2.java

```
01 // while 문을 이용한 구구단 출력 프로그램
02 import java.util.*;
03
04 public class LoopExample2 {
05     public static void main(String[] args) {
06         int n;
07         int i = 1;
08         System.out.print("구구단 중에서 출력하고 싶은 단을 입력하시오: ");
09         Scanner scan = new Scanner(System.in);
10         n = scan.nextInt();
11         while (i <= 9) {
12             System.out.println(n + "*" + i + "=" + n * i);
13             i++;
14         }
15     }
16 }
```

여기서는 먼저 사용자로부터 출력하고 싶은 구구단의 단수를 받아서 변수 n에 저장한다. 여기서의 루프 제어 변수는 i이다. i의 초기값이 0이 아니고 1인 것에 유의하라. 구구단은 1부터 곱해야 하기 때문에 0이 아니고 1로 초기화를 하였다. 그리고 반복 루프도 9보다 작거나 같을 때까지 반복하도록 하였다.



예제: $(1+2+3+\dots+9+10)$ 계산하기



1 2 3 4 5 6 7 8 9 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 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

합계=55



CalSum.java

```
01 public class CalSum {  
02     public static void main(String[] args) {  
03  
04         int i = 1;  
05         int sum = 0;  
06         while (i <= 10) {  
07             sum = sum + i;  
08             i++;  
09         }  
10         System.out.println("합계=" + sum);  
11  
12     }  
13 }
```

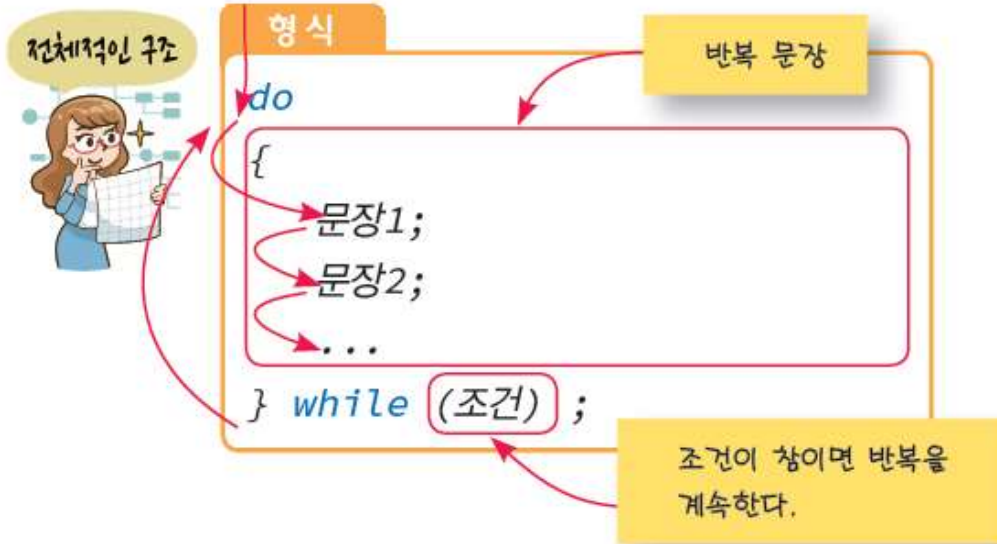
sum은 0으로 초기화되어야 한다.

i는 1부터 시작한다.

i가 10이하이면 i의 값을
sum에 더한다.

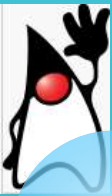


DO-WHILE 문



반복 조건을 가장
아래에서 검사합니다.





DO-WHILE 문의 예

직접 입력
하여 확인



DoWhile1 .java

```
01 public class DoWhile1 {  
02     public static void main(String[] args){  
03         int i = 10;  
04         do {  
05             System.out.println("i의 값: " + i);  
06             i++;  
07         } while (i < 3);  
08     }  
09 }
```

실행결과



i의 값: 10



예제: 정확한 입력받기



올바른 월을 입력하시오 [1-12]: 13

올바른 월을 입력하시오 [1-12]: 14

올바른 월을 입력하시오 [1-12]: 0

올바른 월을 입력하시오 [1-12]: 1

사용자가 입력한 월은 1



직접 입력
하여 확인



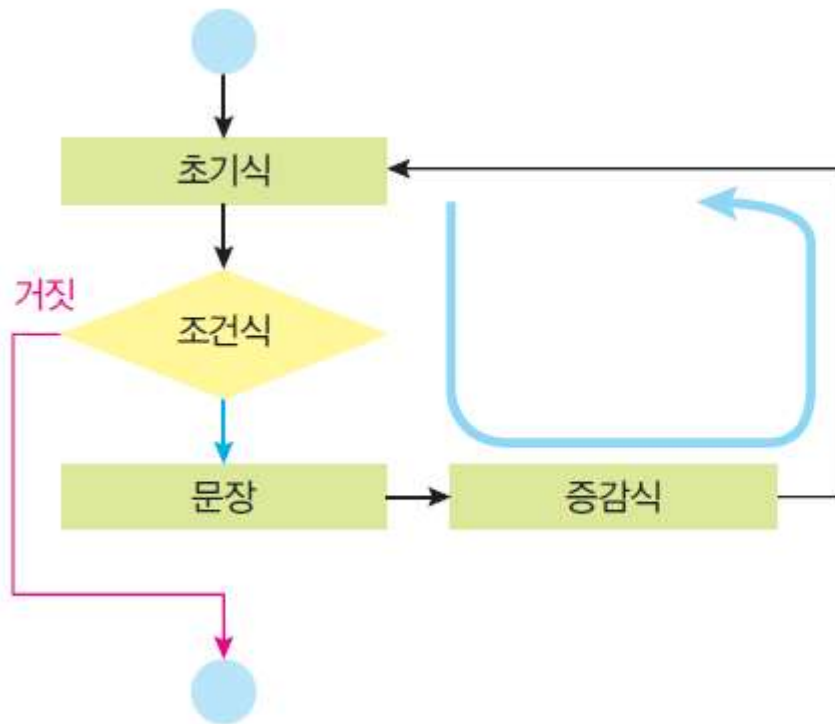
CheckInput.java

```
01 import java.util.Scanner;
02
03 public class CheckInput {
04     public static void main(String args[]) {
05         Scanner input = new Scanner(System.in);
06         int month;
07         do {
08             System.out.print("올바른 월을 입력하시오 [1-12]: ");
09             month = input.nextInt();
10
11             } while (month < 1 || month > 12);
12         }
13     System.out.println("사용자가 입력한 월은 " + month);
14 }
```



FOR 루프

- 정해진 횟수만큼 반복하는 구조





FOR 문의 구조

전체적인 구조



형식

```
for(초기식; 조건식; 증감식) {  
    반복문장;  
}
```

형식

```
for(i=0; i<5; i++) {  
    System.out.print("Hello");  
}
```



예제: 정수의 합 계산하기

실행결과



1부터 10까지의 정수의 합 = 55

직접 입력
하여 확인



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;  
07  
08         System.out.printf("1부터 10까지의 정수의 합 = %d", sum);  
09  
10     }  
11 }
```

for 문 안의 초기식에서
제어 변수를 선언할 수 있다.

예제: 팩토리얼 계산하기

직접 입력
하여 확인



Factorial.java

```
01 import java.util.*;
02
03 public class Factorial {
04     public static void main(String[] args) {
05         long fact = 1;
06         int n;
07
08         System.out.printf("정수를 입력하시요:");
09         Scanner scan = new Scanner(System.in);
10         n = scan.nextInt();
11
12         for (int i = 1; i <= n; i++)
13             fact = fact * i;
14
15         System.out.printf("%d!은 %d입니다.", n, fact);
16
17     }
18 }
```

먼저 변수 fact를 long형으로 정의한다. 팩토리얼의 값은 생각보다 아주 커질 수 있다. 여기서 fact의 초기값은 반드시 1이어야 한다. 0이면 안 된다. 왜냐하면 팩토리얼은 정수를 전부 곱해서 계산하는 것이므로 초기값이 0이면 결과는 0이 되어 버린다. 따라서 반드시 1로 초기화를 시켜야 한다.

실행결과



```
정수를 입력하시요:20
20!은 2432902008176640000입니다.
```



중첩 반복문

전체적인 구조



형식

```
for(i=0; i<5; i++) {  
    for(k=0; k<5; k++) {  
        반복문장;  
    }  
}
```

외부 반복문

내부 반복문



예제 : 사각형 모양 출력하기



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




NestedLoop.java

```
01 import java.util.*;
02
03 public class NestedLoop {
04     public static void main(String[] args) {
05
```

```
06         for (int y = 0; y < 5; y++) {
07             for (int x = 0; x < 10; x++)
08                 System.out.print("*");
09
10             System.out.println("");
11         }
12
13     }
14 }
```

위의 프로그램을 실행하면 50개의 *가 화면에 5X10의 정사각형 모양으로 출력된다. *를 출력하는 문장의 외부에는 두개의 for 루프가 중첩되어 있다. 외부의 for 루프는 변수 y를 0에서 4까지 증가시키면서 내부의 for 루프를 실행시킨다. 내부의 for 루프는 변수 x를 0에서 9까지 증가시키면서 print() 메소드를 호출한다. 내부 for 루프가 한번 실행될 때마다 화면에는 한 줄의 *가 그려진다. 내부 for 루프가 한 번씩 종료될 때마다 줄바꿈 문자가 화면에 출력되어 다음 줄로 넘어가게 된다.



BREAK 문

직접 입력
하여 확인



Averager.java

```
01 import java.util.*;
02
03 public class Averager {
04     public static void main(String[] args) {
05         int total = 0;
06         int count = 0;
07         Scanner scan = new Scanner(System.in);
08         while (true) {
09             System.out.print("점수를 입력하시오: ");
10             int grade = scan.nextInt();
11             if (grade < 0)
12                 break;
13
14             total += grade;
15             count++;
16         }
17         System.out.println("평균은 " + total / count);
18     }
19 }
```



CONTINUE 문

직접 입력
하여 확인



ContinueTest.java

```
01 public class ContinueTest {  
02     public static void main(String[] args) {  
03  
04         String s = "no news is good news";  
05         int n = 0;  
06  
07         for (int i = 0; i < s.length(); i++) {  
08             // n 이 나오는 회수를 센다.  
09             if(s.charAt(i) != 'n')  
10                 continue;  
11  
12             // n의 개수를 하나 증가한다.  
13             n++;  
14         }  
15         System.out.println("문장에서 발견된 n의 개수 " + n);  
16     }  
17 }
```



어서와 JAVA는 처음이지!

제4장 배열



배열이 필요한 이유

- 예를 들어서 학생이 10명이 있고 이들의 성적의 평균을 계산한다고 가정하자. 학생이 10명이므로 10개의 변수가 필요하다.

```
int s0, s1, s2, s3, s4, s5, s6, s7, s8, s9;
```

- 하지만 만약 학생이 100명이라면 어떻게 해야 하는가?

```
int s0, s1, s2, s3, s4, s5, s6, s7, s8, s9,...,s99;
```

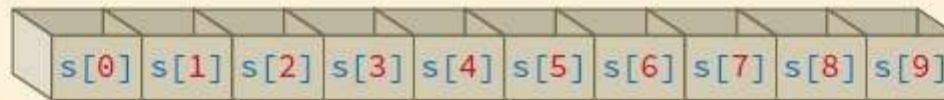


배열의 인덱스

- 다음과 같은 배열을 가정하자.

```
int[] s = new int[10];
```

배열은 하나의 이름을 공유한다.



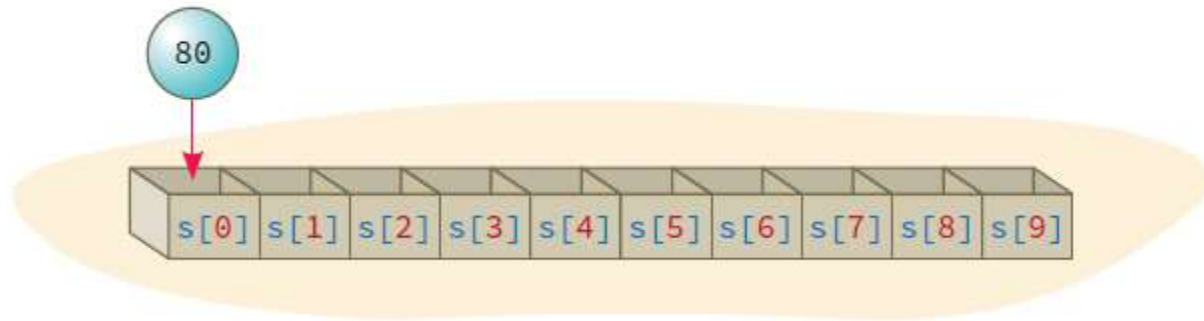
- 배열 요소에는 번호가 붙어 있는데 이것을 인덱스(index)라고 부른다.
- 첫 번째 요소의 번호는 0이고, 마지막 요소의 번호는 9가 된다.



인덱스를 통한 요소의 접근

- 배열은 변수들이 모인 것이니, 배열을 이루고 있는 배열 요소는 하나의 변수로 생각하면 된다.
- 배열의 첫 번째 요소에 80을 저장하려면 다음과 같이 한다.

```
s[0] = 80;
```





배열의 인덱스 범위

- 프로그래머가 인덱스가 범위를 벗어나지 않았는지를 확인하고 책임을 져야 한다.

```
int[] scores = new int[5];  
scores[0] = 10;  
scores[1] = 20;  
scores[2] = 30;  
scores[3] = 40;  
scores[4] = 50;  
scores[5] = 60;
```



Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
at ArrayTest4.main(ArrayTest4.java:16)





배열의 초기화

직접 입력
하여 확인



ArrayTest3.java

```
01 public class ArrayTest3 {  
02     public static void main(String[] args) {  
03         int[] scores = { 10, 20, 30, 40, 50 };  
04         for (int i = 0; i < scores.length; i++)  
05             System.out.print(scores[i]+" ");  
06     }  
07 }
```

각 배열은 length라는 필드를 가지고 있다.
length 필드는 배열의 크기를 나타낸다.
따라서 이것을 이용하면 배열의 크기만큼
반복을 시킬 수 있다.

실행결과



10 20 30 40 50

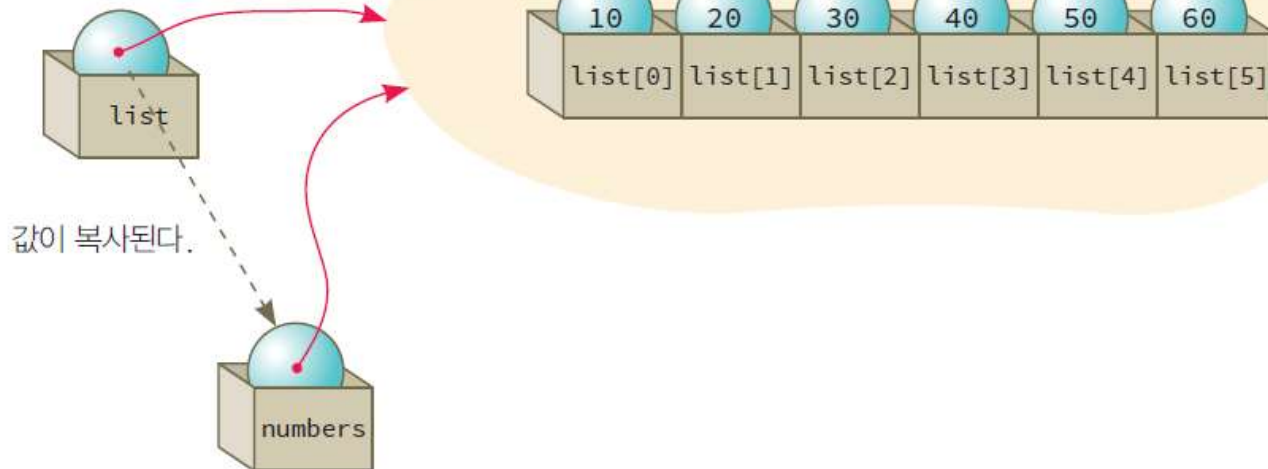
```
int[] scores = { 10, 20, 30, 40, 50 };
```




배열의 복사

○ 배열 참조 변수의 복사

```
int [] list = { 10, 20, 30, 40, 50 };  
int [] numbers = list;
```





배열의 복사

- 한 배열의 모든 값을 다른 배열로 복사하고 싶다면 Arrays 클래스의 `copyOf()` 메소드를 사용
- (예) `int [] list_copy = Arrays.copyOf(list, list.length);`



2차원 배열

전체적인 구조

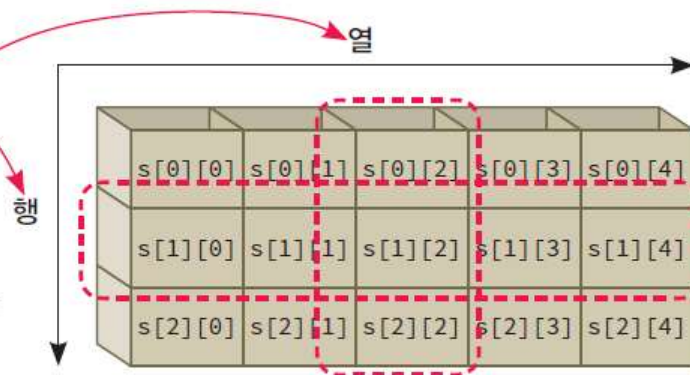


```
int[][] s = new int[3][5];
```

2개의 대괄호가
2차원 배열을
나타낸다.

행의 개수

열의 개수





2차원 배열의 초기화

```
int[][] testArray = {  
    {10, 20, 30},  
    {40, 50, 60},  
    {70, 80, 90}  
};
```



2차원 배열 예제

직접 입력
하여 확인



ArrayTest6.java

```
01 import java.util.Scanner;
02
03 public class ArrayTest6 {
04     public static void main(String[] args) {
05         int[][] array = {
06             { 10, 20, 30, 40 },
07             { 50, 60, 70, 80 },
08             { 90, 100, 110, 120 }
09     };
10
11     for (int r = 0; r < array.length; r++) {
12         for (int c = 0; c < array[r].length; c++) {
13             System.out.println(r + "행" + c + "열:" + array[r][c]);
14         }
15     }
16 }
17 }
```

실행결과



```
0행0열:10
0행1열:20
...
2행2열:110
2행3열:120
```



Q & A

